***EISTI – ING1 Responsable de projet : M Sébastien LABORIE***

2010/2011

**Projet Génie Logiciel**

**Livrable 2 : Version intermédiaire du projet**

**Gallet Mériadec, Klelifa Sarah, Lion Cédric, Razavet Maël**

Sommaire

[Introduction 3](#_Toc289883364)

[Architecture des classes (UML) 4](#_Toc289883365)

[Mode d’emploi de l’application : 5](#_Toc289883366)

[Etape 1 : Création de compte 5](#_Toc289883367)

[Etape 2 : Identification 7](#_Toc289883368)

[Etape 3 : Création d’un nouveau problème : 8](#_Toc289883369)

[Etape 4 : Enregistrer ses problèmes 10](#_Toc289883370)

[Répartition du projet 11](#_Toc289883371)

[Stockage et gestion des données 12](#_Toc289883372)

[Gestion des comptes utilisateurs 13](#_Toc289883373)

[Interface graphique 14](#_Toc289883374)

[Présentation générale de l’interface 14](#_Toc289883375)

[Gestion des problèmes 15](#_Toc289883376)

[Gestion des contraintes 15](#_Toc289883377)

[Avancement global du projet 16](#_Toc289883378)

[Remarques personnelles 17](#_Toc289883379)

[Remarques relatives à Maël 17](#_Toc289883380)

[Remarques relatives à Mériadec 17](#_Toc289883381)

[Remarques relatives à Cédric 18](#_Toc289883382)

[Remarques relatives à Sarah 18](#_Toc289883383)

Introduction

Le projet dont nous sommes en charge et sur lequel nous travaillons actuellement consiste à gérer une interface permettant à un utilisateur de résoudre des problèmes d’optimisation mathématique.

A ce jour, nous vous présentons un rendu intermédiaire qui va vous permettre de vous rendre compte de l’avancement du projet. Toutes les parties que nous avions à réaliser pour ce livrable ont été effectuées et nous vous donnerons d’ailleurs plus de détails sur cela dans ce livrable. Nous avons également avancé plus que prévu pour ce livrable, mais là encore, nous vous expliquerons cela au cours de ce compte rendu intermédiaire.

Mais de quoi est donc composé ce livrable, à quoi va-t-il bien vous servir ?

Ce livrable est composé du diagramme de classes de l’application, de l’application elle-même ainsi que de ce rapport. Il vous détaille la manière dont nous avons raisonné pour construire nos classes JAVA ainsi que la manière dont nous nous sommes répartis les tâches à l’intérieur même de notre groupe. Ce rapport explique l’avancement détaillé du projet, l’avancement de chacune des tâches et leurs explications, le mode d’emploi de l’application ainsi que les remarques personnelles des développeurs.

Architecture des classes (UML)



Ce diagramme UML permet de modéliser tout ce qui concerne la partie modèle de notre projet. Il s’agit de la base principale du projet car il s’agit des méthodes qui vont être appelées par les modules graphiques et de contrôle.

Mode d’emploi de l’application :

Ce paragraphe est consacré à l’utilisation de notre logiciel. Il vous explique en détail, pour chaque étape, comment procéder pour faire ce que vous voulez avec images à l’appui.

## Etape 1 : Création de compte

Lorsque vous arrivez pour la première fois sur notre application, vous ne pouvez pas vous connecter directement. En effet, vous n’êtes pas connu de notre programme et ce dernier ne peut donc vous donner l’accès nécessaire.

Afin de vous connecter, il faut donc vous créer un compte. Lorsque vous arrivez sur la page d’accueil, cliquer sur ‘Créer un compte’ : 

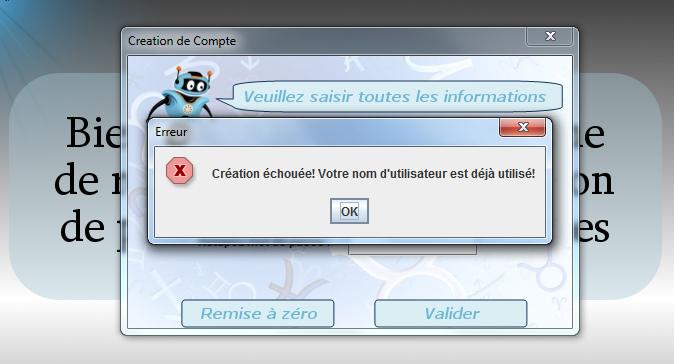
Dès lors que vous avez cliqué dessus, une fenêtre s’ouvre devant vous, mettant ainsi en arrière plan temporairement la fenêtre de départ. Remplissez les champs demandés en suivant l’exemple ci-dessous :



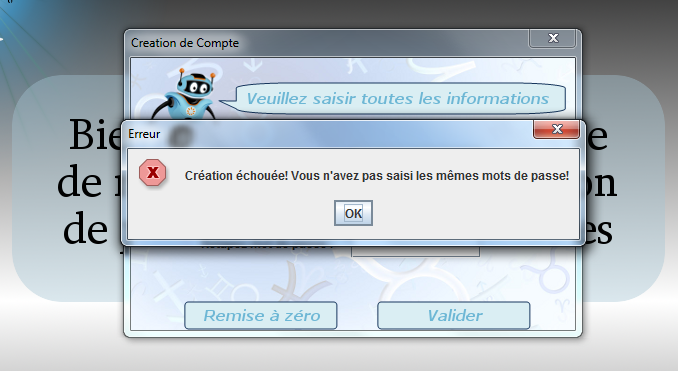
Puis cliquez sur ‘Valider’.

**Note importante !**

En cas de choix d’un nom d’utilisateur déjà utilisé, vous serez confronté à l’erreur suivante :



De même, si vous ne saisissez pas deux fois le même mot de passe, vous verrez apparaître à l’écran :



## Etape 2 : Identification

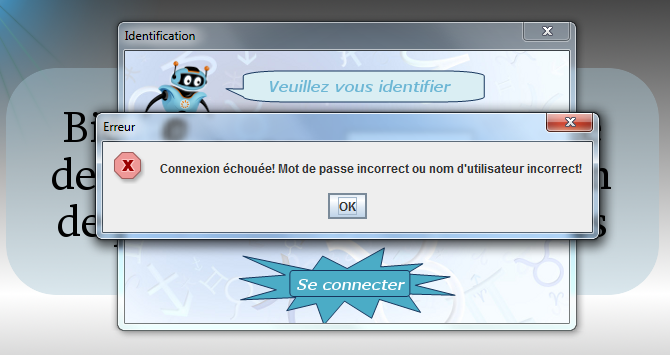
 Une fois cette étape effectuée, la fenêtre se referme, laissant place à l’image de fond. A présent, vous pouvez vous identifier en utilisant votre nom d’utilisateur ainsi que votre mot de passe, précédemment remplis.

Pour cela, cliquer sur ‘S’identifier’ :

Tout comme précédemment, remplissez les champs qui vous sont demandés :

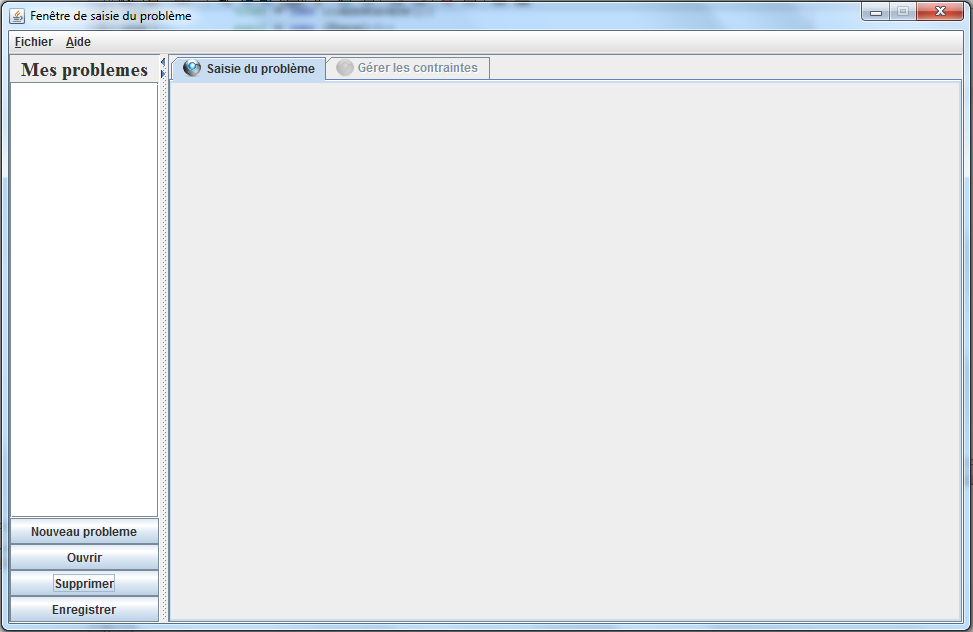


Puis cliquez sur ‘Valider’.



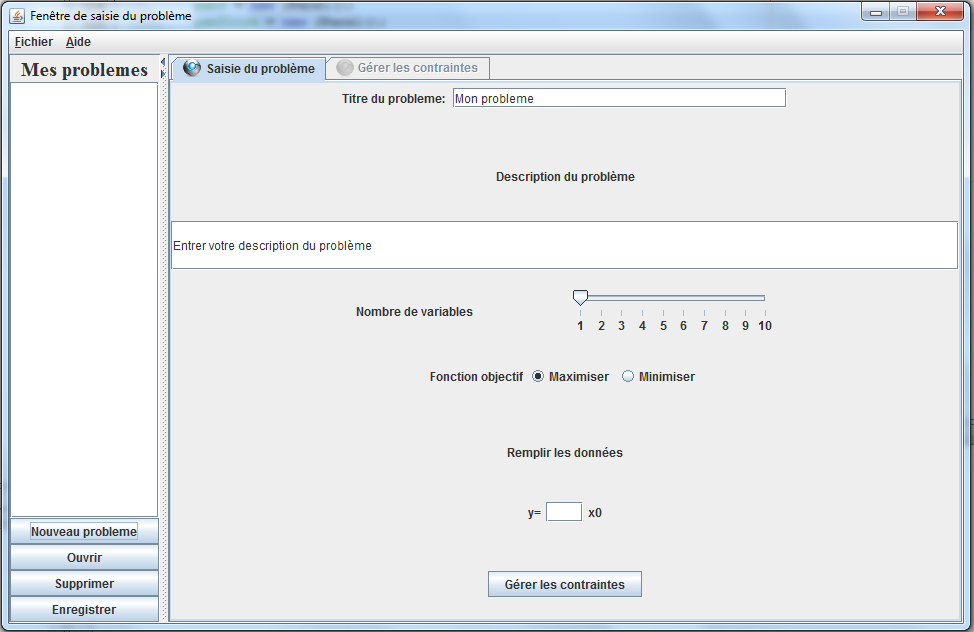
## Etape 3 : Création d’un nouveau problème :

Une fois l’identification faite avec succès, vous arrivez sur une nouvelle page qui se compose de 2 zones (une à droite et une à gauche) ainsi qu’un menu d’édition en haut de celle-ci (voir image ci-dessous).



Pour créer un nouveau problème, rien de plus simple ! Il vous suffit simplement de cliquer sur le bouton *‘Nouveau problème’ :*

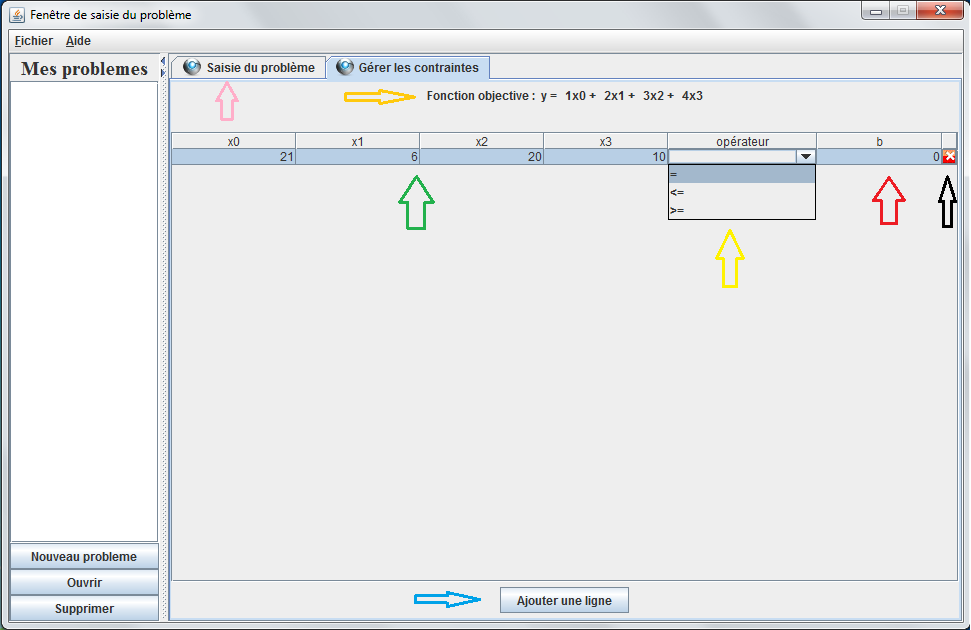
Vous arrivez alors sur cette interface :



Vous n’avez plus qu’à remplir les champs demandés.

Une fois cette tâche effectuée, cliquer sur gérer les contraintes :

Selon le nombre de variables que vous avez demandé, vous aurez dans la table des contraintes ce même nombre de variables (dans le cas ci-dessous avec 4 variables) avec le rappel de la fonction mathématique en haut de l’écran. Remplissez la première ligne à votre goût tel que sur l’image :

La gestion des contraintes de la fonction objective ce fait assez aisément grâce à notre interface représenté ci-dessous.

Pour expliquer cela simplement nous avons fait un screenshot de l’interface, avec l’onglet « Gérer les contraintes »  sélectionné et nous y avons mis des flèches de couleur afin de pouvoir détailler chaque fonctionnalité.

* La flèche **rose** pointe sur l’onglet « Saisie du problème », cet onglet nous permet de revenir à la saisie du problème si jamais nous voulons par exemple modifier le nombre de variable. Ne vous inquiétez surtout pas quant à vos contraintes déjà remplies, elles ne seront pas supprimées. Ceci est dû à un enregistrement automatique des données à chaque fois que l’on modifie le nombre de variable.
* La flèche **orange** pointe sur la fonction objective écrite dans l’onglet « Saisie du problème ». Elle permet de rappeler à l’utilisateur la fonction qu’il a créé et ainsi lui éviter tout passage d’un onglet à un autre pour se rappeler de celle-ci.
* La flèche **verte** pointe sur les cellules se référant aux variables. C’est ici que vous marquerai le coefficient de chaque variable de vos contraintes. Si vous n’écrivez rien dans la cellule d’une variable celle-ci se met automatiquement à zéro.
* La flèche **jaune** pointe sur le choix de l’opérateur. Nous avons préféré y mettre une liste de choix prédéfini, car cela permet à l’utilisateur de ne pas se trompé.
* La flèche **rouge** pointe sur la constante de la contrainte.
* La flèche **noire** permet de supprimer une ligne, il suffit seulement de cliquer sur la croix et la ligne définissant la contrainte disparait. Son utilité ? Permettre à l’utilisateur de moduler plus facilement sa liste de contrainte, s‘il en a mis une en trop ou s’il en a mis une deux fois, peut importe, il peut la supprimer sans problème.
* Enfin la flèche **bleue** permet à l’utilisateur de rentrer une nouvelle ligne. Attention ! Cette ligne ne sera ajouté si et seulement si la première à été rempli correctement.

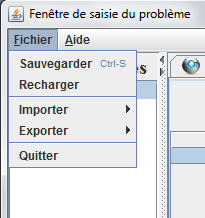
Nous pouvons donc voir que la gestion des contraintes de la fonction devient simple grâce à notre interface.

*Note :* On peut aussi importer des problèmes directement contenus dans un fichier XML, il suffit de cliquer sur importer dans le menu d’édition pour charger tous les problèmes valides contenus dans le fichier dans l’application.

## Etape 4 : Enregistrer ses problèmes

 Pour enregistrer, servez vous du bouton ‘Enregistrer’ présent dans la partie gauche de la fenêtre :

Enfin, servez vous du panneau de gauche si vous souhaitez créer à nouveau un problème …

Pour enregistrer de manière définitive vos problèmes, vous devez aller dans le menu d’édition et cliquer sur *‘Sauvegarder’*.

Voilà, tout est à présent terminé.

Répartition du projet

Au niveau de la répartition du projet, comme indiqué dans le MS Project, chacun a une ou plusieurs tâches à réaliser.

La gestion des utilisateurs (connexion, création de comptes,..) a été faite par Maël.

Le traitement des problèmes de l’utilisateur, a été réalisé par Cédric. De plus, les deux parties citées précédemment utilisent le DOM donc ils ont travaillé en collaboration.

Mériadec et Sarah, quant à eux, se sont chargés de la mise en place de toute l’interface graphique. Plus précisément, Mériadec s’est occupé de mettre en place une interface graphique globale intégrant l’ensemble des modules codés par Maël et Cédric (design de l’application) ainsi que de gérer la gestion des problèmes (excepté les contraintes). Sarah, de son côté, a adapté les contraintes des problèmes avec l’interface graphique. Chaque personne à tout de même pu donner ses idées sur la façon d’organiser les éléments graphiques.

Stockage et gestion des données

Pour commencer, il était nécessaire de trouver une architecture idéale pour modéliser un problème sous forme d’arbre et pour pouvoir sauvegarder toutes les données d’un problème le plus efficacement possible. Un schéma XSD a aussi été créé pour définir de façon sure l’architecture du fichier. Ce fichier XML peut être modélisé à l’aide d’un arbre UML (voir ci-dessous).



Ensuite, grâce à JAVA, nous devions pouvoir interagir avec les fichiers XML. Il s’agit d’une partie assez similaire au niveau des connaissances avec la partie de identification puisqu’il s’agit de trouver un moyen de parser judicieusement le fichier XML. Il a fallut pour cela faire plusieurs recherches pour comprendre et trouver un moyen efficace de réaliser ce traitement. Il existe plusieurs moyens tel que sax, dom ou jdom mais nous avons choisis de charger les fichiers en mémoire sous forme de dom puisque cette librairie est déjà. Toutes les méthodes manipulant les fichiers XML et XSD sont disponibles dans la classe « Utilitaire ». On peut donc charger un fichier XML en mémoire sous forme de document composé de nœuds et vice versa.

On peut aussi valider un document suivant un schéma XSD. Enfin tout ceci a pu aboutir a créer une classe statique BddProbleme qui récupère en utilisant le dom les informations d’un problème pour les stocker dans une variable problème lisible par « la partie graphique » du projet, on peut de même sauvegarder un problème en XML et les supprimer. Il est aussi possible d’importer des problèmes d’un fichier XML externe tout le validant par rapport au schéma défini pour être sur qu’il définit bien un ou plusieurs problèmes.

Gestion des comptes utilisateurs



Concernant la gestion des comptes utilisateurs, il s’agit de pouvoir interagir avec un fichier XML grâce au DOM. Ceci car lorsqu’un utilisateur veut se connecter (si il possède déjà un compte), le programme va vérifier dans le fichier identification.xml (c’est la base de données des comptes utilisateurs existants) si le login et mot de passe entré correspond à un nœud du fichier XML. A noté que le fichier identification.xml est composé d’éléments de type « personne » ayant les attributs login et password. Il faut donc utiliser le DOM afin de parcourir l’arbre de personne et permettre à l’utilisateur de se connecter à sa session. Au niveau de la création de compte, il faut également utiliser le DOM afin d’ajouter le nouveau nœud personne à la fin du fichier identification.xml.

Toutes les fonctions qui servent à la fois à la gestion des problèmes et des utilisateurs (fonctions permettant d’interagir avec des fichiers XML) ont été regroupé dans une seule classe : la classe « Utilitaire». Puis, toutes les fonctions qui permettent de vérifier si un utilisateur possède bien un compte, de vérifier si un utilisateur est déjà dans la base de données (pour la création de compte, deux utilisateurs ne peuvent pas avoir le même nom) ainsi que l’ajout d’un nouveau membre dans la base de données des utilisateurs se trouvent dans le fichier « BddUtilisateur.java ».

Ensuite, l’auto-chargement s’occupe aussi de récupérer des problèmes en fonction de l’utilisateur. Cette partie fonctionne de cette manière : à la première utilisation de l’application par un nouvel utilisateur, un fichier XML lui correspondant est généré. Puis, lorsque l’utilisateur s’identifie et qu’il créé ou modifie ou supprime un problème alors son fichier XML est édité. C’est ainsi que par exemple, l’utilisateur « User» se connecte et grâce à la partie auto-chargement, il peut sélectionner les problèmes et les éditer. Concrètement, l’auto-chargement extrait les problèmes de « User.xml » et les affiche sur la partie gauche de l’application. La liste des problèmes qui lui sont donc associés sont donc visibles à tout moment ….

Interface graphique

## Présentation générale de l’interface

Dès lors que l’on nous a annoncé que le projet était un logiciel, que ce soit pour une entreprise comme pour des particuliers, il nous a semblait indispensable de penser « interface graphique ». C’est véritablement la première chose qui nous est venue à l’esprit, avant même de connaître dans le détail le but de l’application. En effet, pour une entreprise, tout doit être clair, net et précis. On ne doit pas se poser de questions en voyant le programme. Un de nos buts premiers était donc de rendre l’interface accessible à tous, il fallait la rendre intuitive et c’est une des tâches à laquelle il a souhaité s’atteler.

Si l’on rentre un peu plus dans le détail, cette application doit permettre aux personnes qui l’utilisent de saisir un problème d’optimisation mathématique et de le résoudre. La première partie du projet nous a donc amené à nous consacrer au codage du module de saisie du problème. Mais bien entendu, qui dit codage du module de saisie dit également codage du module visuel. En effet, toute la partie de code qui rentre en jeu dans la création d’un problème n’a d’intérêt que si elle se matérialise visuellement. Et il en est de même pour tous les autres morceaux de code de l’application. C’est ainsi que nous avons pris la décision de mettre en place une interface graphique, aussi cohérente que possible afin de ne faire de toutes les parties du programme que nous nous étions partagées qu’une seule et même entité visuelle.

Au-delà même de l’apparence du logiciel, il est important de bien cerner son fonctionnement. Savoir ce qu’il permet de faire et comment me parait être une étape primordiale dans le développement d’un tel projet. Dans cette interface, tout a été étudié afin que l’utilisateur puisse effectuer un maximum d’actions en un minimum de clics et avec un maximum d’interactivité avec la machine. Nous avons pensé « simplicité » et « efficacité ».

Si on se place en tant que développeur de l’application, l’interface graphique est un regroupement de plusieurs classes Java. Toutes sont placées dans le package *« View »* de notre architecture en MVC.

Comme vous avez pu le constater, cette tâche est en fait un regroupement de plusieurs sous-tâches. Voici en fait les différentes parties dont Mériadec s’est occupé :

* Mise en place de l’interface graphique générale (images, gestion du suivi des frames les unes après les autres,…)
* Affichage graphique des modules d’identification et de création des comptes codés par Maël.
* Codage de l’interface permettant la saisie du problème (excepté les contraintes, module codé par Sarah).
* Gestion des interactions entre le panneau de gauche de la fenêtre et le panneau de droite (modification d’un problème existant, rafraichissements des listes, grisage du panel de saisie en cas de suppression du problème courant,…).
* Récupération des données d’un problème de l’interface graphique dans la classe *‘Problème.java’* (faisant partie du package Model) et enregistrement des données dans un fichier XML par l’utilisation des fonctions codées par Cédric.

## Gestion des problèmes

Dès le début, l’interface graphique propose les deux seules actions qui s’offrent à l’utilisateur : se créer un compte ou bien s’identifier. L’identification, si elle est réussie, permet par la suite de créer un problème ou bien de modifier ou supprimer un problème déjà existant. Dès lors qu’une erreur se produit, un message s’affiche à l’écran afin que l’utilisateur soit prévenu que sa requête n’a pu être prise en compte. Le reste se veut être assez instinctif. Il suffit pour créer un problème de remplir les champs demandés puis de valider ses choix en cliquant sur le bouton *‘Enregistrer’*.

La réalisation de cette partie a demandée une quantité de travail assez importante de part le fait notamment qu’elle peut constamment être améliorée. Un petit temps de réflexion ainsi que la mise en place de schémas au préalable nous ont permis de nous donner une idée très précise de ce que nous avions à réaliser en JAVA. Une fois toutes les idées organisées, nous avons donc pu nous consacrer à son développement. Cette étape en elle-même n’a pas posée trop de problèmes, excepté la mise en place du code permettant de mettre une image en arrière plan des composants.

## Gestion des contraintes

La gestion des contraintes se fait grâce à l’interface décrite dans la partie *‘Mode d’emploi de l’application’*. Tout est automatisé grâce à une table qui gère l’ajout et la suppression d’une contrainte. On propose à l’utilisateur un choix d’opérateurs dans une liste déroulante puisqu’un nombre limité est autorisé pour la résolution mathématique de l’équation demandée. Les contraintes sont ensuite enregistrées dans une liste de contrainte et ajoutées au problème si celui-ci a bien été enregistré.

La réalisation de cette partie a été très complexe, notamment de par l’utilisation des JTable. En effet ce composant nécessite de nombreuses modifications pour être personnalisé et utilisé de manière correcte.

Avancement global du projet

Concernant l’avancement du projet, nous avons donc réalisé toutes les tâches prévues pour le 6 avril (livrable 2). L’identification de l’utilisateur et l’enregistrement d’un problème sans son résultat sont donc ce que nous avons réalisés.

De plus, de par notre motivation importante et notre envie d’aller le plus loin possible dans les temps que nous nous sommes fixés pour réaliser ce projet, nous avons pu nous avancer suffisamment de telle sorte que nous nous trouvons actuellement en avance par rapport à notre planning. De ce fait nous avons bien entamé la fonctionnalité de suppression et de modification des problèmes. Cela fait toujours partie intégrante du livrable 3 mais puisque celle-ci semble être fonctionnelle, nous l’avons jointe à ce livrable ci. Cela est représenté par la partie de gauche de la fenêtre : « Mes problèmes ».

En revanches certaines fonctionnalités de cette partie n’ont pas encore été testées. Il se peut donc que tout ne marche pas parfaitement. Tout sera bien évidemment testé et vérifié pour le prochain livrable qui avait pour objectif premier de créer ce module.

L’aspect visuel de l’interface graphique tend à être amélioré, tant au niveau du design qu’au niveau de l’ergonomie mais reste à ce jour grandement opérationnelle.

Concrètement, il reste à faire l’export des problèmes vers Excel et Scilab puis l’affichage des résultats dans une fenêtre JAVA ainsi qu’en HTML. Et enfin, il reste bien évidemment la résolution des problèmes grâce à l’utilisation de la méthode Simplex. En revanche il se peut que le projet ne soit commenté entièrement car nous ne commentons généralement que les fonctions qui sont sures de rester de manière définitive. Bien entendu, dès lors qu’un module sera fonctionnel et non voué à être changé, nous mettrons en place tous les commentaires nécessaires à la bonne compréhension du code par des personnes extérieures au programme.

Remarques personnelles

## Remarques relatives à Maël

Je trouve que le groupe est homogène dans le sens où tout le monde a et veut la même charge de travail. De plus, nous sommes tellement motivés que lorsqu’on avait finit nos tâches pour le livrable 2, nous avons quand même continué à avancer le projet. C’est ainsi que j’ai déjà réalisé la partie auto-chargement des données de l’utilisateur alors que c’était prévu pour le livrable 3.

De plus, je n’ai pas éprouvé de difficultés quant à la réalisation de ma partie java. Cependant, l’UML a été un peu plus compliqué à réaliser car certaines choses telles que l’utilisation d’ArrayList<Integer> n’avait pas été vues en cours. Mais cela nous a permis d’apprendre de nouvelles fonctionnalités de l’UML et d’approfondir nos connaissances.

J’ai trouvé que l’utilisation du XML avec ce projet était intéressante ici, car cela nous a permis d’utiliser le DOM et de pouvoir associer le JAVA avec le XML.

## Remarques relatives à Mériadec

En ce qui me concerne, je suis plutôt satisfait du groupe. Nous sommes tous motivés, conscients de vouloir bien faire et de rendre une application mettant en œuvre l’ensemble de nos connaissances apprises jusqu’à ce jour. Malgré quelques petites tensions par moment, tout se déroule globalement bien. En ce qui concernent ces tensions, celles-ci sont principalement dues à la mise en place d’un système de subversion générant ça et là quelques conflits entre les versions faites par plusieurs d’entre nous. On a ainsi eu droit à quelques apparitions d’erreurs inattendues par moment.

Mais, pour ce qui est du reste, tout se passe bien. Nous avançons assez régulièrement et envoyons nos versions aux autres membres du groupe dès que des modifications (fonctionnelles) sont apportées au projet. Cela permet donc de limiter grandement les erreurs lors de la réunification des codes.

Le projet est intéressant et met en œuvre beaucoup de notions que nous avons vues. Cela est donc d’autant plus motivant pour nous.

## Remarques relatives à Cédric

Malgré quelques conflits, l’entente dans le groupe est bonne et tout le monde a à cœur de travailler pour faire avancer le projet.

Grâce au planning, nous avons pu nous assurer de finir nos tâches dans les temps et même d’être en avance dans le projet.

Le seul outil que je ne maitrisais pas avant ce projet était la façon dont interagir entre java et le XML, cet pour ca que j’ai trouvé intéressent de m’occuper de cette partie pour découvrir et apprendre de nouvelles choses. La librairie Dom fut relativement pratique et intéressent a découvrir.

## Remarques relatives à Sarah

Il est vrai que gérer un projet à quatre n’est pas quelque chose d’aisé. En effet il faut convenablement se répartir les tâches en satisfaisant le plus de personnes possible, mais il faut aussi faire des compromis afin que chaque membre ai la même quantité de travail. Bien sûr, il y a des tâches que tous les membres veulent réalisées car elles sont plus agréables que d’autres et ceci peut être source de conflit.

Dans notre groupe nous avons vraiment voulu que tout le monde soit satisfait. C’est pour cela que l’établissement du planning fut une tâche élémentaire de notre projet.

La réalisation des différentes parties de ce rendu m’a fait remarquer que chaque membre du groupe est prêt à aider les autres à avancer en cas de problèmes.

Pour finir, je dirai que la réalisation de la gestion des contraintes m’a causée pas mal de problèmes mais elle m’aura permis d’approfondir mes connaissances sur les tables en JAVA.