UNIVERSITÉ DE FRIBOURG SUISSE
UNIVERSITÄT FREIBURG SCHWEIZ

Ingénierie du document pour le web

Alina Petrescu
Nevena Radovanovic
(Groupe 10)

27 mai 2013

DEW 2013 - Project Report

Département d'Informatique - Departement für Informatik • Université de Fribourg - Universität Freiburg • Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland

http://diuf.unifr.ch

Table des matières

1 Avant-propos

Notre projet représente un vrai travail de groupe qui nous a tenues occupées durant plusieurs semaines parsemées de quelques nuits blanches inévitables mais inoubliables. La collaboration pour la partie programmation a été facilitée par le système de stockage en ligne *Git* (qui est un système de gestion de versions, concurrent du système SVN) et par son service Web appelé *GitHub* qui héberge notre projet. De plus, nous avons utilisé la base de données XML native *eXist* qui a aussi été notre serveur Web. La sauvegarde des données, l'organisation des pages et la génération du mapping ont été réalisés grâce au framework *Oppidum* écrit en *XQuery/XSLT*. Toute cette infrastructure, qui nous a été mise à disposition gracieusement par l'équipe du cours DEW, a été complétée par l'utilisation, sans modération, des réseaux sociaux. Par contre, aux séances de laboratoire, nous avons participé en chair et en os et avec beaucoup de plaisir.

Concernant le timing, il n'y a pas eu de problèmes grâce à l'efficacité de notre équipe, d'un côté, et à la qualité de l'encadrement dont nous avons pu bénéficier, d'un autre côté.

L'esprit d'équipe a aussi persisté durant la rédaction du rapport final. Nous avons décidé de faire ce rapport exclusivement en français pour faciliter la propagation de notre travail dans le monde francophone.

2 Introduction

2.1 Organisation du document

À part les deux premières sections, l'avant-propos et l'introduction, ce rapport contient encore quatre sections et la bibliographie.

Dans la troisième section 3, on donne des informations concernant les technologies que nous avons utilisées.

Dans la quatrième section 4, on détaille l'implémentation choisie pour chaque onglet de notre site Web dynamique. On indique les paradigmes choisis et les solutions proposées du point de vue de l'ingénierie du document pour le Web.

Dans la cinquième section 5, on évoque les problèmes que nous avons eus tout au long de ce projet, ainsi que les solutions apportées.

Dans la sixième section 6, on fait la synthèse du projet, en suggérant de possibles améliorations du travail.

2.2 Motivations

L'objectif de ce mini projet était de se familiariser avec toutes les technologies du Web vues dans le cours. Notre tâche consistait à mettre en œuvre la vue "étudiant" du site déjà existant *xmoodle*, vue composée de six pages (onglets). Le travail sur le projet nous a permis de mettre en place les notions théoriques qui nous ont été présentées et d'approfondir plus particulièrement des aspects pratiques liés au "monde" XML.

3 Langages et Technologies

Dans notre cours de l'ingénierie du document pour le Web, nous avons vu un grand nombre de technologies Web. Pour la plupart d'entre elles, nous avons trouvé une application dans notre projet :

HTML (en anglais *HyperText Markup Language*) est un langage de balisage (de mise en forme) permettant aux navigateurs d'afficher des pages sur le Web.

XHTML (en anglais *eXtensible HyperText Markup Language*) est le successeur d'HTML. Il est une "réécriture" plus stricte de la version précédente du langage et se base sur la syntaxe XML.

CSS (en anglais *Cascading Style Sheet*) sont des feuilles de style en cascade, permettant de décrire l'esthétique des documents HTML et XML. Elles permettent surtout une séparation entre le contenu et la forme de ces documents.

XML (en anglais *eXtensible Markup Language*) est un langage de balisage plus versatile, un métalangage (au même titre que SGML). Si un document respecte les règles lexicales et syntaxiques du XML, on dit qu'il est bien formé (voir 4).

XML Schema est un langage de description de format pour un document XML. Il permet également de savoir si le document est valide.

DTD (en anglais *Document Type Definition*) est un document permettant de décrire la structure des documents SGML et XML à l'aide d'une liste d'éléments légaux et d'attributs.

XPath est un langage qui permet la localisation des nœuds d'un document XML en utilisant une syntaxe propre, non XML.

XSL-T (en anglais *eXtensible Stylesheet Language Transformations*) est un langage de transformation XML de type fonctionnel permettant de transformer un document XML. Le résultat obtenu peut être en un format quelconque (XML, XHTML, Latex, txt, etc.).

XQuery est un langage de requête permettant non seulement d'extraire des informations d'un document XML (ou plus généralement d'une base de données native XML), mais également d'effectuer des calculs complexes à partir des informations extraites et de reconstruire de nouveaux documents XML.

SVG (en anglais *Scalable Vector Graphics*) est un format de données conçu pour décrire des ensembles de graphiques vectoriels et basé sur XML.

SMIL/Timesheet (en anglais *Synchronized Multimedia Integration Language*) est une langage qui correspond à une spécification W3C permettant de synchroniser des contenus multimedia différents afin de créer des présentations multimedia (voir 9).

Latex est à la fois un langage permettant à l'utilisateur de se concentrer sur la structure logique de son document et un environnement d'édition qui prend en charge le formatage du document lors de la compilation.

Dom (en anglais *Document Object Model*) est un standard W3C permettant à des programmes d'accéder ou de mettre à jour le contenu, la structure ou le style de documents XML et HTML.

SAX (en anglais *Simple API for XML*) a été initialement un API Java pour le XML. Grâce à son succès, cet API a aussi été implémenté dans d'autres langages (voir 10).

JavaScript (ou plus court *js*) est un langage de scripts orienté objet et utilisé dans les pages Web interactives. Il est destiné aux traitements du côté client et est normalement interprété par le navigateur (voir 9).

4 Implémentation

Tous les fichiers correspondant au site Web de notre projet sont regroupés dans le dossier *xmoodle_G10* qui se trouve sur le serveur Web *eXist* dans le container *exist/webapp/projets*. À son tour, le dossier *xmoodle_G10* contient plusieurs sous-dossiers parmi lesquels on peut mentionner :

- le dossier *views* où se trouvent principalement les fichiers *.xsl* qui correspondent aux six pages (onglets) du site ; nous en avons modifiés cinq, afin d'obtenir les vues mentionnées ci-dessous, par des transformations XSLT ;
- le dossier *models* contient des fichiers *.xql* ; à chaque vue on y trouve une requête XQuery correspondante ;
- le dossier *init* abritant surtout le fichier *db.xml* qui représente la base de données interrogée par les requêtes XQuery mentionnées ci-dessus ;
- le dossier *resources* qui regroupe, dans des sous-dossiers distincts, les feuilles de style (des fichiers *.css*), les images (des fichiers *.jpeg*, *.gif* et *.png*), et les scripts (des fichiers *.js*).

Par la suite, on discute ci-dessous l'implémentation de chacune des pages qui forment la vue "étudiant" du site Web.

4.1 Home

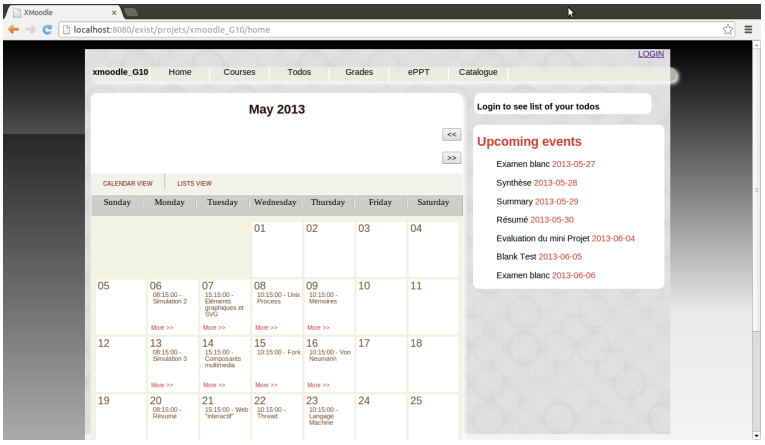

FIGURE 1 – L'onglet HOME

C'est la page d'accueil (la première page) de notre site Web (voir la Figure 1). Nous avons ajouté la *génération dynamique* de la liste des *todos* pour un utilisateur enregistré et connecté à la base de données. De plus, la liste des événements à venir est affichée identiquement pour tous les utilisateurs. Chacune de ces deux listes est présentée dans un "cadre" à part. Pour cela, nous avons modifié la feuille de style *base.css*. Cette approche a été facilitée par la réalisation d'une série d'exercices rendue auparavant. Dans la liste des *todos*, l'utilisateur peut voir les titres de ses activités, les tâches et les délais. Toutes les tâches sont triées par ordre de priorité afin qu'il puisse extraire les tâches les plus importantes. L'obtention de la liste des événements à venir a été un peu plus délicate à cause du formatage des dates à afficher. Cependant, l'utilisation des fonctions pour le traitement des dates (comme *day-from-date()* ou *month-from-date()*) nous a permis de résoudre ce problème. Il convient de remarquer que les couleurs utilisées pour mettre en évidence les données importantes dans nos deux listes ont été choisies en correspondance avec le design du calendrier déjà affiché sur la même page.

4.2 Courses

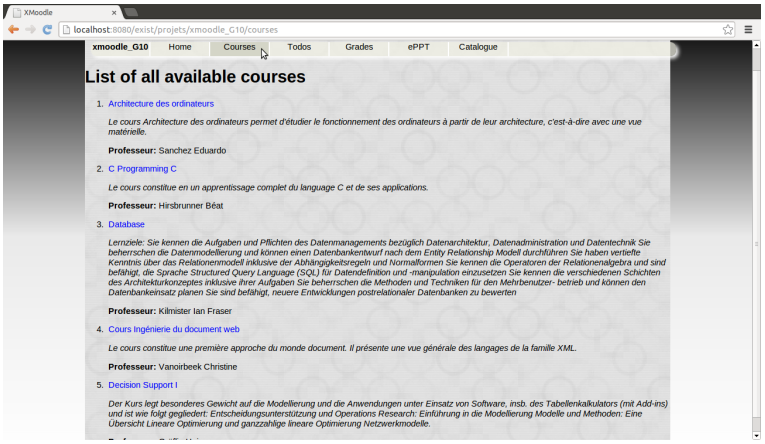

FIGURE 2 – L'onglet Courses

Cette page est une page "générique" qui ne dépend pas de l'utilisateur connecté (voir la Figure 2). Elle affiche une vue de tous les cours disponibles avec leur description et les noms des professeurs. Nous avons aussi colorié les titres des cours afin qu'ils puissent être visibles plus facilement. Les autres éléments de la liste (enseignants, descriptions) sont bien séparés, ce qui améliore la lisibilité des informations. La description des cours est écrite en italique et avant chaque enseignant il y a le titre *Professeur* écrit en gras. En utilisant nos connaissances acquises pendant les travaux en laboratoire, nous avons réussi à mettre en œuvre cette page sans trop de difficultés.

FIGURE 3 – L'onglet Grades, histogrammes

4.3 Grades

Nous avons passé beaucoup de temps pour réaliser cette page. Heureusement, c'est l'une des deux parties les plus intéressantes du projet (voir la Figure 3). Dans un premier temps, nous avons mis au point la démarche pour obtenir, à partir de données indiquées localement, un graphique simple et facile à lire. Nous avons opté pour un graphique en histogrammes (de type *Bar Chart*), où chaque histogramme représente soit une note soit une moyenne. Afin de tracer le graphique, nous avons utilisé, pour toutes les formes, l'élément graphique SVG *path* qui est versatile et facile à manipuler.

Dans un deuxième temps, nous avons renoncé aux données locales et alimenté le graphique avec des valeurs existantes dans un fichier XML. Pour obtenir des informations sur l'utilisateur connecté, nous avons utilisé les variables *user* et *currentCourse*. De leur côté, les notes ont été sauvées dans les variables *mymark* et *othersmarks*. Pour calculer les moyennes, nous avons eu besoin des fonctions *count()*, *div()*, *sum()* et *number()*. Afin d'afficher les acronymes des cours en-dessous des histogrammes, nous avons employé les fonctions *string()* et *substring()* (car certains acronymes étaient trop longs).

Dans un troisième temps, après avoir fait marcher le graphique, nous avons ajouté un bouton pour permettre à l'utilisateur de voir seulement les résultats des cours où il est inscrit ou les résultats de tous les cours. Pour pouvoir différencier ces deux cas, nous avons utilisé la fonction *contains()*. Finalement, nous avons rendu notre approche la plus générale possible afin qu'elle fonctionne pour un tout utilisateur authentifié et inscrit à un nombre quelconque de cours.

En outre, nous avons prévu une animation SVG assez amusante afin de rappeler à un visiteur qu'il doit d'abord se connecter avant de pouvoir consulter ses notes (voir la Figure 4).

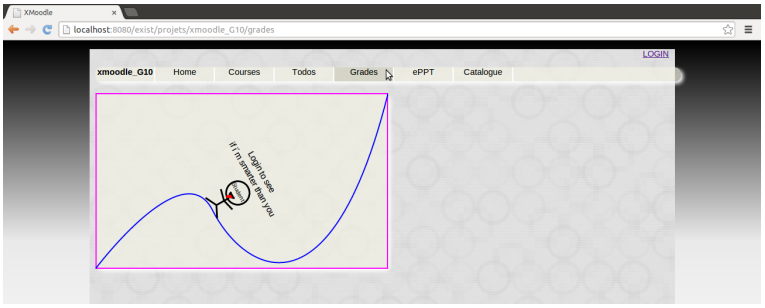

FIGURE 4 – L'onglet Grades, animation amusante

4.4 ePPT

Cette page correspond à l'autre partie très intéressante du projet. Notre but a été d'utiliser le langage *SMIL* et les feuilles de temps (*Timesheet*) afin de réaliser trois productions multimedia par une approche déclarative. Le document créé laisse les media indépendants et assure leur synchronisation temporelle ainsi que leur disposition spatiale. L'utilisateur garde un rôle actif et peut interagir avec l'interface qui intègre les composants multimedia.

En arrivant sur l'onglet *ePPT*, l'utilisateur peut naviguer entre trois présentations qui se succèdent dans la même page dynamique. Un premier scénario correspond à une présentation de type "diaporama", qui affiche une suite de plusieurs "transparents" dont le contenu défile progressivement. Le dernier "slide" contient des liens externes ainsi qu'un bouton permettant de reprendre le "diaporama". Le

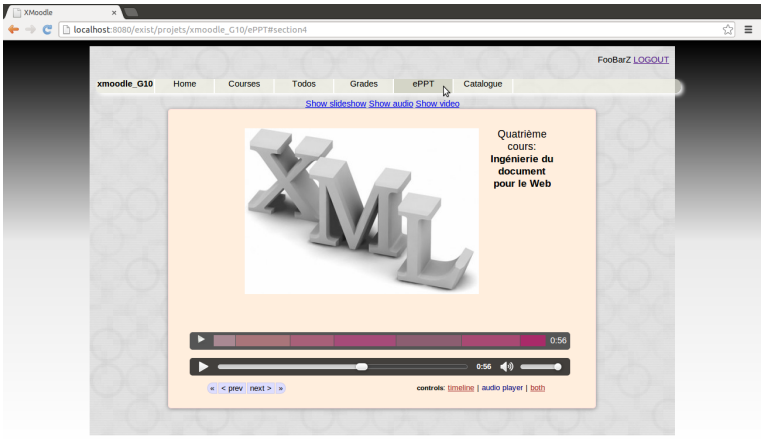

FIGURE 5 – L'onglet ePPT, audio

deuxième scénario est plus complexe et fait intervenir du texte, des images et du son (voir la Figure 5). Le tout est synchronisé et l'enchaînement peut se faire automatiquement. De plus, l'utilisateur a à sa disposition toute une panoplie de possibilités pour piloter le déroulement qui lui convient. Par exemple, la partie audio a été séquencée et l'utilisateur peut choisir à sa guise, à l'aide des contrôles, les passages qui l'intéressent. Bien sûr, ceux-ci restent synchronisés avec les images et le texte.

Le dernier scénario contient principalement une vidéo, mais aussi du texte et du son

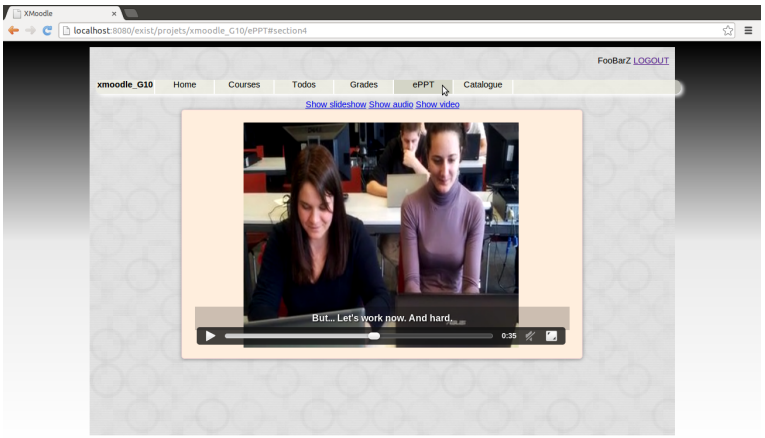

FIGURE 6 – L'onglet ePPT, vidéo

(voir la Figure 6). L'avancement de la vidéo est coordiné avec les sous-titres et peut être contrôlé par l'utilisateur.

Nous avons accordé une attention particulière aux détails. Par exemple, dans la barre correspondant au découpage audio, chaque subdivision a une couleur distincte et sa longueur est proportionnelle à la durée de la séquence appropriée. En outre, la navigation entre les trois scénarios multimedia est cohérente. Par exemple, lorsque l'utilisateur quitte la production vidéo, la progression de l'image et du son s'arrête mais elle est reprise quand l'utilisateur y revient.

Pour l'implémentation de la solution décrite ci-dessus, nous avons principalement utilisé la librairie *Timesheet* écrite en *JavaScript*. Cette solution a l'avantage d'être relativement bien supportée par les navigateurs, par rapport au langage *SMIL* lui-même qui, souvent, a besoin d'un player comme RealPlayer ou QuickTime pour s'exécuter. Ainsi, à part le fichier *timesheets.js*, nous avons ajouté aux ressources du projet le script *timeline.js*, les feuilles de style *audio.css*, *video.css* et *timeline.css*, ainsi que plusieurs fichiers d'images, de son et une vidéo.

Il a fallu rester vigilantes pour ne pas se laisser emporter dans les méandres des sources multimedia. Nous avons passé un temps non négligeable pour retoucher les images avec le logiciel *Fotoxx*, pour convertir les formats vidéo avec *ConvertAviToMp4*, pour mixer le son avec *Audacity*, et pour faire le montage vidéo avec *PhotoFilmStrip*.

Il convient de mentionner que nous avons testé la page *ePPT* avec les navigateurs Google Chrome et Epiphany (l'équivalent Linux de Safari) qui sont actuellement parmi les plus "réactifs" sur le marché.

4.5 Catalogue

Cette page affiche dans sa région centrale le contenu d'un fichier qui est généré à partir de la base de données (voir la Figure 7). Il est écrit en Latex et présente la liste de tous les cours avec leur description. Dans un "cadre" du genre "info-bulle"

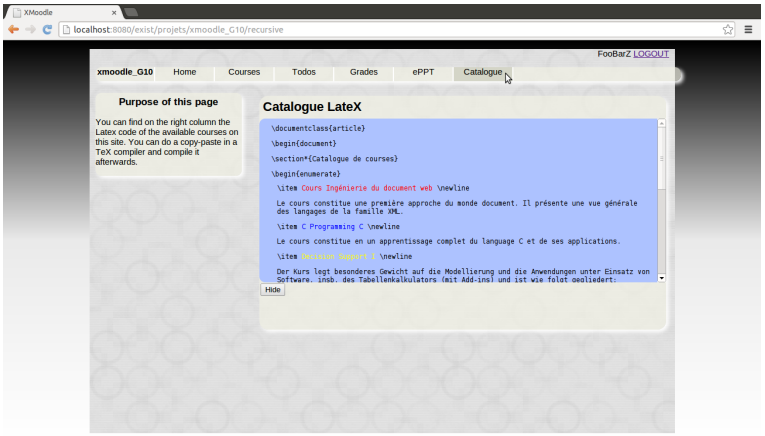

FIGURE 7 – L'onglet Catalogue

placé sur la partie gauche de la page, l'utilisateur est averti qu'il peut récupérer le contenu Latex qui est prêt à l'emploi par un compilateur approprié. À part nos connaissances en Latex, l'emploi de la fonction *position()* et de l'opérateur *mod* s'est avéré très utile pour colorier les titres des cours selon un pattern "rouge-bleu-jaune". Sinon, la technique utilisée est la même que celle mise en place dans la sous-section

5 Problèmes rencontrés

Durant notre travail, nous avons dû résoudre des problèmes inévitables aux implémentations choisies. Cependant, à part ces problèmes, nous avons rencontré toute une série de situations particulières pour lesquelles il a fallu trouver des solutions ad-hoc. Ci-dessous, on présente quelques exemples.

• Sans être des fans de Windows, nous avons essayé au début d'utiliser ce système d'exploitation pour réaliser le projet. Après plusieurs essais infructueux et, grâce à la patience et au pouvoir de persuasion des assistants, nous avons continué l'aventure dans le monde merveilleux de Linux.

• Un autre souci est arrivé aussi au début du projet. Tout de suite, *git-cola* nous a laissé une mauvaise impression. En fait, nous n'avons pas pu travailler sur un même fichier en même temps (à cause d'un cercle vicieux de messages exigeant de faire et refaire les opérations *push* et *pull*). De nouveau, à l'aide des assistants, nous avons pu mettre en place une stratégie pour éviter ces conflits (mais sans comprendre vraiment la cause du phénomène).

• Selon l'adage "Quand on cherche, on trouve", un autre problème est apparu à cause de notre curiosité ☹. En explorant la page *Todos*, nous avons créer un élément *ToDoList* pour un certain utilisateur de la base de données. Ensuite, cet élément s'est montré presque indestructible et nous n'avons pas pu l'effacer sans l'aide des assistants. Cependant, cette péripétie nous a donné l'idée de s'assurer qu'il n'y ait pas de balises *ToDoList* éventuellement vides ☺.

• L'euphorie provoquée par la fin de l'implémentation de la vue *Grades* a été de courte durée. Suite à l'opération "push", le graphique *SVG* qui s'affichait parfaitement sur l'un des ordinateurs est devenu disponible sur l'autre ordinateur. Malheureusement, cette fois, le graphique était tout dispersé. Heureusement, le problème a été rapidement résolu par la modification de la feuille de style *base.css*.

• Concernant l'utilisation du *SMIL* et du *Timesheet*, nous avons eu plusieurs problèmes. Voilà seulement deux exemples. L'accès aux sources multimedia à partir du fichier *.xsl* correspondant à la vue *ePPT* n'a pas été évident. Grâce à l'aide précieuse des assistants, dans la solution retenue, nous avons regroupé toutes les sources dans un dossier qui a été placé au niveau du container *exist/webapp/projets* et les références ont été précisées à partir du *localhost :8080/exist/projets*. Encore plus mystérieux : des balises comme *<link .../>*, *<script .../>*, ** ou *<source .../>* étaient bien acceptées mais elles n'avaient aucun effet. Afin de les rendre opérationnel, il a fallu utiliser des balises fermantes à part entière.

6 Conclusion

6.1 Synthèse

Dans le cadre de ce projet, nous avons réalisé l'implémentation de plusieurs vues dynamiques pour un site Web déjà existant et hébergé par un serveur Web *eXist*.

Il convient de mentionner que toutes les pages réalisées sont générées à la volée, sans qu'elles aient d'équivalents HTML préexistants et figés. Une telle page dynamique est créée chaque fois, par une transformation XSLT, en fonction de certains choix de l'utilisateur connecté et des informations stockées dans une base de données XML native (dans notre cas *eXist* à nouveau qui, de plus, utilise le framework *Oppidum*).

6.2 Améliorations futures

Nous pensons que les objectifs fixés dans le cadre de ce projet ont été bien réalisés et espérons que le jury aura la même opinion. Cependant, nous sommes conscientes que des améliorations peuvent être apportées et nous mentionnons ci-dessous quelques exemples :

- Afficher le graphique de la page *Grades* en 3D en utilisant du SVG et du JavaScript 11 ;
- Prévoir une boîte surgissante (PopUp) obtenue avec du JavaScript et qui avertit l'utilisateur connecté si un délai à respecter arrive à échéance dans moins de 3h ;
- Montrer la valeur de chaque note représentée dans le graphique en histogrammes lorsque la souris défile au-dessus (implémentation en JavaScript) ;
- Utiliser l'expérience gagnée dans la réalisation de l'onglet *ePPT* afin de rendre les autres vues plus conviviales ; par exemple :
  - le visiteur peut choisir d'écouter la liste des tâches qu'il doit remplir ;

- ◇ le visiteur peut avoir un bref aperçu animé du contenu d'un cours qui l'intéresse (à la place d'une description texte comme dans la situation actuelle).

- Ajouter dans la page *Catalogue* un bouton supplémentaire qui apparaît en-dessous du cadre qui affiche le contenu du fichier Latex ; en appuyant sur ce bouton, l'utilisateur peut prévisualiser, sous forme pdf par exemple, le résultat de la compilation du fichier Latex.

6.3 Remerciements

Maintenant que nous approchons la fin de ce projet, nous pouvons dire que nous avons eu beaucoup de plaisir à travailler dessus. Même lorsque nous avons rencontré des difficultés, nous les avons regardées comme un défi et nous avons pu les surmonter.

Tout d'abord, nous voulons remercier Madame la Professeur Christine Vanoirbeek pour son excellent cours "Ingénierie du Document pour le Web". Ce projet nous a permis la découverte et l'appréhension de plusieurs langages orientés document ainsi que la joie du travail en équipe. De plus, l'application d'un projet de programmation document pour la résolution professionnelle d'un "vrai" problème métier nous a rendues plus confiantes en nos capacités et nous a apporté une précieuse expérience.

Plus particulièrement, nous sommes reconnaissantes vis-à-vis des Messieurs Luca Carnevale, Fouad Slimane et Gil Luyet dont la compétence et la disponibilité nous ont beaucoup aidées tout au long de notre travail. Leurs réponses précises et rapides à toutes nos questions ont contribué de manière significative à faciliter notre apprentissage et ont rendu ce projet plus compréhensible et agréable.

Références

[1] Christine Vanoirbeek. *Ingénierie du document pour le web.* EPFL - IC - MEDIA.

[2] Jean-Philippe Forestier. *Technologies XML.* OSYX - Bureaux E.A.M - Version 1.6, 2002.

[3] Jean-Philippe Forestier. *XML & XSL-T.* OSYX - Bureaux E.A.M - Version 2.1, 2007.

[4] `http://www.w3schools.com` - La bible XML.

[5] `http://fr.wikipedia.org` - Souvent utilisé pour les définitions dans la section Technologies 3 et pour des explications simples.

[6] `https://www.ibm.com/developerworks/xml`

[7] `http://stackoverflow.com`

[8] `http://fr.html.net/tutorials`

[9] `http://wam.inrialpes.fr/timesheets` - Site principal pour l'utilisation de *SMIL* et de *Timesheet*

[10] `http://www.saxproject.org` - Tutoriel *SAX*

[11] `http://debeissat.nicolas.free.fr/svg3d.php`