



# PlanIFTicateur

## Travail pratique 4

GLO-2004 – Génie Logiciel Orienté Objet – Hiver 2015

Travail présenté à  
M. Jonathan Gaudreault

Chayer, Philippe	<a href="mailto:Philippe.chayer.1@ulaval.ca">Philippe.chayer.1@ulaval.ca</a>	IFT	PHCHA47
Khouma, Abdou	<a href="mailto:abdou.khouma.1@ulaval.ca">abdou.khouma.1@ulaval.ca</a>	GIF	ABKHO9
Gadoury, Gabriel	<a href="mailto:Gabriel.gadoury.1@ulaval.ca">Gabriel.gadoury.1@ulaval.ca</a>	IFT	GAGAD1
Yéo, Clotioloman	<a href="mailto:Clotioloman.yeo.1@ulaval.ca">Clotioloman.yeo.1@ulaval.ca</a>	GLO	CLYEO1

## Table des matières

Table des matières .....	2
Introduction .....	3
Principales interfaces utilisateurs .....	4
Fenêtre principale .....	4
La fenêtre principale est essentiellement composée des parties suivantes : .....	5
• Une barre de menu au haut de l'écran, permettant l'accès à des fonctionnalités triviales .....	5
Fenêtre statistiques .....	6
Modèle du domaine .....	6
Modèle des cas d'utilisation .....	9
Architecture logique .....	10
Diagrammes de classes de conception .....	10
Diagrammes de séquences .....	12
Déplacement d'un cours .....	12
Affichage de la grille horaire .....	14
Fonctionnalité dont nous sommes les plus fiers.....	15
Gestion de projet .....	17
Conclusion .....	18

## Introduction

---

Ce rapport consiste en une analyse de la faisabilité pour la conception d'un logiciel permettant la gestion et la création d'horaire. Cette analyse sera basée sur les informations fournies par le client. Le but du logiciel est de faciliter la gestion des horaires de session effectuée par la direction de programme.

- **PlanIFTicateur** est un programme apte à construire un horaire de session de façon interactive et peut, si souhaité, faire la génération de votre horaire de façon optimale (rapide, efficace et flexible). Le but primaire est de rendre agréable la construction de l'horaire tout en réduisant le temps consacré à ce dernier.

- Une planification automatique optimale grâce à un algorithme de recherche efficace se basant sur les restrictions et les statistiques de ce qu'est un bon horaire.

- Une interface simple et intuitive visant à aider l'utilisateur dans son travail.

- Un suivi simple et efficace grâce à une fonctionnalité permettant de prendre des notes en lien avec la grille horaire que vous construisez.

- Les horaires créés seront en tous points valides grâce à des fonctionnalités de validations optimales. Ces fonctions rendront facile et stimulante la création d'un horaire grâce à son aspect visuel et dynamique.

- Une importation des données et restrictions des cours instantanée facilitant le travail et limitant les erreurs ou les oublis.

- Des statistiques pertinentes disponibles pour la grille horaire en construction visant à l'optimisation de votre horaire.

- Une exportation de fichiers simple et facile à des fins d'utilisation hors programme.

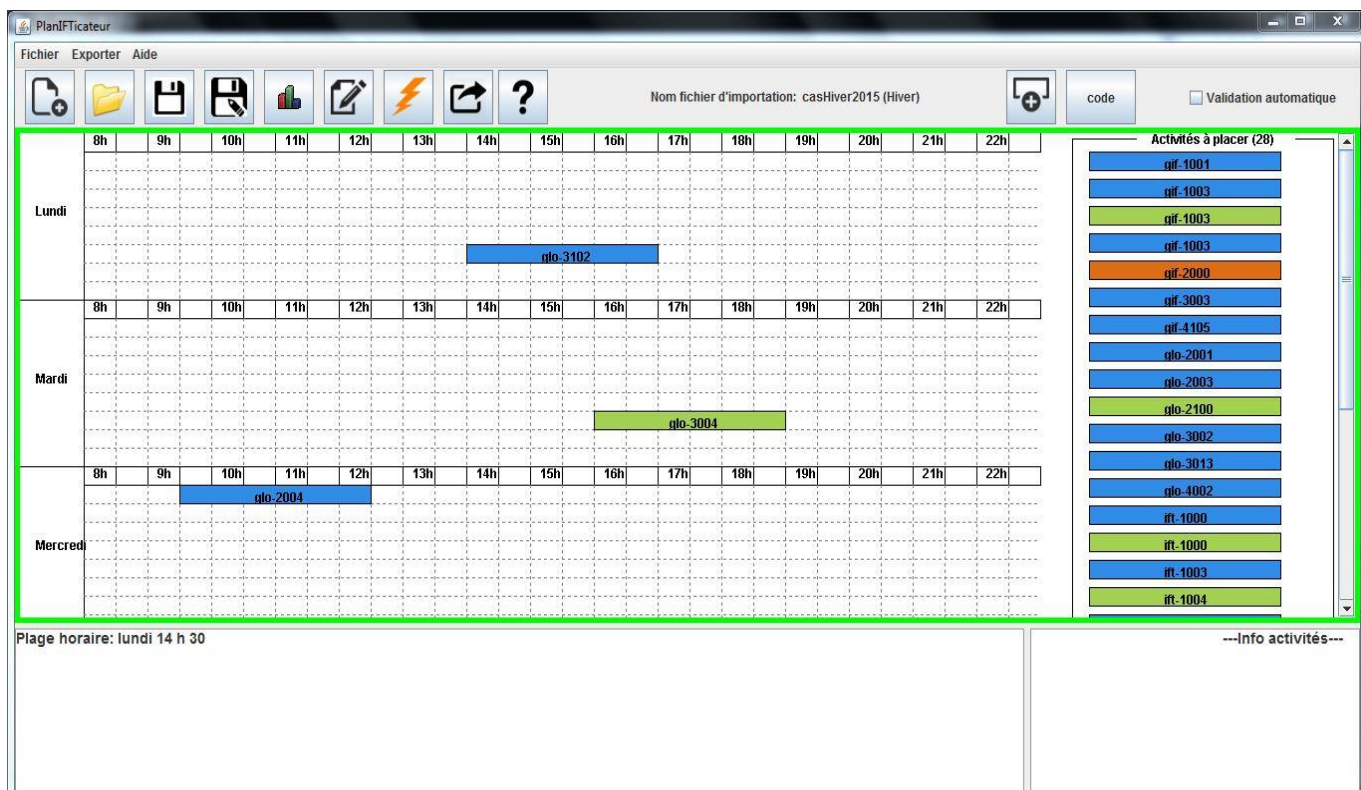
- Un historique est disponible pour un suivi efficace de toutes les modifications réalisées sur votre planification.

## Principales interfaces utilisateurs

Tout d'abord, une interface utilisateur doit miser sur l'aspect ergonomique. Tout en respectant les critères du client, nous avons réussi à rendre simple et intuitif l'interface graphique de notre programme. Les informations généralement recherchées par les utilisateurs sont concentrées au haut de la fenêtre, tandis que les autres sont positionnées de manière à respecter les contraintes du client.

Une fenêtre pour la grille horaire encadrée par une couleur vivante et représentative de l'état du travail indiquera à l'utilisateur s'il y a un problème ou si tout va bien, un champ au bas communiquera avec précision les différentes erreurs au besoin ou encore des messages de confirmation et une barre de menu semblable à toutes les barres représentées dans les différents logiciels connus de tous. Voyez maintenant les différents croquis :

### Fenêtre principale

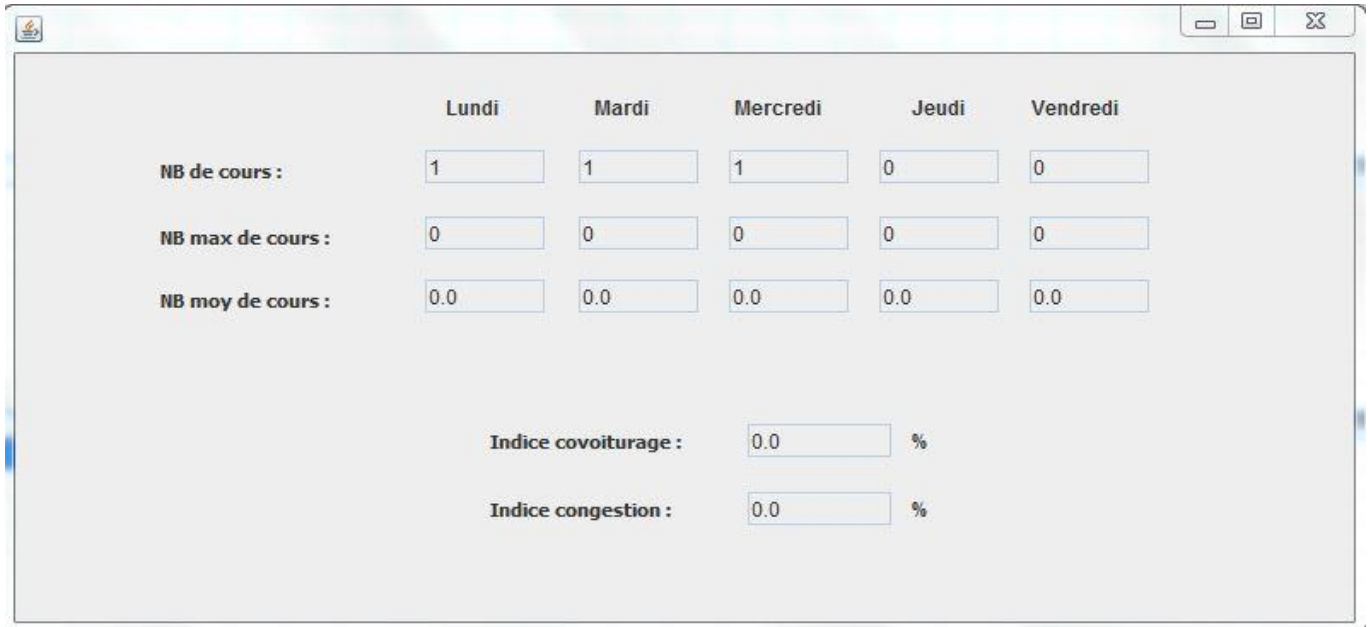


La fenêtre principale est essentiellement composée des parties suivantes :

- Une barre de menu au haut de l'écran, permettant l'accès à des fonctionnalités triviales
- Une grille horaire qui suivra la même logique, c'est-à-dire simple et intuitive.
- Une liste des activités non attribuées (située à droite)
- Un champ pour l'indication d'erreurs
- Un bouton planification automatique permettant de faire l'essai de la fonctionnalité de planification automatique de l'application
- Un bouton note permettant à l'utilisateur de prendre des notes
- Un bouton statistique dont le contenu est détaillé ici-bas.

## Fenêtre statistiques

La liste des statistiques est automatiquement affichée pour tous les jours de la semaine.



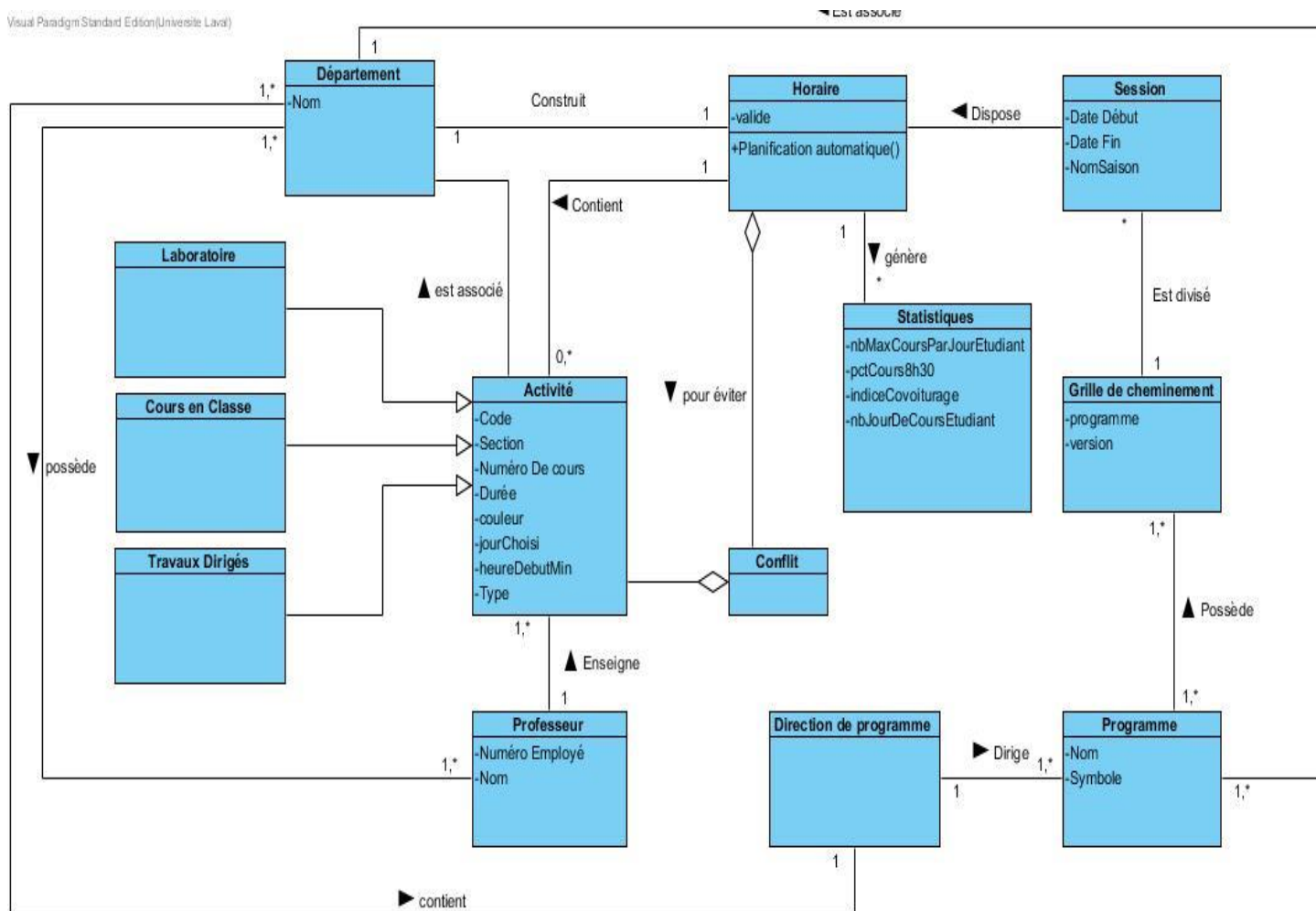
The screenshot shows a software window titled 'Fenêtre statistiques'. It contains a table of statistics for the days of the week (Lundi to Vendredi). The statistics include the number of courses (NB de cours), the maximum number of courses (NB max de cours), and the average number of courses (NB moy de cours). Additionally, there are two summary statistics: 'Indice covoiturage' and 'Indice congestion', both showing a value of 0.0 and a percentage sign.

	Lundi	Mardi	Mercredi	Jeudi	Vendredi
NB de cours :	1	1	1	0	0
NB max de cours :	0	0	0	0	0
NB moy de cours :	0.0	0.0	0.0	0.0	0.0
Indice covoiturage :	0.0 %				
Indice congestion :	0.0 %				

## Modèle du domaine

---

Cette section consiste à vous présenter une perspective générale du fonctionnement de *PlanIFTicateur*. Comme tout bon modèle d'affaires, le schéma ci-dessous contient l'ensemble des informations nécessaires à la compréhension de notre projet, voire notre logiciel. Suite au visionnement du dît schéma, nous pourrons constater l'ampleur du travail accompli.



### Détails du diagramme :

Pour le diagramme de modèle de domaine, plusieurs choix ont été faits afin d'obtenir le diagramme ci-dessus. Tout d'abord, le département est présent, puisque c'est ce dernier qui utilisera l'application (ou quelqu'un mandaté par la direction du département). Ensuite, la direction de programme est directement liée aux programmes, puisque c'est la direction qui dirige ces programmes et qui construit les grilles de cheminement.

Ces grilles sont alors divisées en sessions qui ont tous un horaire à respecter selon certains critères, c'est-à-dire d'éviter les conflits d'horaires entre les cours d'une même session (facteurs fournis par les grilles de cheminement). Dans notre diagramme, l'horaire peut être vu de deux façons :

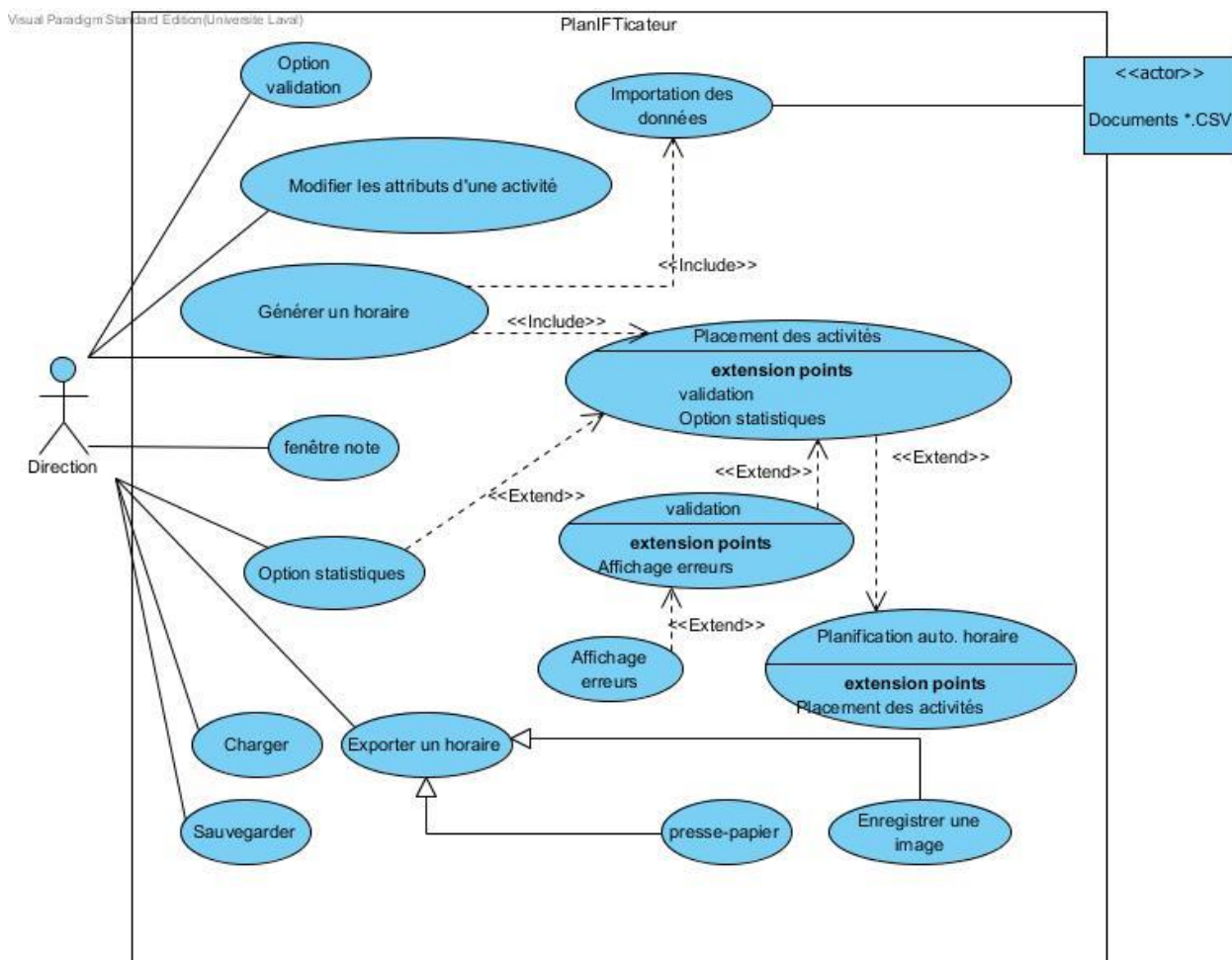
- Soit comme étant un horaire d'étudiant qui contient normalement cinq cours avec, possiblement des laboratoires;
- Soit comme étant un horaire de tous les cours qui sont donnés dans un département. D'ailleurs, c'est de cette façon que la direction planifie les horaires des sessions qui permettra ensuite d'obtenir celle des étudiants.

Par la suite, il y a les activités, qui sont réunies dans une superclasse pour ce diagramme et qui se divise en trois sous-classes : les laboratoires, les cours en classe et les travaux dirigés. Ces sous-classes correspondent aux différents types d'activités qui seront mises à l'horaire. Finalement, il y a les étudiants qui font partie du modèle puisque l'horaire construit concerne directement les étudiants. D'ailleurs, les statistiques générées par l'application sont basées sur ces étudiants.



## Modèle des cas d'utilisation

Maintenant que nous avons une bonne vue d'ensemble, il sera plus simple de comprendre la section ci-présente. Le modèle qui suivra représente le système, *PlanIFTicateur*, abritant tous ses cas d'utilisation. Il y aura également une représentation sommaire des acteurs du système. Suite à ce modèle, suivra l'architecture logique du logiciel.



## Architecture logique

---

### Diagrammes de classes de conception

---

Le diagramme qui suit représente le corps des classes utilisées par notre application. Par rapport précédent, nous avons ajouté plusieurs méthodes et attributs dans les différentes classes, voir toutes. Certains attributs ont été ajoutés suite à plusieurs constations lors de l'implémentation du code.

Nous avons aussi ajouté une classe **ImageExporter** qui est utilisée pour exporter les images de l'horaire dans un format prédéfini. Le contrôleur, nommé ici **HoraireController**, est la seule classe qui échange avec l'interface.

Tout élément de l'interface qui souhaite accéder à une fonction ou un événement doit passer par ce contrôleur qui s'assurera, par la suite, de faire la relation vers le code approprié. Le même phénomène se produit à l'inverse, c'est-à-dire lorsque les éléments du domaine, qui contient le code fonctionnel, désirent interagir avec l'interface utilisateur.

*(voir diagramme sur la page suivante)*



## Diagrammes de séquences

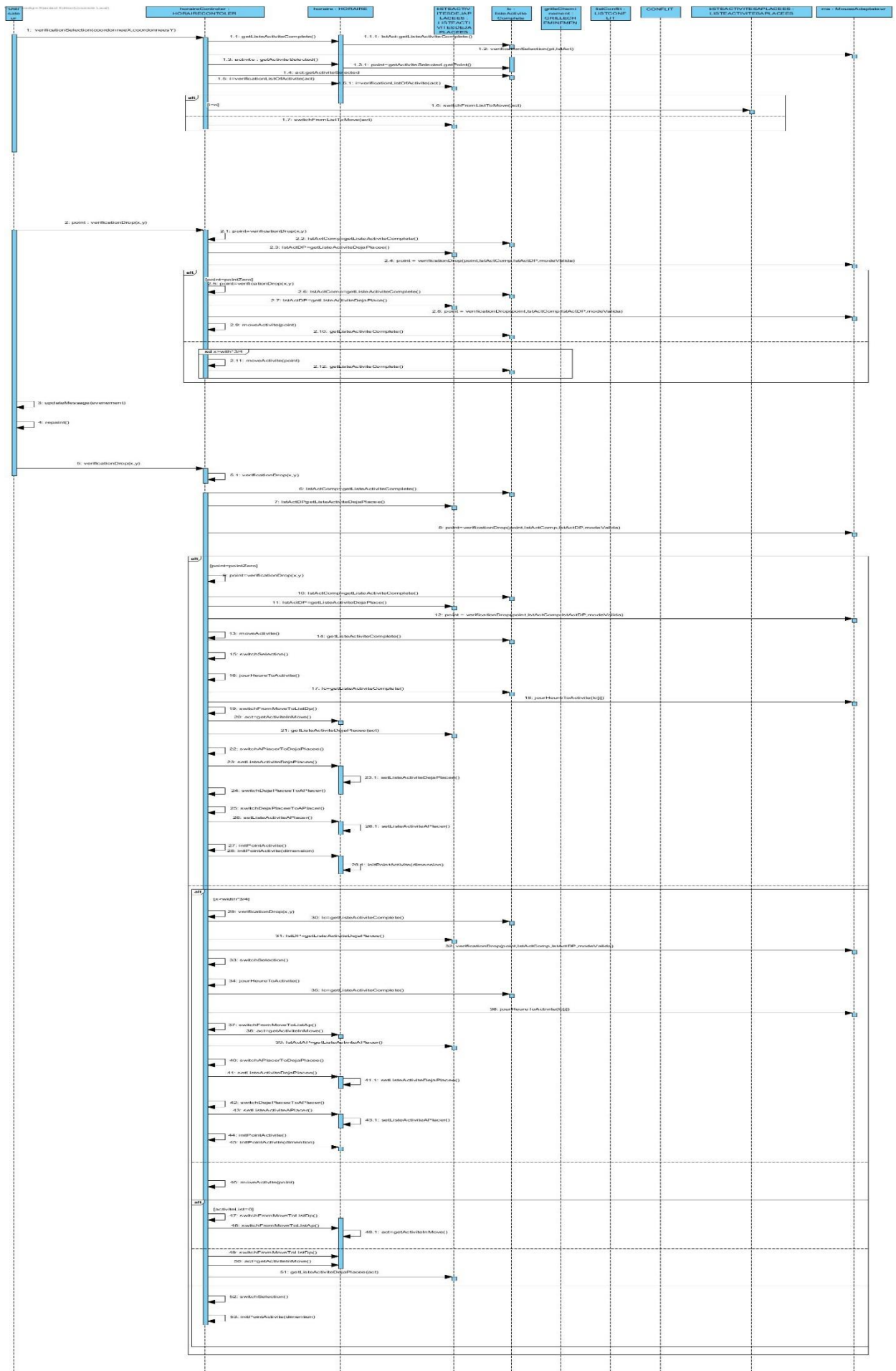
---

### Déplacement d'un cours

Tout d'abord, lors du déplacement d'un cours, une méthode du contrôleur **verificationSelection(coodonneeX,coordonneesY)** est lancée. Cette méthode appelle **getListeActiviteComplete()** de l'horaire. Par la suite, cette méthode de l'horaire appelle une méthode de la classe **listeActiviteComplete** qui retourne la liste des activités complète. Puis, la méthode du contrôleur appelle le **mouseAdapter** et la méthode **verificationSelection(pt,lstAct)**. Puis, la même méthode du contrôleur obtient l'activité qui était sélectionnée à l'aide de la méthode **getActiviteSelected()** qui retourne un point avec lequel l'activité est associée. Après avoir obtenu l'activité sélectionnée, il (le contrôleur) fait une vérification sur la liste des activités en appelant la méthode **verificationListOfActivite(act)**. Celle-ci permet de valider le déplacement et ainsi, déplacer l'activité dans la liste des activités placées ou bien la retourner à l'endroit de départ avant le déplacement. Cette première partie, s'assurer que la sélection est valide (par exemple, que nous avons bien sélectionné une activité en cliquant)

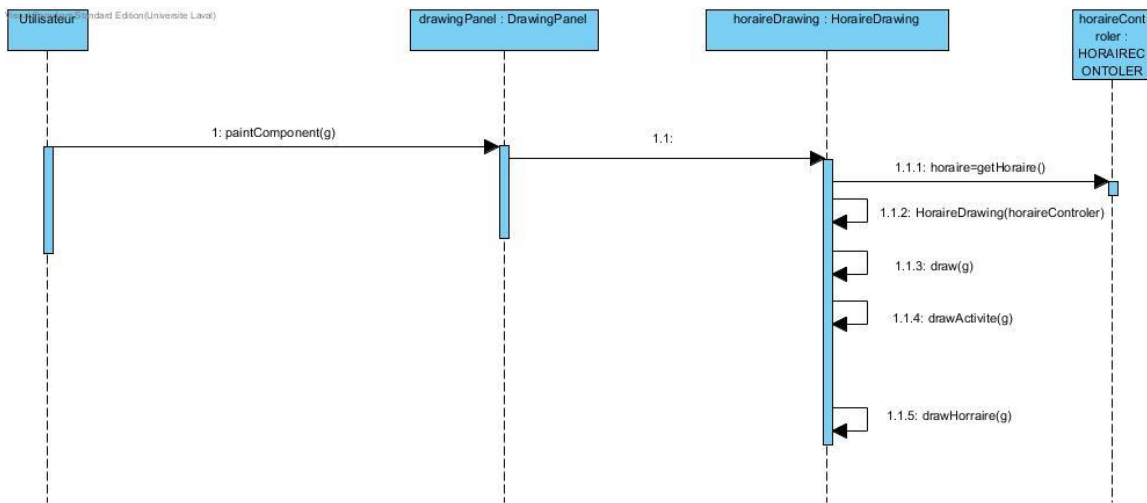
Ensuite, il faut s'assurer que le cours soit accroché à l'horaire. Ceci est exécuté à l'aide de la méthode **verificationDrop(x,y)**. Cette méthode permet de faire en sorte que le cours s'accroche à la grille horaire lors du déplacement. Ainsi, il ne tombe pas à une position entre deux plages horaire, il est toujours à un emplacement qui est valide pour être relâché (drop).

*(Voir diagramme sur la page suivante)*



## Affichage de la grille horaire

L'utilisateur appelle la méthode **paintComponent(g)** de **DrawingPanel** en passant un objet graphique. **DrawingPanel** instancie un objet **HoraireDrawing** qui invoque à son tour **getHoraire** du contrôleur qui retourne un booléen. Selon la valeur de ce booléen, la méthode **draw(g)** invoque à la fois **drawHoraire(g)** et, soit **drawActiveite(g)**, soit **drawActiveite(g)**.



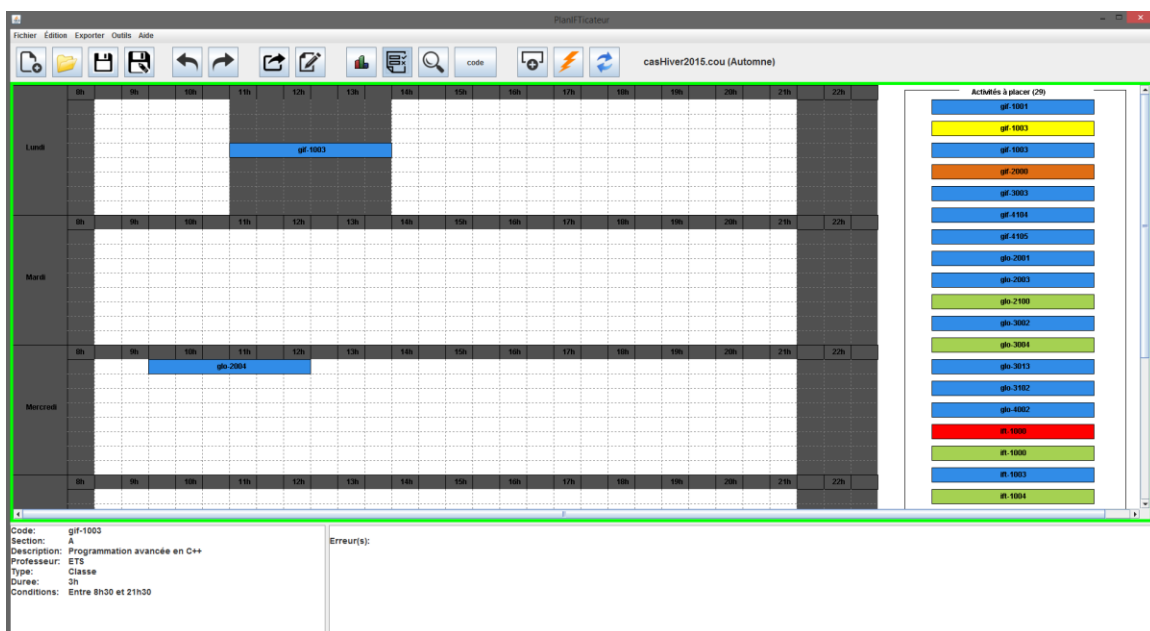
## Fonctionnalité dont nous sommes les plus fiers

Nous avons choisi la fonctionnalité de *drag-and-drop*. Cette fonction a posé un certain défi à implémenter. Nous voulions avoir quelques choses qui montrait visuellement le processus en action, et non seulement un déplacement de la souris sans objet en-dessous. De plus, nous voulions que le cours qui était en train de se faire déplacer par l'utilisateur soit toujours dans une case de l'horaire. Ainsi, lorsque l'utilisateur déplace un cours, dès que le cours est rendu au-dessus de l'horaire, celui-ci est accroché à une case. Lorsque l'utilisateur déplace la souris, le cours reste en place jusqu'au moment où la souris est plus proche de la bordure de la prochaine case.

Cette fonctionnalité nous a aussi permis de faire en sorte que la validation automatique bloque les cases qui n'étaient pas valides. Ainsi, lorsque l'utilisateur sélectionne un cours, les cases invalides (donc, qui feraient en sorte que l'horaire n'est pas valide si le cours était placé à cet endroit) sont inaccessibles par l'utilisateur. Même si la souris passe par-dessus les cases grisées par la validation automatique, le cours reste en place. Il se déplacera seulement lorsque la souris sera dans une position qui permettrait de déposer le cours dans une position qui est considéré comme valide.

Cette fonctionnalité est expliquée plus en détails dans la section : Diagramme de séquences dans la sous-section : Déplacement d'un cours.

Voici des captures d'écran pour illustrer le drag-and-drop avec la validation automatique qui est activée. Un cours a été placé dans l'horaire et le cours en jaune dans la partie de droite est le cours sélectionné (l'utilisateur doit garder le bouton de la souris enfoncé pour que le cours soit sélectionné). La zone grise est la zone où le cours ne peut être positionné. Autrement, il y aurait un conflit d'horaire.





Par la suite, il est possible de constater que la souris se trouve de l'autre côté de la zone grise (la zone de conflit) mais le cours n'a pas suivi. Ceci est causé par notre fonction qui fait les validations lors du *drag-and-drop*. Tant que le cours ne peut être dans une position valide, celui reste à l'endroit où il est actuellement.

Planificateur

Fichier Edition Exporter Outils Aide

cashiver2015.cou (Automne)

Activités à placer (29)

- pf-1001
- pf-1003
- pf-1000
- pf-1002
- pf-1004
- pf-1005
- pf-1006
- pf-1007
- pf-1008
- pf-1009
- pf-1010
- pf-1011
- pf-1012
- pf-1013
- pf-1014
- pf-1015
- pf-1016
- pf-1017
- pf-1018
- pf-1019
- pf-1020
- pf-1021
- pf-1022
- pf-1023
- pf-1024
- pf-1025
- pf-1026
- pf-1027
- pf-1028
- pf-1029

Code: pf-1003  
Section: 8  
Description: Programmation avancée en C++  
Professeur: ETS  
Type: Classe  
Durée: 3h  
Conditions: Entre 8h30 et 21h30

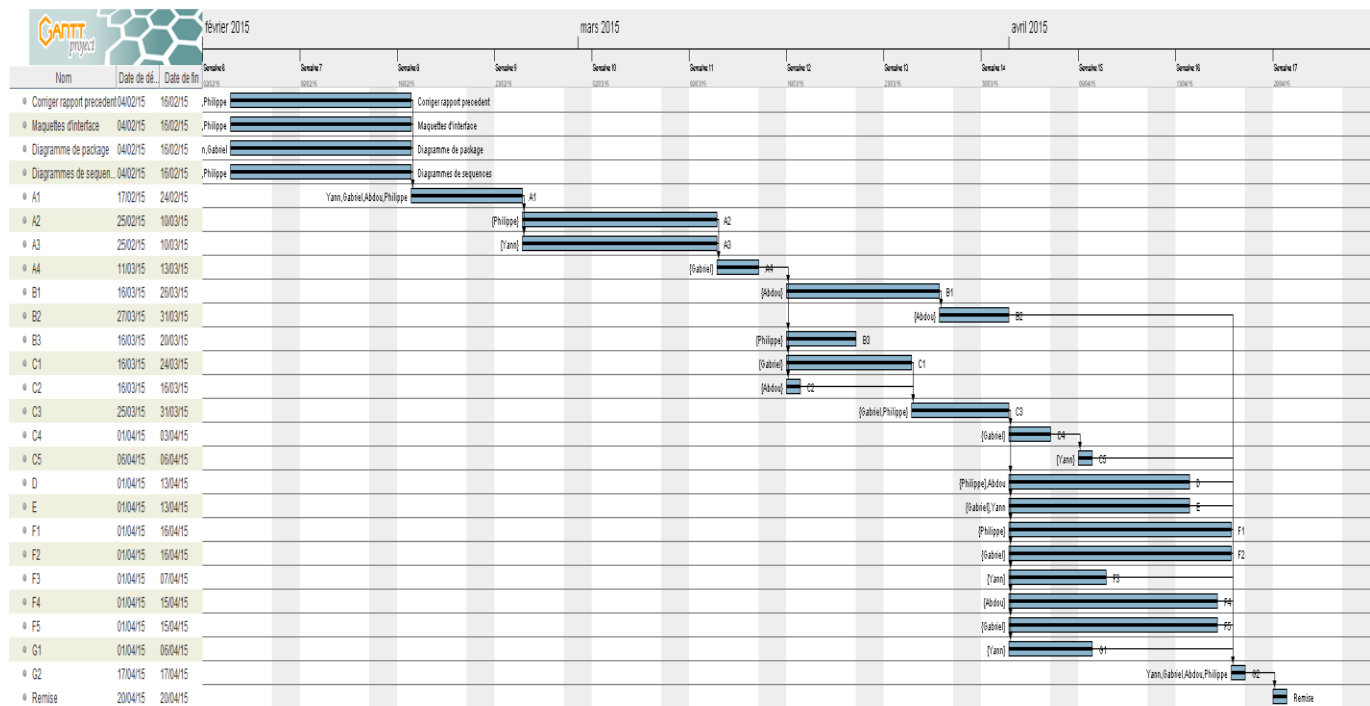
Plage horaire: lundi 10 h

Erreur(s):



# Gestion de projet

Voir [diagrammeDeGant.gan](#) .



## Conclusion

---

Pour conclure ce rapport, nous croyons que nous avons répondu aux exigences du client. Les fonctions qui ont été demandées ont été implémentées dans notre application. Il est possible d'ouvrir un horaire vide, d'y ajouter des cours en les glissant sur l'horaire à la position désiré (Drag and drop), il y a un indicateur de validité de l'horaire qui est disponible en tout temps (cadre qui entoure la zone de l'horaire qui est vert lorsqu'elle est valide et qui devient rouge lorsqu'il y a un conflit).

Une limite de notre application vient du fait que la zone horaire est composée d'une seule zone de dessin (drawing panel). Donc, lorsqu'il y a un grand nombre de cours, il est possible d'aller voir les cours qui sont dans le bas de la fenêtre, cependant, nous ne pouvons les placer dans l'horaire, puisque celle-ci n'est plus visible dans la fenêtre. Par contre, lorsqu'un cours est placé sur l'horaire, tous les cours qui étaient situés en-dessous du cours qui a été placé sont repositionnés (il remonte vers le haut). Donc, en construisant l'horaire, les cours du bas qui étaient trop loin et non disponibles pour être placés sur l'horaire peuvent être facilement assignés une heure et une journée lorsqu'il remonte. Pour éviter ce problème, nous aurions dû utiliser deux drawing panel pour ainsi avoir une *scroll-bar* ajustable par panel.

De façon général, le programme est facile d'utilisation et il est intuitif pour un utilisateur débutant. Nous avons utilisé des symboles familiés pour identifier les boutons de l'application pour permettre à une personne qui n'a jamais utilisé le programme de facilement l'utiliser.