UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Suivi de l'étudiant : analyse et implémentation dans Moodle

Mémoire présenté par
William BONESIRE

en vue de l'obtention du grade de Master
Computer Science and Engineering

Promoteur(s)
Yves DEVILLE

Lecteur(s)
Charles PECHEUR, Isabelle MOTTE , Frédéric FERVAILLE

Année académique 2016-2017

Remerciements

Ce mémoire est l'accomplissement de 10 mois de travail mais surtout de plusieurs années d'apprentissage. Il n'aurait cependant pas pu voir le jour sans l'aide de plusieurs personnes que nous souhaitons remercier.

Notre premier remerciement ira à notre promoteur Monsieur Yves Deville pour nous avoir permit de démarrer ce travail sur des bases solides mais également pour ses précieux conseils et avis pertinents lors de toutes les étapes de ce dernier.

Nous tenons également à remercier Madame Isabelle Motte et Monsieur Frédéric Fervaille pour leur aide précieuse à la découverte de la plateforme Moodle, leur disponibilité, leur implication dans le projet et leurs remarques qui nous ont permit d'avancer de manière efficace et structurée.

Merci à Monsieur Charles Pecheur pour avoir accepter de prendre part à ce mémoire et d'avoir consacré du temps à sa relecture.

Le dernier remerciement ira à Maman pour son soutien et le temps passé à relire cet écrit pour le rendre le plus agréable à lire que possible.

William Bonesire

Résumé

De nos jours, Internet a pris une place importante dans la vie de pas mal de personnes. Parmi toutes ces personnes se trouve un bon nombre d'étudiants et de professeurs qui ont, eux aussi, compris que cette technologie pouvait leur être utile. L'Internet a donc vu l'apparition de systèmes de gestion d'apprentissage pour aider les institutions à procurer des espaces de gestion de cours en ligne. Dans le cas de l'Université Catholique de Louvain, c'est la plateforme Moodle qui est utilisée depuis plusieurs années.

Cette dernière a été choisie pour son approche Open Source et modulable. C'est dans ce contexte que ce mémoire est né. Le but est de développer une extension qui pourrait avoir une place utile comme outil pour les utilisateurs du MoodleUCL mais également pour tous les utilisateurs de Moodle à travers le monde. C'est grâce à sa communauté grandissante et engagée que la plateforme connaît un tel succès.

En nous basant sur des travaux déjà réalisés, nous avons eu le souhait de créer un rapport permettant d'utiliser toutes les informations que collecte la plateforme sur ses utilisateurs et ce, pour être capable de faire du suivi de l'étudiant. Faire du suivi de l'étudiant est une notion assez vague et sera donc pleinement développée au cours de cet écrit.

L'extension finale de ce mémoire permettra aux étudiants d'obtenir des statistiques concernant leur utilisation de Moodle comparées à celle de leurs congénères. Des indicateurs ont été mis en place pour leur indiquer leur état actuel d'implication. Les professeurs, eux, auront un aperçu de l'avancement de leurs classes et pourront également obtenir plus d'informations grâce à différents outils mis en place.

L'outil réalisé est fonctionnel mais connaît des défauts de performances. Ces derniers proviennent de la quantité d'informations stockées dans la base de données, devoir traiter toutes ces informations prend un certain temps. Une partie de cet écrit présentera les différents problèmes mais également les pistes d'améliorations proposées pour produire un outil encore plus efficace.

Table des matières

Chapitre 1

Introduction

L'utilisation d'Internet n'a pas cessé d'augmenter depuis qu'il est accessible à tout le monde. Ce dernier permet une connexion inégalable entre les personnes. Il est évident que le domaine de l'éducation n'a pas été laissé à part dans cette révolution. L'accès à de nombreuses ressources mais également la communication plus facile et l'échange d'informations sont désormais des piliers du système éducatif. Alors qu'autrefois, le seul moyen de s'instruire était de se rendre dans les salles de cours ou encore de consulter des ouvrages dans les bibliothèques, il est désormais possible de trouver tout son bonheur sans bouger de chez soi. Dans cet esprit, il est évident que la relation étudiants/professeurs a également évolué avec le temps. Les professeurs disposent maintenant de documents en ligne, de liens et ressources pour l'apprentissage, plus besoin d'aller jusqu'au bureau pour discuter, il suffit d'envoyer un mail, les travaux sont rendus en ligne, ... Tous ces éléments simplifient grandement la vie de chacun et permettent de réaliser pas mal d'économies.

Depuis plusieurs années maintenant, l'Université catholique de Louvain a décidé d'utiliser la plateforme Moodle comme système de gestion de cours en ligne. Cette dernière permet aux professeurs de créer des espaces en ligne pour y déposer toutes les ressources dont ont besoin les étudiants dans le cadre de leurs cours. Cet outil est totalement Open Source et entièrement modulable. C'est dans cet esprit que ce mémoire est né, dans le but de créer une extension qui permettra à l'université d'enrichir les ressources dont dispose sa plateforme pour fournir plus d'outils à ses utilisateurs et leur proposer un apprentissage personnalisé.

Pour pouvoir commencer le développement, il nous fallait une idée de module à créer qui serait à la fois utile, pertinente et pas encore présente dans l'ensemble des plugins disponibles sur le dépôt de Moodle. C'est un travail réalisé par l'Université de Laval qui nous a donné l'idée. Ces derniers ont réalisé un outil leur permettant de suivre l'évolution des étudiants durant l'année. L'idée de notre extension est donc de créer un rapport Moodle permettant d'effectuer du suivi d'étudiants à l'aide des informations collectées par la plateforme.

Avant de pouvoir réaliser l'extension voulue, il nous a fallu définir ce qu'on entendait réellement par : *faire du suivi de l'étudiant*. L'intérêt premier de ce dernier est de pouvoir proposer un enseignement de qualité à tout le monde, peu importe d'où l'étudiant provient. Dans notre cas, nous nous basons sur une plateforme en ligne que les étudiants sont libres d'utiliser. Grâce aux données récoltées par cette dernière, nous allons être capable de donner des feedbacks à ses utilisateurs et éventuellement d'améliorer la façon de l'utiliser et donc d'enseigner.

L'implémentation au sein de Moodle nous a demandé de d'abord nous plonger dans sa structure et sa syntaxe. Un des gros avantages de la plateforme est qu'elle dispose d'une communauté très active et d'une documentation très bien détaillée. Après avoir prit connaissance des règles à respecter et des différentes API disponibles, nous avons pu nous lancer dans l'implémentation de

l'extension souhaitée avec des bases solides.

Nous verrons également que l'utilisation d'une telle extension dans un système Moodle tel que celui utilisé par l'UCL peut poser problèmes en termes de performances. La quantité d'informations stockées dans la base de données est considérable et être à même de les traiter pour générer des indicateurs pertinents peut poser problème quant au temps nécessaire pour effectuer les calculs. Mais nous proposerons des solutions permettant aux administrateurs de telle structure d'améliorer l'efficacité de la recherche et du calcul.

Les objectifs derrière ce mémoire sont multiples :

1. Une réflexion pour arriver à une idée innovatrice et utile d'extension à la plateforme Moodle.
2. La création d'une extension fonctionnelle et cohérente avec la structure Moodle.
3. L'analyse des améliorations possibles et le retour sur la qualité de l'extension développée.

Ce travail se compose de 4 parties. La première aura pour but de présenter toutes les ressources dont nous disposons pour réaliser l'extension voulue. Nous allons parcourir les différents articles discutant du sujet pour ensuite nous intéresser plus en détail à la plateforme Moodle. La deuxième partie sera consacrée à la définition exacte du but du plugin. Nous allons passer en revue ce qu'on entend par le suivi de l'étudiant pour ensuite discuter de comment nous comptons le réaliser et quelles fonctionnalités seront intégrées à l'extension. La troisième partie sera consacrée aux aspects techniques du développement, c'est dans cette partie que nous allons présenter le travail effectué pour arriver à un produit fonctionnel. Pour terminer, nous allons discuter des performances accomplies par l'extension et proposer des perspectives d'améliorations. Nous présenterons ensuite des fonctionnalités qui pourraient être ajoutées au rapport pour le rendre encore plus agréable à utiliser.

Chapitre 2

État de l'art

La première étape de ce mémoire a été de définir quelle utilité nous souhaitions donner à notre extension. Différents articles nous ont rapidement mis sur la voie que nous voulions suivre. Nous allons présenter ces derniers et ensuite discuter de la plateforme Moodle et tous les outils qu'elle a à proposer.

2.1 Analyse de quelques outils de suivi existants

L'idée de notre plugin a été inspiré par des sources extérieures ayant déjà eu pour idée d'effectuer du suivi d'étudiants. Nous allons présenter deux articles qui nous ont inspiré dans notre création. Le premier est un travail réalisé par l'Université de Laval. Cet article est un peu la principale source d'inspiration de notre travail car ce sont eux qui nous ont donné l'idée d'élaborer un tel rapport. Le deuxième article est un ancien plugin Moodle qui n'est plus mis à jour à l'heure actuelle (et donc obsolète) qui avait pour but de faire une analyse de l'implication des étudiants à une plus petite échelle que ce que nous envisageons.

2.1.1 Le tableau de bord de la réussite de l'Université de Laval

L'Université de Laval [1] a mis au point un outil permettant aux étudiants, aux professeurs et aux directeurs de programmes d'avoir des statistiques sur les étudiants concernant leur avancée au sein des cours. Ce travail nous a donné l'idée de créer notre propre plugin de suivi des étudiants adapté à la plateforme Moodle.

Fonctionnalités

L'outil développé par Laval propose des fonctionnalités différentes selon les profils des différentes personnes connectées :

– **Description des fonctionnalités offertes aux étudiants** : Les outils fournis aux étudiants leur permettent d'avoir une vision de leur implication et de leur progression dans leur cours. Elle leur fournit également des ressources pour leur permettre de s'améliorer.

  En se connectant sur la plateforme, ils obtiennent leur liste de cours, chacun ayant un indicateur de couleur leur donnant leur situation globale au sein du cours (vert = en voie de réussite, jaune = à surveiller, rouge = à risque) (voir figure 2.1). Cette situation est obtenue en comparant les résultats de l'étudiant avec ceux du groupe. Une situation détaillée peut être obtenue en cliquant sur un cours particulier.

FIGURE 2.1 – Page d'accueil de la plateforme.

La vue détaillée offre cinq aspects différents (voir figure 2.2) :

1. Une liste de critères ayant chacun un indicateur de couleur calculé par rapport au groupe.
2. La moyenne qu'ils ont obtenu au sein du cours ainsi qu'une moyenne de session projetée de la fin du quadrimestre.
3. Leurs résultats cumulés pour le cours courant ainsi qu'une cote projetée.
4. Un graphique présentant l'évolution de leurs indicateurs au cours du quadrimestre.
5. Une boite permettant l'échange de messages avec les professeurs et directeurs de programmes.

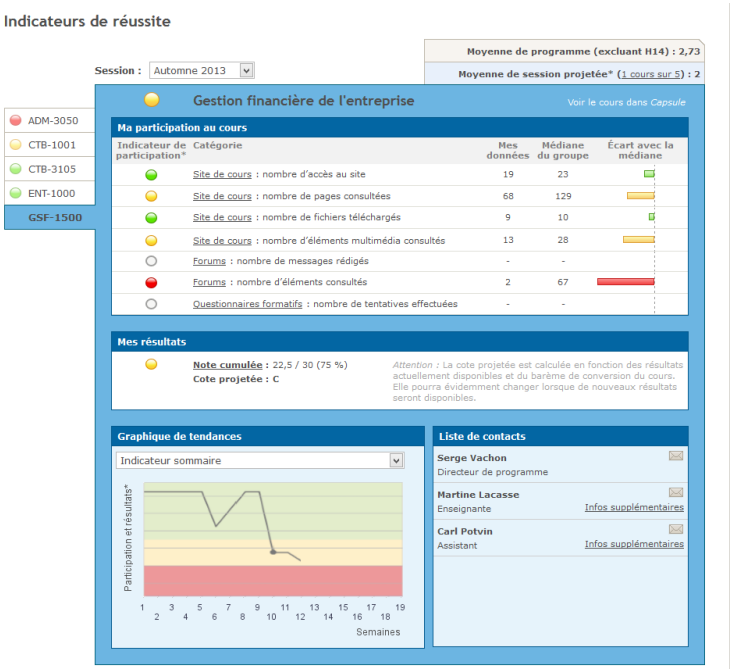


FIGURE 2.2 – Vue détaillée d'un cours.

L'outil propose également aux étudiants des mini-tests et des ressources d'aide (voir figure 2.3). Les mini-tests sont présents pour aider les étudiants à déceler leurs lacunes, des pistes d'améliorations sont alors proposées en fonction des résultats.

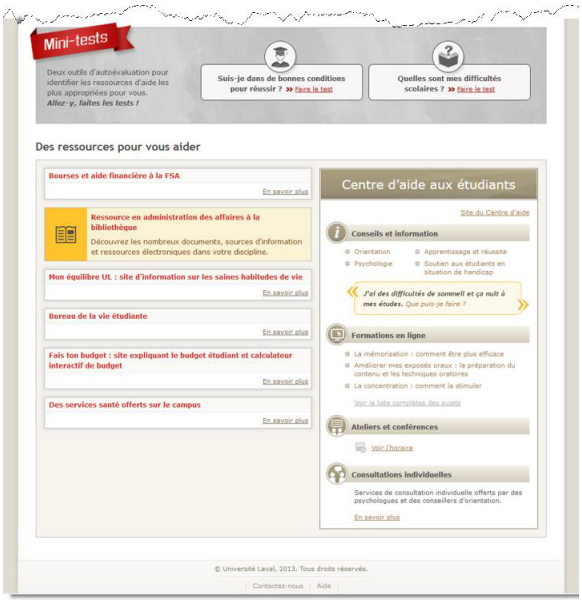


FIGURE 2.3 – Page présentant des mini-test et les ressources d'aides.

– **Guide de l'enseignant** : Les enseignants ont pour chacun de leur cours un tableau reprenant tous les étudiants inscrits (voir figure 2.4). Les étudiants ayant un indicateur de couleur rouge apparaissent au sommet de la liste pour permettre aux professeurs de repérer plus rapidement les cas problématiques. Les différentes colonnes du tableau peuvent être triées par ordre croissant ou décroissant. Il est également possible de filtrer les étudiants selon certains critères.

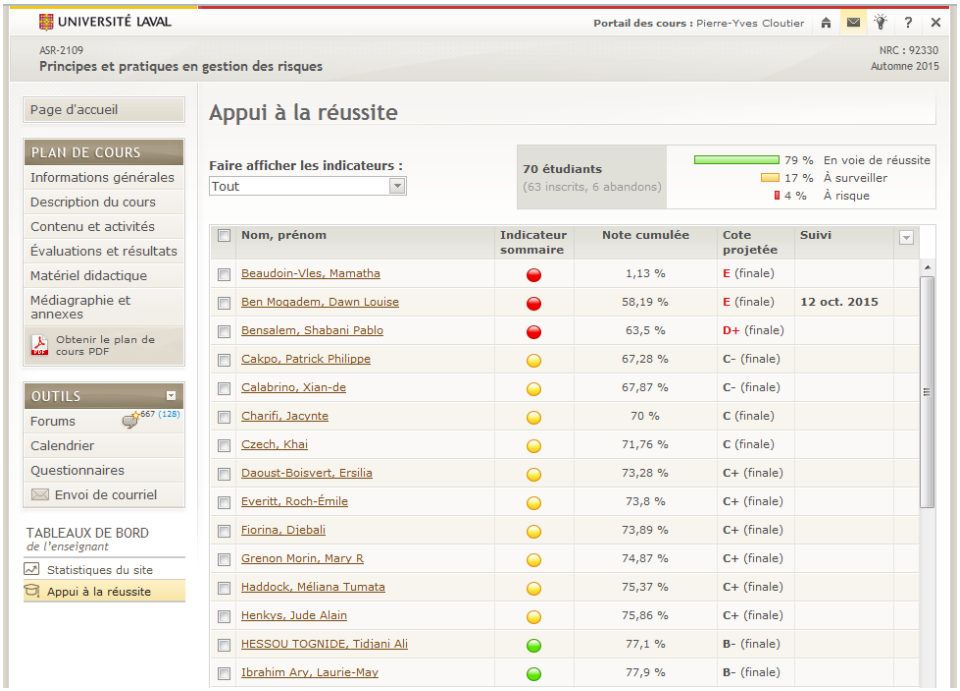


FIGURE 2.4 – Page présentant la liste d'étudiants d'un cours.

Le tableau possède six colonnes :

1. Le nom et prénom de l'étudiant.
2. L'indicateur sommaire de l'étudiant.
3. La note cumulée.
4. La cote projetée de fin de quadrimestre qui n'apparait seulement quand 30% des résultats du cours ont déjà été publiés.
5. La date du dernier suivi effectué pour l'étudiant (échange de mail ou commentaire). Si un nouveau suivi n'est pas encore consulté, la date apparait en gras.
6. Une dernière colonne permet d'exporter les données et modifier la forme des indicateurs.

Il est également possible de cliquer sur le nom d'un étudiant pour obtenir son rapport personnel (voir figure 2.5). On retrouve dans ce dernier les mêmes éléments que présentés ci-dessus bien que l'aspect graphique soit différent.

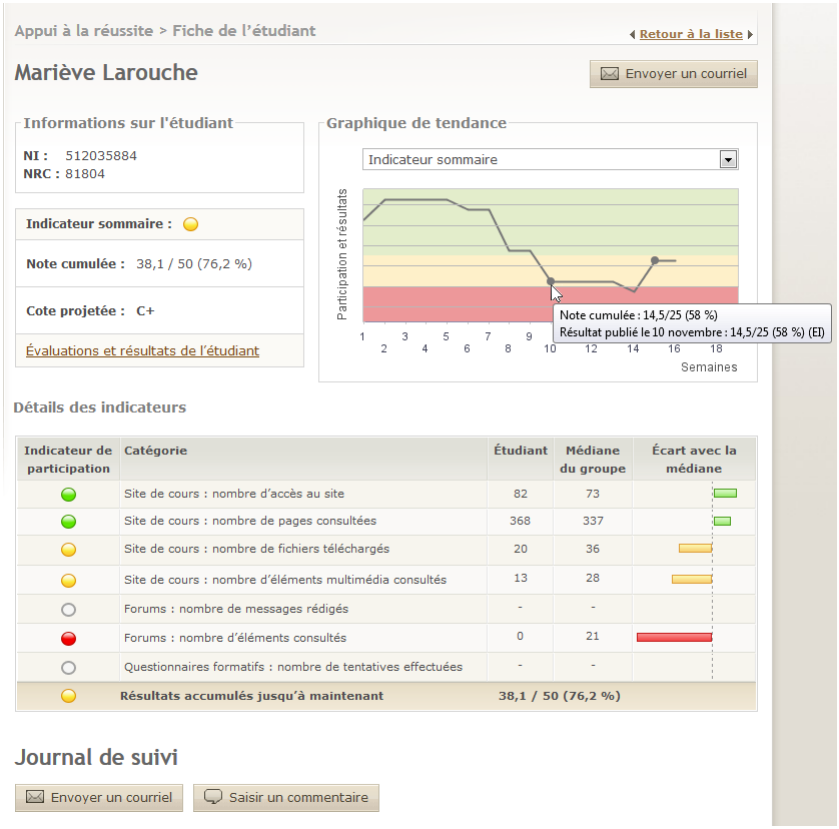


FIGURE 2.5 – Page présentant le rapport détaillé d'un étudiant.

Les professeurs, en plus de pouvoir communiquer avec un étudiant via le journal de suivi, ont également la possibilité d'envoyer des messages à plusieurs étudiants en même temps.

– **Guide du directeur de programme** : Le tableau du directeur de programme est sensiblement similaire à celui du professeur (voir figure 2.6). Les différences sont :

1. Présence de l'historique académique de l'étudiant (nombre de crédits acquis).
2. Présence des indicateurs de couleurs de tous les cours suivis par l'étudiant.
3. Possibilité d'ajouter un étudiant à une liste de surveillance. Cette liste a été mise au point car un directeur de programme a généralement un grand nombre d'étudiants à

surveiller. Les mettre dans la liste lui permet de surveiller les cas à risque de manière particulière.

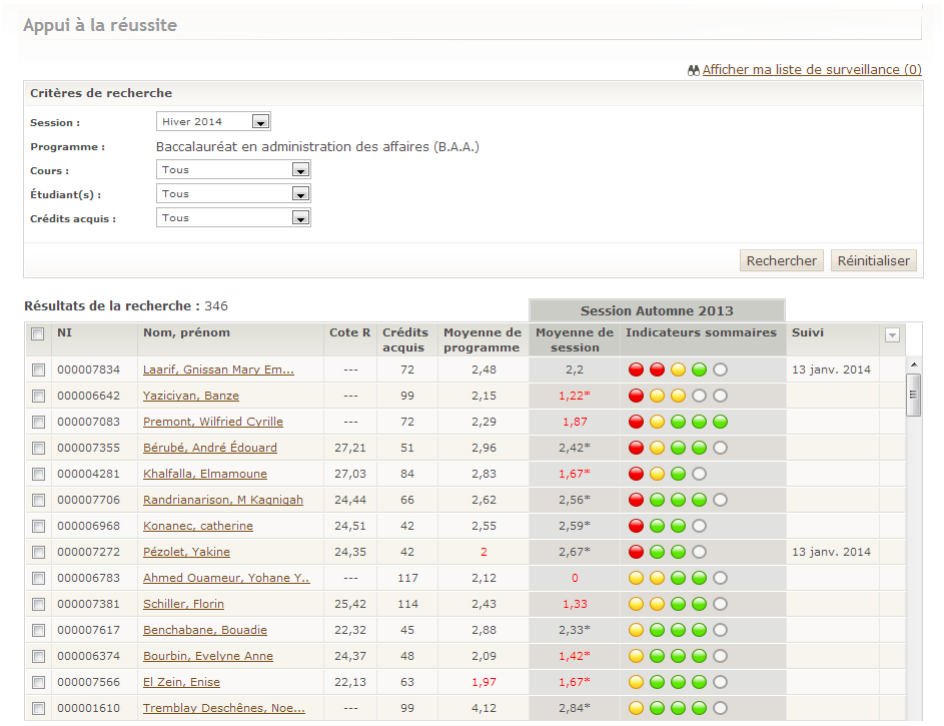


FIGURE 2.6 – Page présentant le tableau des directeurs de programme.

Indicateurs

L'outil développé par Laval se repose sur différents indicateurs. Le premier est la participation au cours. Cette dernière est calculée via différents critères (si un critère n'est pas utilisé au sein d'un cours, il n'est pas pris en compte). On calcule la participation d'un étudiant par rapport à la médiane du groupe pour attribuer une pastille de couleur au travail de l'étudiant. Puisqu'un étudiant se trouvant juste en-dessous de la médiane n'est pas forcément en difficulté, les seuils d'indicateurs sont définis de la sorte :

Couleur	Signification	Écart de la valeur de l'étudiant par rapport à la médiane du groupe
Vert	Bonne à très bonne participation	Participation de 70 % et plus de la médiane
Jaune	Participation inférieure au groupe	Participation de 30 à 69.9 % de la médiane
Rouge	Participation significativement inférieure au groupe	Participation inférieure à 30 % de la médiane
Blanc	Données insuffisantes	--

FIGURE 2.7 – Seuil définissant l'indicateur de participation.

Le deuxième indicateur est les notes obtenues par les étudiants. Ce dernier est obtenu en calculant la moyenne pondérée des résultats sommatifs du cours. Les seuils sont définis à la figure 2.8.

L'indicateur sommaire correspond à la combinaison des deux présentés ci-dessus. Ce dernier est adapté en fonction de l'avancée dans le quadrimestre. Au début, il n'y a encore aucune note et donc, il est égal à l'indicateur de participation. Au plus on avance dans l'année et au plus l'indicateur des notes prendra de l'importance dans ce calcul pour, au final, représenter 100%

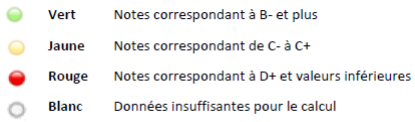


FIGURE 2.8 – Seuil définissant l'indicateur des notes.

de l'indicateur sommaire car ce sont les notes qui permettent le mieux de juger la réussite d'un étudiant.

2.1.2 Le plugin Engagement Analytics de Moodle

Ce plugin [2] avait pour but de donner des informations aux étudiants sur leur progression dans un cours. Il se basait sur les activités ayant un impact dans le succès d'un cours en ligne. Son unique but était de transmettre des informations et non de prendre des décisions. Ces dernières étaient laissées au bon soin des enseignants.

Ce plugin était composé d'un rapport et d'un bloc et n'était accessible que par les professeurs. Le rapport donnait le résumé des indicateurs pour les étudiants d'un cours (voir figure 2.9) et permettait au professeur de changer les seuils et poids accordés à chaque indicateur. Le bloc lui affichait sur la page principale du cours la liste des étudiants avec un indicateur de couleur représentant leur situation générale.

Il utilisait trois indicateurs différents. Ces derniers pouvaient être adaptés avec des seuils et des poids pour améliorer les informations transmises. Les différents indicateurs étaient ensuite combinés pour estimer le risque qu'un étudiant échoue le cours. Le danger était représenté à l'aide d'indicateurs de couleurs.

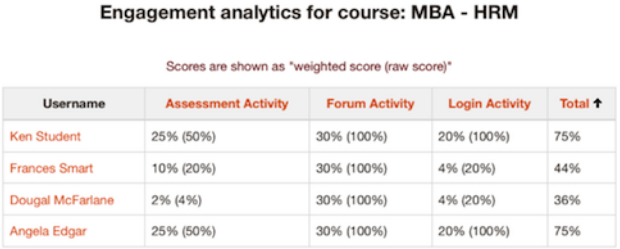


FIGURE 2.9 – Tableau représentant les indicateurs pour les étudiants.

Les trois indicateurs fonctionnait comme suit :

1. **Forum** : Vérifie si l'étudiant lit les messages sur le forum et participe en postant de nouvelles infos :

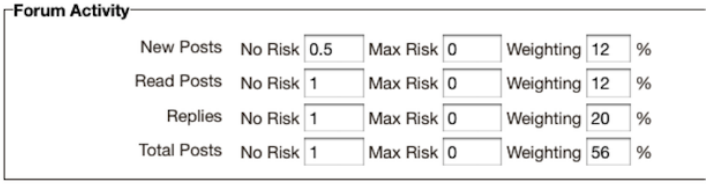


FIGURE 2.10 – Administration de l'indicateur forum.

2. **Connexion** : Vérifie si l'étudiant se connecte sur la plateforme Moodle et s'il passe du temps au sein du cours :

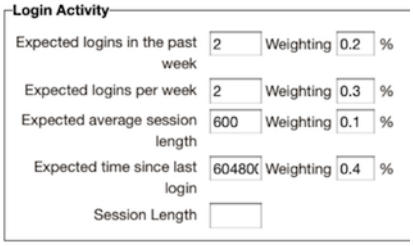


FIGURE 2.11 – Administration de l'indicateur connexion.

3. **Travaux** : Vérifie si l'étudiant remet bien tous les travaux demandés au sein du cours et si aucun de ces derniers n'a été remis en retard :

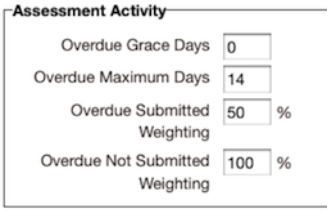


FIGURE 2.12 – Administration de l'indicateur travaux.

Il est également possible au professeur de pondérer chacun de ces indicateurs pour générer l'indicateur de couleur représentant la situation générale de l'étudiant :

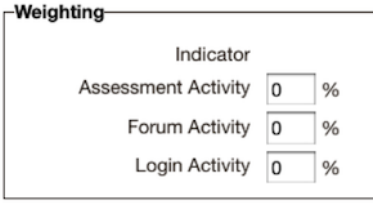


FIGURE 2.13 – Administration de l'indicateur travaux.

2.2 Présentation de notre approche de développement dans Moodle

2.2.1 Présentation de la plateforme

Moodle est une plateforme en ligne qui permet à ses utilisateurs de créer et administrer des cours destinés aux étudiants (LMS : Learning Management System [3]). Cette plateforme a été créée en 2001 en Australie par Martin Dougiamas [4]. Elle est entièrement open source, distribuée sous la licence GPL (General Public License) [5], ce qui assure que tout son contenu peut être modifié et adapté selon les besoins de l'utilisateur. A l'heure actuelle, elle compte plus de 103 millions d'utilisateurs et 81000 sites enregistrés à travers le monde [6]. Le site est présent dans plus de 200 pays et est disponible dans plus de 100 langues différentes. Moodle est l'acronyme

de Modular Object Oriented Dynamic Learning Environment (Environnement d'apprentissage orienté-objet, dynamique et modulaire).

Moodle a pour ambition d'avoir un environnement entièrement modulable [7]. Ce dernier repose sur un ensemble de plugins qui fonctionnent comme des blocs lego que l'on peut assembler comme on le souhaite pour créer la plateforme voulue. La communauté Moodle est très active et de ce fait, de nombreux plugins sont disponibles pour améliorer la plateforme via de nouvelles fonctionnalités. Le dépôt Moodle ne contient pas moins de 1340 plugin [8]. La documentation est en constante amélioration et les forums d'entraide sont très utilisés par les différents membres de la communauté.

La plateforme propose de nombreuses structures différentes, chacune offrant ses avantages et ses fonctionnalités :

- **Les activités [9]** : les activités regroupent l'ensemble des fonctionnalités que l'on peut utiliser au sein d'un cours Moodle. Elle permettent à l'étudiant d'interagir avec le professeur ou avec d'autres étudiants. Elles correspondent aux structures dans lesquels l'étudiant peut directement participer : Assignment, chat, forum, lesson, quizz, wiki, ...
- **Les ressources [10]** : Les ressources sont des éléments créés par le professeur qui permettent un support à l'apprentissage pour l'étudiant. Ce sont des éléments de consultation uniquement : livre, fichier, dossier, page, url, ...
- **Les blocs [11]** : Les blocs sont des éléments qui offrent des vues agglomérées sur des ressources ou des activités. Les administrateurs peuvent installer de nouveaux blocs que les enseignants peuvent choisir d'activer ou non dans leurs cours. Ces derniers peuvent être placés à différents endroits sur la page et peuvent apparaitre à plusieurs niveaux : sur la page d'accueil, dans un cours, ... Il existe une série de blocs standards fournis par la plateforme : calendrier, commentaires, résumé du cours, messages, navigation, ...
- **Les rapports [12]** : Les rapports sont des pages sur le site où l'on peut obtenir des informations particulières sur certains points. A l'origine, il existait des rapports de cours et des rapports d'administration mais ces derniers ont fusionné pour ne créer qu'une catégorie.

2.2.2 Structure choisie pour le plugin : le rapport

Comme on a pu le voir dans la section précédente, Moodle possède un très grand nombre de structures différentes. La première étape, lors de la création de contenu nouveau, est de savoir quelle structure choisir par rapport aux fonctionnalités que l'on souhaite rajouter à la plateforme.

Au départ, nous voulions travailler avec une activité car elle est en lien direct avec le carnet de notes. Mais après réflexion, nous ne souhaitons pas fournir un indicateur chiffré pour caractériser l'implication d'un étudiant au sein d'un cours. Un bloc aurait pu également être envisagé mais ceux-ci offrent un espace de travail assez réduit étant donné que leur but est d'occuper un espace limité d'une page. Or dans notre cas, il y aura une grande quantité d'informations. C'est pourquoi nous souhaitons dédier une page entière à l'affichage des données.

Nous avons choisi de travailler avec la structure rapport. Tout d'abord, cette dernière offre une très grande flexibilité étant donné qu'elle n'est soumise à aucune règle. Un rapport est à la base une simple page blanche où l'on peut afficher une vue choisie sur les données du cours. De plus, les rapports servent à fournir aux utilisateurs des informations particulières sur certains points. Dans notre cas, il s'agit d'informations concernant les étudiants et leur participation/investissement au sein de leurs cours. Parmi toutes les ressources fournies par Moodle, le rapport offre la meilleure structure de travail pour réaliser le plugin que nous souhaitons.

2.2.3 Autres éléments Moodle exploités dans nos développements

Moodle offre un très large panel d'outils et ce dernier ne cesse de croître vu qu'il s'agit d'un logiciel libre avec une communauté dynamique. Nous allons donc présenter ceux utiles à connaitre pour comprendre les actions réalisées lors de ce mémoire.

Carnet de notes

Le carnet de notes [13] est un outil permettant de regrouper toutes les notes obtenues par les étudiants au sein d'un cours. Dès qu'un étudiant réalise une activité qui sera cotée, la note sera automatiquement ajoutée dans le carnet. Au sein de ce dernier, il est possible de consulter les notes (pour les étudiants et les professeurs), de les modifier (pour les professeurs), de les regrouper en catégories, ... En résumé c'est l'endroit où se rendre si l'utilisateur souhaite avoir des informations à propos de ses grades. Grâce à ce dernier, il est possible de récupérer la note totale d'un étudiant au sein d'un cours.

Suivi d'achèvement

L'achèvement des activités [14] permet aux professeurs de définir des activités qui doivent être réalisées pour "réussir" le cours sur la plateforme. Ce dernier permet aux étudiants de voir où ils en sont au sein du cours et de savoir ce qu'il leur reste à faire pour progresser. C'est un outil très utile pour le suivi des étudiants, ce dernier doit cependant être activé manuellement par le professeur dans le menu d'administration du cours. Il est possible pour chaque type d'activités de définir quand l'étudiant a réellement terminé cette dernière.

Il est également possible de coupler à cet outil un plugin additionnel : la barre de progression [15]. Cette dernière est un outil visuel qui donne aux étudiants la possibilité de voir leur avancement dans la liste des tâches à réaliser.

Base de données

La base de données de Moodle [16] est une mine d'informations précieuses au bon fonctionnement de la plateforme. Il y existe pas moins de 200 tables. Il n'y a cependant pas grande nécessité à comprendre toutes ces tables pour pouvoir utiliser la plateforme et faire du développement dessus. Il suffit simplement de se concentrer sur les tables qui sont intéressantes pour notre travail. Par exemple, si on veut travailler sur le forum, il est intéressant de se concentrer sur les 9 tables *forum_xxx* qui sont utiles à son bon fonctionnement. Par contre, si le travail nécessaire ne demande pas d'utiliser ces dernières, ses 9 tables n'ont aucune utilité et donc pas besoin de passer du temps à les regarder.

Dans le cadre de notre travail, il nous a fallu nous intéresser à très peu de tables. La principale qui est le pilier de tout notre rapport est la table *mdl_logstore_standard_log*. C'est dans celle-ci que toutes les informations concernant les utilisateurs connectés vont se retrouver. Chaque action entreprise par les personnes sur le site vont être enregistrées dans cette dernière. C'est évidemment la plus grosse table de la base de données et c'est via cette dernière que nous allons être capable d'effectuer du suivi d'étudiant à l'aide des données qu'elle contient.

Nous avons également dû nous intéresser aux tables concernant les utilisateurs et les cours mais cela ne nous a été utile que pour récupérer certaines données particulières.

Chapitre 3

Le suivi de l'étudiant et ses différentes facettes

Ce chapitre a pour but premier de présenter pourquoi nous souhaitons créer un outil capable de faire du suivi de l'étudiant. Ensuite, nous allons discuter des différents indicateurs que nous avons mit en place à l'aide des ressources fournies par Moodle. Nous terminerons par détailler les différentes fonctionnalités que nous souhaitons implémenter dans notre extension pour arriver à un produit final fonctionnel.

3.1 Suivi de l'étudiant

3.1.1 Motivation pédagogique

La motivation principale du "suivi de l'étudiant" est de pouvoir fournir à chaque personne un apprentissage de qualité peu importe son origine et comment il travaille. De nos jours, les écoles et universités doivent affronter un nombre grandissant d'étudiants d'univers et de capacités différents [17][18]. Dans ce contexte, il est très important de mettre en place différents systèmes et techniques permettant d'effectuer un suivi de ces derniers afin de leur proposer des solutions adaptées si des problèmes apparaissent.

Lors de l'apprentissage groupé (avec un système de classes par exemple), si les étudiants ne donnent pas de bons résultats, cela peut provenir soit d'un problème du côté étudiant ou soit du côté professeur. Disposer de méthodes de suivi permet d'identifier d'où viennent les problèmes et d'adapter la situation. Si tous les étudiants n'arrivent pas à réaliser une certaine tâche, une remise en question du professeur peut être nécessaire. Au contraire, si seulement certains n'y arrivent pas, alors le problème vient plus probablement de leur méthode de travail.

Ne pas effectuer un tel suivi serait une perte d'informations importantes pour le système éducatif. Toutes ces dernières permettent d'améliorer la transmission de savoir et le bien être des acteurs de l'apprentissage.

Le suivi peut être effectué de bien des façons. Il est clair que suivre les résultats des étudiants est une manière efficace de déceler des lacunes. Il suffit dès lors de leur donner un feedback sur mesure pour améliorer leur façon de travailler. La fréquentation des cours est un autre élément qui peut mettre la puce à l'oreille sur un problème d'apprentissage. La prochaine section va décrire plus en détail comment effectuer un tel suivi lorsque la plupart de la communication est réalisée via Internet.

3.1.2 Spécificité du suivi en ligne

De par l'avancée technologique que nous pouvons connaître actuellement, le monde virtuel est de plus en plus utilisé pour véhiculer de l'information. Ceci est dû à son côté pratique et fonctionnel qui offre de nombreux avantages par rapport au format papier anciennement bien plus utilisé.

L'aspect "en ligne" permet aux étudiants d'avoir accès à de nombreuses informations mais également de partager leurs problèmes et solutions de manière rapide et ce, à un grand nombres d'autres personnes. Il n'est pas étonnant que les professeurs se tournent désormais vers des plateformes sur Internet pour transmettre les documents utiles aux cours, faire part des dernières informations, organiser des activités ...

Étant donné que les étudiants passent désormais plus de temps sur Internet que dans les salles de classe, il semble opportun de s'intéresser aux informations récoltées par les plateformes afin de pouvoir obtenir différentes statistiques sur ces derniers. C'est à l'aide de ces données que nous allons mettre au point un outil de suivi de l'étudiant. Le principe est très simple : un LMS (learning management system) comme Moodle enregistre toutes les informations sur les actions des utilisateurs connectés. Pour aller encore plus loin, chaque "clic" réalisé par un utilisateur va être enregistré dans une table de la base de données (table des *logs*). Toutes ces données vont pouvoir être combinées pour former différentes statistiques sur chaque étudiant.

De par cette méthode, nous allons tenter de mettre en évidence les différentes lacunes que les étudiants peuvent avoir : un étudiant qui ne se connecte jamais, un étudiant qui ne télécharge pas les documents utiles, un étudiant qui ne participe pas aux activités proposées, ... Mais également, pouvoir analyser les résultats obtenus par les étudiants qui, de nos jours, sont enregistrés en ligne et sont un grand point de repère de l'avancement et de l'implication d'un élève dans un cours.

3.1.3 Limite de l'approche

La méthode présentée ci-dessus permet d'obtenir des statistiques en fonction de l'utilisation que les étudiants font de la plateforme. Il faut cependant rester vigilant quant à l'interprétation que l'on fait des ces statistiques. En effet, chaque personne est différente et utilise les outils qu'on lui propose de manière personnelle. De ce fait, il ne faut pas tirer des conclusions trop hâtives. Certains élèves vont se connecter quotidiennement tandis que d'autres n'iront que périodiquement sur la plateforme. Certains vont être actifs sur le forum tandis que d'autres vont plutôt discuter de vive voix avec leurs collègues ...

Cette étape d'interprétation est donc une limite du suivi. Aucun modèle ne peut prédire parfaitement où se situe l'utilisateur dans son apprentissage sans savoir à quel type d'étudiant on a affaire. Le but ici est de proposer un outil qui va transmettre certaines informations basées sur un groupe d'étudiants. Libre à chacun de les utiliser et de les interpréter de manière cohérente.

3.2 Indicateurs

3.2.1 Analyse des traces disponibles dans Moodle

Dans le cadre de ce mémoire, la plateforme de communication avec les étudiants est Moodle. La plupart des indicateurs vont être basés sur les informations qui sont collectés lors des visites de ces derniers sur le site. Ces dernières sont appelées des *logs*. Une table entière dans la base de données Moodle est consacrée à ces données (*mdl_logstore_standard_log*).

Ce système de logs a été mis en place lors de la mise à jour 2.6 de Moodle. Il a pour but de donner la possibilité aux utilisateurs de résoudre les problèmes, d'améliorer la plateforme mais également **de donner aux étudiants un meilleur feedback de leur apprentissage** [19]. Tout cela basé sur les informations que laissent les utilisateurs en utilisant la plateforme. Cette table de la base de données a donc été mise en place pour effectuer un tel suivi mais à l'heure actuelle n'est presque pas utilisée. Le but de notre plugin est donc d'utiliser une partie des informations proposées par cette dernière.

La table des logs contient à l'heure actuelle 50 actions différentes. L'annexe A présente ce tableau analysé en fonction des données utiles dans le cadre de notre travail. Nous avons gardé 12 actions permettant d'établir 4 indicateurs différents pour attester de l'implication d'un étudiant au sein d'un cours. Nous avons également utilisé le carnet de notes et le suivi d'achèvement, pour établir deux indicateurs supplémentaires.

3.2.2 Remarque concernant le choix des indicateurs

Les indicateurs proposés dans ce travail ont été produits dans un temps limité afin de pouvoir mettre en place une structure de rapport fonctionnel et délivrant des résultats cohérents. Nous avons conscience qu'ils sont loin d'être les plus pertinents. Ce travail a été réalisé dans le cadre d'un mémoire ingénieur civil en informatique, la réflexion concernant les indicateurs devrait se poursuivre (et il nous semble que ce serait une démarche intéressante) dans un domaine entouré de pédagogues ou d'analystes de l'apprentissage.

Les valeurs que nous avons utilisées au sein des calculs ainsi que les données avancées pour générer les indicateurs sont issus de nos choix personnels afin de produire des indicateurs porteurs de sens. Un prochain travail pourrait porter sur l'amélioration des ces derniers afin de donner encore plus de pertinence au rapport mis au point. Le travail actuellement réalisé a été développé dans le but de pouvoir effectuer des modifications simples et rapides des indicateurs utilisés. La méthode à suivre pour modifier ces derniers sera présentée dans le chapitre 4.

3.2.3 Indicateurs choisis

Les 6 indicateurs que nous utilisons sont répartis en 3 différentes catégories (2 indicateurs par catégorie) :

Notes

Dans la catégorie des notes, les deux indicateurs sont : les résultats obtenus par l'étudiant au sein du cours et le nombre d'activités cotées auxquelles il a participé.

- **Résultats** : Les résultats obtenus par un étudiant sont un indicateur extrêmement important puisqu'il donne un feedback direct et chiffré sur le travail qu'il a effectué. C'est une cotation avec laquelle tout le monde est habitué à travailler et donc facilement compréhensible. C'est le seul indicateur numérique que nous utilisons et également le seul qui provient d'une ressource Moodle (le carnet de notes).
- **Nombre d'activités cotées** : Le nombre d'activités gradées auxquelles participe un étudiant est un bon indicateur de son implication dans un cours. Si ce dernier ne participe à aucune activité, le professeur ne peut pas lui donner un feedback sur son travail et donc l'étudiant ne peut pas savoir s'il a bien compris la matière. D'un autre côté, y participer permet à l'étudiant de connaître son niveau de maitrise de la matière vue.

Participation

La participation est une catégorie plus abstraite à comprendre que les notes. Il s'agit des actions que l'on peut effectuer sur le site qui font que l'étudiant est considéré comme actif sur ce dernier. Dans notre cas, on distingue deux types d'actions possibles : les actions dites de consultation et les actions dites de contribution. Nous allons utiliser le nombre d'actions effectuées par un étudiant pour chacune de ces deux catégories afin de former nos deux indicateurs.

- **Actions de consultation** : Les actions de consultation regroupent toutes les visites passives qu'un étudiant peut effectuer sur le site. C'est-à-dire qu'il ne va pas laisser de traces de son passage. Dans la table des logs, cela correspond à quatre actions différentes : "viewed", "printed", "launched" et "downloaded". On peut voir qu'il s'agit plus d'évènements qui permettent à l'étudiant de récupérer des ressources concernant le cours actuel.
- **Actions de contribution** : Les actions de contribution regroupent toutes les visites actives sur le site. C'est-à-dire que l'étudiant va laisser une trace sur le site. Il aura participé à la vie du cours en ligne. Elle regroupe 6 actions différentes de la table des logs : "added", "assessed", "created", "evaluated", "submitted" et "uploaded". Ces actions correspondent, par exemple, à une remise de devoir, une participation sur le forum, une réalisation d'un quizz, ...

Progression

La progression est l'avancement d'un étudiant au sein d'un cours. Dans le cadre de la plateforme Moodle, on parle de suivi d'achèvement. Pour pouvoir utiliser cette catégorie, il faut que le professeur ait défini les étapes à accomplir pour "terminer" le cours sur la plateforme (comme présenté au chapitre 1). Notre premier indicateur sera donc lié à l'achèvement des activités définies par le professeur, auxquelles on rajoute une notion de ponctualité.

- **Activités complétées** : Il s'agit du nombre d'activités que l'étudiant a déjà accomplies parmi celles définies par le professeur au début de l'année. Elle représente directement l'avancement de l'étudiant au sein du cours et donc sa progression personnelle. Cette information ne provient pas directement de la table des logs mais de la table des modules complétés (*mdl_course_module_completion*). Il nous suffit de compter le nombre de "completionstate" présents dans cette dernière.
- **Ponctualité** : A l'indicateur présenté ci-dessus, on ajoute un indicateur de ponctualité qui va être basé sur le nombre d'activités non réalisés avec une deadline terminée. Ne pas réaliser une activité dans le temps imparti est également un indicateur de progression mais qui agit de manière négative sur l'apprentissage d'un étudiant. Pour cet indicateur, il faut regarder les "becomeoverdue" dans la table des logs.

3.2.4 Conversion en indicateurs de couleurs

Les indicateurs présentés ci-dessus nous donnent à chaque fois une valeur numérique représentant les données propres à un étudiant. Afin de pouvoir générer un rapport aussi clair que possible, nous avons décidé de représenter la signification des données chiffrées sous forme d'indicateurs de couleurs. Pour ce faire, nous allons à chaque fois comparer les données d'un étudiant avec la médiane du groupe (tous les étudiants suivant le cours). De cette façon, nous allons pouvoir tirer des conclusions sur l'avancement de chaque étudiant. Cette idée nous provient directement de l'idée développée par l'Université de Laval.

La médiane est utilisée pour ne pas être influencé par des valeurs extrêmes. Si on avait fait le choix de travailler avec la moyenne, les résultats pourraient être perturbés par des valeurs qui n'ont pas de sens. Par exemple, si un étudiant a abandonné le cours et ne se connecte plus

du tout sur la plateforme ou bien si, au contraire, un étudiant se connecte tous les jours par habitude. Ces types d'étudiants peuvent entrainer des valeurs très basses ou très hautes dans la masse totale. Le choix de la médiane nous parait donc ici opportun pour obtenir une meilleure interprétation des résultats obtenus.

Pour rappel, la médiane est obtenue en classant toutes les valeurs d'un ensemble en ordre croissant ou décroissant et en prenant la valeur au milieu de l'ensemble.

Une fois que l'on a les valeurs correspondant à un étudiant et la médiane du groupe, nous calculons, pour chaque indicateur, un rapport entre les deux valeurs :

$$rapport_i = \frac{ind_{i\ etu}}{ind_{i\ med}} \tag{3.1}$$

C'est la valeur donnée par ce rapport qui va nous donner la couleur à afficher. Si la médiane d'un indicateur est égale à 0 alors l'indicateur sera déclaré non pertinent et la mention "Non pertinent" prendra la place de l'indicateur de couleur. Pour décider de la couleur, il a fallu définir des seuils à utiliser pour définir les trois groupes de données. Le calcul est fait comme ceci :

$$\begin{cases} rapport_i \geq 0.9 \rightarrow \text{Indicateur vert} \\ 0.5 \leq rapport_i < 0.9 \rightarrow \text{Indicateur jaune} \\ rapport_i < 0.5 \rightarrow \text{Indicateur rouge} \end{cases} \tag{3.2}$$

Ces indicateurs sont à interpréter de la sorte : la couleur verte signifie que l'étudiant est dans le bon et qu'il participe de manière cohérente sur la plateforme, la couleur jaune signifie que l'étudiant pourrait être plus investi sur la plateforme malgré le fait qu'il y est déjà présent et la couleur rouge signifie que l'étudiant ne participe pas assez à la vie du cours en ligne et doit absolument changer sa manière de travailler. Bien sûr, chaque indicateur de couleur doit être analysé en fonction du critère qu'il représente.

Il existe juste une exception à ce qui est présenté ci-dessus, dans le cas du rapport concernant la ponctualité. Vu que, dans ce cas-ci, on calcule le nombre de retards, on inverse le rapport pour être cohérent avec l'indicateur :

$$rapport_i = \frac{ind_{i\ med}}{ind_{i\ etu}} \tag{3.3}$$

Lors du calcul de ce rapport, nous devons utiliser une astuce pour éviter le cas où l'étudiant n'aurait aucun retard à son actif. Si ce cas se présente, nous serions dans le cas d'une division par zéro ce qui n'est pas cohérent au niveau mathématique. Pour ce faire, lorsque l'étudiant n'a pas de retard ou lorsque la médiane est à zéro, on écrase la valeur à 0.8 à chaque fois. Cette valeur a été choisie car si un étudiant a 2 "becomeoverdue" et que la médiane est à zéro, on obtient un rapport de $\frac{0.8}{2} = 0.4$, ce qui donne un indicateur rouge. Par contre, si l'étudiant n'a qu'un seul "becomeoverdue", l'étudiant obtient un rapport de $\frac{0.8}{1} = 0.8$, ce qui donne un indicateur jaune.

Une fois que les rapports des 6 différents indicateurs ont été obtenus, nous pouvons calculer un indicateur global qui donne une idée de la situation générale d'un étudiant. Cet indicateur est obtenu en faisant la moyenne des 6 indicateurs :

$$ind_{global} = \frac{\sum_{i=1}^{6} min(rapport_i; 2)}{6} \tag{3.4}$$

Une petite subtilité a été introduite dans le calcul de cette moyenne. On prend à chaque fois le minimum entre la valeur du rapport et la valeur 2. Ceci est dû au fait que l'on ne veut pas être influencé par un rapport qui serait prédominant aux autres. Si un rapport a une valeur plus élevée que les autres, l'indicateur global ne serait pas représentative de la situation réelle de l'étudiant.

En plus de cet indicateur global, on calcule deux indicateurs supplémentaires qui seront utilisés pour générer le rapport des professeurs. Ces deux indicateurs correspondent à la moyenne entre les deux indicateurs de participation et la moyenne entre les deux indicateurs de progression. On suit la même logique que pour l'indicateur global afin de les calculer :

$$ind_{participation} = \frac{min(rapport_3; 2) + min(rapport_4; 2)}{2} \tag{3.5}$$

$$ind_{progression} = \frac{min(rapport_5; 2) + min(rapport_6; 2)}{2} \tag{3.6}$$

3.3 Fonctionnalités

Nous allons maintenant présenter les différentes fonctionnalités que l'on pourra retrouver au sein du rapport. Deux aspects seront présentés indépendamment : le rapport étudiant et le rapport professeur. Il est évident que le rapport du côté professeur sera composé de bien plus de fonctionnalités que celui du côté étudiant étant donné le nombre d'informations bien plus important qui s'y trouvent.

3.3.1 Côté étudiant

Le côté étudiant du rapport se veut épuré mais efficace (voir figure 3.1). Toutes les informations sont présentées sous forme d'un grand tableau. Les indicateurs sont regroupés par deux en fonction de la catégorie qu'il représente. Chaque ligne du tableau reprend à chaque fois la valeur de l'étudiant, la médiane du groupe et l'indicateur de couleur représentant le rapport entre les deux premières valeurs. Au-dessus du tableau se trouve l'indicateur global de l'étudiant.

Nous avons également placé des liens après chaque catégorie vers des ressources fournies par Moodle pour que l'étudiant puisse avoir plus de renseignements sur ces informations :

- **Notes** : Pour la catégorie des notes, on redirige l'étudiant vers le carnet de notes où sont stockées toutes les informations concernant les travaux gradés qu'il a effectué au sein du cours.
- **Participation** : Pour la participation, on le redirige vers le rapport de participation. Ce dernier présente toutes les activités auxquelles l'utilisateur a participé lors de ces visites.
- **Progression** : Pour la progression, on le redirige vers la barre de progression qui affiche les différentes étapes à accomplir pour "réussir" le cours sur Moodle. Cette dernière doit avoir été activée par le professeur pour être utilisable au sein du cours. Si cela n'a pas été fait, l'étudiant n'aura accès à aucune information.

Comme vous pouvez le voir, le rapport du côté étudiant est un rapport de pure consultation, l'étudiant ne peut effectuer aucune action particulière hormis cliquer sur les liens qui sont mis à sa disposition.

Rapport détaillé d'un étudiant (visible par l'enseignant et l'étudiant)

Indicateur global de réussite : feu de couleur lié à ind_{global}

Notes

	Résultat étudiant	Résultat médian	Indicateur visuel
Note totale	$ind_{1\ etu}$	$ind_{1\ med}$	feu de couleur lié à ind_1
Nombre d'activités évaluées	$ind_{2\ etu}$	$ind_{2\ med}$	feu de couleur lié à ind_2

Détail des notes (lien vers le carnet de notes de l'étudiant)

Participation

	Résultat étudiant	Résultat médian	Indicateur visuel
Nombre d'actions de consultation	$ind_{3\ etu}$	$ind_{3\ med}$	feu de couleur lié à ind_3
Nombre d'actions de contribution	$ind_{4\ etu}$	$ind_{4\ med}$	feu de couleur lié à ind_4

Détail de la participation (lien vers le rapport de participation de l'étudiant)

Progression

	Résultat étudiant	Résultat médian	Indicateur visuel
Nombre d'activités achevées	$ind_{5\ etu}$	$ind_{5\ med}$	feu de couleur lié à ind_5
Nombre d'activités en retard	$ind_{6\ etu}$	$ind_{6\ med}$	feu de couleur lié à ind_6

Détail de la progression (lien vers la barre de progression de l'étudiant si configurée sinon message « La barre de progression n'est pas configurée dans ce cours » pour stimuler son utilisation).

FIGURE 3.1 – Schéma de la structure du rapport étudiant

3.3.2 Côté professeur

Du côté professeur, le rapport a dû être adapté car, en raison du nombre d'étudiants qui peuvent parfois suivre certains cours, il nous fallait générer un tableau où chaque étudiant n'occuperait qu'une seule ligne (voir figure 3.2). C'est dans cet esprit de simplicité que nous avons créé les deux indicateurs pour la participation et la progression comme présenté dans la section précédente.

Le tableau du rapport professeur se présente de la sorte : sur chaque ligne, on retrouve le nom de l'étudiant, sa note totale au sein du cours et puis trois indicateurs de couleurs (indicateur de participation, indicateur de progression et indicateur global). Nous avons choisi de ne pas tenir compte du nombre d'activités gradées et de juste garder la note totale pour la catégorie notes car cette dernière est bien plus représentative du travail de l'étudiant.

Au sein de ce tableau, le professeur a la possibilité de cliquer sur le nom d'un étudiant pour accéder à son rapport complet. Le rapport complet contient les mêmes informations que celles présentées dans la section étudiante hormis le fait que nous avons retiré l'accès aux diverses ressources parce que, seule la personne connectée peut avoir accès à ses propres informations et donc les liens ne seraient d'aucune utilité pour un professeur.

Pour que le professeur ait une idée claire de la situation actuelle de sa "classe", nous avons disposé au dessus du tableau trois graphiques. Ces derniers représentent le pourcentage de chaque

indicateur de couleurs (participation, progression et global) au sein du cours. Cela permet au professeur d'avoir une vision claire de la répartition des étudiants au sein de ces 3 catégories et d'avoir une idée de la situation dans laquelle se trouve le cours.

En plus de tous ces aspects d'affichage des informations, nous avons ajouté une page d'administration accessible par les professeurs. Cette dernière permet de modifier le poids que possède chaque indicateur dans le calcul de l'indicateur global et dans le calcul des indicateurs de participation et de progression. Elle donne accès à un simple formulaire à compléter avec le poids de chaque indicateur.

Rapport global d'un enseignant

	Note globale	Participation	Progression	**Réussite globale**
Nom1 Prénom1	$ind_{1\ etu}$	feu de couleur lié à $ind_{participation}$	feu de couleur lié à $ind_{progression}$	feu de couleur lié à ind_{global}
Nom2 Prénom2	...			
Nom3 Prénom3				

FIGURE 3.2 – Schéma de la structure du rapport professeur

Chapitre 4

Implémentation du suivi de l'étudiant dans Moodle

Maintenant que nous avons posé les bases nécessaires à la bonne compréhension du travail à réaliser, nous allons pouvoir nous pencher sur l'implémentation au sein de Moodle. Ce chapitre va présenter le chemin parcouru pour obtenir une extension fonctionnelle. Nous allons d'abord nous attarder un peu sur les différents outils fournis par la plateforme pour les développeurs et nous allons présenter les différentes API qui nous ont été utiles. Après, nous allons nous plonger dans la structure d'un rapport. La première partie de notre réalisation présentée sera le rapport étudiant pour ensuite parler du côté professeur et ses différentes fonctionnalités. La dernière partie expliquera comment installer le rapport sur la plateforme.

4.1 Développement dans Moodle

4.1.1 Installation de Moodle

Afin de pouvoir développer et tester sans créer de problèmes et en toute liberté, nous avons choisi de travailler sur une version locale de cette dernière. Afin de pouvoir commencer à travailler dessus, il nous a fallu installer certaines technologies.

Nous avons choisi d'utiliser un environnement Ubuntu car ce dernier facilite largement l'installation mais surtout la maintenance de Moodle. Afin d'avoir une base de données fonctionnelle et un environnement de travail compatible, nous avons téléchargé Apache, MySQL et PHP. Une fois toutes ces différentes technologies mises en place, nous avons pu installer Moodle sur notre machine personnelle.

L'installation de Moodle est très intuitive et aisée via les lignes de commande. Il suffit de télécharger la dernière version et de la mettre dans le bon dossier sur la machine (*/var/www/html/*) [20]. Après une simple organisation de la base de données, il suffit de se connecter à l'aide d'un navigateur sur l'adresse locale : *http ://localhost/moodle* et de suivre les différentes pages pour configurer la structure correctement.

Dans un premier temps, nous avons ajouté manuellement des étudiants, professeurs et différents cours dans lesquels nous avons inséré des activités. Tout ceci dans le but d'avoir de la matière sur laquelle travailler. Par la suite, nous avons pu avoir accès au cours test utilisé par les équipes de gestion Moodle de l'UCL.

4.1.2 Outils de développement

La structure Moodle propose énormément d'outils pour faciliter le développement. La première chose à faire quand on commence à coder sur Moodle est d'activer le mode "debugging" [21] (*Administration > Site administration > Development > Debugging*). Ce mode permet d'obtenir des messages clairs sur les erreurs qui peuvent survenir. Si ce mode n'est pas activé, Moodle ne retourne aucun message d'erreur et le programmeur se retrouve généralement devant des pages blanches ou incomplètes.

Nous avons également utilisé un autre outil bien pratique : Moodle Adminer [22]. Il s'agit ici d'un plugin que les administrateurs peuvent installer et qui leur permet de gérer la base de données de manière très simple. Il s'agit d'un substitut à PHPMyAdmin qui est adapté à la plateforme. Une fois installée, la structure est accessible via le lien suivant : *Settings > Site administration > Server* et permet d'avoir une vue complète des éléments présents dans la base de données.

Au cours de notre travail, en plus de devoir avoir accès à la base de données, nous avons dû en modifier le contenu (ajout de tables). Pour ce faire, nous avons utilisé le XMLDB editor [23]. Il s'agit d'un outil fourni par Moodle (*Site administration > Development > XMLDB editor*) qui permet de générer les fichiers qui spécifient comment la base de données doit être structurée. Il permet donc d'ajouter des éléments, d'en modifier certains ou encore de supprimer des données inutiles. Tout ceci est réalisé via la création automatique de fichiers XML qui iront directement se placer dans les bons dossiers.

4.1.3 Présentation des API utilisées

La plateforme Moodle se comporte comme un framework répondant à ses propres règles et ses propres conventions. Cette dernière dispose de nombreuses API [24] chacune offrant des outils particuliers en fonction de ce que l'on souhaite faire. Nous allons présenter les principales API que nous avons utilisées au cours du développement.

Data Manipulation API

L'API consacrée à la manipulation des données [25] est celle dont nous avons eu le plus besoin. En effet, elle permet d'interagir avec toutes les informations contenues dans la base de données et, étant donné que tout le principe de notre plugin repose sur les informations que la plateforme collecte, il semble clair que nous avons dû l'utiliser pour les récupérer.

Cette dernière est présente depuis la version 2.0 de Moodle. Il faut bien entendu rester conscient de ce que l'on fait puisqu'on travaille à un niveau d'abstraction très élevé en manipulant directement les données.

Afin de pouvoir utiliser l'API, il faut créer une instance de la base de données dans le fichier sur lequel on travaille. Toutes les fonctions peuvent être appelées en utilisant cette instance. Il existe plusieurs manières de récupérer les données. On peut soit utiliser des fonctions prédéfinies par Moodle on bien utiliser des requêtes SQL complètes. Afin que les fonctions soient correctement effectuées, il faut mentionner les tables sans leur préfixe (**mdl_**). Les arguments des fonctions peuvent être passés à l'aide de tableaux ce qui permet de modifier aisément les recherches que l'on fait.

La liste complète des fonctions disponibles peut être trouvée sur la page de l'API. Il faut bien faire attention à la manière dont les données sont récupérées pour y accéder correctement.

Access API

Cette API [26] permet de savoir quel est l'utilisateur actuellement connecté et de définir des modules définissant les différentes capacités d'une page.

Moodle est basé sur un modèle d'accès par rôle d'utilisateur. En fonction de votre rôle sur le site, vous avez accès à certains contenus et d'autres non. Cela permet, entre autres, d'afficher des choses différentes sur une même page en fonction de qui la regarde. Pour ce faire, il faut définir des capacités différentes pour déterminer qui peut voir quoi.

Les privilèges sont définies dans le tableau *capabilities* (**/db/access.php**). Ils contiennent différents paramètres tel que les risques associés, le type de capacité (lecture ou écriture), le niveau du contexte où l'on se trouve, le type de rôle qui peut avoir accès à cette capacité, ... Une fois que les privilèges sont définis, il faut obtenir le contexte dans lequel on est entrain de travailler pour pouvoir vérifier que les informations sont correctes.

Il existe encore d'autres fonctions présentes au sein de l'API que l'on peut retrouver sur la page associée. Une dernière qui est obligatoire lorsque l'on développe un plugin est la fonction *require_login* qui permet de vérifier si l'utilisateur est bien connecté avant d'afficher quoi que se soit.

Form API

Tous les formulaires de la plateforme sont générés par l'API Form [27]. L'utiliser donne pour avantage de ne pas se soucier du design, d'avoir une gestion sécurisée des informations transmises, permet d'utiliser des outils Moodle, ...

La première étape pour créer un formulaire est de définir une classe dans votre plugin qui étend la classe *moodleform* et qui redéfinit la fonction *definition* pour y ajouter les éléments du formulaire à créer. Une fois que cette classe est créée et que la forme du formulaire est mise dans la fonction *definition*. Il suffit d'instancier cette classe à l'endroit souhaité pour afficher le formulaire. Plusieurs options sont présentes pour gérer les différents cas qui peuvent survenir lorsqu'un utilisateur l'utilise.

String API

Cette API [27] a pour utilité de généraliser l'affichage du texte présent dans le plugin. Dans le but de permettre l'affichage d'une page dans plusieurs langues différentes, on stocke tout le texte dans un seul fichier et utilise des fonctions et mot-clés pour l'afficher au bon endroit. De ce fait, il suffit simplement de traduire le fichier pour que le texte change de langue partout dans le plugin.

Afin de pouvoir afficher du texte au bon endroit, il suffit d'une fonction et de deux mots-clés : un pour localiser le fichier du texte et l'autre pour localiser l'entrée particulière à afficher.

Chart API

Cette API [28] est toute récente dans l'activité de Moodle. Pour pouvoir l'utiliser, nous avons dû mettre à jour notre plateforme vers la version 3.2. Elle propose aux utilisateurs de générer des graphiques dynamiques de manière extrêmement simples à l'aide de quelques méthodes. Elle propose 7 sortes différentes de graphiques aux développeurs, dans notre cas c'est le graphique en forme de camembert qui nous intéresse (Pie chart).

Pour générer un graphique, il ne nous faut qu'une seule information : la série de données qu'il doit représenter. Pour se faire, il suffit de créer une instance de chart_serie et de lui donner en paramètre le tableau contenant toutes les données à afficher. Il est évidemment possible d'utiliser d'autres méthodes pour rendre le graphique plus simple à comprendre. On peut donner un titre, nommer les différentes classes de données, éditer les axes, ... Il est également possible de superposer différents types de graphiques.

Navigation API

L'API consacrée à la navigation [29] est utilisée pour apporter une structure au site. On peut l'utiliser via la variable générique **$PAGE**. Elle permet de définir le titre, l'en-tête, l'url, ... des différentes pages du site.

4.1.4 Découverte de la syntaxe

Moodle possède son propre environnement de développement [30] et donc ses propres règles et sa propre syntaxe. Pour pouvoir l'exploiter, il faut être à même de comprendre les mots-clés et autres fonctions utilisés au sein des fichiers. Pour ce faire nous allons présenter les différents éléments qui nous ont été utiles au sein de nos fichiers. Nous allons nous appuyer sur le fichier **index.php** (voir annexe B.1) comme exemple car il est le fichier le plus complet.

Tout fichier Moodle doit démarrer avec le même commentaire présentant le projet général de Moodle ainsi que l'auteur du fichier, le nom de ce dernier et la licence GNU GPL [5] qui assure un partage libre des différents éléments créés. Ensuite, on trouve tous les imports qui sont nécessaires au bon fonctionnement de la page courante. Moodle possède un système de lien générique qui permet de trouver directement le chemin vers le dossier général ou bien le dossier des librairies. Les utiliser permet de toujours trouver les fichiers peu importe l'endroit où sont stockés (local, serveur, ...) tous les dossiers utiles au bon fonctionnement de la plateforme.

Ensuite on récupère les paramètres qui nous seront utiles. Dans notre cas, il s'agit de l'id du cours courant qui est récupérée via l'URL ainsi que celui de l'utilisateur courant. Ce dernier est récupéré via une variable générique qui permet de récupérer les données de la personne connectée. Ces données nous permettent de vérifier que l'utilisateur est bien connecté et est autorisé à accéder au cours en question. On peut en dernier obtenir le contexte du cours.

Le contexte [31] est une notion assez particulière dans Moodle. La plateforme est une hiérarchie de contextes qui contiennent chacun leurs propres propriétés. Par exemple, le site a son propre contexte qui contient les contextes des catégories qui contiennent aux mêmes les contextes des cours et ainsi de suite. Ceux-ci servent à définir dans quel espace on travaille, les contextes de niveau bas reçoivent les informations des contextes au dessus d'eux. Par exemple un étudiant membre d'un cours X sera également un étudiant d'un quizz créé dans le cours X. Ces derniers permettent également de définir des rôles propres au contexte. Cela permet entre autre d'avoir un étudiant dans un cours qui peut être manager d'un autre cours. Il faut donc toujours récupérer le contexte dans lequel on travaille pour ne pas commettre d'erreur.

Après tout cela, on définit l'url de la page courante en lui donnant l'id du cours comme paramètre. On lui donne un titre et on récupère le nom du cours. On définit toutes ces informations sur la page en utilisant une variable générique comme présenté dans l'API navigation. De cette manière, toutes les informations sont bien placées (la barre de navigation, l'en-tête, l'url). On définit également le type d'affichage que l'on souhaite avoir. Dans notre cas, on le met sur

"report" afin qu'il aie la même forme que les autres rapports déjà présents sur le site.

Pour terminer, il ne nous reste plus qu'à afficher l'en-tête, à placer le code souhaité pour la page et ensuite afficher le bas de page.

4.1.5 Structure d'un plugin : rapport

Nous avons vu que Moodle possède beaucoup de structures différentes. Chacune ayant ses propres particularités. Pour développer une vue sur les activités des étudiants, la plus appropriée est le rapport. Le code des fichiers présentés ci-dessous peut être trouvé dans l'annexe B.

Pour créer un rapport, il faut ajouter un nouveau sous-dossier dans le dossier *report/* de Moodle. Dans notre cas, il s'appelle followup (suivi). Dans ce dossier, il suffit d'un seul fichier *index.php* pour que le rapport soit déjà fonctionnel. Mais le minimum recommandé est d'avoir le fichier index ainsi qu'un fichier *version.php* et un dossier pour les différentes langues (lang/en/report_followup.php). Il est évidemment possible de créer plusieurs fichiers php et ensuite mettre des liens vers ceux que vous souhaitez utiliser. Nous en avons créé quelques-uns et allons les expliquer par la suite.

Le dossier **lang** a déjà été présenté via l'API correspondant donc on ne va pas s'y attarder ici. Dans le fichier *index.php*, il faut y mettre la structure du rapport : c'est le fichier qui sera généré quand on se rend sur le rapport. Pour le fichier *version.php*, il contient les informations importantes concernant le rapport. Tel que le nom de ce dernier, la version dans laquelle on se trouve et la version nécessaire de Moodle pour que le plugin fonctionne.

Notre rapport contient plusieurs autres fichiers que l'on a ajoutés pour des facilités ou parce qu'on en avait besoin pour faire fonctionner certaines fonctionnalités :

- **lib.php** : Contient toutes les fonctions que l'on utilise pour récupérer ou pour utiliser les données utiles à la génération du rapport.
- **infos.php** : Permet de fournir le rapport détaillé d'un étudiant au professeur.
- **admin.php** : Permet aux professeurs d'administrer le rapport en modifiant les différents coefficients des indicateurs.
- **form.php** : Définit la forme du formulaire qui est utilisé dans la page d'administration.

En plus de ces différents fichiers, nous avons ajouté deux dossiers pour faire fonctionner le rapport :

- **images** : Contient les images des indicateurs de couleurs.
- **db** : Contient les fichiers nécessaires à l'utilisation de la base de données :
  1. **access.php** : Définit les différentes permissions d'accès.
  2. **install.xml** : Permet la création d'une nouvelle table dans la base de données pour stocker les différents coefficients (générer via XMLDB editor).

4.2 Système de view

Comme présenté dans le chapitre 3, notre extension présente deux rapports différents adaptés à la personne qui le génère. La première question à se poser est *comment afficher des informations différentes en fonction des personnes qui consultent la page ?* Pour pouvoir différencier les deux rapports en se connectant sur la même page, nous avons utilisé l'API access. Cette dernière propose un système de priorités qui permettent de définir ce que les gens peuvent voir en fonction

du rôle qu'ils occupent sur la plateforme.

Étant donné que, dans notre cas, il y a deux types de rapports à afficher sur la page, nous avons créé une autorisation particulière pour les professeurs. Cette dernière est présente dans le fichier */db/access.php*. Elle donne accès aux informations à toutes les personnes possédant le statut de 'teacher', 'editingteacher' et 'manager'.

Il suffit simplement ensuite de créer les deux rapports sur la même page (*index.php*) et d'utiliser deux conditions pour tester le statut de la personne connectée. On utilise la condition pour le rapport professeur et sa négation pour le rapport étudiant.

Nous avons fait le choix de n'utiliser qu'une seule autorisation et non pas deux car il est coutume de donner à une personne possédant une autorisation également toutes celles de niveau inférieur. Si ce système est utilisé et que nous définissons deux autorisations alors, un professeur souhaitant générer le rapport de sa classe obtiendrait également son propre rapport, et ce car il possède également le statut d'étudiant. Évidemment, ce genre de situation est à éviter car elle n'est d'aucune utilité.

4.3 Rapport étudiant

La première partie réalisée a été la vue étudiante du rapport de suivi (voir figure 4.1). Nous avons choisi de commencer par là étant donné le nombre de fonctionnalités réduites que propose cette vue du rapport. Il y a deux aspects qui nous permettent de générer le rapport étudiant : les fonctions qui récoltent les données utiles à la génération des indicateurs et le code permettant d'afficher de manière cohérente les informations récoltées.

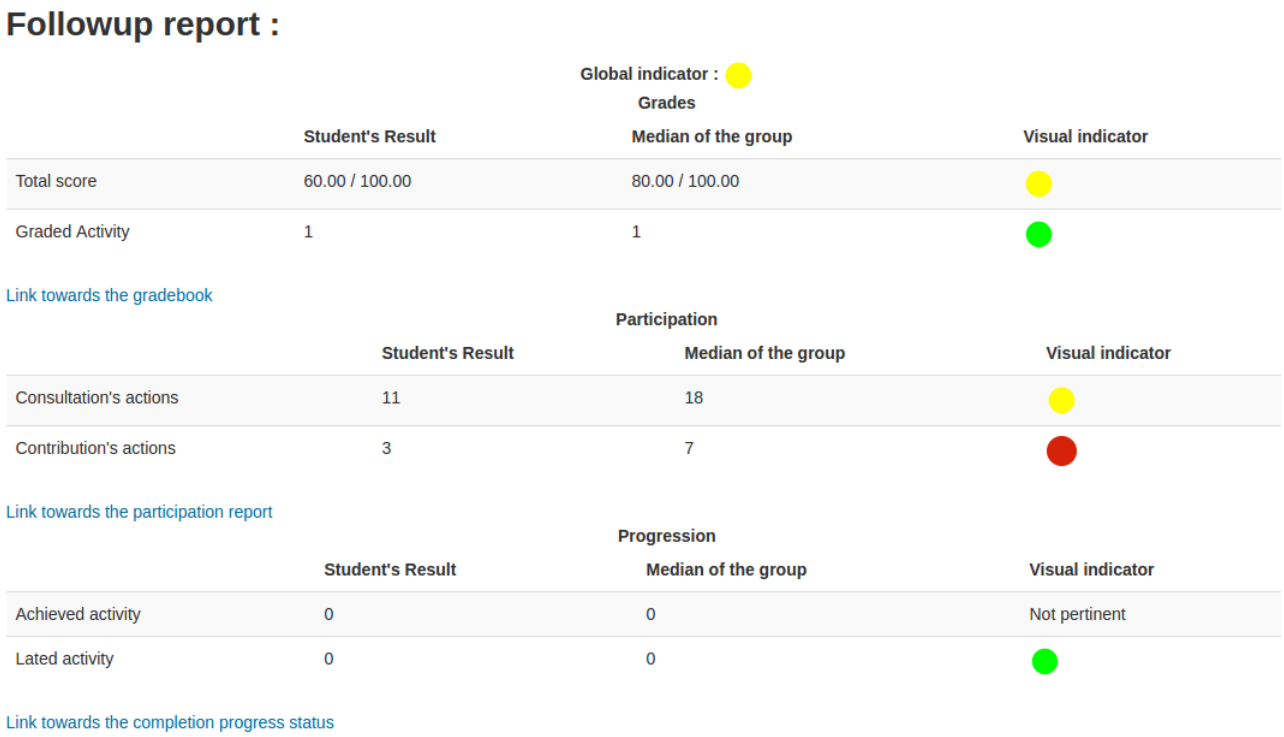


FIGURE 4.1 – Rendu du rapport étudiant.

4.3.1 Fonctions utiles

Toutes les fonctions utilisées sont présentes dans le fichier *lib.php*. Pour produire le rapport étudiant, il nous a fallu implémenter 7 fonctions qui sont réparties comme ceci : 2 pour les indicateurs, 1 pour l'indicateur global, 1 pour les médianes et 3 pour les indicateurs couleurs.

Les premières méthodes créées nous ont permis de récupérer les différentes valeurs pour un étudiant particulier. Le premier indicateur est récupéré à l'aide d'une fonction Moodle permettant d'obtenir le score total d'un étudiant au sein d'un cours (*grade_get_course_gade($user, $id);*). Pour les autres indicateurs, nous avons crée deux fonctions : une pour les indicateurs provenant de la table des logs et une pour l'indicateur d'activités complétées qui est récupéré depuis une autre table. La méthode est la même pour les deux méthodes. On utilise des requêtes SQL pour récupérer l'ensemble des actions que l'étudiant a réalisé, correspondant à l'indicateur. On utilise ensuite la méthode **count()** pour obtenir le nombre total à chaque fois. C'est ce nombre qui est renvoyé par la méthode.

Pour le calcul des médianes, une seule méthode a été implémentée, on utilise une variable pour définir sur quel indicateur on est entrain de travaillé. Le principe est très simple : on récupère tous les étudiants du cours, on calcule ensuite leurs valeurs d'indicateurs personnels que l'on stocke dans un tableau. Une fois toutes les valeurs récoltées, on va simplement ordonner le tableau et retourner la valeur se trouvant au milieu pour obtenir la médiane.

Il est intéressant de signaler que ces méthodes ont été implémenté pour permettre aux futurs utilisateurs de modifier rapidement les indicateurs. Pour le calcul de ceux des étudiants deux cas de figures peuvent apparaitre. Soit on souhaite rajouter un indicateur basé sur les logs, dans ce cas, il faut modifier la fonction *calculate_indicators_log($userid, $course, $indicator)* en lui rajoutant une condition pour obtenir les bonnes actions à chercher dans la base de données. Si on souhaite en ajouter un qui ne provient pas des logs, il faut créer une nouvelle méthode de manière similaire à *calculate_completion($userid, $course)*. Pour le calcul des médianes, peu importe le type d'indicateur ajouté, il suffit de mettre une condition supplémentaire dans la fonction *median_indicators($course, $indicator)* et utilisé la méthode correspondante pour récupérer les valeurs des étudiants. Une fois ces quelques modifications faites, le nouvel indicateur est prêt à être utilisé.

Une fois que l'on est capable de récupérer les indicateurs et les médianes de chaque étudiant, on peut se lancer dans le calcul de l'indicateur global (voir équation 3.4). On va récupérer les 6 rapports correspondants aux 6 indicateurs (voir équation 3.1). Pour ce faire, nous allons utiliser les fonctions expliquées ci-dessus. Il faut bien prendre garde à imposer les bonnes conditions pour que les rapports soient valides. On vérifie donc que les médianes soient bien différentes de 0 et s'il se trouve qu'il y en ait qui ne respectent pas cette condition, on met le rapport égal à 0 pour ne pas en tenir compte. En fonction du nombre de rapport égal à zéro, on adapte le dénominateur de l'indicateur pour ne diviser que par le nombre d'indicateurs utiles. Une seule exception à ce système est le calcul du rapport de ponctualité (voir équation 3.3). Nous devons vérifier à chaque fois la médiane et l'indicateur personnel pour écraser ces valeurs à 0.8 si elles sont égales à 0. Une fois tous les rapports obtenus, on récupère les coefficients définis par le professeur à l'aide du formulaire et on calcule la valeur de l'indicateur global à l'aide de la formule présentée dans le chapitre précédent.

Une fois que nous avons récupéré toutes les valeurs chiffrées des différents indicateurs, il ne nous reste plus qu'à les convertir en indicateurs de couleur (voir équation 3.2). Pour ce faire, nous avons dû créer trois méthodes pour gérer les différentes situations présentes. La première sert à convertir les 5 premiers indicateurs. Pour ce faire, nous vérifions que la médiane est bien différente

de 0 (si ce n'est pas le cas, nous renvoyons la mention "Pas pertinent") et ensuite en fonction de la valeur du rapport, nous retournons la couleur associée. La deuxième est pour l'indicateur de ponctualité, ce dernier se calcule de manière un peu différente des 5 autres. Nous vérifions d'abord les valeurs et les écrasons à 0.8 si elles sont égales à 0 et ensuite nous retournons la couleur associée à la valeur du rapport. La dernière méthode est pour la conversion de l'indicateur global qui lui ne nécessite aucune vérification particulière. Les indicateurs de couleurs sont représentés par trois images (un rond rouge, jaune et vert). Ces images sont présentes dans le dossier **/images**.

Le fichier *lib.php* contient également la fonction permettant d'ajouter le lien vers le rapport dans le menu du cours. Pour ce faire, on crée un fonction qui étend le menu de navigation du cours. Et on ajoute le lien vers le cours en ajoutant l'id du cours dans lequel on se trouve comme paramètre.

4.3.2 Système d'affichage

Après avoir correctement récupéré toutes les informations nécessaires à la génération du rapport d'un étudiant, il nous faut les afficher dans un esprit d'efficacité, de facilité et de compréhension. Nous avons choisi de représenter les données à l'aide de tableaux. Ce système d'affichage est simple à utiliser et à comprendre.

Pour avoir un rendu aussi clair que possible nous avons choisi d'utiliser 3 tableaux différents (voir figure 4.1) (un par catégorie d'indicateurs : notes, participation, progression). Au dessus de ces tableaux se trouvent l'en-tête de la page ("Rapport de suivi") et l'indicateur de couleur correspondant à l'étudiant en question.

Les tableaux suivent tous la même logique et sont générés à l'aide des méthodes des tableaux HTML pour PHP :

- D'abord un en-tête avec les titres de colonnes : **valeur de l'étudiant**, **médiane** et **indicateur de couleur**
- Ensuite viennent les valeurs des différents indicateurs (un indicateur par ligne de tableau).
- Et en dessous du tableau un lien vers la ressource correspondant à la catégorie.

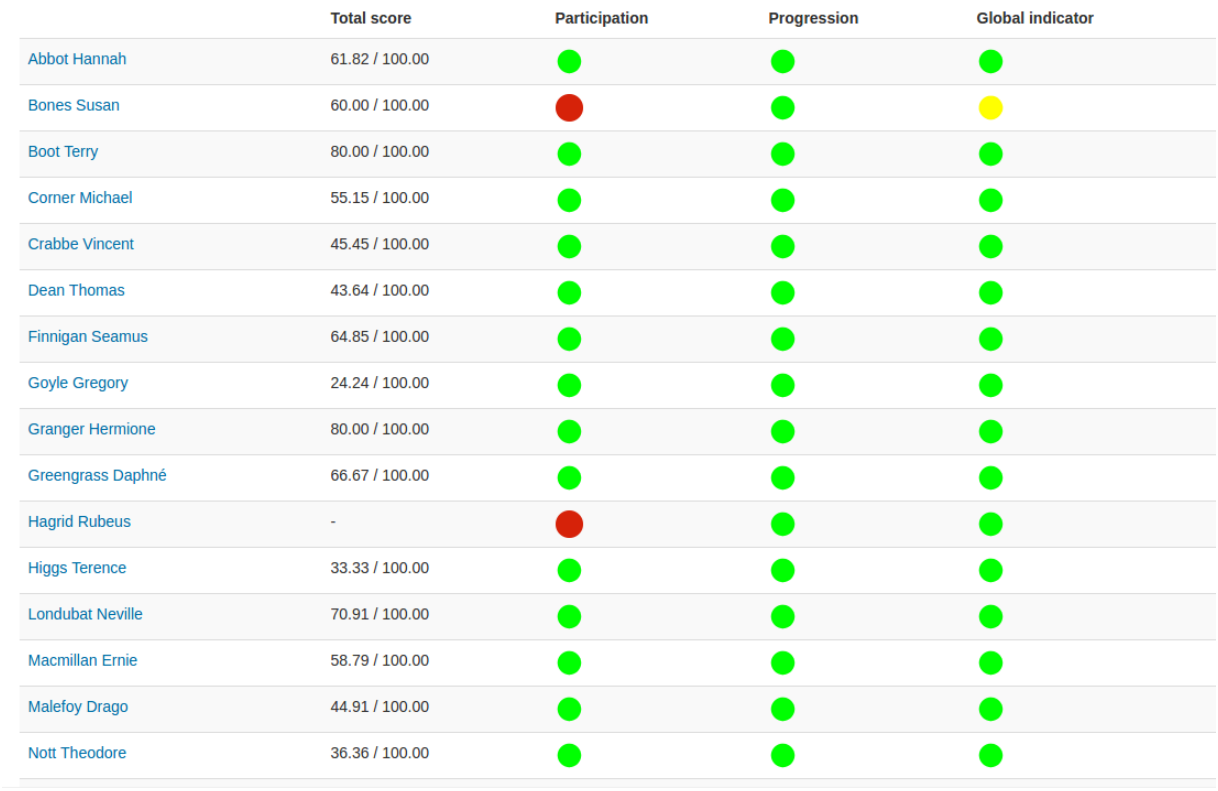
4.4 Rapport professeur

Ensuite nous nous sommes intéressés au rapport du professeur (voir figure 4.2). Les informations à afficher sont différentes du rapport étudiant mais l'affichage suit la même logique. La majeure différence par rapport à ce qui a été présenté plus haut est que le professeur dispose de fonctionnalités supplémentaires qui lui permettent d'avoir plus d'informations sur la situation des étudiants et de sa classe en général.

4.4.1 Fonctions utiles

Pour la génération du rapport professeur, nous avons dû créé deux nouvelles fonctions dans le fichier *lib.php*. Ces fonctions nous sont utiles pour générer les deux indicateurs propres aux professeurs : la participation et la progression.

Ces deux indicateurs fonctionnent exactement de la même manière. On récupère les coefficients définis pas le professeur et ensuite, on récupère les deux rapports qu'il nous faut pour calculer les indicateurs, les actions de contributions et consultation pour la participation (voir équation 3.5) et les activités complétées et la ponctualité pour la progression (voir équation 3.6). On



	Total score	Participation	Progression	Global indicator
Abbot Hannah	61.82 / 100.00			
Bones Susan	60.00 / 100.00			
Boot Terry	80.00 / 100.00			
Corner Michael	55.15 / 100.00			
Crabbe Vincent	45.45 / 100.00			
Dean Thomas	43.64 / 100.00			
Finnigan Seamus	64.85 / 100.00			
Goyle Gregory	24.24 / 100.00			
Granger Hermione	80.00 / 100.00			
Greengrass Daphné	66.67 / 100.00			
Hagrid Rubeus	-			
Higgs Terence	33.33 / 100.00			
Londubat Neville	70.91 / 100.00			
Macmillan Ernie	58.79 / 100.00			
Malefoy Drago	44.91 / 100.00			
Nott Theodore	36.36 / 100.00			

FIGURE 4.2 – Rendu du rapport professeur.

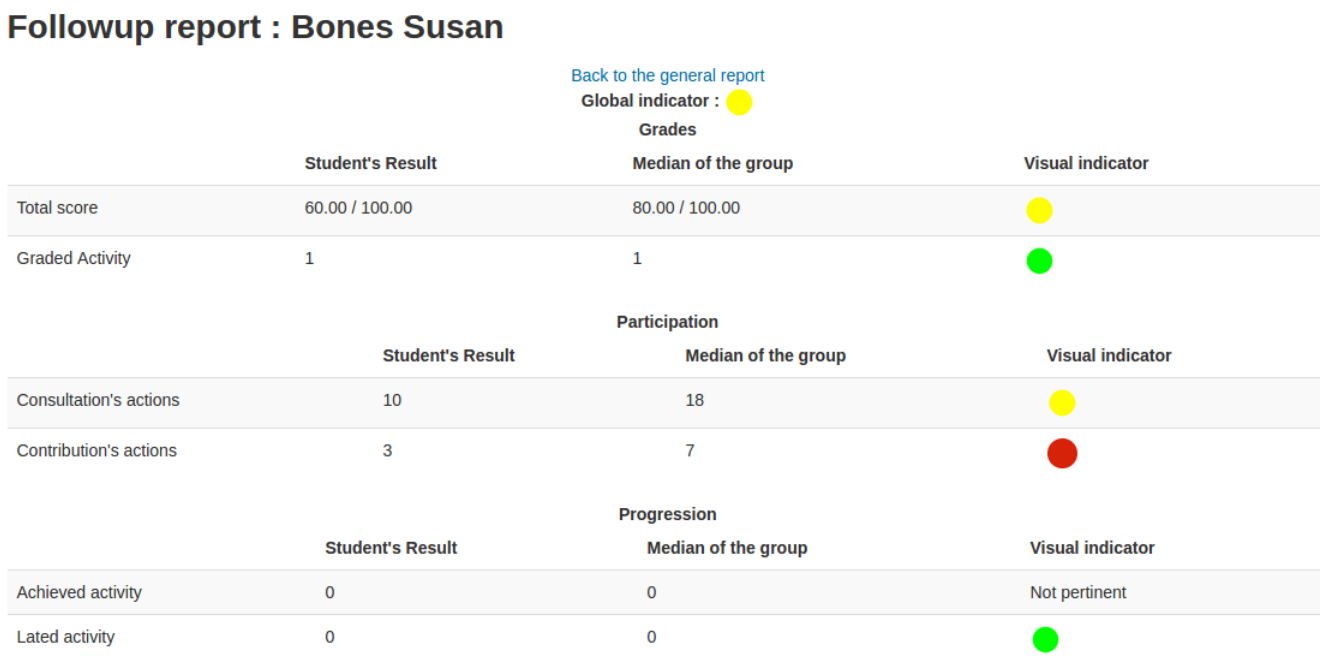
fait également bien attention à ne prendre en compte que les indicateurs dont la médiane est différente de zéro et on adapte le dénominateur si besoin.

4.4.2 Système d'affichage

Le système d'affichage est sensiblement similaire à celui du rapport étudiant mais d'autres éléments viennent s'y ajouter. Tout d'abord, on met l'en-tête du rapport suivi par un lien permettant d'accéder à la page d'administration. Ensuite on affiche les graphiques. Il faut savoir que ces derniers ont été conçus pour s'adapter à l'espace dans lequel ils sont placés. Nous avons donc choisi de les mettre dans un tableau pour éviter que ceux-ci ne prennent trop de place au sommet de la page mais surtout pour qu'il soit plus lisible par l'utilisateur.

Ensuite, on crée le tableau reprenant toutes les infos des étudiants. Pour ce faire, on utilise une boucle **for** qui va boucler sur tous les étudiants du cours en affichant sur chaque ligne : leur nom, leur score total, leur indicateur de participation, leur indicateur de progression et leur indicateur global.

Il est également possible pour le professeur de cliquer sur le nom d'un étudiant pour obtenir son rapport détaillé (voir figure 4.3). Pour rendre cette option viable, nous avons dû créer une nouvelle page (*infos.php*). Il est possible de récupérer les informations d'un étudiant en particulier en transmettant son id via le lien menant à la page infos. Avec cet id transmis via l'url, nous sommes capables de récupérer les données sans que l'utilisateur soit véritablement connecté. On les affiche exactement comme présenté lors du chapitre étudiant. La seule différence est qu'on rajoute le nom de l'étudiant dans l'en-tête pour que le professeur puisse s'en souvenir et on supprime les liens vers les ressources qui ne sont accessibles que pour les personnes connectées.



Followup report : Bones Susan

Back to the general report

Global indicator :

Grades

	Student's Result	Median of the group	Visual indicator
Total score	60.00 / 100.00	80.00 / 100.00	
Graded Activity	1	1	

Participation

	Student's Result	Median of the group	Visual indicator
Consultation's actions	10	18	
Contribution's actions	3	7	

Progression

	Student's Result	Median of the group	Visual indicator
Achieved activity	0	0	Not pertinent
Lated activity	0	0	

FIGURE 4.3 – Rendu du rapport détaillé accessible par le professeur.

4.4.3 Graphique

Les graphiques ont été ajoutés au-dessus du tableau présentant les données des étudiants pour donner au professeur des statistiques sur les indicateurs de sa classe (voir figure 4.4). Ceci lui permet de voir directement la situation générale dans laquelle se trouvent les élèves. Nous avons donc créé trois graphiques représentant les trois indicateurs de couleurs (participation, progression et global). Ceux-ci sont des graphiques en camembert présentant le pourcentage de couleurs de chaque indicateur.

Ces graphiques sont générés à l'aide de l'API chart. Pour pouvoir générer des graphiques cohérents par rapport à nos données, nous avons dû créer de nouvelles fonctions dans le fichier *lib.php*. Il s'agit de trois fonctions ayant pour but de récupérer les proportions de chaque indicateur pour les 3 catégories. Chaque fonction procède de la même manière, pour chaque étudiant, on calcule son indicateur et on incrémente une variable en fonction de la valeur de ce dernier. La fonction retourne trois valeurs qui correspondent au nombre de chaque couleur.

Une fois que ces valeurs sont récupérées, il est extrêmement facile de générer les graphiques correspondants. Il suffit de leur fournir les séries générées à l'aide des fonctions et puis on peut les modifier comme on le souhaite. Nous leur ajoutons donc un titre ainsi qu'une légende pour les rendre facilement compréhensibles par l'utilisateur. Une seule particularité est à noter ici, nous avons dû modifier le fichier *config.php* présent à la racine du dossier Moodle pour donner les bonnes couleurs correspondant aux couleurs des indicateurs.

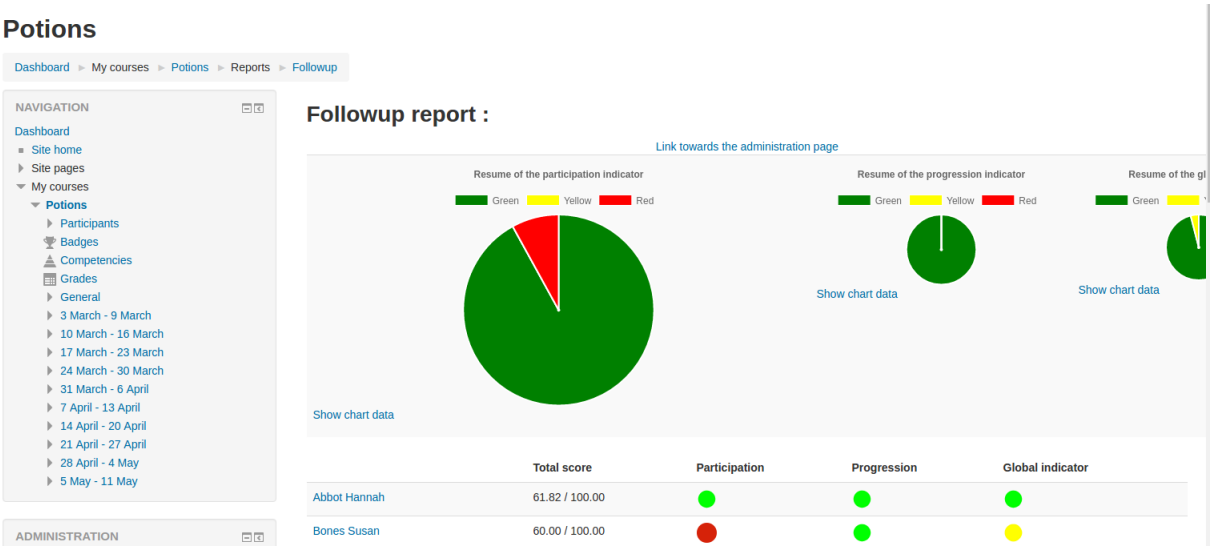


FIGURE 4.4 – Rendu des graphiques au sommet de la page.

4.5 Page d'administration

La page d'administration (*admin.php*) a été créée pour donner aux professeurs la possibilité d'adapter le poids de chaque indicateur dans le calcul des indicateurs visuels (voir figure 4.5). Il semble clair qu'en fonction de l'utilisation qu'un professeur va faire de la plateforme, certains éléments seront plus primordiaux que d'autres. En donnant des poids plus importants à certains critères, le professeur va pouvoir "customiser" les statistiques que le rapport lui renvoie.

Le système proposé est assez simple. Le formulaire permet aux professeurs d'introduire un nombre qui sera le coefficient par lequel sera multiplié l'indicateur dans le calcul d'un indicateur global. La page admin, en plus du formulaire, contient également un lien permettant de retourner directement à la page du rapport.

4.5.1 Nouvelle table

Afin de mettre en place ce système de coefficient, il nous fallait un endroit où stocker les données entrées par les professeurs. Nous avons donc ajouté une nouvelle table à la base de données existante de Moodle. Cette table est uniquement dédiée au stockage des coefficients pour les différents cours. Pour la créer, nous avons utilisé XMLBD editor comme présenté dans une section précédente. Cette table contient donc l'id du cours courant ainsi que les 6 coefficients enregistrés par les enseignants. Nous avons mis comme précaution que les id de cours doivent être tous différents pour éviter d'avoir plusieurs jeux de données associés à un même cours.

4.5.2 Gestion du formulaire

Pour mettre en place le formulaire, il nous a fallu créer un nouveau fichier (*form.php*). Ce fichier contient une classe qui étend la classe moodleform qui est le fichier de base pour la création d'un nouveau formulaire. Elle contient une seule méthode : **definition()** pour définir la forme du formulaire.

Dans notre cas, on souhaite générer un formulaire contenant 6 champs pouvant accueillir des valeurs numériques. Ces 6 champs sont repartis en 3 catégories. La première étape est de placer un élément caché dans la méthode **definition()** afin de pouvoir récupérer l'id du cours dans

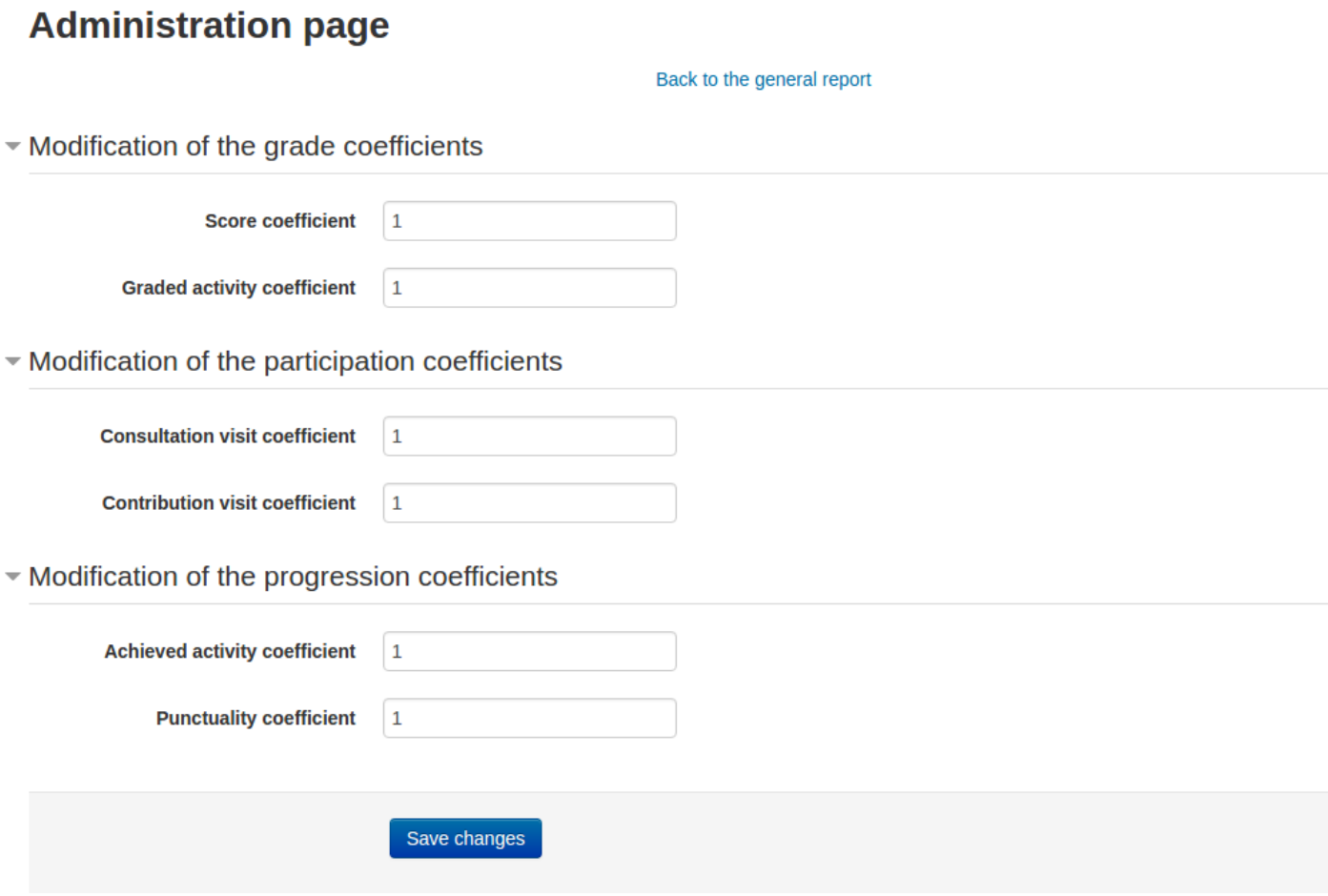


FIGURE 4.5 – Rendu de la page d'administration avec son formulaire.

lequel on travaille. Ensuite le schéma est similaire pour les 3 catégories.

On met d'abord un en-tête pour définir dans quelle catégorie on se trouve (notes, participation ou progression). Ensuite on ajoute l'élément et on définit son type. Nous avons également fait le choix de pré-remplir le champ avec la valeur actuellement présente dans la base de données. Une fois les 6 champs créés, on ajoute un bouton qui permet de valider le formulaire.

Une fois que la classe permettant de générer notre formulaire est définie, il nous faut instancier ce dernier à l'endroit nécessaire. L'instanciation est extrêmement simple, il suffit de donner comme paramètre l'id du cours courant. Ensuite, il faut définir les actions à réaliser dans trois cas de figures différents :

- Le formulaire est annulé : on recharge la page d'administration.
- Les données sont envoyées et sont cohérentes : on enregistre les coefficients dans la table. Si le cours est déjà présent dans cette dernière, on met à jour les infos et sinon on crée une nouvelle entrée avec le nouveau cours.
- Le formulaire est affiché pour la première fois : on affiche le formulaire.

4.6 Installation du plugin

L'installation d'un plugin est extrêmement aisée sous Moodle, la première étape est de se rendre sur le dépôt Moodle (https ://moodle.org/plugins/). Une fois en possession de l'archive .zip

contenant tous les fichiers nécessaires au bon fonctionnement du rapport, il faut l'installer dans le dossier correspondant au type de structure installée (/report dans notre cas). Il suffit ensuite de se rendre dans le menu administration->notifications. L'installation se fera automatiquement et si la bonne version de Moodle est présente, aucun problème ne devrait survenir.

Il est possible que le dossier contenant les rapports dans le menu du cours ne soit pas présent par défaut. Pour le rendre visible par les utilisateurs il faut se rendre dans l'administration du cours->Utilisateurs->Permissions et définir les permissions comme souhaité par l'administrateur du cours.

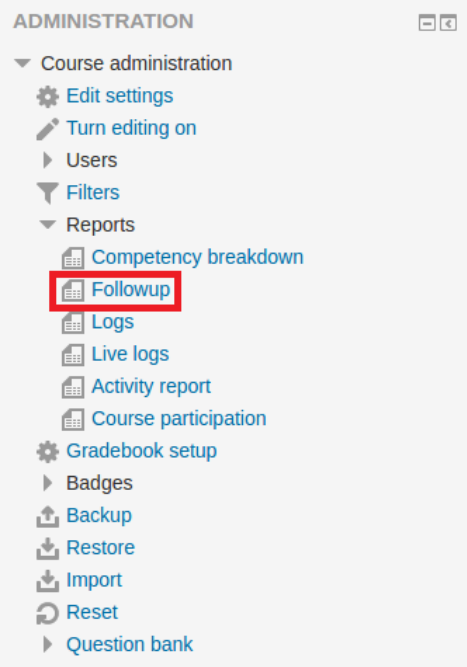


FIGURE 4.6 – Présence du lien dans le menu du cours.

Chapitre 5

Évaluation

L'extension souhaitée est fonctionnelle au sein de la plateforme Moodle. Il est cependant intéressant de se plonger sur les performances obtenues mais également sur les limitations rencontrées et le futur du rapport produit. Ce chapitre va présenter tous ces éléments et également proposer des solutions lorsque des problèmes seront rencontrés. Un large panel d'améliorations futures va également être présenté.

5.1 Performance et évaluation

5.1.1 Outils de vérification Moodle

Comme nous l'avons observé dans le chapitre 4, la plateforme Moodle répond à des règles de structure et syntaxe très précises. Fort heureusement, elle dispose d'outils de vérification permettant aux développeurs de mettre leur code à l'épreuve. Deux de ses outils sont conseillés sur la page dédiée au style de codage de la documentation Moodle [32]. Il est intéressant de ne pas en utiliser qu'un seul car ceux-ci effectuent différents types de tests.

Les deux testeurs sont à installer comme des plugins Moodle (Code checker et Moodle PHPdoc checker). Il suffit de leur spécifier l'endroit où se trouvent les fichiers à tester et puis analyser le résultat. Nous avons d'abord utilisé le plugin Code checker, ce dernier a trouvé 1999 erreurs et 136 avertissements. Ces nombres peuvent paraître énormes mais il faut bien se rendre compte que chaque erreur de style est détectée par le logiciel. Les deux erreurs les plus courantes sont des fautes d'indentation ainsi que des fautes de retour à la ligne. L'utilisation du deuxième plugin a trouvé 39 erreurs supplémentaires.

Il peut sembler futile de corriger ce genre de problèmes étant donné que le plugin fonctionne très bien malgré ces erreurs mais cette mise en forme particulière du code permet à d'autres développeurs de se plonger rapidement dans tous les fichiers présents dans le module. Rappelons que Moodle est sous licence libre et donc que ses utilisateurs sont autorisés à réutiliser le travail d'autrui pour s'inspirer/l'améliorer. Si tout le monde suit quelques règles, cela permet à tous de comprendre facilement des fichiers remplis de lignes de codes.

Les fichiers finaux de ce mémoire ont passé les tests des deux plugins et sont syntaxiquement corrects par rapport aux règles de la plateforme.

5.1.2 Optimisation de la génération du rapport

Notre rapport actuel connaît un large problème un terme de performance. Les informations que nous récoltons se trouvent dans la table des logs, malheureusement cette dernière stocke énormément de données et arrive généralement à un nombre de lignes énormes (plus de 200

millions pour l'UCL). Les recherches effectuées en son sein prennent donc un certain temps à être exécutées et dès lors la page du rapport a un temps de génération élevé. Nous avons entrepris de diminuer ce temps avec certaines méthodes d'optimisation. Tous les temps présentés ont été mesurés à l'aide de l'outil de développement proposé par Google Chrome.

Pour montrer l'efficacité de l'optimisation effectuée, nous allons nous appuyer sur la base de données du Moodle local sur lequel nous avons travaillé. La table des logs de notre plateforme contient (au moment du test) 8922 lignes. Lorsque toutes les fonctionnalités souhaitées ont été implémentées, la page prenait 27.24 secondes pour être générée. Pour diminuer ce temps, nous nous sommes plongé dans la documentation concernant les requêtes SQL [25] et nous avons trouvé une méthode permettant de diminuer le nombre d'appels à la base de données. Le deuxième test fait avec ces nouvelles méthodes a donné un temps de 11.47 secondes, ce qui constitue une diminution non-négligeable.

Afin d'aller encore plus loin dans la démarche, nous avons essayé d'exporter les données utiles de la table des logs vers une autre table pour diminuer le nombre d'informations dans celle où nous allons chercher l'information. Nous avons également utilisé des index pour augmenter la vitesse de recherche au sein des tables (un index est un fichier auxiliaire qui rend la recherche pour des records plus efficace dans la table [33]). L'index permet une recherche sur les trois paramètres dont nous avons besoin : l'id du cours, l'id de l'étudiant et l'action que nous souhaitons retrouver. Voici les résultats que nous avons obtenus :

	Avec index	Sans index
Table des logs	8.11 s	9.94 s
Nouvelle table	7.34 s	50.48 s

TABLE 5.1 – Temps de génération du rapport.

Il y a plusieurs éléments à considérer ici. Tout d'abord, on voit qu'il est toujours préférable de travailler avec un index, sans lui la recherche prend un temps considérable. La différence entre la table des logs et la nouvelle est énorme (± 40s), tout simplement parce que la table des logs possède déjà une série d'index prédéfinis par la plateforme. Pour ce qui concerne la différence entre les deux tables, on voit qu'elle est minime. Cela provient du fait que notre base de données n'est pas représentative car personne ne l'utilise hormis l'admin ce qui fait, qu'à part les actions pré enregistrées, elle ne grandit pas du tout. Comme annoncé au début, la table des logs contient 8922 lignes et lorsque l'on extrait seulement les informations nécessaires, la nouvelle table contient 7191 lignes. Il va sans dire qu'effectuer la même opération sur la base de données d'une plateforme utilisée donnerait des résultats bien différents. Le problème serait que cette technique ne ferait qu'agrandir l'espace occupé par une telle base.

Il semble peu probable de pouvoir encore optimiser l'opération de recherche du côté client. Mais il faut se rappeler que l'on travaille sur une machine locale et donc que les performances sur un serveur ne pourront être que meilleures car ces machines sont conçues pour la communication entre la base de données et la plateforme. Une machine locale connaît beaucoup d'autres opérations fonctionnant en même temps ce qui ralentit ses performances.

La section suivante va présenter des pistes de réflexion qui pourraient rendre le rapport viable dans un environnement d'utilisation réel de la plateforme.

5.2 Autres perspectives

Pour pouvoir utiliser le rapport au sein du MoodleUCL, il faut arriver à optimiser le temps de génération de ce dernier. Le problème actuel provient de la quantité d'informations qu'il faut traiter pour obtenir les indicateurs. Trois pistes vont être explorées ici, ces dernières ne sont que des pistes de réflexion car nous n'avons pas eu accès à des structures permettant de les tester. Il semble clair que pour des raisons de respect de la vie privée et pour éviter tout problème sur la structure actuelle, nous n'avons pas pu utiliser le Moodle mis en place par l'Université, tous nos tests se sont déroulés sur un environnement local qui n'est pas comparable à une structure utilisée quotidiennement par des centaines d'utilisateurs.

5.2.1 Acquisition évolutive des données

Actuellement, à chaque génération du rapport, les indicateurs sont calculés avec toutes les données présentes dans la base de données. Une autre idée serait d'utiliser des techniques de data mining pour ne pas avoir à récupérer toutes les informations à chaque fois. Ce genre de solution pourrait apporter plusieurs avantages : une amélioration du temps de génération du rapport, une sauvegarde régulière de l'état des étudiants, un suivi continu, ...

L'idée serait d'avoir un état de départ du rapport. A chaque nouvelle génération de ce dernier, on calculerait la différence par rapport à l'état précédent à l'aide des données ajoutées dans la base de données pour afficher la version mise à jour du rapport. Ce système nous permet qu'une fois le rapport généré, on ne doive plus se soucier des informations précédemment utilisées mais juste se focaliser sur celles qui se sont rajoutées dans la base de données suite à l'utilisation de la plateforme par les étudiants.

Le principe serait donc bien différent de ce qui est fait actuellement. Au lieu d'aller chercher dans la base de données les informations pour calculer les indicateurs et les afficher à l'utilisateur, on appliquerait une structure classique de Data Mining (voir figure 5.1). Les calculs pourraient être effectués à une période définie par l'administrateur (durant la nuit idéalement pour limiter l'utilisation de la plateforme), ce qui permet aux utilisateurs d'obtenir directement les résultats lors de la génération du rapport.

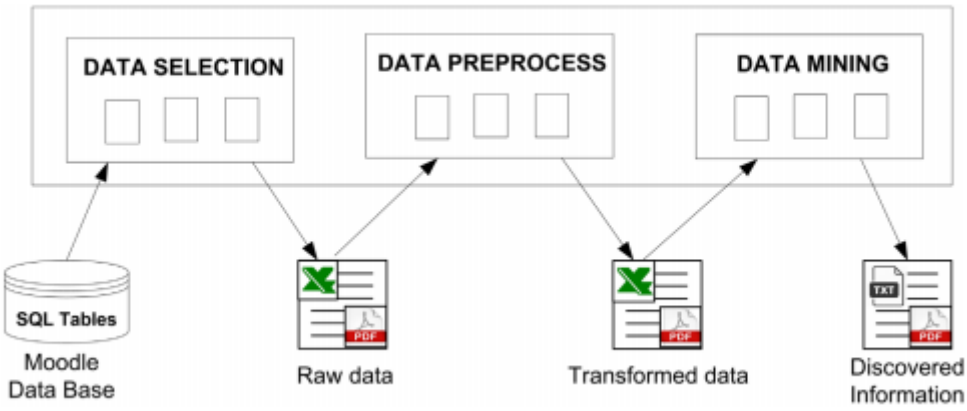


FIGURE 5.1 – Processus de data mining.

Il existe déjà pas mal d'exemples de plugin utilisant du Data Mining pour obtenir des informations sur les utilisateurs. Un bon exemple bien détaillé est le MDM tool [37] qui est un bloc qui a pour but de proposer aux professeurs toutes sortes de statistiques sur les différentes données présentes dans la base de données. Plusieurs algorithmes sont proposés lors de l'analyse.

Ce genre d'analyse est bien plus poussée que celle que nous avons mise en place actuellement mais n'aurait pas pu être développée dans le temps imparti. Nous avons créé une base stable et efficace de travail qui permettrait à d'éventuelles personnes intéressées par le sujet de proposer des améliorations qui pourront le rendre plus performant.

5.2.2 Amélioration de la table des logs

Les requêtes ont été optimisées dans notre code mais il est également possible que les problèmes de performance proviennent de la source où l'on va chercher les données. La table des logs a pour ambition de conserver toutes les traces que laissent les utilisateurs en parcourant la plateforme. De ce fait, elle est également la plus grande table de la base de données de toute instanciation de Moodle. Ce phénomène génère un problème sans fin : on a besoin d'informations pour faire du suivi de l'étudiant mais récupérer ces dernières sur une grande table prend beaucoup de temps.

L'optimisation de la table des logs semble donc une alternative intéressante à envisager pour permettre de rendre la génération du rapport plus efficace. Cependant, ce genre de problèmes ne peut pas être réglé via notre rapport mais bien par l'administrateur de la plateforme.

Une première option qui peut être intéressante à envisager est de faire des backup de cette table tous les quadrimestres. De ce fait, il y aura beaucoup moins de données à traiter lors des recherches et on ne perd aucune information concernant les cours, étant donné que ces derniers sont également donnés sur un seul quadrimestre. Il semble donc inutile de conserver toutes les données de l'année dans la table de base. De plus, en les gardant à l'aide d'un backup, il est toujours possible d'aller chercher des informations en cas de besoin.

Une autre solution possible serait d'utiliser deux bases de données indépendantes. Une de ces deux bases aurait pour unique but de gérer les logs enregistrés par la plateforme. Pour cette option, il serait intéressant d'utiliser une base NoSQL [38]. Ces dernières permettent d'obtenir de meilleures performances pour obtenir les valeurs nécessaires à la génération des indicateurs. Pour enregistrer les données dans une autre table, il serait intéressant de regarder à la page de gestion des logs [39] et de rajouter à cette dernière un script permettant d'effectuer l'enregistrement des logs dans une base externe.

5.2.3 Sauvegarde du rapport

La génération du rapport à chaque consultation via le lien prend du temps dû à la quantité de données à traiter. Il serait envisageable de pouvoir "figer" l'état du rapport à un moment donné et pouvoir le consulter sans devoir effectuer à nouveau tous les calculs nécessaires. Lors de la consultation du rapport, la version sauvegardée serait dès lors affichée et une option permettrait de calculer un nouvel état de ce dernier. Cette idée rejoint l'idée d'utiliser une dynamique de data mining pour obtenir des informations. L'utilisation de cette technique donnerait la possibilité d'avoir un suivi de l'évolution.

5.3 Limitations du module

Il faut se souvenir que notre plugin a été développé sur une plateforme test ne contenant que très peu de données en son sein. Il semble clair que des ajustements devront être effectués au fur et à mesure de son utilisation en temps réel. L'utilisation du cours test de l'UCL nous a semblé être une base solide pour affirmer que le rapport est fonctionnel et utile.

5.3.1 Cadre de Moodle

Notre environnement de développement est la plus simple version de Moodle que l'on puisse connaître. Comme présenté dans le chapitre 2, l'espace de travail de Moodle est entièrement personnalisable. Nous n'avons pas pu tester toutes les possibilités graphiques que propose la plateforme.

Les utilisateurs peuvent choisir de nouveaux thèmes pour modifier l'apparence du site. Depuis la version 2.0 de Moodle, il existe une structure appelée "render" [34] qui permet d'adapter l'affichage des éléments en fonction du thème choisi par l'utilisateur. Ces derniers permettent aux développeurs de pouvoir séparer la logique de l'affichage. Nous ne les avons pas utilisés dans l'implémentation de notre rapport car nous sommes partis dans une autre optique au départ.

Un autre problème serait que l'utilisateur utilise un placement des blocs différents et donc que la taille allouée au rapport sur la page soit plus petite que celle sur laquelle nous l'avons testé. L'affichage pourrait donc être corrompu et moins clair. Malgré tout, les éléments affichés par Moodle savent majoritairement s'adapter à l'environnement dans lequel ils sont affichés.

Tous ces éléments ne posent problème que pour l'aspect graphique du rapport. Les fonctionnalités ne seront, bien entendu, par affectées par les différentes modifications graphiques que peut connaître la plateforme.

5.3.2 Utilisation de la plateforme

Durant ce travail, nous nous sommes limités à créer un nouveau plugin indépendant et fonctionnel. Moodle a pour but d'être entièrement modulable. Il serait donc opportun d'utiliser au mieux toutes les ressources qu'elle met à disposition. Actuellement, le rapport est un affichage statique des données que nous récoltons via nos différentes méthodes. Avec tous les plugins qui se trouvent dans le dépôt moodle, notre création pourrait être améliorée en combinant plusieurs structures pour pouvoir, en plus de seulement consulter des informations, interagir avec le site et offrir du dynamisme à l'utilisateur.

Une partie des différentes options possibles pour utiliser au maximum les possibilités offertes par Moodle vont être présentées dans la section suivante.

5.4 Améliorations possibles du module

5.4.1 Nouvelles fonctionnalités

Notre plugin est le fruit de plusieurs mois de travail, il est clair qu'avec plus de temps, il serait possible d'ajouter de nombreuses fonctionnalités supplémentaires. Nous avons cependant accompli les objectifs fixés au départ de ce mémoire. Malgré tout, ces dernières peuvent rendre l'utilisation du rapport plus simple mais également proposer plus d'outils pour gérer le suivi des étudiants.

La plupart des ajouts que nous allons évoquer nous proviennent du travail réalisé par l'Université de Laval [1]. Ces derniers, ayant réalisé un outil bien plus complet que le nôtre, disposent d'outils supplémentaires très intéressants qui pourraient être ajoutés à notre plugin.

Tri des étudiants

Il est clair que certains cours peuvent contenir des centaines d'étudiants et les afficher par ordre alphabétique dans un grand tableau n'est pas la solution la plus adaptée à la situation. L'idéal serait de fournir plusieurs options aux professeurs pour personnaliser l'affichage des

étudiants de la classe. Une classification par indicateur est des plus appropriées pour pouvoir repérer directement les cas problématiques. Une autre option serait de n'afficher qu'une certaine classe d'étudiants basée sur leurs données. Cette option permettrait également d'effectuer des actions avec seulement certains étudiants (envoyer un message, exporter les données, ...).

Tous ces changements sont faciles à mettre en place, il suffirait de placer une série de menus déroulants, chacun proposant différents paramètres (ordre d'affichage, critères à afficher, données à afficher, ...) et en fonction des résultats obtenus, on pourrait générer le rapport correspondant. Adaptez le rapport avec ce système demanderait une refonte complète du système d'affichage du côté professeur mais ce n'est pas un travail de titan. La plupart des rapports sont d'ailleurs présentés sous la forme de menu sélectif pour ensuite afficher les données des paramètres sélectionnés.

Boîte de communication

Il existe déjà un système de communication propre à Moodle. Il autorise les personnes à communiquer soit par messages ou bien directement via un chat [35]. L'idée serait de permettre aux étudiants et aux professeurs de se contacter directement depuis le rapport. Il semble inutile de redéfinir toute une structure, étant donné que celle déjà présente est fonctionnelle. Cependant, il semblerait opportun d'ajouter un lien permettant à l'étudiant d'envoyer un message au professeur du cours courant et également au professeur de contacter un étudiant via son rapport détaillé. Une dernière piste à envisager serait de permettre aux professeurs de communiquer avec plusieurs étudiants en même temps et même de pouvoir envoyer des messages automatiques. Combiné avec l'amélioration présentée juste avant, il serait également intéressant de pouvoir contacter tous les étudiants d'une certaine catégorie d'indicateur. Par exemple, envoyé un message à tout les étudiants ayant un indicateur rouge.

Exporter les données

Il est parfois utile de pouvoir récupérer les données sous un autre format que celui présenté par notre module. De ce fait, une option permettant aux professeurs d'exporter les données du rapport dans un autre format (CSV, ODS, ...) peut être utile. Cela pourrait permettre de travailler dans un autre environnement mais également de faire de l'analyse de données plus poussée avec des outils qui ne sont pas forcément présents sur la plateforme Moodle.

Un défi concernant cette option est que les données sont calculées à chaque génération du rapport et ne sont enregistrées nulle part. Pour pouvoir les exporter, il faut avoir un endroit où les récupérer. Il faudrait donc repenser cet aspect pour pouvoir mettre au point cette fonctionnalité.

Graphique d'évolution

Un graphique d'évolution permettrait de suivre le changement dans les indicateurs d'un étudiant. Cela donne également la possibilité de voir à quels moments la plateforme est plus ou moins utilisée. La création de graphique a été présentée dans le chapitre 4 et est plutôt simple sur Moodle. Le problème dans cette fonctionnalité est qu'il faut fournir des données pour le créer. Pour pouvoir présenter une évolution d'indicateur, il faut que les précédentes données soient enregistrées quelque part pour pouvoir les récupérer.

Pour résoudre ce problème, il est possible d'envisager une nouvelle table dans la base de données qui enregistrerait les indicateurs des étudiants à un moment précis, de manière périodique. De cette manière, il est possible de générer un graphique avec toutes les données enregistrées.

Lien vers des ressources

Laval propose des mini-tests aux étudiants en fonction des difficultés qu'ils ont dans certains cours. Un système similaire pourrait être envisagé dans le cadre du rapport Moodle. Plusieurs liens peuvent être proposés aux étudiants connaissant des problèmes. Tout d'abord des ressources d'informations fournies par le professeur ou encore des quizz supplémentaires qui permettraient aux étudiants de se confronter à certains points de la matière. Mais aussi des informations externes à Moodle.

Le plus difficile dans cette fonctionnalité est de proposer des aides adaptées au profil de l'étudiant. Pour cela il faudrait être capable d'analyser les difficultés de cette dernière en fonction de son utilisation de la plateforme. Cependant, à l'heure actuelle, les indicateurs produits ne permettent pas de faire une telle analyse car ils ne se portent pas directement sur la matière présente dans le cours mais plus sur l'utilisation de la plateforme. Laval utilise cette fonctionnalité car ils disposent de critères bien plus ciblés sur l'évaluation de leurs étudiants.

Ajout d'un bloc associé au rapport

Comme présenté dans le chapitre 2, les blocs sont une structure Moodle qui peut être placée sur n'importe quelle page et occuper une petite place de celle-ci. Dans l'idée du plugin engagement analytics [2], un bloc reprenant un résumé des indicateurs globaux des étudiants serait un plus pour les professeurs. Simplement en accédant à leur page de cours, il aurait instantanément accès à la situation générale des étudiants de la classe sans pour autant devoir générer le rapport complet. Du côté étudiant, un bloc sur leur tableau de bord pourrait leur présenter un résumé de leur situation dans les différents cours suivis. Ils pourraient dès lors, simplement en se connectant sur Moodle, voir où ils en sont par rapport aux différents groupes.

La création d'un bloc dans Moodle suit la même démarche que la création d'un rapport [36]. De plus, si les deux ressources sont installées sur la même plateforme, on peut directement récupérer les méthodes du rapport pour obtenir les informations nécessaires à la génération des données du bloc. Seulle la gestion de l'affichage et la récupération des données doit être travaillées, ce qui limite donc la quantité de travail à effectuer pour avoir des blocs fonctionnels.

5.4.2 Informations transmises

Notre rapport actuel est une structure de consultation, aucune action n'est prise en fonction des résultats présentés. Des fonctionnalités futures pourraient changer ce système en ajoutant de l'interaction avec l'utilisateur. On pourrait imaginer l'envoi d'un mail automatique dès qu'un indicateur arrive dans le rouge ou bien encore une notification envoyée à l'étudiant.

De plus, comme présenté dans le chapitre 3, les données présentées ne sont pas la solution finale de l'évaluation de l'implication des étudiants. Ces dernières peuvent être largement améliorées pour transmettre encore plus de sens au travail que font les étudiants sur la plateforme. Plusieurs moyens peuvent permettre d'améliorer cet aspect. Une analyse plus approfondie des informations stockées dans la base de données Moodle peuvent nous donner encore plus de moyen d'évaluation. Et d'autre part, une analyse pédagogique plus poussée de ce qui représente réellement l'implication d'un étudiant dans un cours pourrait aider à améliorer les indicateurs produits.

Chapitre 6

Conclusion

Le développement d'une extension à la plateforme Moodle est un travail laborieux et instructif. Nous pouvons affirmer que les objectifs fixés au départ de ce mémoire ont été remplis (idée d'extension utile, développement d'un rapport fonctionnel, analyse du travail fourni et améliorations possibles). L'extension est fonctionnelle et dispose de tous les outils nécessaires pour fournir une base solide pour effectuer du suivi d'étudiant. Il reste encore du chemin à parcourir pour obtenir un outil complet et interactif mais le résultat accompli est encourageant.

La première étape de ce mémoire fut de décider l'utilité que nous voulions donner à l'extension. Le suivi de l'étudiant apparu rapidement comme un sujet d'une part extrêmement intéressant à développer et d'autre part manquant dans les outils fournis par Moodle. Il semble dommage que ce dernier récolte tellement d'informations sur les utilisateurs mais que ces dernières ne soient pas exploitées à leur juste valeur. Le travail de l'Université de Laval nous a rapidement mis sur la voie d'un plugin qui pourrait être celui utilisé par l'UCL. Le nôtre contient moins de fonctionnalités mais est sur la bonne voie pour devenir un outil complet.

Au départ, notre connaissance de la plateforme était très limitée vu que la seule utilisation que l'on en avait fait consistait uniquement en la consultation des cours suivis. Nous n'imaginions pas toutes les possibilités que cette dernière avait à offrir à ses utilisateurs. Malgré ce fait, la prise en main n'a pas été des plus difficiles. Il faut reconnaitre que l'aspect open source du système ainsi que l'énorme communauté qui l'entoure nous a permis de nous diriger dans la bonne voie sans perdre de temps. Les nouveaux développeurs qui souhaitent commencer à créer un plugin peuvent se plonger dans la documentation fournie par le site. Il y a beaucoup de subtilités à comprendre pour bien maitriser les règles et codes de la plateforme, il suffit de bien se documenter et ne pas hésiter à lire les APIs avec de se lancer dans l'inconnu.

Nous avons observé qu'il reste encore de nombreuses modifications à apporter à l'application pour que cette dernière puisse être utilisée couramment par les membres de l'Université. Notamment, la performance qui pose de gros problèmes à l'heure actuelle. Il n'est pas concevable de devoir attendre plusieurs dizaines de secondes pour pouvoir générer un simple rapport statique. Un travail sur la gestion des données doit encore être effectué pour obtenir un travail plus efficace.

Nous sommes fier du chemin parcouru et de l'apprentissage que celui-ci nous a apporté. Mais nous espérons encore plus que ce dernier ne s'arrêtera pas là. Il y a tellement d'aspects qui peuvent mener vers quelque chose de réellement utilisable tel que la réflexion sur les indicateurs pour fournir un suivi plus pertinent des étudiants mais également d'un point de vue technique via toutes les améliorations et pistes de réflexion proposées à la fin de ce travail.

Bibliographie

[1] Appui à la réussite, 2016. www.portaildescours.ulaval.ca.

[2] Adam Olley Dr Phillip Dawson, Ashley Holman. Engagement analytics plugin, nov 2012. https ://docs.moodle.org/22/en/Engagement_Analytics_Plugin.

[3] Learning management system, mai 2017. https ://fr.wikipedia.org/wiki/Learning_management_system.

[4] Moodle history, nov 2015. https ://docs.moodle.org/32/en/History.

[5] Licence publique générale gnu, nov 2016. http ://www.gnu.org/copyleft/gpl.html.

[6] Moodle statistics. https ://moodle.net/stats/.

[7] Moodle overview. https ://moodle.com/moodle-lms/.

[8] Moodleplugins. https ://moodle.org/plugins/.

[9] Activities, nov 2015. https ://docs.moodle.org/31/en/Activities.

[10] Ressources, oct 2015. https ://docs.moodle.org/31/en/Resources.

[11] Blocs, may 2016. https ://docs.moodle.org/31/en/Blocks.

[12] Reports, may 2015. https ://docs.moodle.org/dev/Reports.

[13] Carnet de notes, aout 2016. https ://docs.moodle.org/3x/fr/Carnet_de_notes.

[14] Achèvement des activités, feb 2017. https ://docs.moodle.org/3x/fr/Ach%C3%A8vement_des_activit%C3%

[15] Michael de Raadt. Progress bar, aout 2016. https ://moodle.org/plugins/block_progress.

[16] Moodle database diagram. http ://www.examulator.com/er/.

[17] CRef Conseil des recteurs. Aperçu statistique sur les étudiants et sur le personnel des universités de la fédération wallonie-bruxelles. 2014.

[18] Observatoire de l'Enseignement Supérieur. 2.1 les effectifs dans l'enseignement supÉrieur, 2014. http ://www.oes.cfwb.be/index.php ?id=915.

[19] Logging 2, may 2015. https ://docs.moodle.org/dev/Logging_2.

[20] Step-by-step installation guide for ubuntu, sept 2016. https ://docs.moodle.org/31/en/Step-by-step_Installation_Guide_for_Ubuntu.

[21] Debugging, dec 2015. https ://docs.moodle.org/32/en/Debugging.

[22] Adminer, feb 2015. https ://docs.moodle.org/32/en/Adminer.

[23] Xmldb editor, june 2015. https ://docs.moodle.org/dev/XMLDB_editor.

[24] Core apis, may 2017. https ://docs.moodle.org/dev/Core_APIs.

[25] Data manipulation api, sept 2016. https ://docs.moodle.org/dev/Data_manipulation_API.

[26] Access api, jan 2017. https ://docs.moodle.org/dev/Access_API.

[27] String api, april 2017. https ://docs.moodle.org/dev/String_API.

[28] Chart api, april 2017. https ://docs.moodle.org/dev/Charts_API.

[29] Navigation api, may 2017. https ://docs.moodle.org/dev/Navigation_API.

[30] Output api, march 2016. https ://docs.moodle.org/dev/Output_API.

[31] Context, nov 2014. https ://docs.moodle.org/33/en/Context.

[32] Moodle coding style, sept 2016. https ://docs.moodle.org/dev/Coding_style.

[33] Bernard Lambeau. *LINGI2172 – Databases - Indexing*. UCLouvain / ICTEAM, 2013-2014.

[34] Renderer, dec 2015. https ://docs.moodle.org/dev/Renderer.

[35] Messaging, april 2017. https ://docs.moodle.org/33/en/Messaging.

[36] Blocks, dec 2016. https ://docs.moodle.org/dev/Blocks.

[37] C. ROMERO J. M. LUNA, C. CASTRO. Mdm tool : A data mining framework integrated into moodle. Department of Computer Science and Numerical Analysis, University of Cordoba, Cordoba 14071, Spain, july 2016.

[38] Margaret Rouse. Nosql (not only sql database), march 2017. http ://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL.

[39] Logging, may 2015. https ://docs.moodle.org/33/en/Logging.

Annexe A

Tableau présentant les différentes actions de la table des logs

Annexe 1: Tableau des types d'actions dans la table des logs et classification

verb	Explanation	Suivi ?
~~abandoned~~	When a attempt is abandoned by user (Quiz attempt)	non
~~accepted~~	Example: Accepting a statement when submitting an assignment.	non
added	Used to represent "something that already exists is now part of/bound to another entity". Examples: "Admin added role to user X", "Admin added user X to group A". Wrong example: "User added course in category" because it is a 'move' action, except if a course can be part of multiple categories. The good examples work because: A user can have multiple roles, a user can be in multiple groups.	contribution car utilisé en cas de choix de groupe
~~answered~~	Indicates the actor responded to a Question	non, cela donnerait trop de poids aux tests
assessed	Some submitted material has been assessed	contribution, pertinent dans le cadre d'un atelier d'évaluation par les pairs
~~assigned~~	Assign some privilege or role to user.	non
~~attempted~~	Trying to do an activity. Example: attempting a Math class.	non, statut d'une activité non finalisée
~~awarded~~	ex:-teacher awarded student a badge.	non, mais il est intéressant (l'étudiant a reçu un badge)
~~backedup~~	When a backup has been performed.	non
becomeoverdue	When an activity is overdue Example: Quiz attempt is overdue	non, mais on pourrait l'utiliser pour l'indicateur d'engagement
~~called~~	When a call to something is made like an API @see unknown_service_api_called.php	non
~~commented~~	Offered an opinion or written experience of the activity. Can be used with the learner as the actor or a system as an actor. Comments can be sent from either party with the idea that the other will read and react to the content.	non (pas trouvé d'activités associées)

~~completed~~	To experience the activity in its entirety. Used to affirm the completion of content. This can be simply experiencing all the content, be tied to objectives or interactions, or determined in any other way. Any content that has been initialized, but not yet completed, should be considered incomplete. There is no verb to 'incomplete' an activity, one would void the statement which completes the activity."	intéressant mais uniquement lié à la complétion du cours, peu utilisé
created	Used to represent "something new has been created".	contribution car lié par exemple à la soumission d'un devoir, d'un article de glossaire, d'un post dans un forum ...
~~deleted~~	Used to indicate the object in context was deleted.	non, associé à la suppression d'un devoir, d'une page dans un wiki, ...
~~disabled~~	When an activity is disabled. Example: forum read tracking disabled.	non, action prof
downloaded	When a user download file from user. Example submission/report downloaded.	consultation
~~duplicated~~	For something that has been copied.	non, action prof
~~enabled~~	When some setting is enabled. Example: forum read tracking enabled.	non, action prof
~~ended~~	When a process ends. Example: Lesson ended or course reset ended.	non
evaluated	Material has been evaluated.	contribution, pertinent dans le cadre d'un atelier d'évaluation par les pairs
~~exported~~	When a report is exported in certain format.	non, action prof
~~failed~~	Learner did not perform the activity to a level of pre-determined satisfaction. Used to affirm the lack of success a learner experienced within the learning content in relation to a threshold. If the user performed below the minimum to the level of this threshold, the content is 'failed'. The opposite of 'passed'. This is also used in case when message sending is failed.	intéressant mais suppose de réfléchir comment l'exploiter

graded	Used to represent an activity was graded.	permet de mieux expliciter la note totale
~~granted~~	User is granted some extension or capability. Example: extension granted for submission.	non, action prof
~~imported~~	The act of moving an object into another location or system.	non
launched	When an external object is launched. Try consider started if there is related stopped event.	consultation, action associée à la lecture d'un paquet SCORM
~~locked~~	When an activity is locked. Should have a related unlocked event.	non
~~loggedin/loggedout~~	For login and logout.	non
~~loggedinas~~	If user is logged in as different user. This is used by only one event (user_loggedinas). Adding this verb makes event name more clear, then using loggedin verb.	non
~~moved~~	Used to indicate the object in context was moved.	non
~~passed~~	Used to affirm the success a learner experienced within the learning content in relation to a threshold. If the user performed at a minimum to the level of this threshold, the content is 'passed'. The opposite of 'failed'.	intéressant mais suppose d'y réfléchir plus
printed	Something is printed.	consultation, action associée aux livres
~~reassessed~~	Submitted material has been assessed again.	non, action associée à l'activité dans un atelier, donnerait tro de poids aux ateliers
~~reevaluated~~	Material has been evaluated again.	idem
~~removed~~	By opposition to "Added". This does not mean that the object has been deleted, but removed from the entity, or not bound to it any more.	non
~~reset~~	Sets one or more properties back to the default value.	non
~~restored~~	When restoring a backup. Rolling back to a previous state.	non
~~revealed~~	Some identity is revealed. Example: Identities revealed after blind marking.	non
~~searched~~	Something is searched. Example: searched in course.	non, pas clair
~~sent~~	Message sent.	non

~~started~~	Some activity started	intéressant mais à réfléchir
submitted	This is very close to "Attempted". Depends on context which one should be used. For example:- "Admin submitted a form. Student attempted a quiz." is correct, however some cases might not be as clear as the previous example. We can say both "Student submitted an assignment" or "student attempted an assignment". We need to make the difference clear.	contribution
~~suspended~~	Suspend something. (example a user)	non
~~switched~~	Something has been switched. For example:- The workshop phase has been switched to assessment"	non, action prof
~~unassigned~~	As opposed to assigned. When some role is unassigned.	non
~~unlocked~~		non
~~upgraded~~	Something was upgraded, some module probably	non
~~updated~~	Used to indicate the object in context was updated. Simple example is "Admin updated course xyz".	non
uploaded	When an assignment is uploaded.	contribution
viewed	Something has been viewed. For example:- "Student viewed chapter 1 of book 1."	consultation, action associée à toutes les ressources et activités

Annexe B

Présentation des différents codes

B.1 index.php

```php
<?php
// This file is part of Moodle - http://moodle.org/
//
// Moodle is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Moodle is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Moodle.  If not, see <http://www.gnu.org/licenses/>.

/**
 * This page generates the report for teacher and student
 *
 * @package   report_followup
 * @copyright 2017 Bonesire William
 * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */

require(dirname(__FILE__).'/../../config.php');
require_once($CFG->dirroot.'/course/lib.php');
require_once($CFG->libdir.'/adminlib.php');
require_once($CFG->dirroot.'/grade/querylib.php');
require_once($CFG->libdir.'/gradelib.php');
require_once($CFG->dirroot.'/report/followup/lib.php');

$id = required_param('id', PARAM_INT); // Course ID.
$user = $USER->id; // User ID.

$params = array('id' => $id);
$course = $DB->get_record('course', $params, '*', MUST_EXIST);
require_login($course);
$context = context_course::instance($course->id);
$currentuser = optional_param('user', null, PARAM_INT);

$urlparams = array('id' => $id);
```

```
$navurl = new moodle_url('/report/followup/index.php', $urlparams);
$urlparams['user'] = $currentuser;
$url = new moodle_url('/report/followup/index.php', $urlparams);

$title = get_string('pluginname', 'report_followup');
$coursename = format_string($course->fullname, true, array('context' =>
    $context));

$PAGE->navigation->override_active_url($navurl);
$PAGE->set_url($url);
$PAGE->set_title($title);
$PAGE->set_heading($coursename);
$PAGE->set_pagelayout('report');

echo $OUTPUT->header() ;

/*
 * STUDENT VIEW
*/
if (!has_capability('report/followup:teacherview', $context)) {

    /*
     * VARIABLES AREA
     */

    // Get the grade for a particular student in a particular course.
    $grade = grade_get_course_grade($user, $id);

    // Get the graded activity.
    $graded = calculate_indicators_log($user, $id, 'graded');

    // Get the consulted actions.
    $consulted = calculate_indicators_log($user, $id, 'consulted');

    // Get the contributed actions.
    $contributed = calculate_indicators_log($user, $id, 'contributed');

    // Get the completions status.
    $completion = calculate_completion($user, $id);

    // Get the punctuality.
    $punctuality = calculate_indicators_log($user, $id, 'punctuality');

    // Set the threshold for the grade.
    if ($grade->grade / $grade->item->grademax < 0.5) {
        $gradedisplay = '<img␣src="images/red.png"␣heigth="25"␣width
            ="25">';
    } else {
        $gradedisplay = indicators1($grade->str_long_grade,
            median_indicators($id, 'grade'));
    }

    /*
     * TABLE PRINT AREA
     */

    // Print the report title.
    echo $OUTPUT->heading(get_string('header', 'report_followup'));
```

```
97
    // Print the global indicator (color based).
99  $image = indicators3(calculate_global($user, $id));
    echo '<b><center> ' . get_string('global', 'report_followup') . ' : '
    . $image . '</center></b>';
101
    // Print the notes indicator.
103 echo '<b><center>' . get_string('grades', 'report_followup') . '</
        center></b>';

105 $table1 = new html_table();
    $table1->head = array('', get_string('title_result', '
        report_followup'), get_string('title_median', 'report_followup'),
         get_string('title_visual', 'report_followup'));
107 $table1->data[] = array(get_string('score', 'report_followup'), $
        grade->str_long_grade, median_indicators($id, 'grade'), $
        gradedisplay);
    $table1->data[] = array(get_string('graded', 'report_followup'), $
    graded, median_indicators($id, 'graded'), indicators1($graded,
    median_indicators($id, 'graded')));
109 echo html_writer::table($table1);

111 echo "<a href='http://localhost/moodle/grade/report/user/index.php?
    id=$id'>" . get_string('link_gradebook', 'report_followup') . "</a>";

113 // Print the actions indicator.
    echo '<b><center>' . get_string('participation', 'report_followup')
        . '</center></b>';
115
    $table2 = new html_table();
117 $table2->head = array('', get_string('title_result', '
        report_followup'), get_string('title_median', 'report_followup'),
         get_string('title_visual', 'report_followup'));
    $table2->data[] = array(get_string('consultation', 'report_followup'
        ), $consulted, median_indicators($id, 'consulted'), indicators1($
        consulted, median_indicators($id, 'consulted')));
119 $table2->data[] = array(get_string('contribution', 'report_followup'
        ), $contributed, median_indicators($id, 'contributed'),
        indicators1($contributed, median_indicators($id, 'contributed')))
        ;
    echo html_writer::table($table2);
121
    echo "<a href='http://localhost/moodle/report/participation/index.
    php?id=$id'>" . get_string('link_participation', 'report_followup') .
     "</a>";
123
    // Print the activity indicator.
125 echo '<b><center>' . get_string('progression', 'report_followup') .
        '</center></b>';

127 $table3 = new html_table();
    $table3->head = array('', get_string('title_result', '
        report_followup'), get_string('title_median', 'report_followup'),
         get_string('title_visual', 'report_followup'));
129 $table3->data[] = array(get_string('achieved', 'report_followup'), $
        completion, median_indicators($id, 'completion'), indicators1($
        completion, median_indicators($id, 'completion')));
```

```
    $table3->data[] = array(get_string('lated', 'report_followup'), $
        punctuality, median_indicators($id, 'punctuality'), indicators2($
        punctuality, median_indicators($id, 'punctuality')));
    echo html_writer::table($table3);

    echo "<a href='http://localhost/moodle/blocks/completionstatus/
    details.php?course=$id'>" . get_string('link_completion', '
    report_followup') . "</a>";
}

/*
 * TEACHER VIEW
*/

if (has_capability('report/followup:teacherview', $context)) {

    // Print the report title.
    echo $OUTPUT->heading(get_string('header', 'report_followup'));

    // Add a link to the administration page.
    echo "<center><a href='http://localhost/moodle/report/followup/admin
        .php?id=$id'>" . get_string('link_administration', '
        report_followup') . "</a></center>";

    // Get all the students of a course.
    $sql = 'SELECT u.id, u.firstname, u.lastname, u.username FROM
        mdl_role_assignments ra JOIN mdl_user u ON u.id = ra.userid JOIN
        mdl_role r ON r.id = ra.roleid JOIN mdl_context cxt ON cxt.id =
        ra.contextid JOIN mdl_course c ON c.id = cxt.instanceid WHERE ra.
        userid = u.id AND ra.contextid = cxt.id AND cxt.contextlevel =50
        AND cxt.instanceid = c.id AND  roleid = 5 AND c.id = ? ORDER BY
        lastname ASC';
    $users = $DB->get_records_sql($sql, array($id));

    // Prepare the three different chart.
    $chartparticipation = new \core\chart_pie();
    $chartprogression = new \core\chart_pie();
    $chartglobal = new \core\chart_pie();

    $serieparticipation = new \core\chart_series('Series participation',
         chart_serie_participation($users, $id));
    $serieprogression = new \core\chart_series('Series progression',
    chart_serie_progression($users, $id));
    $serieglobal = new \core\chart_series('Series global',
    chart_serie_global($users, $id));

    $chartparticipation->add_series($serieparticipation);
    $chartprogression->add_series($serieprogression);
    $chartglobal->add_series($serieglobal);

    $chartparticipation->set_title(get_string('resume_participation', '
    report_followup'));
    $chartprogression->set_title(get_string('resume_progression', '
        report_followup'));
    $chartglobal->set_title(get_string('resume_global', 'report_followup
        '));

    $chartparticipation->set_labels(['Green', 'Yellow', 'Red']);
```

```
    $chartprogression->set_labels(['Green', 'Yellow', 'Red']);
171 $chartglobal->set_labels(['Green', 'Yellow', 'Red']);

173 // Table for the chart.
    $tablechart = new html_table();
175 $tablechart->data[] = array($OUTPUT->render($chartparticipation), $
    OUTPUT->render($chartprogression), $OUTPUT->render($chartglobal))
    ;
    echo html_writer::table($tablechart);
177
    // Start the table.
179 $table = new html_table();
    $table->head = array('', get_string('score', 'report_followup'),
    get_string('participation', 'report_followup'), get_string('
    progression', 'report_followup'), get_string('global', '
    report_followup'));
181
    // Prints the data for the different students.
183 foreach ($users as $user) {

185     /*
         * VARIABLES AREA
187      */

189     // Get the grade for a particular student in a particular course
        .
        $grade = grade_get_course_grade($user->id, $id);
191
        /*
193      * TABLE PRINT AREA
         */
195
        // Print the table.
197     $sql = "SELECT id FROM mdl_user WHERE lastname = ? AND firstname
        = ? AND username = ?";
        $studentid = $DB->get_record_sql($sql, array($user->lastname, $
        user->firstname, $user->username));
199     $table->data[] = array("<a href='http://localhost/moodle/report/
    followup/infos.php?id=$id&amp;studentid=$studentid->id'>$user->
    lastname $user->firstname</a>", $grade->str_long_grade, indicators3(
    calculate_participation($user->id, $id)), indicators3(
    calculate_progression($user->id, $id)), indicators3(calculate_global(
    $user->id, $id)));
     }
201
     echo html_writer::table($table);
203 }

205 echo $OUTPUT->footer();
```

B.2 lib.php

```
1 <?php
  // This file is part of Moodle - http://moodle.org/
3 //
  // Moodle is free software: you can redistribute it and/or modify
```

```
5  // it under the terms of the GNU General Public License as published by
   // the Free Software Foundation, either version 3 of the License, or
7  // (at your option) any later version.
   //
9  // Moodle is distributed in the hope that it will be useful,
   // but WITHOUT ANY WARRANTY; without even the implied warranty of
11 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   // GNU General Public License for more details.
13 //
   // You should have received a copy of the GNU General Public License
15 // along with Moodle.  If not, see <http://www.gnu.org/licenses/>.

17 /**
    * This file contains functions used by the report.
19  *
    * @package   report_followup
21  * @copyright 2017 Bonesire William
    * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
23  */

25 defined('MOODLE_INTERNAL') || die();

27 require_once('/var/www/html/moodle/config.php');

29 /**
    * This function add the link in the course menu
31  *
    * @param array $navigation Give the navigation structure
33  * @param array $course Give the id of the course
    * @param array $context Give the context of the current course
35  */
   function report_followup_extend_navigation_course($navigation, $course,
      $context) {
37     if (has_capability('report/followup:view', $context)) {
           $url = new moodle_url('/report/followup/index.php', array('id'
              => $course->id));
39         $navigation->add(get_string('pluginname', 'report_followup'), $
      url, navigation_node::TYPE_SETTING,
           null, null, new pix_icon('i/report', ''));
41     }
   }
43
   /*
45  *
    * INDICATORS COMPUTATION
47  *
    */
49
   /**
51  * This function compute the indicators based on the log table
    *
53  * @param int $userid ID of the user
    * @param int $course ID of the current course
55  * @param string $indicator Named of the indicator we want to calculate
    */
57 function calculate_indicators_log($userid, $course, $indicator) {
       global $DB;
59
```

```
    if ($indicator == 'graded') {
        $param = array('graded');
    } else if ($indicator == 'consulted') {
        $param = array('downloaded', 'launched', 'printed', 'viewed');
    } else if ($indicator == 'contributed') {
        $param = array('added', 'assessed', 'created', 'evaluated', '
            submitted', 'uploaded');
    } else if ($indicator == 'punctuality') {
        $param = array('becomeoverdue');
    } else {
        return null;
    }

    list($insql, $params) = $DB->get_in_or_equal($param);

    $sql = "SELECT id FROM mdl_logstore_standard_log WHERE action $insql
    AND userid = $userid AND courseid = $course";

    $records = $DB->get_records_sql($sql, $params);

    return (count($records)) ;
}

/**
 * This function compute the fifth indicator based on completed activity
 *
 * @param int $userid ID of the user
 * @param int $course ID of the current course
 */
function calculate_completion($userid, $course) {
    global $DB;

    $sql = "SELECT * FROM mdl_course_modules_completion mc JOIN
    mdl_course_modules m ON mc.coursemoduleid = m.module WHERE mc.userid
    = $userid AND mc.completionstate = 1 AND m.course = $course";

    $records = $DB->get_records_sql($sql);

    return (count($records)) ;
}

/**
 * This function compute the participation indicator for teachers
 *
 * @param int $userid ID of the user
 * @param int $course ID of the current course
 */
function calculate_participation($userid, $course) {
    global $DB;

    // Get the coefficients define by the teacher.
    $sql = 'SELECT consulted, contributed FROM mdl_report_followup WHERE
        courseid = ?';
    if ($DB->record_exists_sql($sql, array($course))) {
        $query = 'SELECT consulted, contributed FROM mdl_report_followup
            WHERE courseid = ?';
        $sql = $DB->get_record_sql($query, array($course));

```

```
        $coef = array(
113         "consulted" => $sql->consulted,
            "contributed" => $sql->contributed);
115     } else {
            $coef = array(
117         "consulted" => 1,
            "contributed" => 1);
119     }

121     // Define the 2 indicators needed.
        if (median_indicators($course, 'consulted') == 0) {
123         $report3 = 0;
        } else {
125         $report3 = calculate_indicators_log($userid, $course, 'consulted
                ') / median_indicators($course, 'consulted');
        }
127
        if (median_indicators($course, 'contributed') == 0) {
129         $report4 = 0;
        } else {
131         $report4 = calculate_indicators_log($userid, $course, '
                contributed') / median_indicators($course, 'contributed');
        }
133
        // Don't take into account 0 report.
135     $array = array($report3, $report4);
        $count = 0;
137
        for ($i = 0; $i < 2; $i++) {
139         if ($array[$i] == 0) {
                $count = $count + 1;
141         }
        }
143
        if ($count != 2) {
145         // Min allow us to not be influenced by one indicator.
            return $indicator = ($coef['consulted'] * min($report3, 2) + $
                coef['contributed'] * min($report4, 2)) / (2 - $count);
147     } else {
            return 'Not Pertinent';
149     }
}
151
/**
153  * This function compute the progression indicator for teachers
     *
155  * @param int $userid ID of the user
     * @param int $course ID of the current course
157  */
function calculate_progression($userid, $course) {
159     global $DB;

161     // Get the coefficients define by the teacher.
        $sql = 'SELECT completion, punctuality FROM mdl_report_followup
            WHERE courseid = ?';
163     if ($DB->record_exists_sql($sql, array($course))) {
            $query = 'SELECT completion, punctuality FROM
                mdl_report_followup WHERE courseid = ?';
```

```
165         $sql = $DB->get_record_sql($query, array($course));

167         $coef = array(
            "completion" => $sql->completion,
169         "punctuality" => $sql->punctuality);
    } else {
171         $coef = array(
            "completion" => 1,
173         "punctuality" => 1);
    }
175
    // Define the 2 indicators needed.
177 if (median_indicators($course, 'completion') == 0) {
        $report5 = 0;
179 } else {
        $report5 = calculate_completion($userid, $course) /
            median_indicators($course, 'completion');
181 }

183 if (calculate_indicators_log($userid, $course, 'punctuality') == 0
        && median_indicators($course, 'punctuality') != 0) {
        $report6 = median_indicators($course, 'punctuality') / 0.8;
185 } else if (calculate_indicators_log($userid, $course, 'punctuality')
        != 0 && median_indicators($course, 'punctuality') == 0) {
        $report6 = 0.8 / calculate_indicators_log($userid, $course, '
            punctuality');
187 } else if (calculate_indicators_log($userid, $course, 'punctuality')
        == 0 && median_indicators($course, 'punctuality') == 0) {
        $report6 = 1;
189 } else {
        $report6 = median_indicators($course, 'punctuality') /
            calculate_indicators_log($userid, $course, 'punctuality');
191 }

193 // Don't take into account 0 report.
    $array = array($report5, $report6);
195 $count = 0;

197 for ($i = 0; $i < 2; $i++) {
        if ($array[$i] == 0) {
199         $count = $count + 1;
        }
201 }

203 // Min allow us to not be influenced by one indicator.
    return $indicator = ($coef['completion'] * min($report5, 2) + $coef
        ['punctuality'] * min($report6, 2)) / (2 - $count) ;
205 }

207 /**
 * This function compute the global indicator
209 *
 * @param int $userid ID of the user
211 * @param int $course ID of the current course
 */
213 function calculate_global($userid, $course) {
    global $DB;
215
```

```
    // Get the coefficients define by the teacher.
217 $sql = 'SELECT score, graded, consulted, contributed, completion,
        punctuality FROM mdl_report_followup WHERE courseid = ?';
    if ($DB->record_exists_sql($sql, array($course))) {
219     $query = 'SELECT score, graded, consulted, contributed,
            completion, punctuality FROM mdl_report_followup WHERE
            courseid = ?';
        $sql = $DB->get_record_sql($query, array($course));
221
        $coef = array(
223     "score" => $sql->score,
        "graded" => $sql->graded,
225     "consulted" => $sql->consulted,
        "contributed" => $sql->contributed,
227     "completion" => $sql->completion,
        "punctuality" => $sql->punctuality);
229 } else {
        $coef = array(
231     "score" => 1,
        "graded" => 1,
233     "consulted" => 1,
        "contributed" => 1,
235     "completion" => 1,
        "punctuality" => 1);
237 }

239 // Define the 6 rapports between the indicators and their
        corresponding medians.
    if (median_indicators($course, 'grade') == 0) {
241     $report1 = 0;
    } else {
243     $report1 = grade_get_course_grade($userid, $course)->
            str_long_grade / median_indicators($course, 'grade');
    }
245
    if (median_indicators($course, 'graded') == 0) {
247     $report2 = 0;
    } else {
249     $report2 = calculate_indicators_log($userid, $course, 'graded')
            / median_indicators($course, 'graded');
    }
251
    if (median_indicators($course, 'consulted') == 0) {
253     $report3 = 0;
    } else {
255     $report3 = calculate_indicators_log($userid, $course, 'consulted
            ') / median_indicators($course, 'consulted');
    }
257
    if (median_indicators($course, 'contributed') == 0) {
259     $report4 = 0;
    } else {
261     $report4 = calculate_indicators_log($userid, $course, '
            contributed') / median_indicators($course, 'contributed');
    }
263
    if (median_indicators($course, 'completion') == 0) {
265     $report5 = 0;
```

```
    } else {
267         $report5 = calculate_completion($userid, $course) /
                median_indicators($course, 'completion');
    }
269
    if (calculate_indicators_log($userid, $course, 'punctuality') == 0
        && median_indicators($course, 'punctuality') != 0) {
271         $report6 = median_indicators($course, 'punctuality') / 0.8;
    } else if (calculate_indicators_log($userid, $course, 'punctuality')
         != 0 && median_indicators($course, 'punctuality') == 0) {
273         $report6 = 0.8 / calculate_indicators_log($userid, $course, '
                punctuality');
    } else if (calculate_indicators_log($userid, $course, 'punctuality')
         == 0 && median_indicators($course, 'punctuality') == 0) {
275         $report6 = 1;
    } else {
277         $report6 = median_indicators($course, 'punctuality') /
                calculate_indicators_log($userid, $course, 'punctuality');
    }
279
    // Don't take into account 0 report.
281 $array = array($report1, $report2, $report3, $report4, $report5, $
        report6);
    $count = 0;
283
    for ($i = 0; $i < 6; $i++) {
285     if ($array[$i] == 0) {
            $count = $count + 1;
287     }
    }
289
    // Min allow us to not be influenced by one indicator.
291 return $indicator = ($coef['score'] * min($report1, 2) + $coef['
        graded'] * min($report2, 2) + $coef['consulted'] * min($report3,
        2) + $coef['contributed'] * min($report4, 2) + $coef['completion
        '] * min($report5, 2) + $coef['punctuality'] * min($report6, 2))
        / (6 - $count) ;
}
293
/*
295  *
 * MEDIAN COMPUTATION
297  *
 */
299
/**
301  * This function compute the median of the indicators
 *
303  * @param int $course ID of the current course
 * @param string $indicator Named of the indicator we want to calculate
305  */
function median_indicators($course, $indicator) {
307     global $DB;

309     // Get all the students of a course.
    $sql = 'SELECT␣u.id,␣u.firstname,␣u.lastname␣FROM␣
        mdl_role_assignments␣ra␣JOIN␣mdl_user␣u␣ON␣u.id␣=␣ra.userid␣JOIN␣
        mdl_role␣r␣ON␣r.id␣=␣ra.roleid␣JOIN␣mdl_context␣cxt␣ON␣cxt.id␣=␣
```

```
        ra.contextid␣JOIN␣mdl_course␣c␣ON␣c.id␣=␣cxt.instanceid␣WHERE␣ra.
        userid␣=␣u.id␣AND␣ra.contextid␣=␣cxt.id␣AND␣cxt.contextlevel␣=50␣
        AND␣cxt.instanceid␣=␣c.id␣AND␣␣roleid␣=␣5␣AND␣c.id=␣?';
311     $users = $DB->get_records_sql($sql, array($course));

313     $list = array();

315     if ($indicator == 'grade') {

317         foreach ($users as $user) {
                $sql = grade_get_course_grade($user->id, $course)->
                    str_long_grade;
319             $list[] = $sql;
            }
321
        } else if ($indicator == 'graded') {
323
            foreach ($users as $user) {
325             $sql = calculate_indicators_log($user->id, $course, $
                    indicator);
                $list[] = $sql;
327         }

329     } else if ($indicator == 'consulted') {

331         foreach ($users as $user) {
                $sql = calculate_indicators_log($user->id, $course, $
                    indicator);
333             $list[] = $sql;
            }
335
        } else if ($indicator == 'contributed') {
337
            foreach ($users as $user) {
339             $sql = calculate_indicators_log($user->id, $course, $
                    indicator);
                $list[] = $sql;
341         }

343     } else if ($indicator == 'completion') {

345         foreach ($users as $user) {
                $sql = calculate_completion($user->id, $course);
347             $list[] = $sql;
            }
349
        } else if ($indicator == 'punctuality') {
351
            foreach ($users as $user) {
353             $sql = calculate_indicators_log($user->id, $course, $
                    indicator);
                $list[] = $sql;
355         }

357     } else {
            return null;
359     }
```

```
361     // Sort the array.
        asort($list);
363
        $size = count($list);
365
        return $list[round($size / 2)];
367 }

369 /*
     *
371  * VISUAL INDICATORS
     *
373  */

375 /**
     * This function returns the visual indicator for the 5 first indicators
377  *
     * @param int $indstud Value of the current student's indicator
379  * @param int $indmedian Value of the median of the current indicator
     */
381 function indicators1($indstud, $indmedian) {

383     // Avoid zero division.
        if ($indmedian == 0) {
385         return 'Not pertinent';
        } else {
387         $indicator = $indstud / $indmedian;

389         if ($indicator >= 0.9) {
                return '<img src="images/green.png" heigth="25" width="25">'
                    ;
391         } else if ($indicator >= 0.5 && $indicator < 0.9) {
                return '<img src="images/yellow.png" heigth="25" width="25">
                    ';
393         } else {
                return '<img src="images/red.png" heigth="25" width="25">';
395         }
        }
397 }

399 /**
     * This function returns the visual indicator for the sixth indicator
401  *
     * @param int $indstud Value of the current student's indicator
403  * @param int $indmedian Value of the median of the current indicator
     */
405 function indicators2($indstud, $indmedian) {

407     // These two lines are there to prevent the zero division and make
            the indicator pertinent in regards to 0 becomeoverdue median.
        if ($indstud == 0) {
409         $indstud = 0.8;
        }
411     if ($indmedian == 0) {
            $indmedian = 0.8;
413     }

415     $indicator = $indmedian / $indstud;
```

```
417     if ($indicator >= 0.9) {
            return '<img src="images/green.png" heigth="25" width="25">';
419     } else if ($indicator >= 0.5 && $indicator < 0.9) {
            return '<img src="images/yellow.png" heigth="25" width="25">';
421     } else {
            return '<img src="images/red.png" heigth="25" width="25">';
423     }
}
425
/**
427  * This function returns the visual indicator for the global indicators
     *
429  * @param int $indicator Value of the current student's indicator
     */
431 function indicators3($indicator) {

433     if ($indicator >= 0.9) {
            return '<img src="images/green.png" heigth="25" width="25">';
435     } else if ($indicator >= 0.5 && $indicator < 0.9) {
            return '<img src="images/yellow.png" heigth="25" width="25">';
437     } else {
            return '<img src="images/red.png" heigth="25" width="25">';
439     }
}
441
/*
443  *
     * CHART
445  *
     */
447
/*
449  * Serie for participation chart
     */
451
/**
453  * This function returns the serie used to create the chart for
         participation
     *
455  * @param array $users List of all the users registred in the course
     * @param int   $courseid Id of the current course
457  */
function chart_serie_participation($users, $courseid) {
459
    $a = 0;
461 $b = 0;
    $c = 0;
463
    foreach ($users as $user) {
465     $indstud = calculate_participation($user->id, $courseid);

467     if ($indstud >= 0.9) {
            $a = $a + 1;
469     } else if ($indstud >= 0.5 && $indstud < 0.9) {
            $b = $b + 1;
471     } else {
            $c = $c + 1;
```

```
473             }
        }
475
        return array($a, $b, $c) ;
477 }

479 /**
     * This function returns the serie used to create the chart for
         progression
481  *
     * @param array $users List of all the users registred in the course
483  * @param int   $courseid Id of the current course
     */
485 function chart_serie_progression($users, $courseid) {

487     $a = 0;
        $b = 0;
489     $c = 0;

491     foreach ($users as $user) {
            $indstud = calculate_progression($user->id, $courseid);
493
            if ($indstud >= 0.9) {
495             $a = $a + 1;
            } else if ($indstud >= 0.5 && $indstud < 0.9) {
497             $b = $b + 1;
            } else {
499             $c = $c + 1;
            }
501     }

503     return array($a, $b, $c) ;
    }
505
    /**
507  * This function returns the serie used to create the chart for global
     *
509  * @param array $users List of all the users registred in the course
     * @param int   $courseid Id of the current course
511  */
    function chart_serie_global($users, $courseid) {
513
        $a = 0;
515     $b = 0;
        $c = 0;
517
        foreach ($users as $user) {
519         $indstud = calculate_global($user->id, $courseid);

521         if ($indstud >= 0.9) {
                $a = $a + 1;
523         } else if ($indstud >= 0.5 && $indstud < 0.9) {
                $b = $b + 1;
525         } else {
                $c = $c + 1;
527         }
        }
529
```

```
    return array($a, $b, $c);
}
```

B.3 version.php

```
<?php
// This file is part of Moodle - http://moodle.org/
//
// Moodle is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Moodle is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Moodle.  If not, see <http://www.gnu.org/licenses/>.

/**
 * Version info
 *
 * @package   report_followup
 * @copyright 2017 Bonesire William
 * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */

defined('MOODLE_INTERNAL') || die;

$plugin->version   = 2017041102;    // The current plugin version (Date:
    YYYYMMDDXX)
$plugin->requires  = 2016051900;    // Requires this Moodle version
$plugin->component = 'report_followup';  // Full name of the plugin (
   used for diagnostics).
```

B.4 admin.php

```
<?php
// This file is part of Moodle - http://moodle.org/
//
// Moodle is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Moodle is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Moodle.  If not, see <http://www.gnu.org/licenses/>.
```

```php

/**
 * Administration page
 *
 * @package   report_followup
 * @copyright 2017 Bonesire William
 * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */

require(dirname(__FILE__).'/../../config.php');
require_once($CFG->dirroot.'/course/lib.php');
require_once($CFG->libdir.'/adminlib.php');
require_once($CFG->dirroot.'/grade/querylib.php');
require_once($CFG->libdir.'/gradelib.php');
require_once($CFG->dirroot.'/report/followup/lib.php');
require_once($CFG->dirroot.'/report/followup/form.php');

$id = required_param('id', PARAM_INT); // Course ID.
$user = $USER->id; // User ID.

$params = array('id' => $id);
$course = $DB->get_record('course', $params, '*', MUST_EXIST);
require_login($course);
$context = context_course::instance($course->id);
$currentuser = optional_param('user', null, PARAM_INT);

$urlparams = array('id' => $id);
$navurl = new moodle_url('/report/followup/admin.php', $urlparams);
$urlparams['user'] = $currentuser;
$url = new moodle_url('/report/followup/admin.php', $urlparams);

$title = get_string('pluginname', 'report_followup');
$coursename = format_string($course->fullname, true, array('context' =>
    $context));

$PAGE->navigation->override_active_url($navurl);
$PAGE->set_url($url);
$PAGE->set_title($title);
$PAGE->set_heading($coursename);
$PAGE->set_pagelayout('report');

echo $OUTPUT->header() ;

// Print the title.
echo $OUTPUT->heading(get_string('header_admin', 'report_followup'));

// Link to previous page.
echo "<center><a href='http://localhost/moodle/report/followup/index.php
    ?id=$id'>" . get_string('back', 'report_followup') . "</a></center>";

// Instantiate form.
$mform = new form();
$formdata = array('id' => $id);
$mform->set_data($formdata);

// Form processing and displaying is done here.
if ($mform->is_cancelled()) {
```

```
71      // Handle form cancel operation, if cancel button is present on form
            .
        redirect($url);
73  } else if ($fromform = $mform->get_data()) {
        // In this case you process validated data. $mform->get_data()
            returns data posted in form.
75      $data = $mform->get_data();

77      // Check if the course is already in the database.
        $query = "SELECT␣id␣FROM␣mdl_report_followup␣WHERE␣courseid␣=␣?";
79      $sql = $DB->get_record_sql($query, array($id));

81      $record = new stdClass();
        $record->id = $sql->id;
83      $record->courseid = $id;
        $record->score = $data->score;
85      $record->graded = $data->graded;
        $record->consulted = $data->consulted;
87      $record->contributed = $data->contributed;
        $record->completion = $data->completion;
89      $record->punctuality = $data->punctuality;

91      if ($sql) {
            $DB->update_record('report_followup', $record);
93      } else {
            $DB->insert_record('report_followup', $record);
95      }

97      redirect($url);
    } else {
99      $mform->display();
    }
101 echo $OUTPUT->footer();
```

B.5 form.php

```
1   <?php
    // This file is part of Moodle - http://moodle.org/
3   //
    // Moodle is free software: you can redistribute it and/or modify
5   // it under the terms of the GNU General Public License as published by
    // the Free Software Foundation, either version 3 of the License, or
7   // (at your option) any later version.
    //
9   // Moodle is distributed in the hope that it will be useful,
    // but WITHOUT ANY WARRANTY; without even the implied warranty of
11  // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    // GNU General Public License for more details.
13  //
    // You should have received a copy of the GNU General Public License
15  // along with Moodle.  If not, see <http://www.gnu.org/licenses/>.

17  /**
     * Set up the form
19   *
     * @package   report_followup
```

```
21  * @copyright 2017 Bonesire William
    * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
23  */

25 // Moodleform is defined in formslib.php.
   require_once($CFG->libdir.'/formslib.php') ;
27
   /**
29  * Allow to define a new form
    *
31  * @package    report_followup
    * @copyright  2017 Bonesire William
33  * @license    http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
    */
35 class form extends moodleform {

37      /**
         * This function defines the form
39       */
        public function definition() {
41          global $CFG, $DB;

43          // Present by default.
            $mform = $this->_form;
45
            // Hidden parameter to transmit mandatory parameter.
47          $mform->addElement('hidden', 'id');
            $mform->setType('id', PARAM_INT);
49          $id = required_param('id', PARAM_INT);

51          // Default values.
            $sql = "SELECT score, graded, consulted, contributed, completion
                , punctuality FROM mdl_report_followup WHERE courseid = ?";
53          $default = $DB->get_record_sql($sql, array($id));
            // Grade indicator.
55          $mform->addElement('header', 'description_grade', get_string('
                grade_form', 'report_followup'));

57          $mform->addElement('text', 'score', get_string('score_coef', '
                report_followup')); // Add elements to your form
            $mform->setType('score', PARAM_NOTAGS); // Set type of element
59          $mform->setDefault('score', $default->score); // Default value.

61          $mform->addElement('text', 'graded', get_string('graded_coef', '
                report_followup'));
            $mform->setType('graded', PARAM_NOTAGS);
63          $mform->setDefault('graded', $default->graded);

65          // Participation indicator.
            $mform->addElement('header', 'description_participation',
                get_string('participation_form', 'report_followup'));
67
            $mform->addElement('text', 'consulted', get_string('
                consultation_coef', 'report_followup'));
69          $mform->setType('consulted', PARAM_NOTAGS);
            $mform->setDefault('consulted', $default->consulted);
71
```

```
        $mform->addElement('text', 'contributed', get_string('
    contribution_coef', 'report_followup'));
73      $mform->setType('contributed', PARAM_NOTAGS);
        $mform->setDefault('contributed', $default->contributed);
75
        // Progression indicator.
77      $mform->addElement('header', 'description_progression',
            get_string('progression_form', 'report_followup'));

79      $mform->addElement('text', 'completion', get_string('
            achieved_coef', 'report_followup'));
        $mform->setType('completion', PARAM_NOTAGS);
81      $mform->setDefault('completion', $default->completion);

83      $mform->addElement('text', 'punctuality', get_string('
    punctuality_coef', 'report_followup'));
        $mform->setType('punctuality', PARAM_NOTAGS);
85      $mform->setDefault('punctuality', $default->punctuality);

87      // Button.
        $buttonarray = array();
89      $buttonarray[] = &$mform->createElement('submit', 'submitbutton'
            , get_string('savechanges'));
        $mform->addGroup($buttonarray, 'buttonar', '', array('␣'), false
            );
91      $mform->closeHeaderBefore('buttonar');

93    }
}
```

B.6 infos.php

```
<?php
2 // This file is part of Moodle - http://moodle.org/
  //
4 // Moodle is free software: you can redistribute it and/or modify
  // it under the terms of the GNU General Public License as published by
6 // the Free Software Foundation, either version 3 of the License, or
  // (at your option) any later version.
8 //
  // Moodle is distributed in the hope that it will be useful,
10 // but WITHOUT ANY WARRANTY; without even the implied warranty of
  // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12 // GNU General Public License for more details.
  //
14 // You should have received a copy of the GNU General Public License
  // along with Moodle.  If not, see <http://www.gnu.org/licenses/>.
16
  /**
18  * This page is for information about one special student
    *
20  * @package   report_followup
    * @copyright 2017 Bonesire William
22  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
    */
24
```

```
require(dirname(__FILE__).'/../../config.php');
require_once($CFG->dirroot.'/course/lib.php');
require_once($CFG->libdir.'/adminlib.php');
require_once($CFG->dirroot.'/grade/querylib.php');
require_once($CFG->libdir.'/gradelib.php');
require_once($CFG->dirroot.'/report/followup/lib.php');

$id = required_param('id', PARAM_INT); // Course ID.
$studentid = required_param('studentid', PARAM_INT);

$params = array('id' => $id);
$course = $DB->get_record('course', $params, '*', MUST_EXIST);
require_login($course);
$context = context_course::instance($course->id);
$currentuser = optional_param('user', null, PARAM_INT);

$urlparams = array('id' => $id);
$navurl = new moodle_url('/report/followup/infos.php', $urlparams);
$urlparams['user'] = $currentuser;
$url = new moodle_url('/report/followup/infos.php', $urlparams);

$title = get_string('pluginname', 'report_followup');
$coursename = format_string($course->fullname, true, array('context' =>
    $context));

$PAGE->navigation->override_active_url($navurl);
$PAGE->set_url($url);
$PAGE->set_title($title);
$PAGE->set_heading($coursename);
$PAGE->set_pagelayout('report');

echo $OUTPUT->header();

if (has_capability('report/followup:teacherview', $context)) {

    /*
     * VARIABLES AREA
     */

    // Get the student information.
    $sql = "SELECT lastname,firstname FROM mdl_user WHERE id=?";
    $student = $DB->get_record_sql($sql, array($studentid));

    // Get the grade for a particular student in a particular course.
    $grade = grade_get_course_grade($studentid, $id);

    // Get the graded activity.
    $graded = calculate_indicators_log($studentid, $id, 'graded');

    // Get the consulted actions.
    $consulted = calculate_indicators_log($studentid, $id, 'consulted');

    // Get the contributed actions.
    $contributed = calculate_indicators_log($studentid, $id, '
        contributed');

    // Get the completions status.
    $completion = calculate_completion($studentid, $id);
```

```
82      // Get the punctuality.
        $punctuality = calculate_indicators_log($studentid, $id, '
            punctuality');
84
        // Set the threshold for the grade.
86      if ($grade->grade / $grade->item->grademax < 0.5) {
            $gradedisplay = '<img␣src="images/red.png"␣heigth="25"␣width
                ="25">';
88      } else {
            $gradedisplay = indicators1($grade->str_long_grade,
                median_indicators($id, 'grade'));
90      }

92      /*
         * TABLE PRINT AREA
94       */

96      // Print the report title.
        echo $OUTPUT->heading(get_string('header', 'report_followup') . '␣'
            . $student->lastname . '␣' . $student->firstname);
98
        // Back to previous page
100     echo "<center><a␣href='http://localhost/moodle/report/followup/index
            .php?id=$id'>" . get_string('back', 'report_followup') . "</a></
            center>";

102     // Print the global indicator (color based).
        $image = indicators3(calculate_global($studentid, $id));
104     echo '<b><center>␣' . get_string('global', 'report_followup') . '␣:␣
    ' . $image . '</center></b>';

106     // Print the notes indicator.
        echo '<b><center>' . get_string('grades', 'report_followup') . '</
            center></b>';
108
        $table1 = new html_table();
110     $table1->head = array('', get_string('title_result', '
            report_followup'), get_string('title_median', 'report_followup'),
             get_string('title_visual', 'report_followup'));
        $table1->data[] = array(get_string('score', 'report_followup'), $
            grade->str_long_grade, median_indicators($id, 'grade'), $
            gradedisplay);
112     $table1->data[] = array(get_string('graded', 'report_followup'), $
    graded, median_indicators($id, 'graded'), indicators1($graded,
    median_indicators($id, 'graded')));
        echo html_writer::table($table1);
114
        // Print the actions indicator.
116     echo '<b><center>' . get_string('participation', 'report_followup')
            . '</center></b>';

118     $table2 = new html_table();
        $table2->head = array('', get_string('title_result', '
            report_followup'), get_string('title_median', 'report_followup'),
             get_string('title_visual', 'report_followup'));
120     $table2->data[] = array(get_string('consultation', 'report_followup'
            ), $consulted, median_indicators($id, 'consulted'), indicators1($
```

```
        consulted, median_indicators($id, 'consulted')));
    $table2->data[] = array(get_string('contribution', 'report_followup'
        ), $contributed, median_indicators($id, 'contributed'),
        indicators1($contributed, median_indicators($id, 'contributed')))
        ;
    echo html_writer::table($table2);

    // Print the activity indicator.
    echo '<b><center>' . get_string('progression', 'report_followup') .
        '</center></b>';

    $table3 = new html_table();
    $table3->head = array('', get_string('title_result', '
        report_followup'), get_string('title_median', 'report_followup'),
         get_string('title_visual', 'report_followup'));
    $table3->data[] = array(get_string('achieved', 'report_followup'), $
        completion, median_indicators($id, 'completion'), indicators1($
        completion, median_indicators($id, 'completion')));
    $table3->data[] = array(get_string('lated', 'report_followup'), $
        punctuality, median_indicators($id, 'completion'), indicators2($
        punctuality, median_indicators($id, 'completion')));
    echo html_writer::table($table3);

}

echo $OUTPUT->footer();
```

B.7 lang/en/report_followup.php

```
<?php
// This file is part of Moodle - http://moodle.org/
//
// Moodle is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Moodle is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Moodle.  If not, see <http://www.gnu.org/licenses/>.

/**
 * Lang strings
 *
 * @package   report_followup
 * @copyright 2017 Bonesire William
 * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */

$string['followup:view'] = 'View␣Followup␣report';
$string['reportsettings'] = 'Report␣settings';
$string['intro'] = 'Select␣the␣course␣from␣which␣you␣want␣some␣infos';
```

```
$string['eventreportviewed'] = 'Report viewed';
$string['getreport'] = 'Get the report';
$string['selectcourse'] = 'Select a course';
$string['reportfornone'] = 'You have selected no course';
$string['reportforcourse'] = 'Report for course \'{$a}\'';

$string['achieved'] = 'Achieved activity';
$string['achieved_coef'] = 'Achieved activity coefficient';
$string['back'] = 'Back to the general report';
$string['consultation'] = 'Consultation\'s actions';
$string['consultation_coef'] = 'Consultation visit coefficient';
$string['contribution'] = 'Contribution\'s actions';
$string['contribution_coef'] = 'Contribution visit coefficient';
$string['global'] = 'Global indicator';
$string['grade_form'] = 'Modification of the grade coefficients';
$string['graded'] = 'Graded Activity';
$string['graded_coef'] = 'Graded activity coefficient';
$string['grades'] = 'Grades';
$string['header'] = 'Followup report :';
$string['header_admin'] = 'Administration page';
$string['heading'] = 'Moodle : Followup Report';
$string['lated'] = 'Lated activity';
$string['link_administration'] = 'Link towards the administration page';
$string['link_completion'] = 'Link towards the completion progress
    status';
$string['link_gradebook'] = 'Link towards the gradebook';
$string['link_participation'] = 'Link towards the participation report';
$string['participation'] = 'Participation';
$string['participation_form'] = 'Modification of the participation
    coefficients';
$string['pluginname'] = 'Followup';
$string['progression'] = 'Progression';
$string['progression_form'] = 'Modification of the progression
    coefficients';
$string['punctuality_coef'] = 'Punctuality coefficient';
$string['report'] = 'Report';
$string['resume_global'] = 'Resume of the global indicator';
$string['resume_participation'] = 'Resume of the participation indicator
    ';
$string['resume_progression'] = 'Resume of the progression indicator';
$string['score'] = 'Total score';
$string['score_coef'] = 'Score coefficient';
$string['title_median'] = 'Median of the group';
$string['title_result'] = 'Student\'s Result';
$string['title_visual'] = 'Visual indicator';
```

B.8 db/access.php

```
<?php
// This file is part of Moodle - http://moodle.org/
//
// Moodle is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
```

```
// Moodle is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Moodle.  If not, see <http://www.gnu.org/licenses/>.

/**
 * Capabilities
 *
 * @package    report_followup
 * @copyright  2017 Bonesire William
 * @license    http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */

defined('MOODLE_INTERNAL') || die();

$capabilities = array(

    'report/followup:view' => array(
        'captype' => 'read',
        'contextlevel' => CONTEXT_COURSE,
        'archetypes' => array(
            'student' => CAP_ALLOW,
            'teacher' => CAP_ALLOW,
            'editingteacher' => CAP_ALLOW,
            'manager' => CAP_ALLOW
        ),
    ),
    'report/followup:teacherview' => array(
        'captype' => 'read',
        'contextlevel' => CONTEXT_COURSE,
        'archetypes' => array(
            'teacher' => CAP_ALLOW,
            'editingteacher' => CAP_ALLOW,
            'manager' => CAP_ALLOW
        ),
    )
);
```

B.9 db/install.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<XMLDB PATH="report/followup/db" VERSION="20170321" COMMENT="XMLDB file
    for Moodle report/followup"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../../../lib/xmldb/xmldb.xsd"
>
  <TABLES>
    <TABLE NAME="report_followup" COMMENT="Table for the coefficients">
      <FIELDS>
        <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE=
            "true"/>
        <FIELD NAME="courseid" TYPE="int" LENGTH="20" NOTNULL="true"
            SEQUENCE="false"/>
```

```
        <FIELD NAME="score" TYPE="int" LENGTH="20" NOTNULL="false"
           SEQUENCE="false"/>
        <FIELD NAME="graded" TYPE="int" LENGTH="20" NOTNULL="false"
           SEQUENCE="false"/>
        <FIELD NAME="consulted" TYPE="int" LENGTH="20" NOTNULL="false"
           SEQUENCE="false"/>
        <FIELD NAME="contributed" TYPE="int" LENGTH="20" NOTNULL="false"
            SEQUENCE="false"/>
        <FIELD NAME="completion" TYPE="int" LENGTH="20" NOTNULL="false"
           SEQUENCE="false"/>
        <FIELD NAME="punctuality" TYPE="int" LENGTH="20" NOTNULL="false"
            SEQUENCE="false"/>
      </FIELDS>
      <KEYS>
        <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
      </KEYS>
    </TABLE>
  </TABLES>
</XMLDB>
```

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**