IUT Informatique – La Rochelle

Projet PTS2 - Gestionnaire d'emploi du temps

Cahier des charges fonctionnel

Michaël LUCAS
Benjamin VALOGNES
Romain DOUTEAU
Antoine FOUCHE

Tuteur : Mme LASSUS **Responsable :** M.MARCHAND

I- Présentation du projet : EDT Project

A- Explication du projet : vue d'ensemble

B- Les fonctionnalités importantes

C- Analyse du besoin : problématique du client

D- Pour qui et pour quoi ?

E- Délimitation du projet

F- État de l'art

II- Architecture & Conception

A- Cas d'utilisations

B- Scénarios

C- Diagramme de conception

D- Diagramme de séquence

E- Wireframe

III- Organisation & Planning

A- Planification du projet : Excel

B- Communication : Slack

IV- Réalisation & Programmation

A- Modification de la conception (modèle du domaine)

B- Ajout de nouvelles fonctionnalités

C- Utilisation du GIT

D- Problèmes rencontrés en programmation

V- Bilan : rendu final

A- Fenêtre de connexion au gestionnaire d'emploi du temps

B- Fenêtre de saisie des groupes

C- Fenêtre de saisie des salles

D- Fenêtre de saisie des cours

E- Fenêtre d'affichage de l'emploi du temps

VI- Extensions possibles

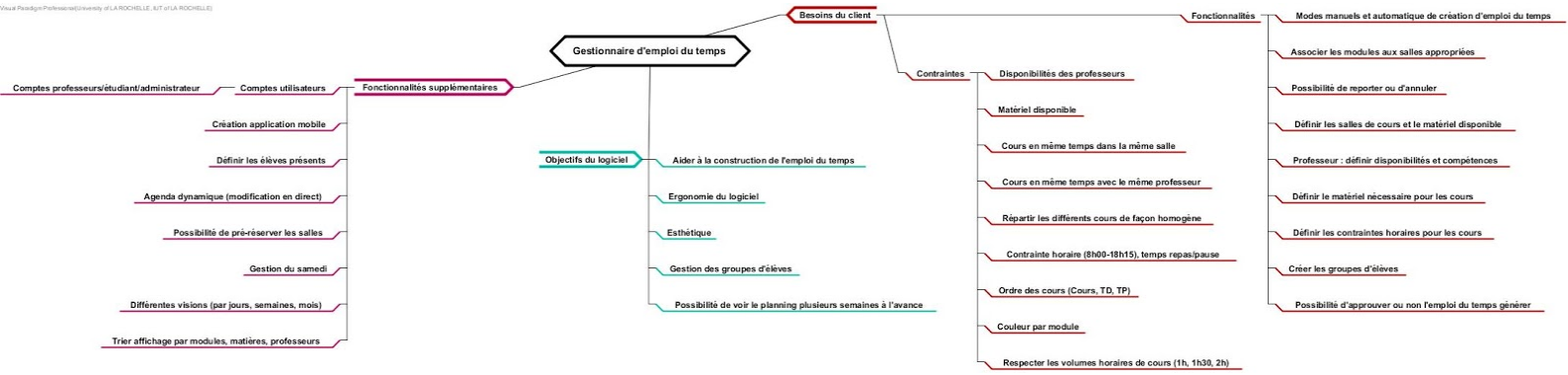
I- Présentation du projet : EDT Project

A- Explication du projet : vue d'ensemble

Le projet qui nous a été confié consiste en la création d'un programme permettant au directeur des études de l'IUT de créer l'emploi du temps avec plus de facilité, comparé à la solution logicielle proposée actuellement, qui repose sur plusieurs tableaux Excel et GPU.

B- Les fonctionnalités importantes

Le but est donc de créer un programme facilitant le travail du directeur des études dans la création et la mise en place des différents cours pour constituer l'emploi du temps, sur plusieurs semaines. Nous avons donc pu déterminer différentes fonctionnalités à intégrer au logiciel à concevoir, fonctionnalités que nous détaillerons dans les partie Conception.



Une des demandes importantes, sur laquelle nous avons été amené à nous concentrer est la partie affichage de l'emploi du temps, saisies des données associées à l'affichage (Cours, Salles, Professeurs, Heures) et l'affichage d'éventuelles contraintes lors du placement d'un cours dans l'emploi du temps.

C- Analyse du besoin : problématique du client

Afin de répondre correctement à la demande du client, il a d'abord fallu analyser la demande de manière précise, afin de définir les différentes fonctionnalités à implémenter au programme.

D- Pour qui et pour quoi ?

Nous avons d'abord commencé par reprendre la demande dans les grandes lignes afin d'en extraire les fonctionnalités importantes, que nous devions impérativement intégrer.

Nous avons donc commencé par définir différents acteurs : le logiciel doit s'adresser à la fois aux étudiants (en consultation), au directeur des études (pour consulter et créer l'emploi du temps) et aux professeurs afin qu'ils puissent indiquer leurs contraintes et consulter leur emploi du temps respectif.

Nous avons dans le même temps définis des fonctionnalités connexes :
- possibilité de filtrer l'affichage de l'emploi du temps par Professeurs, Cours et Salles
- définition de différents affichages : par semaines et par jours

E- Délimitation du projet

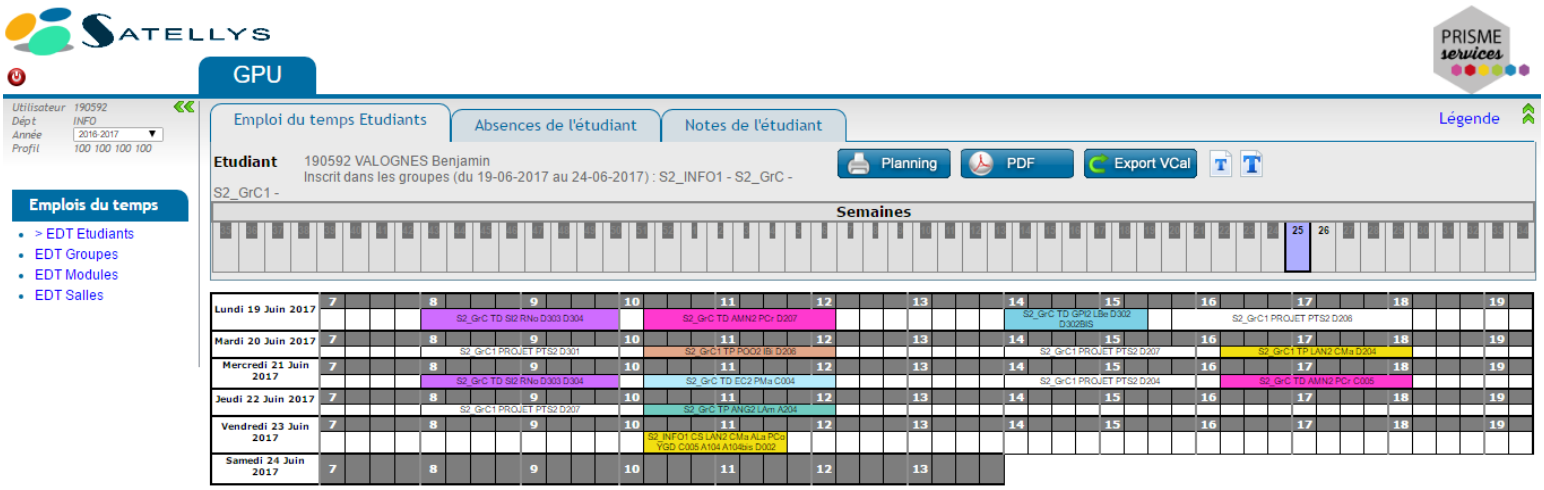
La demande du client était assez vaste, car elle suggérait la construction plus ou moins automatique de l'emploi du temps, et, l'aide à la création de celui-ci à travers une interface soignée et plus agréable à utiliser que l'existant, à savoir GPU.

En terme de compréhension du projet, nous étions donc d'abord partis dans l'optique d'une création automatique de l'emploi du temps, concept auquel nous avons rapidement renoncé aux vues des difficultés futures que nous allions rencontrer.

Nous avons dû diminuer nos ambitions en terme de fonctionnalités à implémenter au logiciel et nous sommes concentrés sur la saisie des données (Cours, Salles, Professeurs) ainsi que de l'affichage de l'emploi du temps pour les différents acteurs concernés.

F- État de l’art

Avant de nous lancer dans la conception du programme, nous avons d’abord réalisé des recherches vis à vis de l’existant. Nous avons commencé par analyser les différentes fonctionnalités offertes par GPU du côté étudiant et directeur des études.



Afin d’avoir une vision plus globale du marché, nous avons élargi nos recherches et avons trouvé d’autres solutions alternatives à GPU, certaines payantes et d’autres open-source. Voici donc quelques unes des solutions existantes : *Pronote* (payant), *EDT Soft* (payant), *Free Timetabling Software* (open-source).

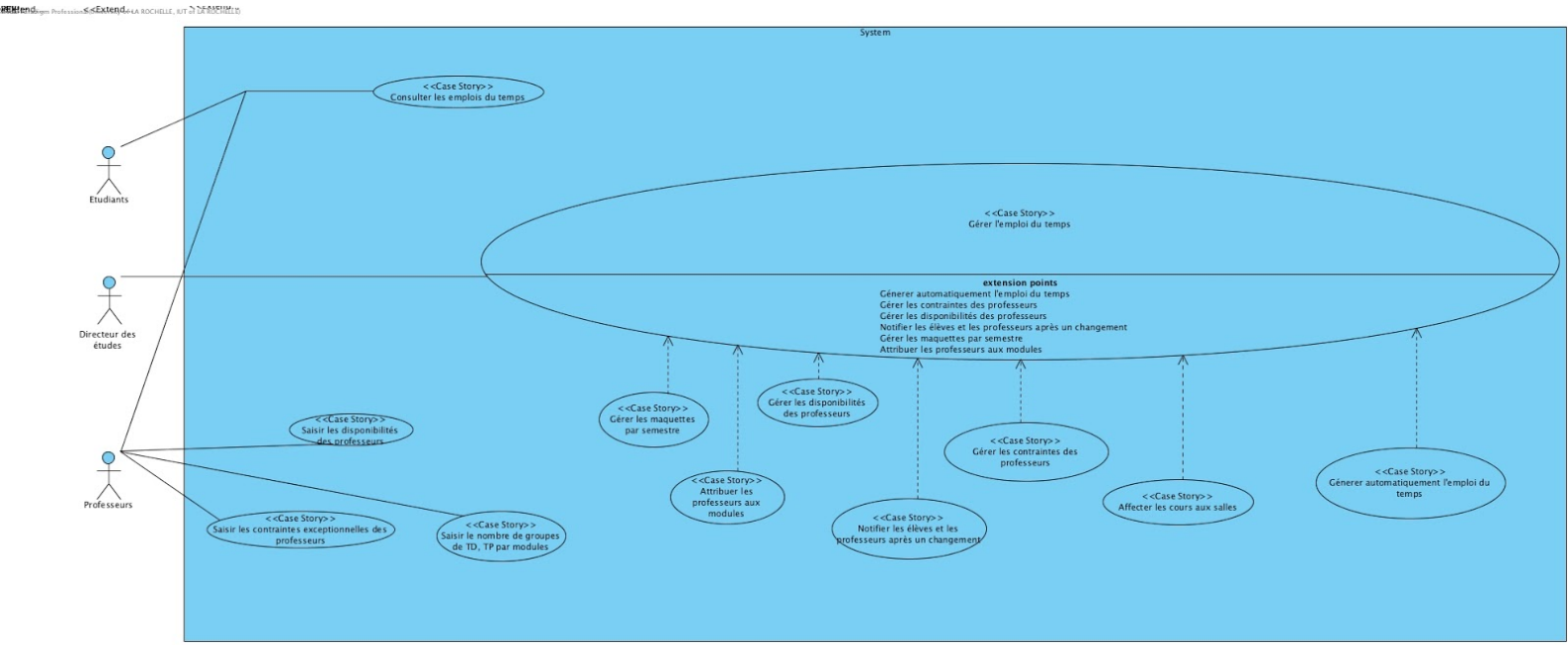
Dans le même temps, nous avons recherché différentes librairies Java pouvant permettre la création d’emploi du temps. Nous n’en avons finalement pas intégrées au projet final.

II- Architecture & Conception

Une fois la phase d'analyse du besoin terminée, nous nous sommes concentrés sur la conception du programme, cela impliquait d'avoir une vision claire des limites / périmètre du projet et des fonctionnalités à intégrer.

A- Cas d'utilisations

Comme outil de conception, nous avons utilisé Visual Paradigm, afin de concevoir dans un premier temps des diagrammes de cas d'utilisations. Le diagramme de cas d'utilisation que nous avons conçu regroupe les trois acteurs : Étudiant, Professeur et Directeur des études.



A ces trois acteurs nous avons relié les différentes fonctionnalités que chacun peut effectuer au sein du programme.

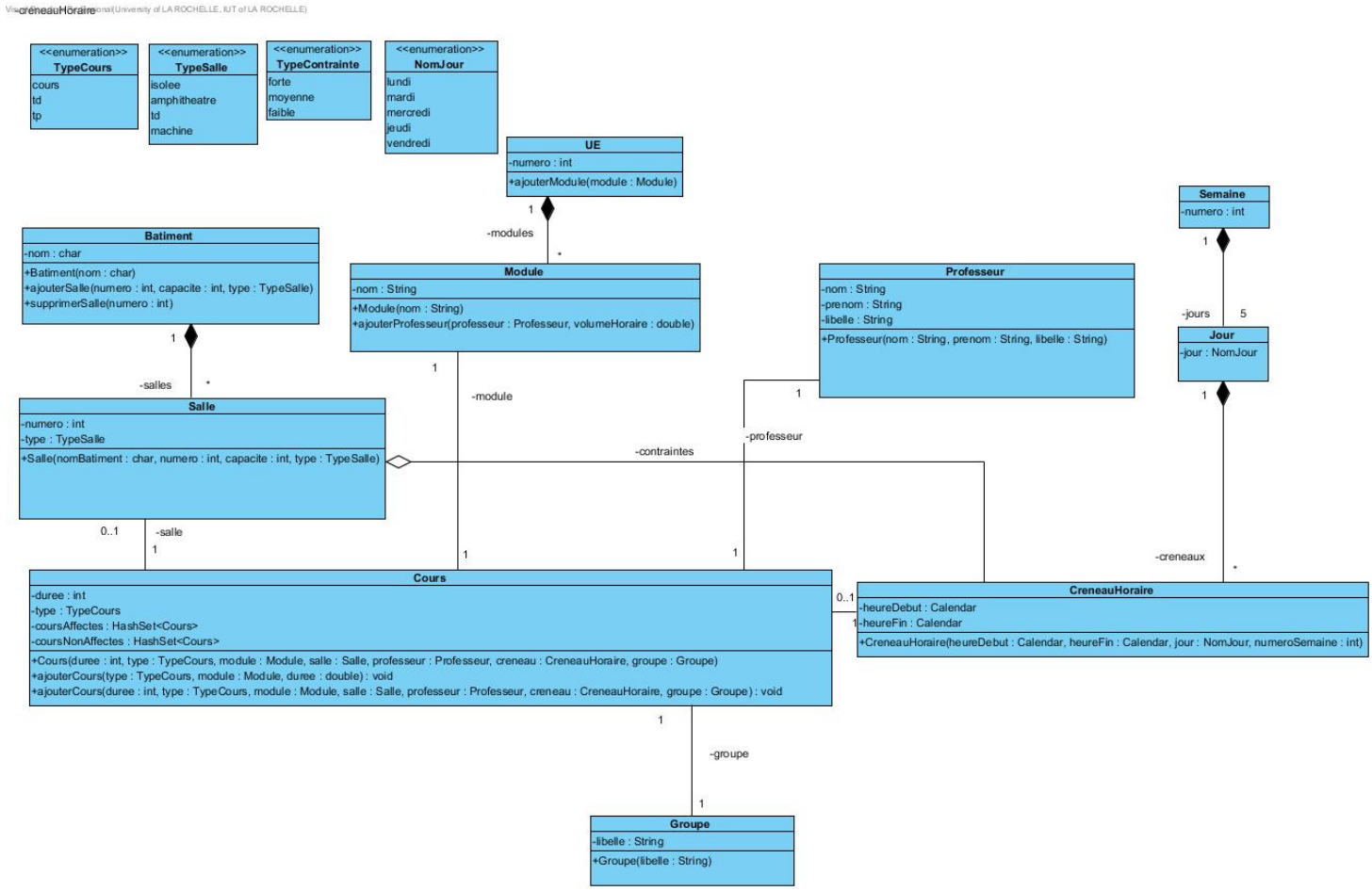
B- Scénarios

A partir des cas d'utilisations définis précédemment, nous avons choisi de les détailler à l'aide de scénario.
Les scénarios nous ont permis de visualiser les fonctionnalités à implémenter et à entrevoir un début de code (objets, acteurs, actions).

Nous avons donc créé plusieurs User Story, un pour chaque scénario. Nous avons choisi de décrire le scénario nominal, mais avons aussi pensé à quelques scénarios alternatifs, ce qui a fait émerger des parties de codes (notamment pour les structures conditionnelles).

C- Diagramme de conception

Une fois les cas d'utilisations en place, nous avons démarré la conception du programme via la mise en place d'un diagramme de conception.



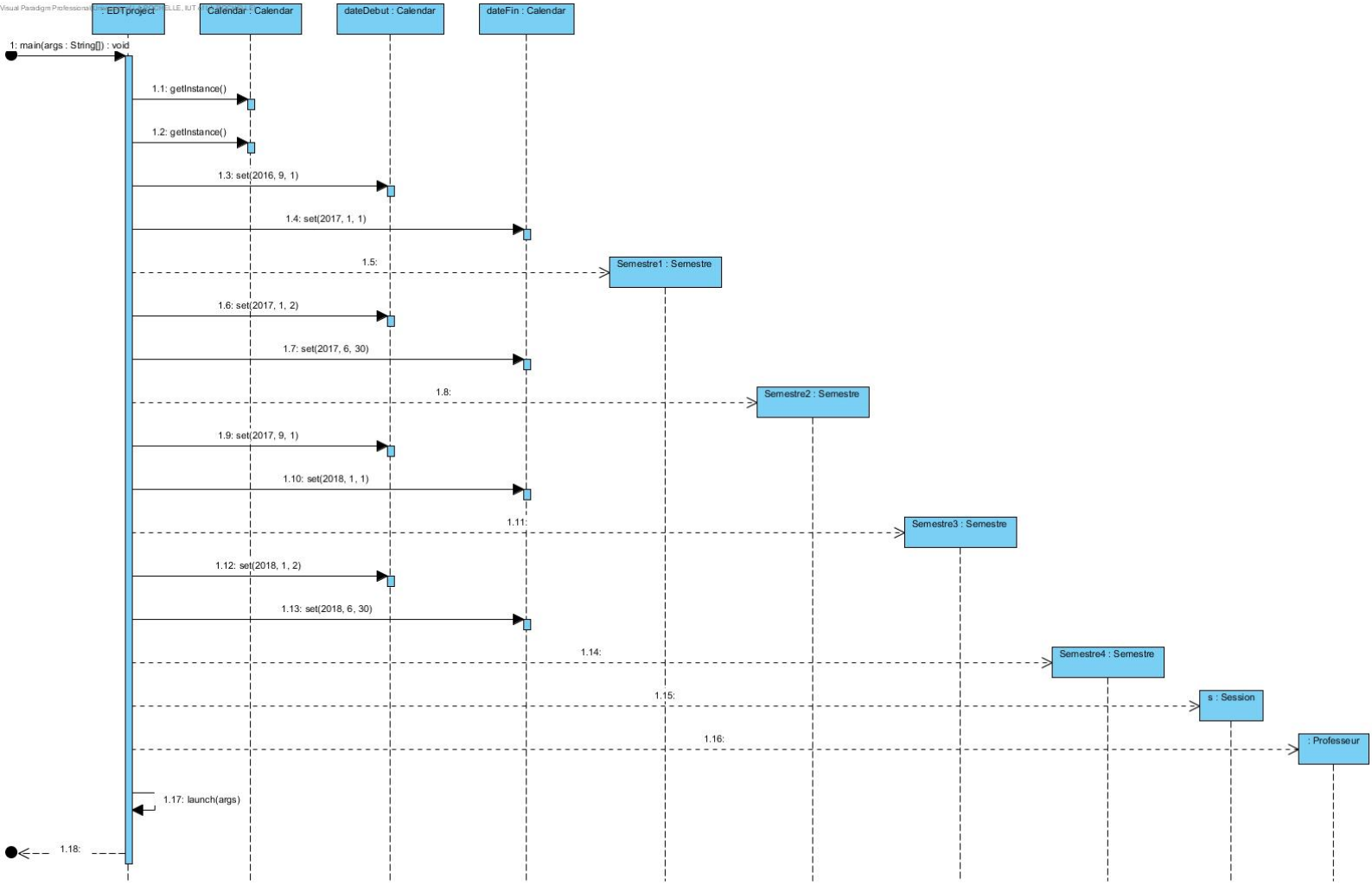
Nous avons réalisé une première version du diagramme de conception, incluant toutes les classes et objets nécessaires à la création complète de l'emploi du temps (Semestres, Jours, Professeurs…).

Cependant, après révision, nous nous sommes rendus compte que notre diagramme de conception allait au-delà des limites que nous avions

préalablement fixé. Nous avons donc revu ce diagramme et n'avons traité, en programmation, qu'une partie des classes nécessaires à la réalisation des fonctionnalités définies.

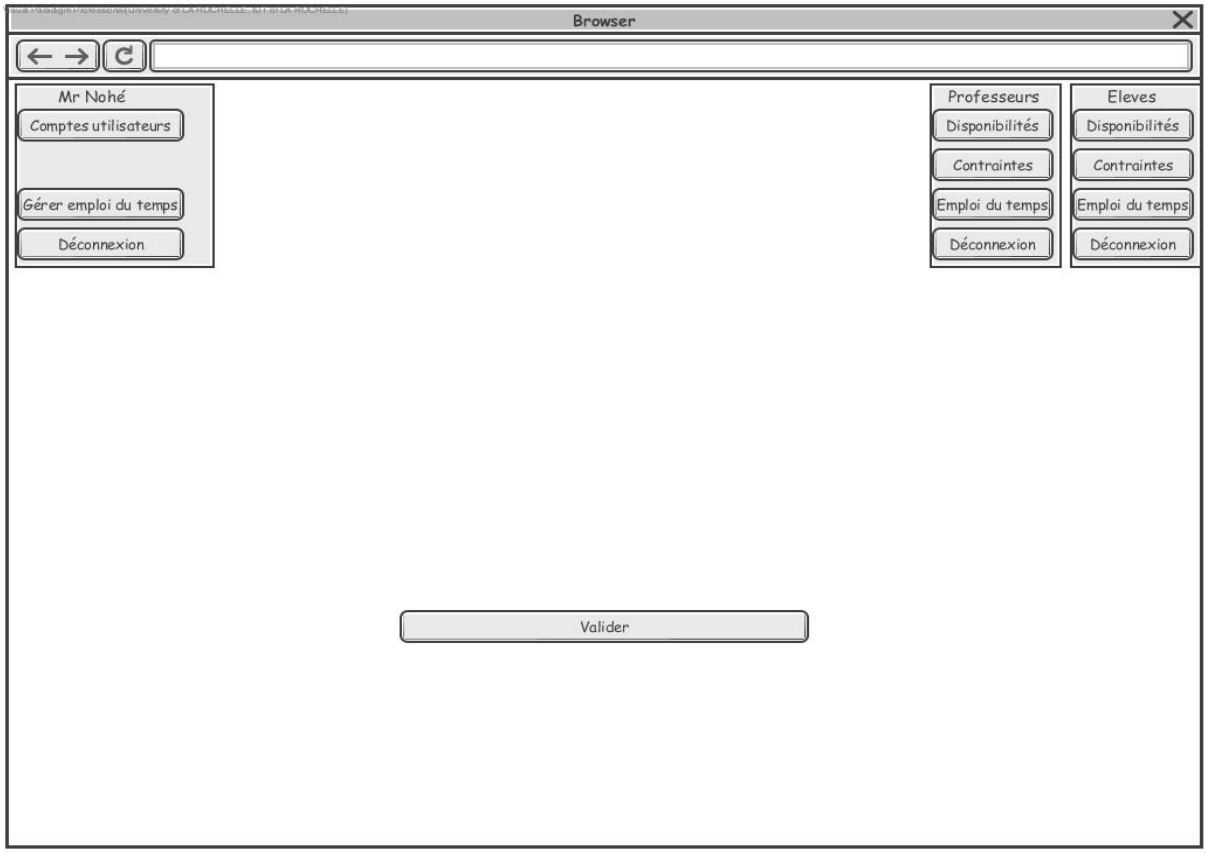
D- Diagramme de séquence

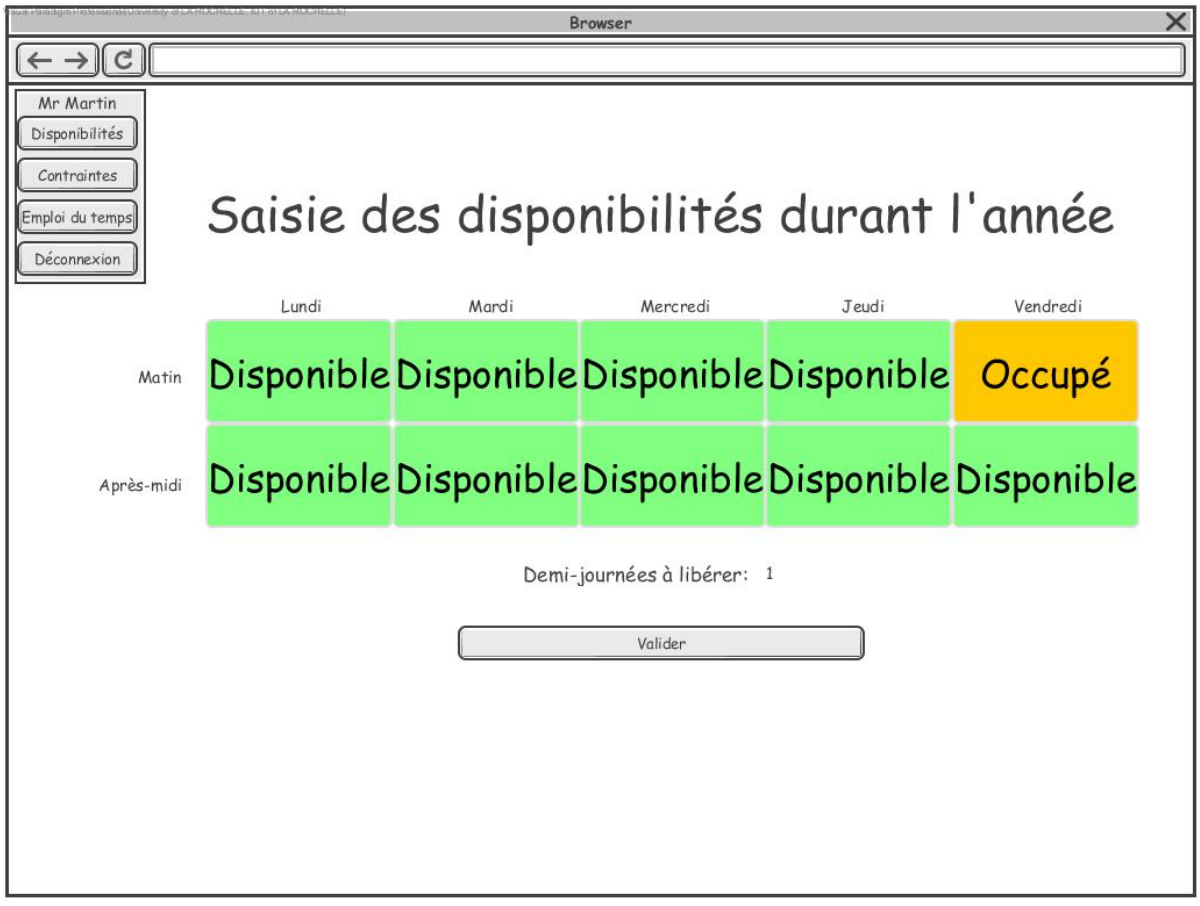
Toujours en conception, une fois le diagramme de conception finalisé, nous avons généré quelques diagrammes de séquences afin de mieux visualiser les différents appels de méthodes.



E- Wireframe

Le projet reposant en grande partie sur une IHM, nous avions prévu de multiples wireframe pour prévisualiser l'IHM à concevoir. En voici quelques captures:


Browser
Mr Nohé
Comptes utilisateurs
Gérer emploi du temps
Déconnexion
Professeurs
Disponibilités
Contraintes
Emploi du temps
Déconnexion
Eleves
Disponibilités
Contraintes
Emploi du temps
Déconnexion
Valider


Browser
Mr Martin
Disponibilités
Contraintes
Emploi du temps
Déconnexion
Saisie des disponibilités durant l'année
Lundi
Mardi
Mercredi
Jeudi
Vendredi
Matin
Disponible
Disponible
Disponible
Disponible
Occupé
Après-midi
Disponible
Disponible
Disponible
Disponible
Disponible
Demi-journées à libérer: 1
Valider

III- Organisation & Planning

A- Planification du projet : Excel

Afin d'organiser le travail de programmation, nous avons mis en place un tableau Excel, sur lequel sont inscrites les différentes tâches de chacun,que les membres de l'équipe sont amenés à effectuer dans une période prédéfinie (plus ou moins longue en fonction de la difficulté de la tâche).

Chaque tâche à réaliser est donc inscrite dans ce fichier, avec son membre attribué ainsi que la durée prévisionnelle de réalisation de la tâche. Une section spéciale est prévue pour suivre l'avancement de la tâche, en pourcentage.

	A	B	C	D	E	F G	H	I	J	K
1	Tâche	Lucas Michaël	Douteau Romain	Valognes Benjamin	~~Fouche Antoine~~	Période Début	Dépendance(ligne tableau	Stade d'avancement (en%)	Priorité	Date fin de tache réel
2	Mise en place d'un filtre pour trier par semestre	X				01/06 - 15/06	0	70	4	
3	Mise en place d'un DatePicker pour afficher l'emploi du temps	X	X			01/06 - 06/06	0	~~100~~	4	11/06
4	Terminer le controller permettant d'afficher l'emploi du temps pour un professeur	X				06/06 - 08/06	0	~~100~~	5	11/06
5	Bug fixe de la fonctionnalité "consulter emploi du temps par un professeur"	X				08/06 - 11/06	8-9-10	~~100~~	5	
6	Creation d'un bloc FXML qui va contenir uniquement la fonctionnalité de l'affichage de l'emploi du temps	X				08/06 - 11/06	4	~~100~~	3	
7										
8	Nouveau bloc FXML pour la création d'un cours par le directeur des études			X		01/06 - 11/06	0	~~100~~	5	11/06
9	~~Mise en place d'un fichier model avec variables, constructeurs, getter et setter~~				~~X~~	~~01/06 - 08/06~~	~~0~~	~~0~~	~~5~~	
10	~~Mise en place du lien entre le controller et le fichier model~~			~~X~~		~~11/06 - 18/06~~	~~8-9~~	~~0~~	~~5~~	
11										
12	Bloc FXML racine	X				01/06 - 06/06	0	70	2	
13										
14	~~Mise en place de la liaison java DDD~~		~~X~~	~~X~~		~~01/06 - ?~~	~~0~~	~~0~~	~~2~~	
15	Bloc FXML pour la gestion des sessions			X		01/06 - 11/06	0	~~100~~	1	

Le fichier Excel se trouvant sur un Drive commun à tous les membres du projet, chacun peut visualiser l'avancement de chacun des membres et ainsi pouvoir proposer de l'aide ou des modifications dans l'ordre des tâches à réaliser.

B- Communication : Slack

Comme moyen de communication supplémentaire, nous avons aussi utilisé la plate-forme Slack. Une manière simple de pouvoir échanger entre les différents membres de l'équipe.

IV- Réalisation & Programmation

A- Modification de la conception (modèle du domaine)

La phase de conception terminée et approuvée par toutes les parties prenantes, nous avons pu commencer la phase de réalisation, qui se traduit par l'écriture du modèle de conception en langage de programmation Java.

Comme dit précédemment, nous n'avons pas repris l'entièreté du modèle de conception, au vu du fait que celui-ci était en dehors du nouveau périmètre définit pour le projet. Nous avons donc dans un premier temps été fidèles au modèle de conception (aux classes concernées et leur structure) puis avons, petit à petit, été amenés à modifier ou ajouter certaines fonctionnalités, pour faire face à différents problèmes.

Exemple de modifications : Problème d'objets de tests

Dans la classe EDTproject.java, nous initialisions des objets afin de tester le programme. Un problème est survenu lors de l'exécution de classes non jointes à la classe principale.

La solution que nous avons apporté pour résoudre ce problème est la création de tableau statiques dans la classe EDTproject et la modification des classes Professeur, Salle, Session avec l'ajout dans leur constructeur respectif d'une ligne de code permettant l'ajout automatique au tableau de chaque type (cités précédemment).

Nous avons donc ajouté des tableaux Professeurs, Salles, Sessions qui regroupent automatiquement tous les objets créés, nous permettant ainsi d'appeler ces différents tableaux dans d'autres classes, sans devoir recopier le code de création des objets.

B- Ajout de nouvelles fonctionnalités

Nous avons globalement respecté le périmètre du projet. Nous n'avons donc pas été amené à ajouter de nouvelles fonctionnalités.

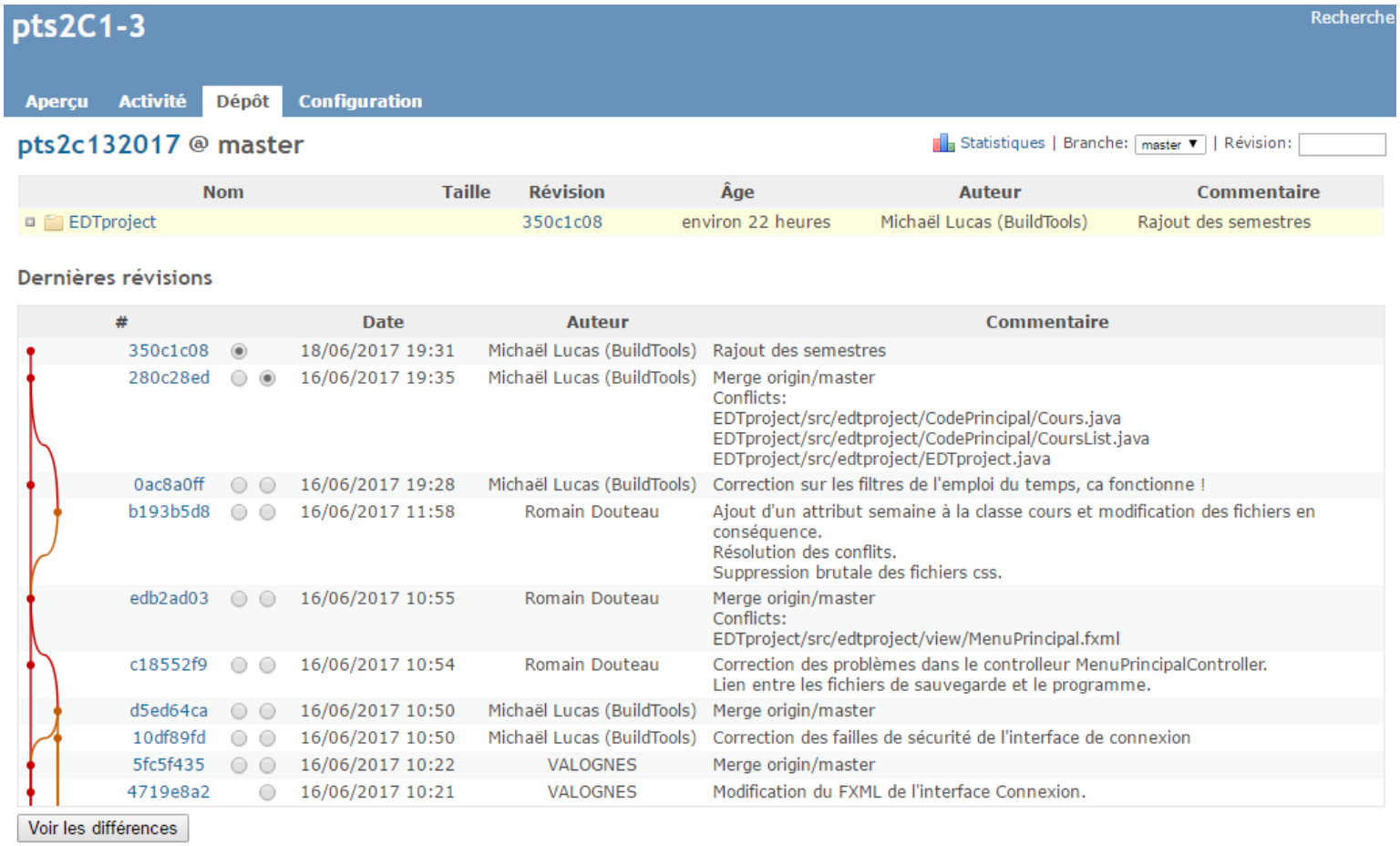
Pour être certain d'avoir un livrable, nous avons procédé par "méthode agile".

C’est à dire que nous avons d’abord concentré nos efforts sur l’élaboration de la vue de l’emploi du temps (vue professeur, puis étudiant et directeur des études).

Ainsi, nous avons procédé par ajout : une fois les étapes clés prévues terminées, nous avons progressivement ajouté des fonctionnalités supplémentaires, inclues dans le périmètre mais optionnelles. Après la mise en place des différentes vues, nous avons ajouté les fonctionnalités d’ajout des salles, d’ajout des professeurs, d’ajout des sessions utilisateur puis le placement des salles de la réserve (cours créés mais non placés sur l’emploi du temps) sur l’emploi du temps.

C- Utilisation du GIT

Dans le cadre du projet, nous avons été amené utiliser GIT, afin de pouvoir travailler en équipe et de manière simultanée sur le code Java. Nous avons eu quelques problèmes d’utilisations du GIT (comme le montre les plus de cent push sur le dépôt), notamment au niveau des fonctionnalités Merge et Rebase.



Nous avons, au fil de l'utilisation du GIT et de la confrontation à ces problèmes, appris à les résoudre sans créer d'incompatibilités entre les versions des uns des autres.

Nous avons donc organisé de façon plus précises les modifications que nous allions apporter au programme de manière à ne pas modifier chacun le même fichier, et bien travailler sur des parties différentes du programme.

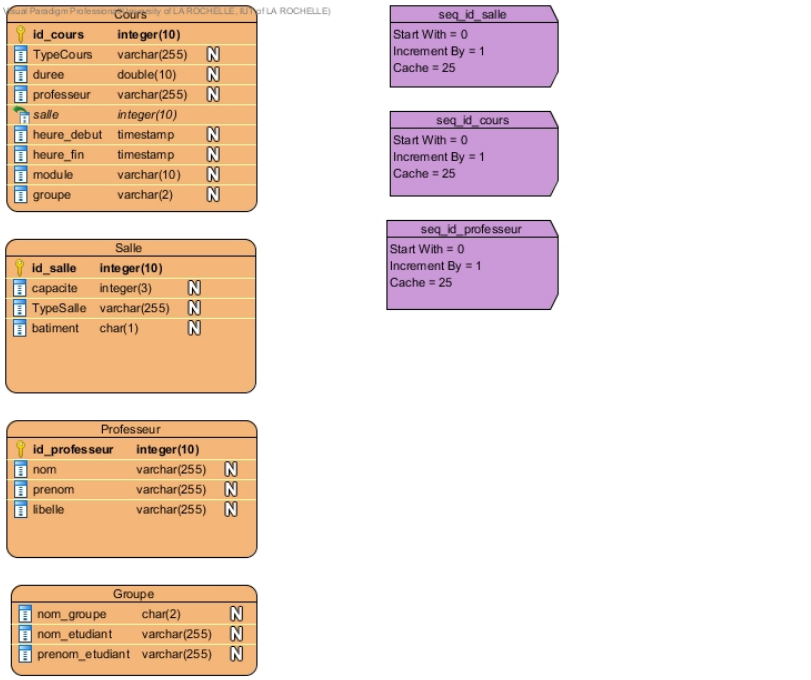
D- Problèmes rencontrés en programmation

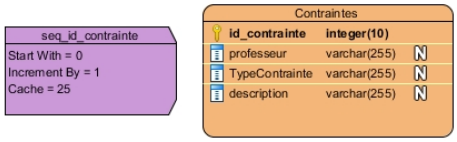
Lors de la phase de réalisation du projet, plusieurs problèmes se sont posés, liés à des problèmes techniques et de compétences. Ces problèmes ont été soit résolus au fil de nos recherches soient tout simplement remplacer par des solutions alternatives plus ou moins proches de la solution souhaitée.

Exemples de problèmes rencontrés :

1. Mise en place de la base de données

Au départ, nous avions prévu de stocker les différents objets dans une base de données Oracle (celle de l'IUT ou sur un autre serveur). Malheureusement, nous n'avons pas pu mettre en place cette partie du projet à cause de trop nombreux problèmes, à la fois d'accès à la base depuis l'extérieur et de liaison entre la base de données et Java.





Nous avons donc opté pour une solution intermédiaire, qui est l’enregistrement des données dans un fichier et le chargement de celui-ci au lancement du programme. Ainsi, nous avons pu mettre en place un système d’enregistrement de données, mais il reste cependant un problème d’ordre fonctionnel : le fichier reste en local, et par conséquent, n’est pas accessible pour les autres utilisateurs.

2. Problèmes liés au FXML

Malgré l’accessibilité du FXML par l’intermédiaire de Scene Builder, nous avons rapidement rencontré de nombreux problèmes pour rassembler les fenêtres sur une seule interface. En effet, nous voulions faire en sorte de proposer à l’utilisateur une seule interface, et non une succession de fenêtres apparaissant pour chaque fonctionnalité demandée. Ainsi, de part ce choix, nous nous sommes rendus compte que nombre de parties de notre IHM ne pouvaient pas être redimensionnées, la seule solution que nous avons trouvé aurait été de rassembler tout nos blocs FXML dans un seul, ce qui aurait diminué de façon importante la lisibilité et la fiabilité de notre code, sachant que nous n’avons pas trouvé de moyens pour lier plusieurs controller à une seule interface FXML sans passer par la balise <fx:include>

Certains choix ont du être fait concernant le FXML, ainsi, afin d’avoir une interface dynamique, nous avons mis en place un Canvas c’est a dire une zone où le programme va dessiner automatiquement les cours, cependant cela rend l’utilisation de Drag/Drop très compliquée à mettre en place, en plus de ne pas pouvoir interagir directement avec les cours dans cette zone. Nous étions aussi parti lors de la création des salles, modules, groupes et sessions utilisateurs sur l’idée de créer un tableau permettant de visualiser en temps réel la saisie et la modification de ces différents éléments. Cependant, à cause des restrictions sur l’utilisation des TableView nous avons décidé de laisser cette fonctionnalité en périmètre et de redéfinir nos priorités. Nous avons aussi eu beaucoup de difficultés pour effectuer la liaison entre le controller et le FXML, pour effectuer des appels de méthodes et transferts de variables entre les différents éléments.

3. Problème avec l'IDE Netbeans et JDK8 (la classe java.io.File)

Suite aux différents problèmes techniques lors de la mise en place d'une base de données, nous avons fait le choix d'utiliser un fichier pour sauvegarder et charger les données à partir de l'interface graphique du gestionnaire

d'emploi du temps. Mais, après la mise en place du système de fichier, un problème d'accessibilité à la classe jave.io.File est survenu dans l'IDE Netbeans empêchant l'enregistrement des données de manière persistante.

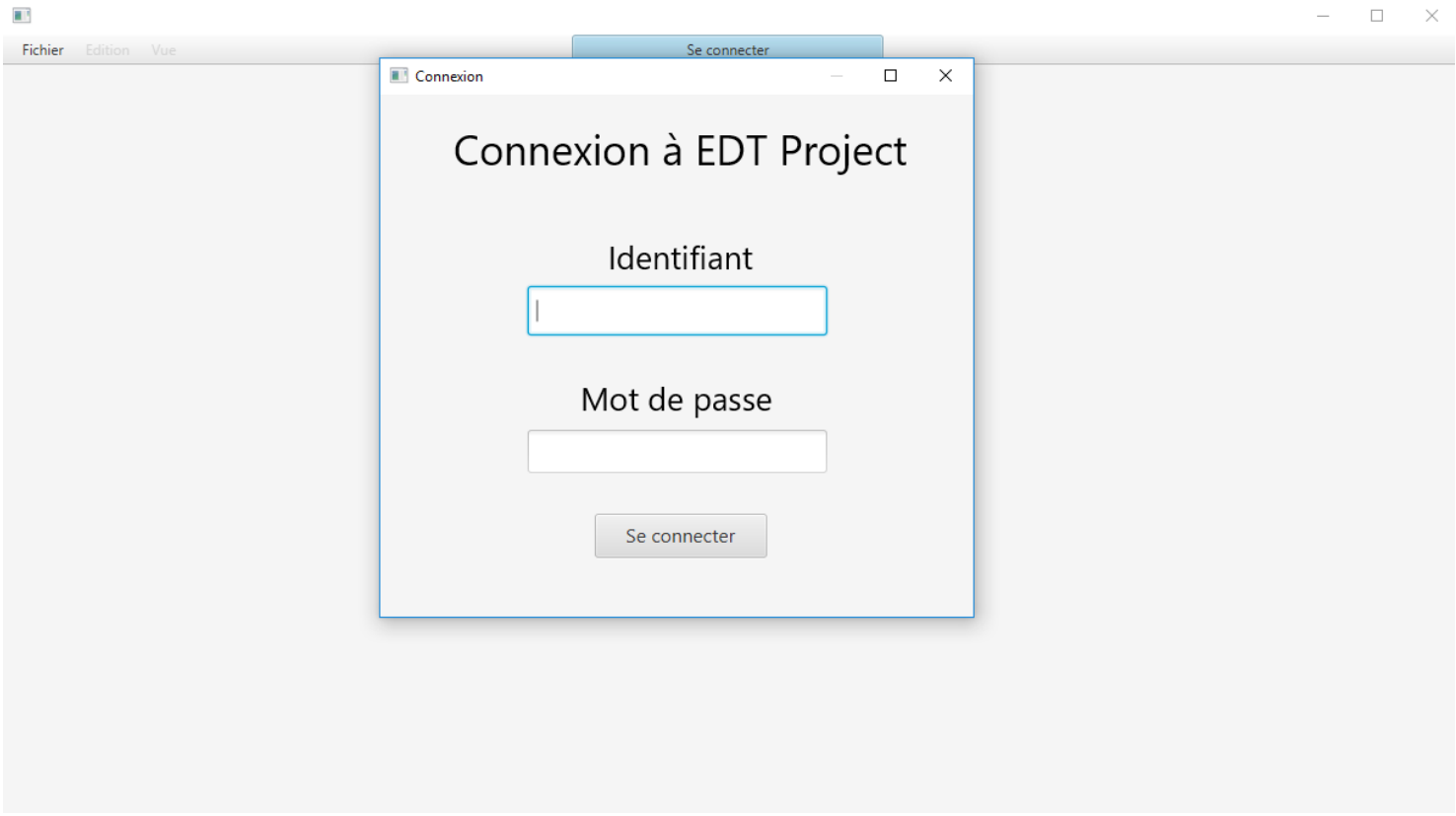
4. Problème de compréhension et d'utilisation de l'outil git

Pour mutualiser les sources du programme, nous avons utilisé l'outil git. Mais nos connaissances insuffisantes de cet outil ont nettement compromis notre efficacité à son utilisation. Pour récupérer les données envoyées sur le git avec la commande pull, nous avions le choix pour régler les conflits entre les fichiers entre utiliser la méthode de Rebase ou celle du Merge mais nous n'avons jamais compris la différence entre les deux. Nous avons donc principalement utiliser la méthode Merge car celle-ci nous permettait de comparer les différences entre les fichiers sans écraser nos modifications.

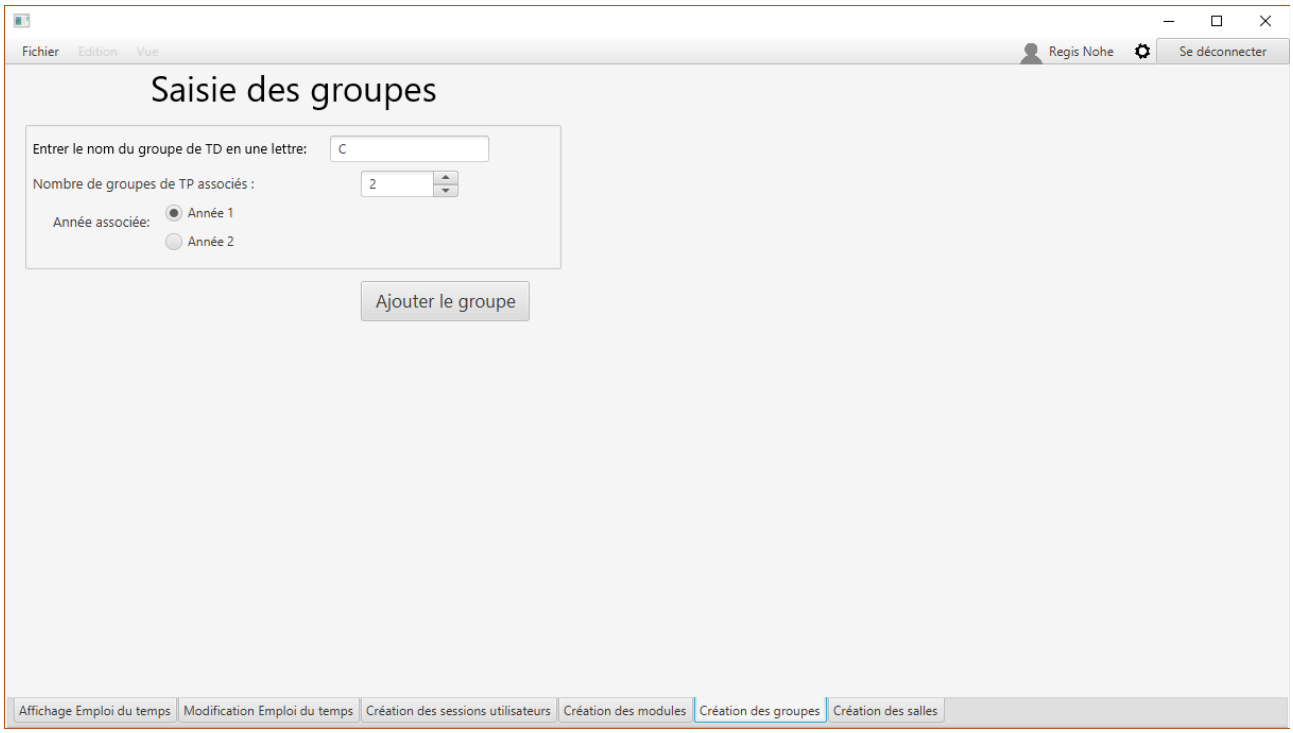
V- Bilan : rendu final

Après avoir implémenter les différentes fonctionnalités demandées, nous pouvons maintenant vous proposer quelques aperçus de l'IHM.

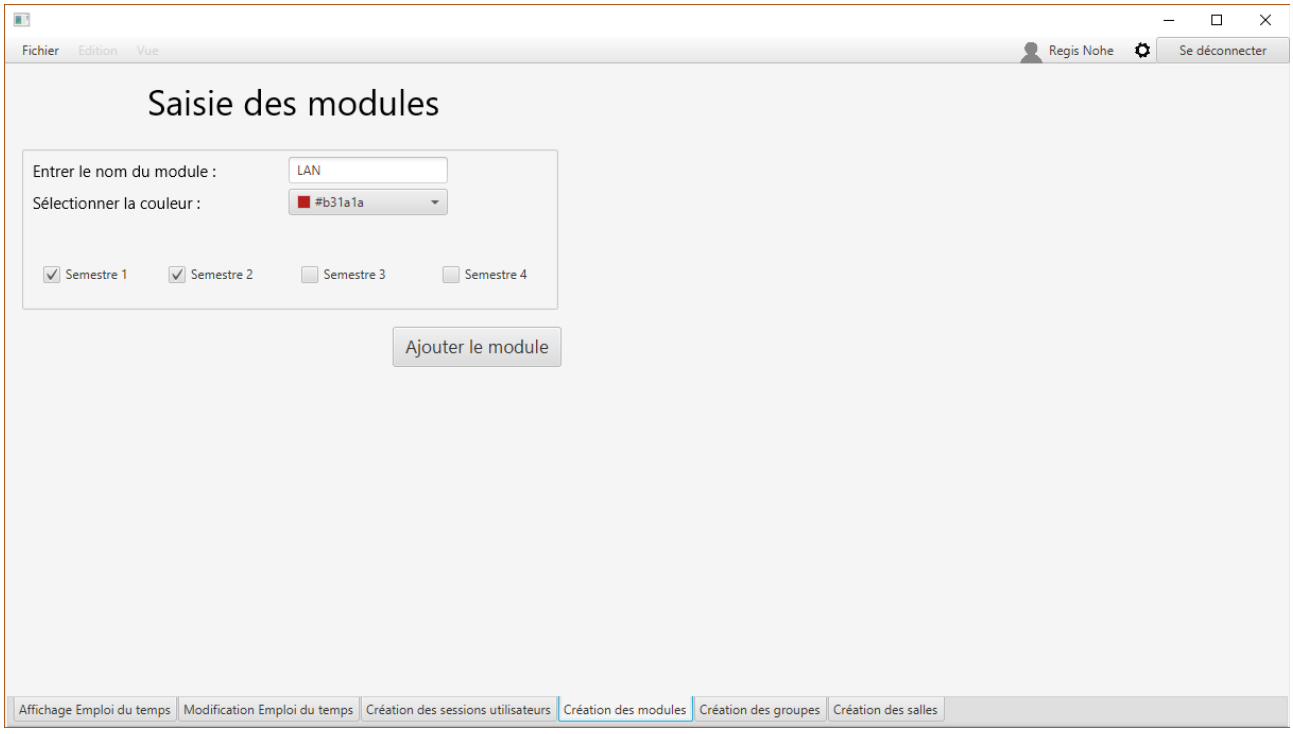
A- Fenêtre de connexion au gestionnaire d'emploi du temps :



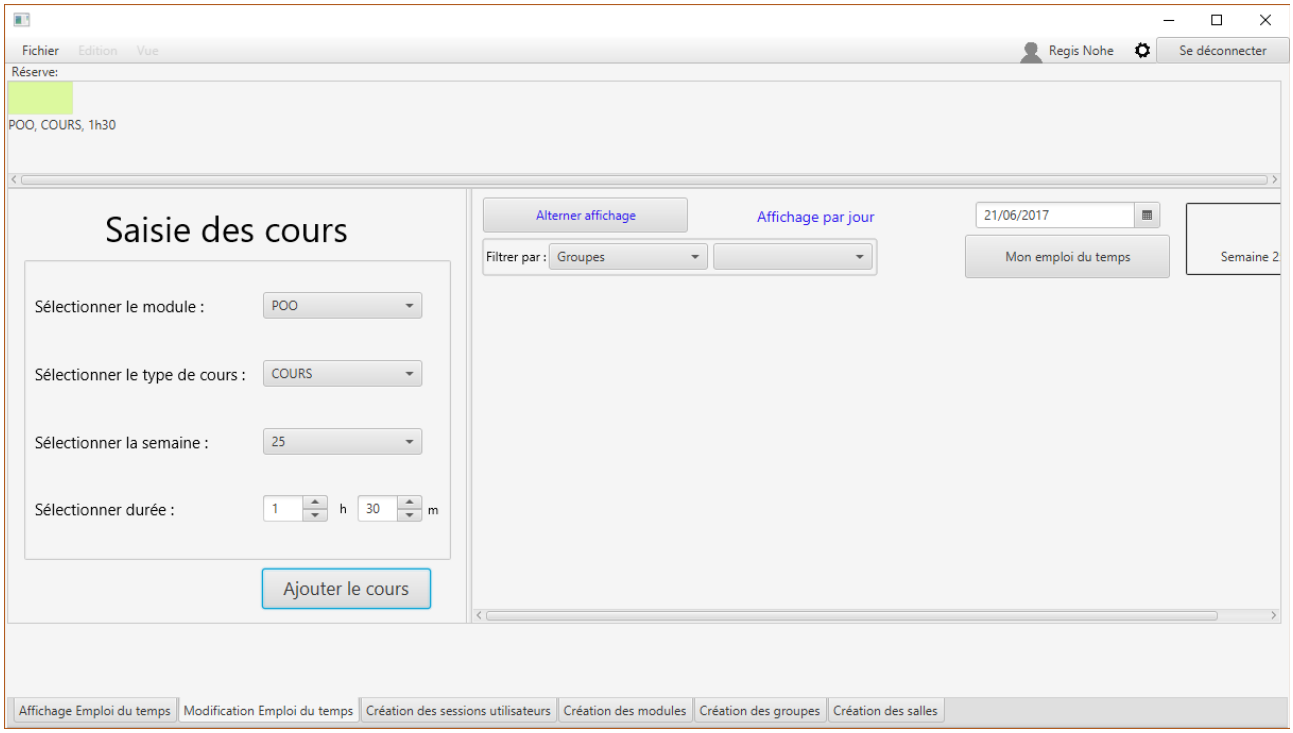
B- Fenêtre de saisie des groupes :



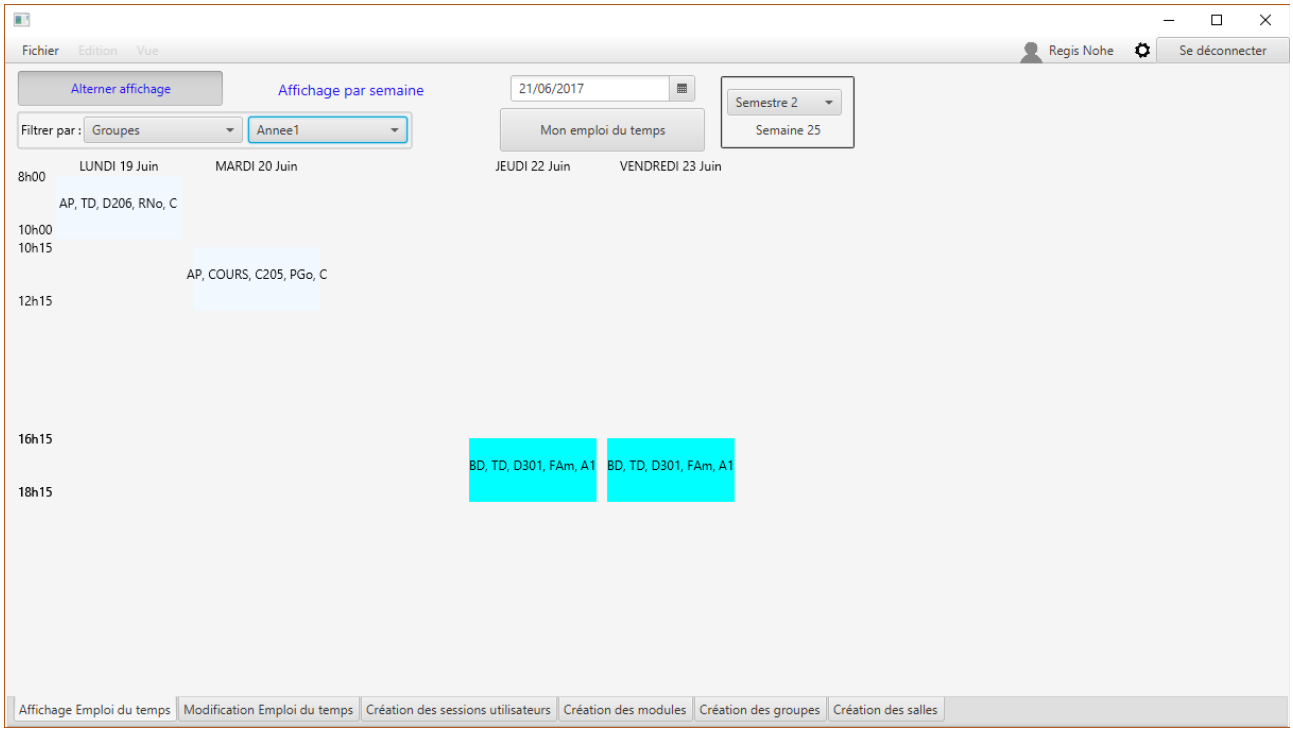
C- Fenêtre de saisie des modules:



D- Fenêtre de saisie des cours :



E- Fenêtre d'affichage de l'emploi du temps :



VI- Extensions possibles

En terme de fonctionnalités que nous pourrions ajouter au logiciel afin de le compléter, on peut citer :

- l’ajout des contraintes professeur (et le types de contraintes associées)
- mise en place d’une version mobile
- l’ajout d’une liaison à une base de données
- la possibilité de générer automatiquement l'emploi du temps