

2012

EFREI

Ilan SASPORTAS
Cyril SAYADA

22

31-mai-12

JEUDI

	L 1					L 2					L 3			L3 Asy	M1 Asy	M2 SI-E	WM
	A	B	C	D	PL1	A	B	C	D	PL2	A	B	C				
1																	
2																	
3																	
4																	
5																	
6																	
7	30																
8																	
9	30:00	A	C	E	F						A	4					
10		BA	QU	MO	PER												
11	30	RR	INT	NC	ILH												
12		EJ	ER	YS	ON												
13	30:00	ON	O														
14																	
15	30	FH	FH	E	FH												
16		22	11	34	32												
17	10:00	B	D	G	J												
18																	
19	30	BA	QU														
20		RR	INT														
21	12:00	EJ	ER														
22		ON	O														
23	30																
24																	
25	13:00	FH	FH														
26		22	11														
27	30																
28																	
29	14:00																

[DOSSIER DE CONCEPTION]

PROJET C++ - GESTION DE PLANNING – EFREI PROMO 2015

Table des matières

Introduction	2
Projet – Spécifications fonctionnelles générales	3
Description sommaire	3
Utilisateurs	3
Type.....	3
Equipe	3
Fonctionnalités proposées	4
Analyse – Spécifications fonctionnelles détaillées	5
Date.....	6
Horaire	6
Salle.....	6
Professeur	7
Matiere.....	8
Cours	8

Introduction

Dans ce projet, l'objectif est de fournir une aide à l'optimisation du planning dans une grande école d'ingénieurs en informatique. Nous proposons ainsi de réaliser une application à l'aide de l'outil Qt du langage C++, permettant de pallier à certains problèmes que peut rencontrer une grande école, au niveau des imperfections sur le planning. En effet, l'erreur est humaine, mais elle ne l'est pas pour l'ordinateur. Un programme pourra dans tous les cas vérifier l'heure qu'il transmet aux élèves, et aux professeurs, il pourra aussi vérifier qu'on n'affecte pas la même salle à deux cours différents. Ce n'est pas toujours le cas pour l'humain... A l'aide du fourni par la MOE et du cahier des charges, nous avons pu mettre au point un programme qui permettra une optimisation de planning.

Projet – Spécifications fonctionnelles générales

Description sommaire

Le projet consiste à fournir à l'administrateur (qu'on appellera ici à tout hasard par le trigramme WBS) une interface, gérant les contraintes, permettant de définir et d'ajouter des cours sans avoir de conflit. Les élèves aussi pourront voir l'emploi du temps généré.

Utilisateurs

Nous distinguons deux types d'utilisateurs pour notre application :

- L'administrateur, WBS : il peut ajouter des cours à l'emploi du temps
- L'élève : il peut seulement visualiser l'emploi du temps

Type

Application informatique, développée à l'aide du langage C++. On a utilisé la bibliothèque Qt pour gérer l'aspect graphique de notre application.

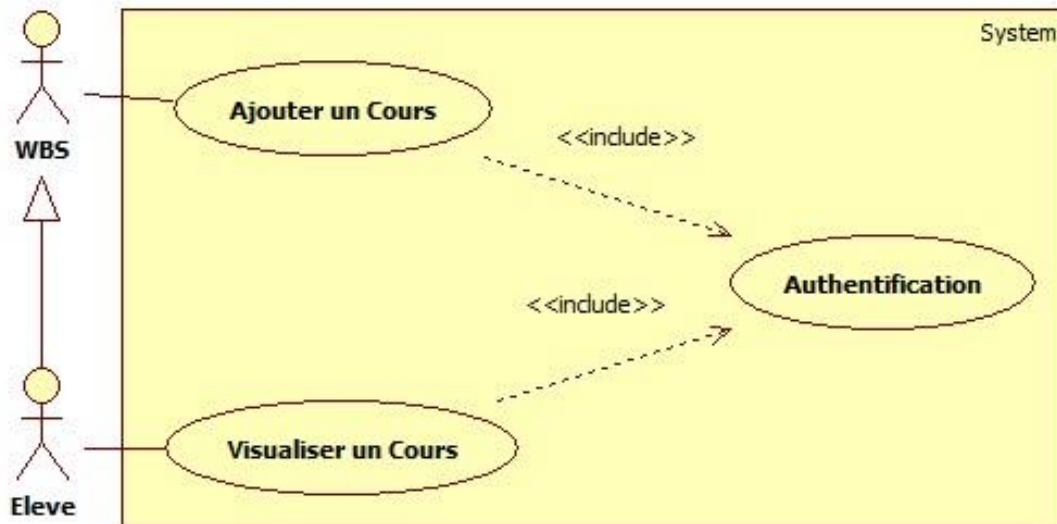
Equipe

Ilan SASPORTAS : Conception, programmation

Cyril SAYADA : Conception, programmation

Fonctionnalités proposées

Dans notre application, après l'authentification, nous proposons deux cas d'utilisation, dont un majoritaire qui représente l'ajout de cours.



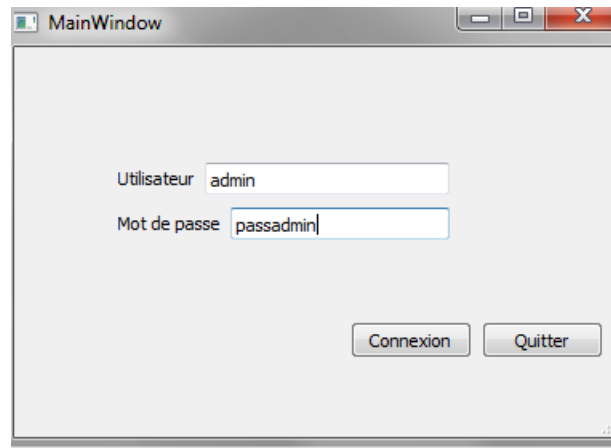
Tout d'abord, nous chargeons à partir de 3 fichiers textes nos données, il s'agit de :

- Salles.txt : on définit les salles sous la forme
 - **_typeSalle _nomSalle**
- Professeurs.txt : on définit les différents professeurs intervenants sous la forme
 - **_nom _module**
- Matieres.txt : on définit les différentes matières sous la forme
 - **_promo _module _nbAmphi _nbTD _nbTP**

Authentification

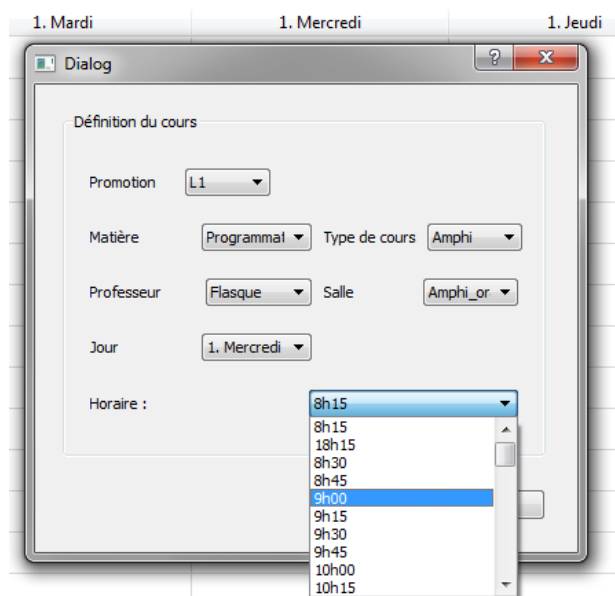
Une boîte d'authentification avec login/mot de passe permet de s'authentifier au système.

Voici la boîte de dialogue :



Ajouter un cours

Pour ajouter un cours, nous avons créé cette fenêtre de dialogue qui permet de sélectionner les options du cours à ajouter.



Analyse – Spécifications fonctionnelles détaillées

Ici , nous détaillons les différentes classes créées pour les utiliser dans notre application :

Date

-

Date
-_jour: int -_mois: int +int:Date()

Dans cette classe, nous définissons une classe pour la date. Celle-ci est composée d'un membre donnant le jour(*_jour*), et d'un autre le mois(*_mois*). Elle définit aussi le constructeur de la classe date permettant d'initialiser ses attributs.

Horaire

-

Horaire
-int:_heure -int:_min +Horaire()

Le premier membre permet de stocker l'heure (*_heure*). Le second stocke les minutes (*_min*). Ensuite, nous définissons les constructeurs de la classe qui permettront d'initialiser un objet.

Salle

Salle
-QString:_typeSalle -QString:_nomSalle
+Salle() +void:setTypeSalle(QString) +void:setNomSalle(QString)

La classe Salle permet de stocker deux attributs, l'un définissant le type de la salle (*_typeSalle*) , soit un amphi, salle de TD ou TP ; l'autre donnant le nom de la

salle(*_nomSalle*). Les méthodes définies sont le constructeur, et une méthode permettant de définir le type et une dernière le nom de la salle.

Professeur

Professeur
+QString: _nom +QString: _module +int: _faitPromoL1 +int: _faitPromoL2 +int: _faitAmphi +int: _faitTD +int: _faitTP
+Professeur() +void: setNom(QString) +void: setModule(QString) +void: setFaitPromoL1(int) +void: setFaitPromoL2(int) +void: setFaitAmphi(int) +void: setFaitTD(int) +void: setFaitTP(int)

La classe professeur stocke elle dans des QString les noms et module affectés au professeur (*_nom* et *_module*). Ensuite, des entiers déterminant si les professeurs sont affectés à la promo L1/L2, s'ils font des amphi/TP/TD. (*_faitPromoL1*, *_faitPromoL2*, *_faitAmphi*, *_faitTD*, *_faitTP*). On définit également son constructeurs, et des méthodes de type « set » pour définir les attributs précédents.

Matiere

Matiere
-int: _promo -QString: _module -int: _nb_TD -int: _nb_TP -int: _nb_Amphi -int: _duree_TD -int: _duree_TP -int: _duree_Amphi +Matiere() +void(): setPromo(int) +void(): setModule(QString) +void(): setNbTD(int) +void(): setNbTP(int) +void(): setNbAmphi(int) +QString(): getModule() const +int(): getPromo() const

Pour la classe *matiere*, on définit les attributs représentant la promo, le module, le nombre de TD/TP/Amphi, et leurs durées respectives. On fait ensuite des méthodes de type « set » pour définir ces attributs, et une méthode *getModule()* pour récupérer le module correspondant à l'objet défini. On fait de même pour *getPromo()*.

Cours

Cours
-Matiere: matiereDuCours -Professeur: professeurDuCours -int: promotionDuCours -Salle: salleDuCours -Date: dateDuCours -Horaire: dateDuCours +Cours() ~Cours() +void(): setMatiere() +void(): setProfesseur() +void(): setPromotionDuCours() +void(): setSalle() +void(): setDate() +void(): setHoraire()

Dans la classe *Cours*, on définit la *matiere* correspondante, le professeur affecté, et la promotion dans laquelle sera donnée ce cours, la salle utilisée, la date et l'horaire. On

utilisera les classes définie auparavant. Ensuite, les méthodes set définies permettent de changer les attributs.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hendrerit hendrerit consectetur. Nulla facilisi. Donec vitae sollicitudin risus. Fusce non libero et ante pellentesque varius eget nec ligula. Fusce luctus ipsum sollicitudin orci adipiscing quis tempor lorem accumsan. In egestas purus et massa hendrerit feugiat. Aenean quis odio non odio dapibus adipiscing. Curabitur id magna eu magna ullamcorper vulputate vel id arcu. Morbi turpis orci, porttitor a aliquet id, dictum eu risus. Sed facilisis lorem quis est mattis elementum. Sed faucibus vehicula tempor. Nam fringilla facilisis lacinia.¹

¹ Un petit easter egg pour la route ☺

Conclusion

Au regard de la difficulté, de la longueur, du temps imparti et des autres projets/examens/partiels en cours, nous ne pouvions pas, malgré toute la bonne volonté que nous y avons mis, terminer ce programme.

Mis à part ce que nous avons essayé de terminer, beaucoup d'autres améliorations seraient possibles :

- Mise en place d'une base de données au lieu de fichier texte(SQL)
- Gestion de contraintes
- Interface plus développée
- ...

Malgré tout, nous avons énormément progressé dans le langage C++, et avons appris pour ce projet l'utilisation de la bibliothèque Qt.