Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное
учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий
Кафедра Компьютерные интеллектуальные технологии

«Допустить к защите»
Зав. каф. КИТ, к.т.н.
_____ А.В. Речинский
«____» _____ 20___ г.

ДИПЛОМНЫЙ ПРОЕКТ СПЕЦИАЛИСТА

АВТОМАТИЗАЦИЯ ОРГАНИЗАЦИОННОЙ РАБОТЫ В СИСТЕМЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ MOODLE

специальность: 010503 «Математическое обеспечение и администрирование информационных систем»

Выполнил(а):
Ветров Даниил Юрьевич
Подпись_____

Руководитель:
доцент каф. КИТ ИКНТ, к.т.н.,
Щукин Александр Валентинович
Подпись_____

Рецензент:
директор по разработке ПО
ООО «ГЕТ бизнес консалтинг»
Трофимчук Константин Петрович
Подпись_____

Санкт-Петербург
2015

СОДЕРЖАНИЕ

Реферат

дипломной работы специалиста на тему

«Автоматизация организационной работы в системе дистанционного обучения Moodle»

Название на англ.:

Automation of organizational work in distance education system Moodle

Работа содержит: стр. 80, ил. 8, табл. 1, библ.: 12 названий.

Ключевые слова: автоматизация организационной деятельности, система дистанционного обучения, Moodle, веб-разработка, PHP, модуль.

Тема дипломного проекта относится к области дистанционного обучения. Была реализована платформа, позволяющая автоматизировать организационную деятельность деканата для создания аналитических отчётов и массовых действий над группами студентов. На базе этой платформы была реализована возможность генерирования отчёта в виде ежемесячной аттестации и система оповещения студентов о неудовлетворительной успеваемости по результатам аттестаций.

Работа имеет логическую структуру, состоящую из вводного раздела, пяти основных глав и заключения. В водном разделе описана актуальность работы. В первой главе поставлены цели и задачи. Во второй главе изучаются исходные данные в виде бизнес-процессов сотрудников деканата и устройства LMS Moodle, так же поставлены функциональные требования к проекту. В третей главе производится обзор возможных методов разработки приложений, которые взаимодействуют с Moodle, и на основе этих рассуждений строится архитектура будущего приложения. В четвёртой и пятой главах рассматриваются разработка и тестирования проекта. В заключительной части сформулированы выводы по работе. В приложении приведён исходный код получившейся программы.

Введение

Автоматизация организационной работы в обучении – востребованная функция, которая позволяет уменьшить участие сотрудников деканата и сократить их время и усилия, потраченные на обработку и проверку данных.

Moodle – система управления обучением, которая используется для управления учебными материалами с обеспечением совместного доступа. Работа в дистанционной системе подразумевает выполнение функциональных действий пользователей разных ролей: администратора, преподавателя, студента, сотрудника деканата. Деятельность сотрудника деканата характеризуется тем, что она подразумевает работу с целыми группами студентов и курсов и направлена как на получение аналитических отчётов об успеваемости, посещаемости и прочих метриках для управления учебным процессом, так и на групповые операции: оповещение, отчисление и т.д.

Однако в самом Moodle средств, помогающих в автоматизации именно организационной работы, присутствует мало, а с ростом количества студентов и курсов организационная работа усложняется. Поэтому существует крайняя необходимость в автоматизации организационного процесса.

1. Цели и задачи

Цель работы – автоматизация бизнес-процессов сотрудников деканата в части выполнения действий по управлению учебным процессом:

- консолидация данных о результатах аттестации по различным учебным дисциплинам;
- подготовка аналитических отчетов по академической успеваемости;
- выполнение массовых операций над группами студентов;

Необходимые условия реализации:

- интеграция разрабатываемой бизнес-логики в систему дистанционного обучения Moodle;
- выполнение операций со студентами первого высшего образования;

Исходными данными для разрабатываемого проекта:

- сведения о ведении учета академической успеваемости и аттестации студентов;
- организация учебного процесса в рамках дистанционной системы dl.spbstu.ru;
- техническая документация по LMS Moodle;

Для достижения цели, было необходимо выполнить следующие задачи:

- анализ бизнес-процессов сотрудников деканата;
- разработка концепции веб-приложения;
- автоматизация централизованного сбора данных;
- автоматизация составления отчётов;
- функциональное и нагрузочное тестирование веб-приложения;

2. Обзор исходных данных

2.1. Система дистанционного обучения Moodle

Moodle является аббревиатурой от англ. Modular Object-Oriented Dynamic Learning Environment (модульная объектно-ориентированная динамическая обучающая среда), и даёт возможность создавать сайты для онлайн-обучения. Moodle относится к свободно распространяемому программному обеспечению (по Стандартной общественной лицензии GNU, созданной в рамках проекта по свободному распространению программного обеспечения). В основном это означает, что Moodle охраняется авторскими правами, но имеются дополнительные права. Разрешается копировать, использовать и модифицировать Moodle при условии, что вы согласны [4]:

- предоставлять ваши дополнения другим;
- не модифицировать и не удалять оригинальную лицензию и авторские права;
- а также применять эту же лицензию к любым вторичным работам;

Moodle может быть инсталлирована на любом компьютере или сервере, который может запустить веб-сервер, PHP и может поддерживать базы данных типа SQL (например, MySQL). Он может быть запущен под операционными системами Windows и Mac и многих разновидностях Linux. [9]

2.1.1. История Moodle

Разработка Moodle началась в 1999 году, однако Moodle 1.0 была выпущена в августе 2002 года. Пользователи обсуждали Moodle на форуме, переводили Moodle на разные языки и создавали темы. Год спустя, был выпущен первый предлагаемый модуль (Семинар).

Moodle быстро росла: впервые в 2004 году в Оксфорде был проведены академические обсуждения Moodle, и компании стали становиться партнёрами Moodle.

С улучшенной документацией и новой сертификацией Moodle зарекомендовала себя к 2007 году как ведущая и «Отмеченная наградами система управления обучением с открытым исходным кодом». От 1000 зарегистрированных Сайтов в 2004 году она дошла до полумиллиона пользователей в 2008 году и более миллиона пользователей в 2010 году, с более чем 50 партнёрами Moodle. Её репозиторий переводов «AMOC» поддерживает более 100 языков. Долгожданная «Moodle 2.0» вышла в ноябре 2010 года и теперь, регулярные выпуски приносят расширенные средства каждые шесть месяцев. В настоящее время всё сфокусировано на мобильной технологии: «Официальное приложение HTML5» было выпущена в 2013 году и версия «Moodle 2.5», включает в себя настраиваемые темы клиентов, которые подходят для экранов всех размеров. [6]

2.1.2. Инструментарий Moodle

Сердце Moodle – это курсы, которые содержат интерактивные действия и ресурсы. Существует более 20 типов возможных действий (форумы, глоссарии, вики, задания, экзамены, варианты ответов (выборы), базы данных и т.д.), и каждое может использоваться пользователем многократно.

Основная мощь этой основанной на интерактивных действиях модели обучения связана с комбинированием интерактивных действий в последовательности и группы, которые могут помочь вам вести участников по путям обучения. Таким образом, каждое интерактивное действие может основываться на результатах предыдущих интерактивных действий.

Существует ряд других инструментов, облегчающих сбор обучающихся общин, в том числе блоги, обмен сообщениями, перечни участников и т.д., а также другие полезные средства, подобные оцениванию, отчётам, интеграции с

другими системами и т.д. Поэтому его можно рассматривать как набор инструментов, с помощью которых можно просто и естественно начать, а затем продвигаться с ускорением ко всё более эффективному учебному процессу.

2.1.3. Архитектура Moodle

Moodle является платформой обучения во всём мире, которая поддерживает открытые стандарты, и по своему замыслу и построению способна к многостороннему управлению и интеграции внешних приложений и информации в единой платформе.

Moodle позволяет создавать модули подобных типов:

- элементы курса;
- отчеты администратора;
- типы заданий;
- плагины аутентификации;
- блоки;
- форматы курсов;
- отчеты по курсам;
- плагины подписки на курсы;
- фильтры;
- отчеты по оценкам;
- форматы экспорта оценок;
- форматы импорта оценок;
- портфолио;
- типы вопросов в тестах;
- форматы импорта/экспорта тестов;
- отчеты по тестам;
- хранилища файлов;
- типы ресурсов;

– плагины поиска;
– и ещё 30 разных типов;

Архитектура Moodle типична для приложений подобного масштаба и назначения и состоит из 3 слоёв: слоя пользовательского интерфейса (UI), набора библиотек для работы (Libraries) и библиотеки для работы с базой данных и файлами, которые по сути являются расширениями php (DB libs, File libs) (рис. 2.1)[10]

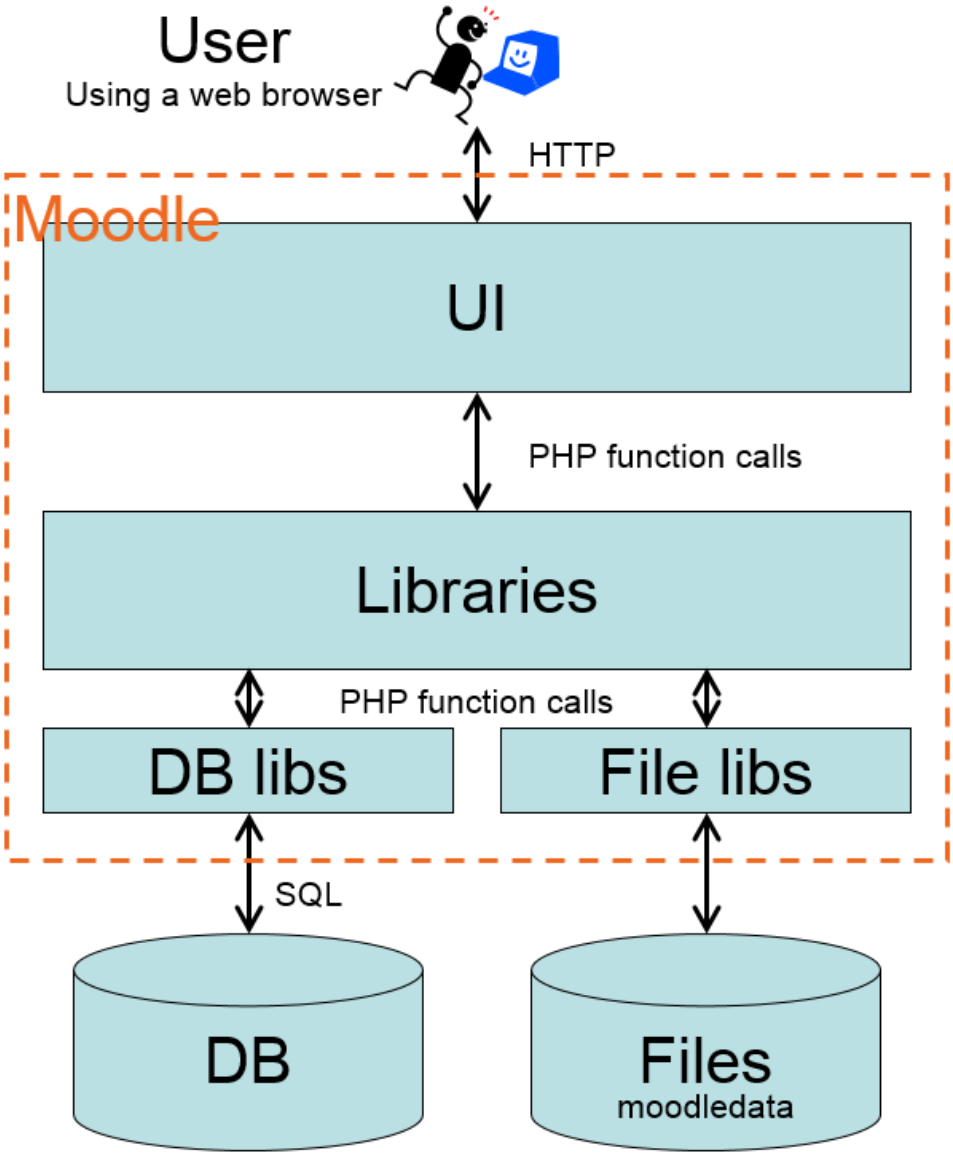

Рис. 2.1. **Архитектура Moodle**

2.2. Анализ бизнес-процессов сотрудников деканата

Деканат - административно-учебное управление факультета в высшем учебном заведении. Деканат выполняет функции координации и административного обеспечения учебного процесса, ведения делопроизводства.

Основные функции деканата:

- организация учебно-методической и воспитательной работы;
- организация по выполнению образовательно-профессиональных программ по направлениям и специальностям;
- учет студентов и их успеваемость;
- контроль за состоянием помещений и имущества, находящихся в распоряжении деканата;
- делопроизводство и документооборот на факультете;
- проведение мероприятий по обеспечению здоровья и безопасности жизни студентов и сотрудников факультета;

Функция учёта успеваемости студентов является востребованной: в данный момент существует 10 институтов и 981 групп, подсчёт успеваемости которых надо производить каждый месяц. Процесс аттестации устроен следующим образом:

1. Сотрудник деканата передаёт ответственному в группе студентов аттестационный лист. Аттестационный лист представляет из себя таблицу, со списком предметов и списком студентов, где каждому студенту необходимо проставить оценку по успеваемости за заданный период. Обычно период составляет месяц.
2. Преподаватель при помощи ответственного студента выставляет каждому учащемуся оценку за заданный период.

3. После того, как лист будет полностью заполнен, студент относит лист обратно в деканат для учёта успеваемости.
4. По итогам аттестации выносится выговор студентам, у которых 3 и более неаттестованных предмета за месяц. При повторении ситуации выносится строгий выговор. При 3-ем повторении – студент может быть представлен к отчислению.

Из описания процесса видно, что процесс очень зависим от организованности студента и наличия в необходимый момент преподавателя. Как и любой процесс, в котором участвую много людей, в нем присутствует человеческий фактор, а значит цель – анализ успеваемости – может быть не достигнута, что может повлиять на выполнение других функций деканатом.

Для иллюстрации процесса я построил диаграмму использования (рис. 2.2). В жёлтых кругах обозначено действие, действия связаны прямыми линиями с теми, кто учувствует в их исполнении, пунктиром показана зависимость действий.

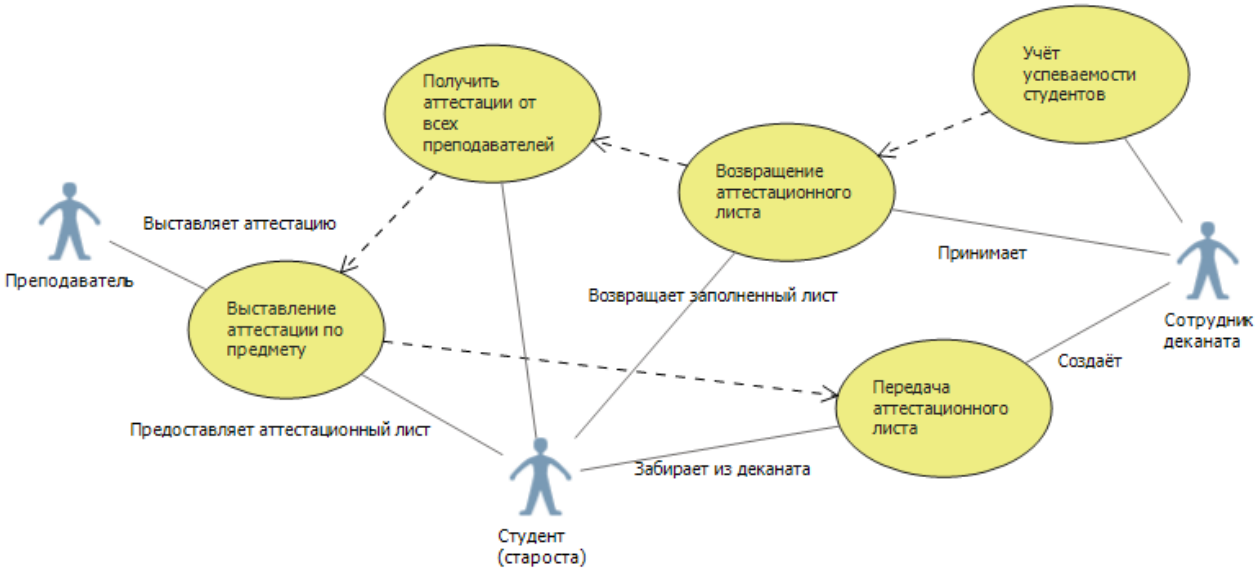

Рис 2.2 **Диаграмма использования, поясняющая процесс аттестации**

Мною был разработан альтернативный вариант аттестации с использованием системы Moodle. Так как сейчас и так существует большое

количество предметов, где в образовательном процессе уже используется Moodle, то можно учёт аттестации и анализ результатов переложить на него.

Для этого преподавателю необходимо настроить свой курс таким образом, чтобы каждое задание, которое будет учитываться при выставлении аттестации, было помечено специальной меткой – категорией оценки. Так приложение, ответственное за сбор данных и их анализ, будет получить информацию по результатам аттестаций автоматически. А по результатам аттестаций оно сможет оповещать студентов о плохой успеваемости.

Я построил диаграмму использования для этого метода (рис. 2.3.).

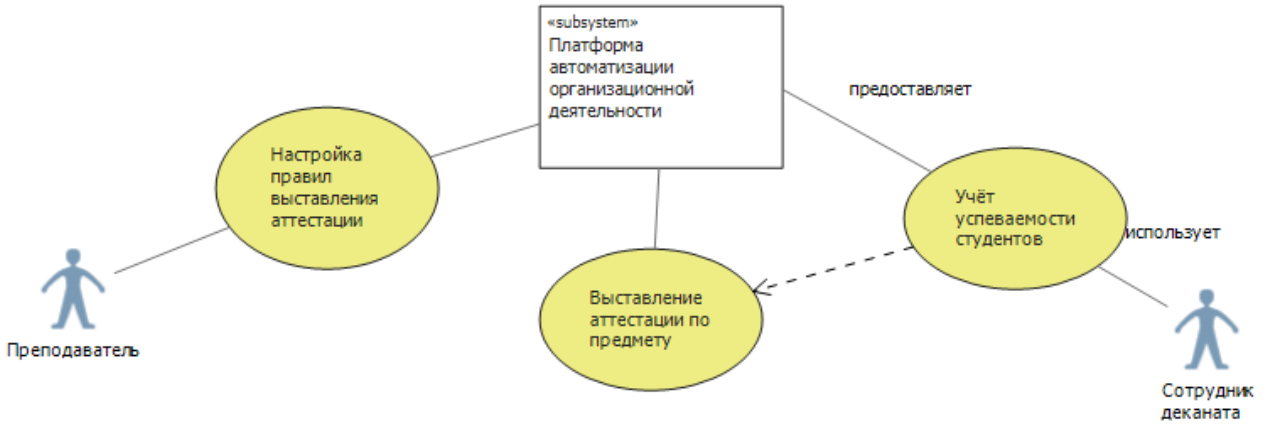

Рис. 2.3. **Диаграмма использования альтернативной аттестации с использованием Moodle**

Легко заметить, что с использованием такой схемы снижается зависимость учёта успеваемости от других процессов, так же уменьшается количество участников. По сути весь процесс будет завесить от стабильности и быстродействия системы.

2.3. Определение требований к проекту

Исходя из анализа бизнес-процессов сотрудников деканата были поставлены следующие требования к проекту:

– Взаимодействие с системой дистанционного обучения Moodle

- Формировать аттестационные листы по заранее заданным группам и курсам
- Формировать информацию о выговорах, строгих выговорах и возможном отчислении
- Реализовать массовые операции со списком студентов, полученном по заданным условиям. Например, оповещать студентов о неудовлетворительной успеваемости
- Давать возможность авторизованным пользователям настраивать аттестационные периоды, количество предметов для неаттестации, пороговое значение для учёта неаттестации, дату оповещения неаттестованных студентов

3. Методы разработки приложений для взаимодействия с Moodle

Взаимодействие с Moodle можно организовать несколькими способами:

- Как стороннее приложение
- Как расширение для Moodle

Каждый вариант имеет свои плюсы и минусы, и я рассмотрел оба варианта, постарался выбрать наиболее подходящий для поставленной задачи.

3.1. Взаимодействие в виде стороннего приложения

Данное архитектурное решение заключается в том, чтобы использовать ту же саму базу данных, что и Moodle, получать рабочие данные и использовать их в своих целях (рис. 3.1.).

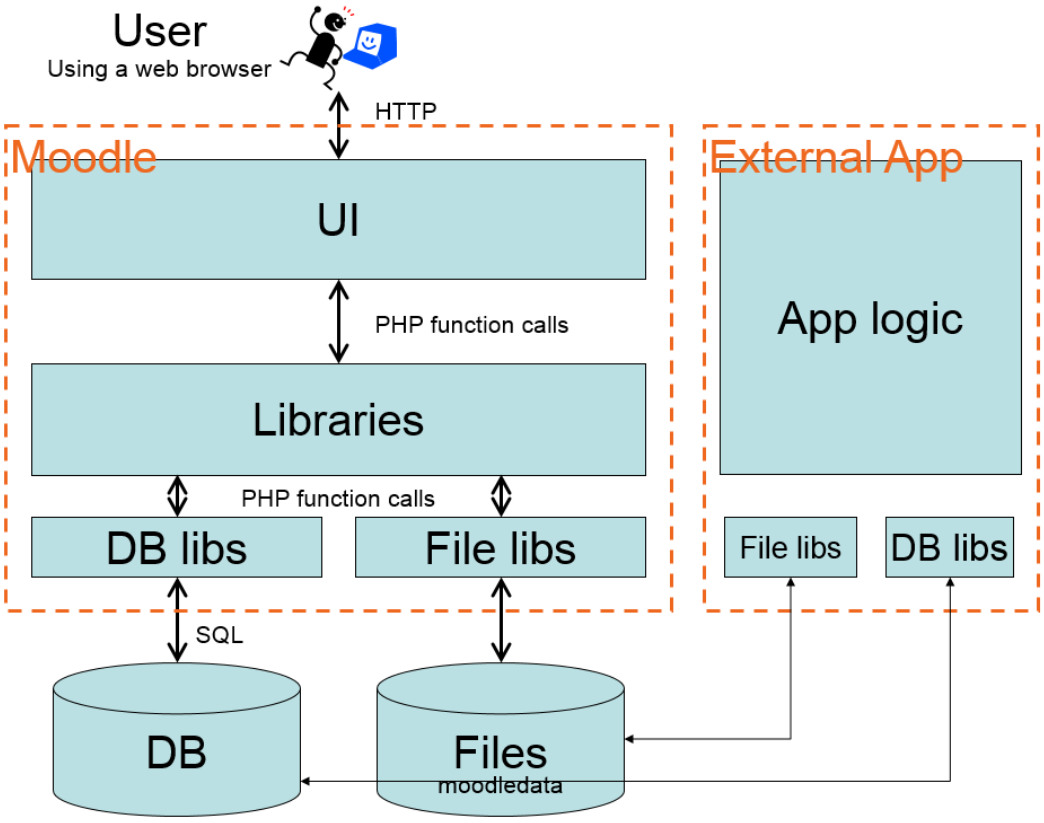

Рис. 3.1. **Архитектура взаимодействия внешнего приложения с Moodle**

Чтобы учитывать моменты, связанные с безопасностью и производительностью, доступ к базе данных даётся только на чтение.

В таком случае, не важно, на каком языке разработано приложение, не важныа платформа, ОС и т.д., важна лишь возможность подключения к базе данных.

Само приложение может кэшировать полученные данные для уменьшение количества запросов и нагрузки на СУБД. Сами настройки приложения могут храниться как на компьютере, где оно было развёрнуто, так же и в базе данных на том же сервере, где развёрнут Moodle, или на каком-либо другом.

При таком подходе мы объязаны задавать настройки подключение к БД и создавать нового пользователя для работы с базой, наделив его правами только на чтение данных.

Так же возможны пробемы, связанные с изменением структуры базы с изменением moodle. С каждым новым крупным релизом появляются новые возможности и изменния касаются не только исходного кода продукта, но и формата хранения данных. [10] Для решения этой задачи при обнолвении moodle приходилос бы заново переписывать части приложения, ответственного за работу с базой данных, или же делать настройки для формирования соотвествия таблиц и полей базы на определённые структуры данных в приложении, что может вызвать неопраданную сложность разработки и тестирования.

Ещё один минус такого подхода в том, что при расширении сайта, увеличения нагрузки и притока пользователей, один из методов оптимизации - это масштабирование базы данных, когда некотоыре таблицы хранятся на разных серверах. В таком случае надо учитывать, что таблицы, ответственные за работу приложения должны находиться на одном сервере, или, опять же, необходимо ввести поддержку подобного функционала.

3.2. Взаимодействие через плагин Moodle

Moodle имеет модульную архитектуру, что значит, что кроме встроенных модулей возможна ещё и разработка сторонних расширений (Рис. 3.2.).[2]

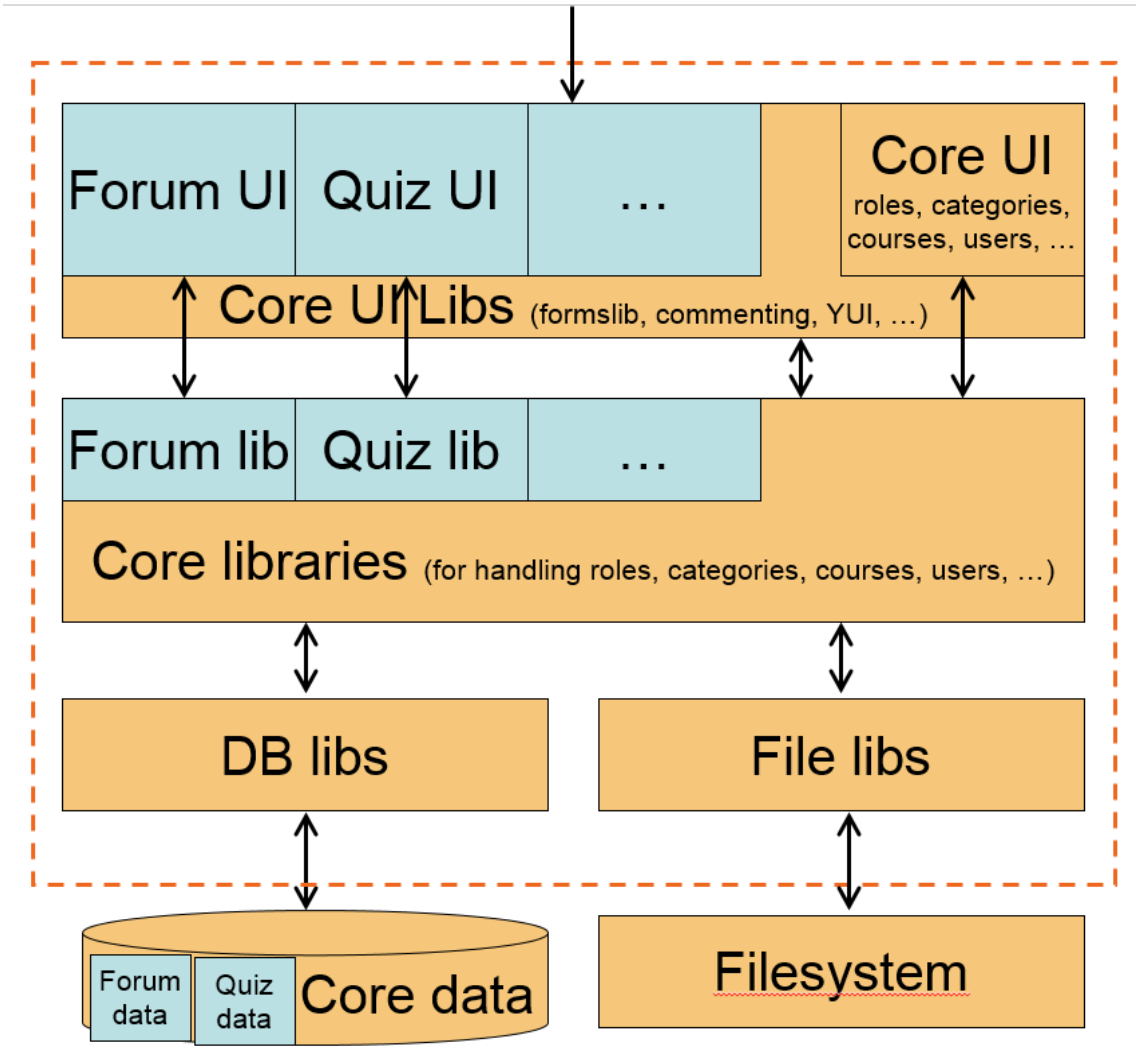

Рис. 3.2. **Архитектура модулей в Moodle**

Добавление новых возможностей идёт путём добавления кода в логические слоя Moodle: в UI, и в библиотеки. При этом модуль так же может использовать код другого модуля, если тот предоставляет такую возможность.

Так же модуль может использовать базу данных Moodle напрямую, но при этом исполнение всего SQL тогда будет исполнятся с правами системы:

разрешён полный доступ ко всем данным на запись и чтение. При наличии соответствующего API, подход с применением прямых запросов считается неудачным и опасным как для архитектуры самого модуля, так и системы в целом.[2]

Однако при такой реализации приложения не ясно, где лучше всего хранить настройки, введённые пользователем. Существует 2 варианта хранения пользовательских данных:

- Хранить данные в базе данных Moodle, в заранее созданной таблице
- Хранить данные в альтернативном источнике. В качестве альтернативного источника может выступать как другая база данных, так и вовсе другой тип хранения: кэш, другая СУБД, текстовые файлы и т.д.

Первый подход имеет несравнимый плюс – при создании резервной копии, Moodle будет копировать в том числе и данные из таблицы с настройками. Однако при переходе на другую версию, очищения БД или исправлении каких-либо ошибок, необходимо проверять, существует ли уже такая таблица в базе данных.

Второй подход обладает таким же недостатком – при изменении конфигурации сервера или развёртывании новой копии moodle, необходимо иметь в виду, что должны быть созданы дополнительные условия для работы модуля в виде создания и подготовки к использованию альтернативного источника данных. Плюсом является то, что не будет задета системная база данных и уменьшается риск повреждения данных.

3.3. Проектирование функциональной и логической схемы приложения

Для разработки я решил использовать архитектуру взаимодействия с помощью модуля, потому что так плагин будет частью сайта и частью Moolde,

он будет легче переноситься и расширяться. Так же с точки зрения использования удобнее, когда, всё находится в рамках одной системы.

В рамках этого проекта можно использовать разные типы модулей [7], но логичнее всего использовать тип модуля – отчёт по тестам. Данный модуль характеризуется тем, что при его запуске ему сразу передаются объекты, помогающие в получении результатов по заданиям и категориям оценок. С использованием этого модуля удобного генерировать аттестационные листы по заданным параметрам.

Так же необходимо использовать задачи Cron [8], потому что их можно настраивать на автоматическое исполнение по расписанию. Использование этого типа необходимо для реализации функционала оповещения студентов об их неуспеваемости по предметам.

Для хранения настроек я решил использовать вариант с таблицей в существующей базе данных Moodle, так как способ с отдельной базой второй сложнее в реализации и требует дополнительных настроек, дополнительного подключения к другому источнику данных и, соответственно, требует дополнительных ресурсов. Поэтому в данном случае, целесообразно использовать первый вариант – как самый производительный и простой в реализации. [1]

Далее было необходимо проанализировать требования к разработке и понять, какими встроенными средствами Moodle можно воспользоваться, а какие необходимо создать самостоятельно.

Анализируя требования к сайту стало ясно, что пользователь модуля будет оперировать несколькими сущностями: группа студентов, группа курсов и месяц аттестации.[3]

Для организации группы студентов лучше всего подходит когорт (cohort) в Moodle. Это глобальная на весь сайт группа студентов, с которой можно производить различные действия, например, записывать на курс

одновременно всех студентов внутри. В Moodle так же есть роли, но создавать отдельную роль для каждой группу студентов – нецелесообразно. Так же есть группы внутри курсов, но их организацией занимается преподаватель и в разных курсах они могут быть разные.

Для организации групп курсов я решил использовать категории курсов. Для решения бизнес-требования, необходимо разбить все курсы в категории курсов по семестрам обучения. Например, можно создать такую структуру:

Математическое обеспечение информационных систем

1 курс

1 семестр

Математический анализ (часть 1)

Информатика

Программирование (часть 1)

Я решил, что удобнее всего в настройках модуля задать соответствие каждому семестру обучения на группу, чтобы в интерфейсе создания отчёта использовать выбор только группы и периода аттестации.

В качестве объекта периода аттестации удобнее всего использовать категорию оценок. Категория оценок позволяет добавлять в неё определённые задания, подсчитывать итог по формуле и получать результат. То есть преподаватель добавляет задания в категорию, указывает, по каким параметрам будет считаться оценка и далее модуль сам будет получать задания из нужной категории и выводить в аттестационном листе. Однако у этого подхода есть небольшой технический минус – категории оценок должны называться везде одинаково. Это связано с тем, что категории оценок – объект локальный для каждого курса. Для решения этой проблемы я придумал такой алгоритм: выбрать все категории оценок из каждого курса в категории курсов, группировать по названию и выводить список для выбора. А дальше по

выбранному названию категории оценок получать в каждом курсе информацию, если такая категория существует.

4. Практическая реализация

4.1. Инструменты разработки

Moodle написан на языке программирования PHP [1] и может работать с разными базами данных MySQL, PostgreSQL, MSSQL, Oracle, SQLite.

Поэтому для разработки я создал виртуальное окружение, полностью имитирующее самую популярную серверную конфигурацию: Linux + Apache + MySql + PHP. По данным Worldwide Quarterly Server Tracker доля серверов на Linux с установленным веб-сервером Apache занимает 46.96% и является самой большой. [1]

Для создания среды я использовал Vagrant — свободное и открытое программное обеспечение для создания и конфигурирования виртуальной среды разработки. Является обёрткой для программного обеспечения виртуализации, например, VirtualBox, и средств управления конфигурациями, такими как Chef, Salt и Puppet. Он удобен тем, что для создания среды необходимо просто указать пакеты, которые должны быть установлены, и указать некоторые моменты их конфигурации. Их установкой и настройкой программа займётся сама.

После того, как виртуальная среда была развёрнута, я установил начал выбирать среду разработки. PHP (англ. PHP: Hypertext Preprocessor — «PHP: препроцессор гипертекста») - скриптовый язык общего назначения, часто применяемый для разработки веб-приложений. Как и для других скриптовых языков, для разработки под PHP можно использовать любой текстовый редактор с базовой подсветкой синтаксиса. Несмотря на то, что файлы в них быстро открываются, это неудобно для больших проектов, так как в них отсутствует автодополнение и навигация по коду.

Среди популярных и актуальных сред разработки с возможностью автодополнения, навигации оп коду и отладки на данный момент есть:

- Komodo IDE / Edit
- PhpStorm
- NetBeans
- Zend Studio
- Eclipce

Среди этого списка быстро работает и удобно устроена среда PhpStorm. Однако, надо заметить, что выбор среды и веб-сервера, с помощью которого будет обрабатываться php код не влияет на результаты разработки.

4.2. Реализация генерирования отчёта

Плагины в Moodle необходимо создавать со специально заранее заданной структурой, описанной в документации [11]:

- /version.php – версия модуля
- /db/install.xml – исполняется во время установки
- /db/install.php – исполняется сразу после install.xml
- /db/uninstall.php – исполняется во время удаления
- /db/upgrade.php – исполняется после того, как версия изменяется
- /db/access.php – определение прав доступа к БД
- /db/events.php – обработчик событий
- /cron.php – задачи cron
- /lib.php – файл с основными функциями для модуля
- /settings.php – файл, позволяющий создать страницу настроек

Создание плагина начинается с создания заполнения install.xml, install.php, uninstall.php, upgrade.php. С их помощью я учёл возможные варианты с обновлением как версии moodle, так и версии модуля, чтобы избежать случая, когда специальная таблица для хранения настроек модуля может быть потеряна.[12]

Далее началась разработка index.php. Изначально должны быть определить, какие должны быть предустановки для работы с плагином. Это делается вызовами:

```
require_login();
require_capability('gradereport/customgrader:view', $context);
require_capability('moodle/grade:viewall', $context);
```

Из которых видно, что для работы с плагином нужно, нужно иметь права на просмотр отчётов и специальные права из модуля, который мы создаём – это права на чтение из базы данных.

Для конфигурирования прав на использование БД для построения отчёта достаточно использовать права на чтение информации о курсах [7] в файле /db/access.php:

```
$capabilities = array(
    'gradereport/customgrader:view' => array(
        'riskbitmask' => RISK_PERSONAL,
        'captype' => 'read',
        'contextlevel' => CONTEXT_COURSE,
        'archetypes' => array(
            'teacher' => CAP_ALLOW,
            'editingteacher' => CAP_ALLOW,
            'manager' => CAP_ALLOW
        )
    )
```

Из приведённого куска кода видно, что доступ будет только для чтения и информации по курсам

```
'captype' => 'read',
'contextlevel' => CONTEXT_COURSE,
```

Далее в index.php выводится шапка сайта, меню и пр. верхняя часть сайта с помощью функции print_grade_page_head.

Далее возможны два варианта исхода: или параметры для генерирования отчёта не заданы и тогда мы должны вывести список из двух пунктов, предлагающий выбрать группу (когорт) и период аттестации (категорию оценки) или же мы должны генерировать отчёт.

Выборку когортов и категорий оценок я проводил прямо в index.php с помощью:

- Для когортов: функций $DB->get_records и передачи нужного типа, а в дальнейшем проверял пара на возможность использования нужного типа через has_capability('moodle/cohort:assign', $context)
- Для категорий оценок: grade_category::fetch_all([])

Далее, выводил форму для выбора (рис. 4.1) через написанный мною класс customgraderreport_form.

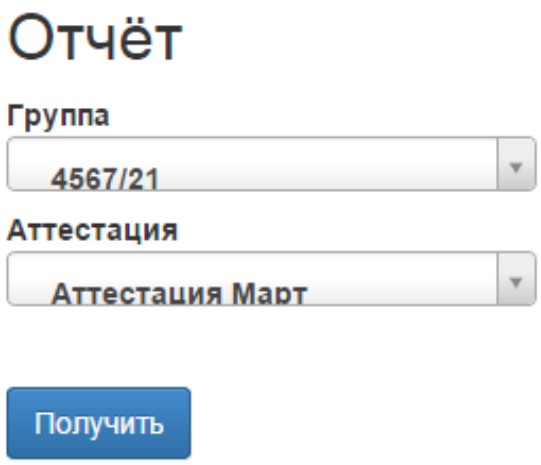

Рис. 4.1. **Форма генерирования отчёта**

Он наследуется от moodleform, который реализует обработку форм.

Если же параметры формы уже отправлены, то сначала для полученных курсов из категории производился пересчёт оценок с помощью функции grade_regrade_final_grades. Далее для генерации отчёта использовался класс grade_report_customgrader, который наследуется от класса grade_report. В этом

классе уже есть все объекты, помогающие сбору информации, необходимо было правильно ими воспользоваться, соблюдая API.

Точкой входа для генерации отчёта служила функция get_final_grades, в которую передавался id курса и массив студентов для оценок. Далее эта информация объединялась в целый массив и представлял собой массив массивов, полностью соответствующей аттестационному листу (рис. 4.2.).

Аттестация Март для группы 4567/21

	Тестовый предмет 1	Тестовый предмет 3	Тестовый предмет 4	Тестовый предмет 5	Тестовый предмет 10
Фамилия 1 Имя	76	81	69	100	100
Тест Тестов	100	100	100	100	100
Фамилия 2 Имя (н.а. №2)	51	97	0	0	78
Фамилия 5 Имя	65	54	32	85	85

Рис. 4.2. **Сгенерированный аттестационный лист**

На аттестационном листе красным обозначен студент, который не аттестован в данном месяце и в скобках указано число, соответствующее количеству месячных неаттестаций.

Для оповещения студентов использовалась cron задача, описанная в файле cron.php. В ней процесс устроен так же, как в index.php за исключением нескольких вещей - группы заданы с самого начала, а месяц аттестации используется текущий, и этот файл не выводит никаких данных как результат своей деятельности. Такая задача запускается по расписанию, дата запуска настраивается. Полный текст кода, который генерирует отчёт, приведён в Приложении.

4.3. Реализация настроек модуля

Настройки приложения добавляют специальный пункт в меню через $ADMIN->add и передают страницу, которая генерирует (рис. 4.3.) и результат этой формы сохраняет в специально заранее созданную таблицу из install.xml.

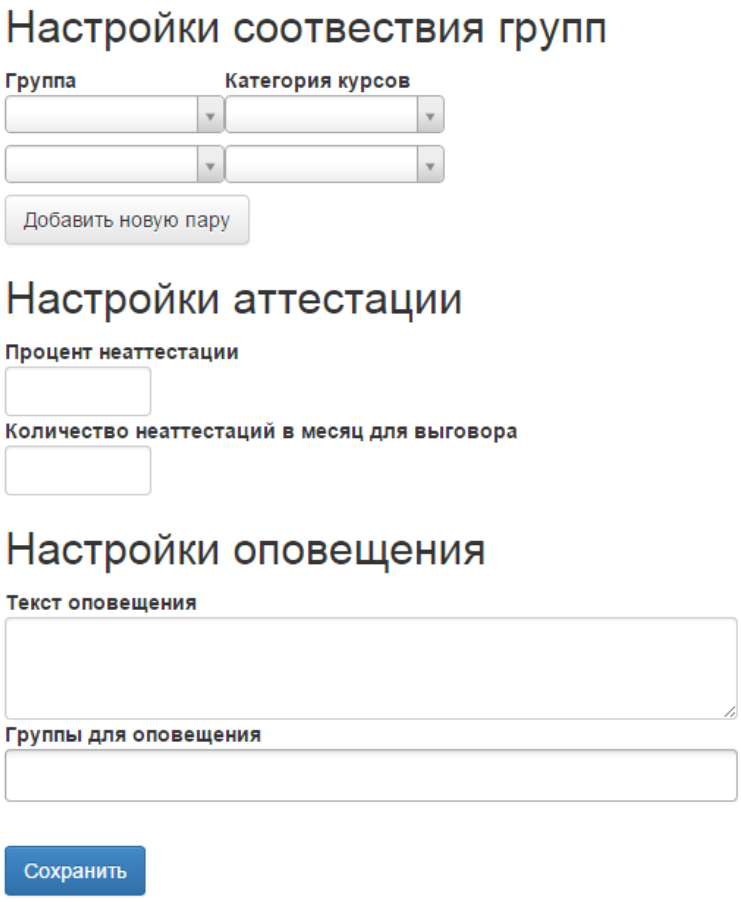

Рис. 4.3. **Настройки модуля**

Эта форма customgradersettings_form наследуется от moodleform. Внутри этого класса реализуется работы формы настроек: там задаются поля, получаются их значения, сохраняются в БД. Полный текст кода, отвечающий за настройки, приведён в Приложении.

5. Тестирование и внедрение

5.1. Функциональное тестирование

Функциональное тестирование — это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

Функциональные требования включают в себя:

- Функциональная пригодность
- Точность
- Способность к взаимодействию
- Соответствие стандартам и правилам

В рамках работы я подразумеваю, что созданный модуль соответствует тем правилам и требованиям, которые я описывал в главе «Концепция веб-приложения».

По требованиям функциональной пригодности и точности, в первую очередь были проверены возможности модуля:

- Различные настройки и их комбинации
- Генерирования отчётов на разных категориях с разным количеством студентов и предметов
- Так же проверка того, что различные настройки не влияют на работоспособность генерирования отчётов

Далее было проверено, модуль способен взаимодействовать с остальными частями системы, то есть что установка модуля и его использование не мешает базовым функция Moodle, а именно: [5]

- Просмотр, создание, редактирования курса и его элементов
- Прохождению тестов и использованию прочих учебных элементов, выставлению оценок
- Администрированию курса, добавлению, удалению новых студентов в курс
- Управлению категориями курсов
- Управлению категориями оценок

К сожалению, протестировать весь Moodle не представляется возможным, так как это очень большая система и на её тестирование может уйти не одна неделя. Поэтому тестирование проводилось только тех элементов, которых касаются вызовы API модуля. Все остальные элементы Moodle остаются нетронутыми.

5.2. Нагрузочное тестирование

Нагрузочное тестирование — подвид тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе. В общем случае означает практику моделирования ожидаемого использования приложения с помощью эмуляции работы нескольких пользователей одновременно. Основная цель нагрузочного тестирования заключается в том, чтобы, создав определённую ожидаемую в системе нагрузку (например, посредством виртуальных пользователей) и, обычно, использовав идентичное программное и аппаратное обеспечение, наблюдать за показателями производительности системы.

В рамках нагрузочного тестирования я создал SQL скрипт, который генерирует 1000 пользователей, добавляет их в базу данных и подписывает на курсы. Далее была разработана программа на языке C#, которая многопоточно

одновременно производила 100 запросов на генерирование аттестационного листа, дожидалась ответов, замеряла среднее время ответа. Конфигурация сервера, на котором происходило тестирование, является средняя по современным меркам: CPU: 2Gh Core i5, RAM: 500 Mb. Данная конфигурация соответствует бюджетной конфигурации выделенных серверов на сегодняшнем рынке хостингов и вычислительных мощностей.

За удовлетворительные результаты нагрузочного тестирования я принял 1 с. – время, которое психологически принимается как довольно-таки быстрый ответ от сайта. Среднее время, которая показал модуль по результатам 10 запусков нагрузочного теста: 0,1074 с., которое я принял за удовлетворительное. Минимальным значением являлось 0,8278 с., а максимальным – 1,307 с. Эти результаты я счёл удачными и считаю, что плагин может работать под нагрузкой такого сайта, как dl.spbstu.ru.

5.3. Анализ экономической эффективности

Разработка модуля для автоматизации организационной работы заняла 145 рабочих часов. С учётом средней заработной платы по данным «Яндекс.Работа» по вакансии «Программист PHP» на основе 128 вакансий, она может быть оценена в 79 750 рублей. Работа проделана одноразово, не нуждается в дополнительной разработке, так же не нуждается в дополнительном администрировании, кроме как администрировании всего ресурса дистанционного образования и не нуждается в дополнительном обучении сотрудников.

Обработка одной бумажной ведомости занимает 10 минут времени. На обработку 981 ведомости для всех групп первого высшего образования в Политехническом Университете ежемесячно уйдёт 9810 минут или 163,5 часов. В семестр это будет 654 часа. При оценке одного месяца оплаты сотрудника деканата в 20 000 рублей, получаются затраты в 3 270 000 рублей за семестр.

Из этого следует, что использование данного модуля сможет сэкономить более 3 000 000 рублей за семестр обучения.

5.4. Техника безопасности

Так как проект представляет из себя расширения для веб-приложения, то техника безопасности сводится к технике безопасности работы за компьютером.

Требования безопасности, направленные на предотвращение неблагоприятного влияния на здоровье человека вредных факторов производственной среды и трудового процесса при работе с персональными электронно-вычислительными машинами (ПЭВМ) содержатся в СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы», утвержденного Постановлением Главного санитарного врача Российской Федерации от 03.06.2003 г. №118 (в редакции от 03.09.2010 г.).

При работе с ПЭВМ (компьютерами) необходимо соблюдать следующие меры безопасности и охраны труда:

- эксплуатация ПЭВМ должна осуществляться в помещениях с естественным и искусственным освещением;
- оконные проемы должны быть оборудованы регулируемыми устройствами типа жалюзи, занавесей, внешних козырьков, позволяющих исключить прямую блескость, создаваемую солнечными лучами;
- площадь на одно рабочее место пользователей, работающих с видеодисплейными терминалами (мониторами) на базе электронно-лучевой трубки должна составлять не менее 6 квадратных метров, с мониторами на базе плоских дискретных

экранов (жидкокристаллические, плазменные) – не менее 4,5 квадратных метров;

- светильники местного освещения должны иметь не просвечивающий отражатель с защитным углом не менее 40 градусов;
- расстояние от глаз до экрана видеодисплейного терминала должно находится в пределах 600-700 мм, но не ближе 500 мм;
- в помещении с ПЭВМ должна производиться ежедневная влажная уборка пола и мебели. Помещения с работающими ПЭВМ необходимо проветривать после каждого часа работы;
- женщины со времени установления беременности переводятся на работы, не связанные с использованием ПЭВМ, или для них ограничивается время работы с ПЭВМ (не более 3 часов за рабочую смену) при условии соблюдения гигиенических требований;
- продолжительность непрерывной работы с видеодисплейным терминалом без регламентированного перерыва не должна превышать 1 час;
- для предупреждения преждевременной утомляемости рекомендуется организовывать рабочую смену путём чередования работ с использованием ПЭВМ и без неё;
- при работе с текстовой информацией рекомендуется выбирать наиболее физиологичный режим представления черных символов на белом фоне;
- если работник во время перерыва в работе с ПЭВМ вынужден находиться в непосредственной близости от него (менее 2 метров), то необходимо отключить питание монитора.

В зависимости от категории трудовой деятельности и уровня нагрузки за рабочую смену при работе с ПЭВМ устанавливается суммарное время регламентированных перерывов, приведенное в следующей таблице:

Таблица 5.1.

Уровень нагрузки в зависимости от рабочей группы

Категория работы с ПЭВМ	Уровень нагрузки за рабочую смену при видах работ с ПЭВМ			Суммарное время регламентированных перерывов при 8-часовой смене, мин.
	группа А, количество знаков	группа Б, количество знаков	группа В, часов	
I	до 20 000	до 15 000	до 2	50
II	до 40 000	до 30 000	до 4	70
III	до 60 000	до 40 000	до 6	90

Требования безопасности при эксплуатации электрооборудования регламентируются следующими нормативными актами:

- Правилами устройства электроустановок (издание шестое с отдельными разделами и главами в издании седьмом), утвержденными Главтехуправлением, Госэнергонадзором Минэнерго СССР 05.10.1979 г.;
- Правилами технической эксплуатации электроустановок потребителей, утверждёнными Приказом Минэнерго России от 13.01.2003 г. №6;
- Межотраслевыми правилами охраны труда (правилами безопасности) при эксплуатации электроустановок (ПОТ РМ

016-2001), утвержденными Постановлением Минтруда России от 05.01.2001 г. №3.

При эксплуатации офисного электрооборудования необходимо выполнять следующие требования:

- применяемое электрооборудование должно быть заводского изготовления и соответствовать требованиям государственных стандартов и технических условий (что подтверждается в документах завода-изготовителя);
- при эксплуатации и обслуживании электрооборудования необходимо соблюдать требования:
  - настоящей инструкции;
  - паспорта и руководства (инструкции) по эксплуатации электрооборудования, разработанных заводом-изготовителем электрооборудования (при их наличии);
- Работники при эксплуатации электрооборудования могут производить простейшие операции по его обслуживанию:
  - подключение и отключение разъемов ПЭВМ и оргтехники (принтеров, факсов, копировальных аппаратов);
  - установку и удаление бумаги в печатающие и копирующие устройства (в предусмотренные лотки для бумаги);
  - выемку, установку, замену картриджей в печатающих и копирующих устройствах;
  - выемку застрявшей бумаги в печатающих и копирующих устройствах;
  - удаление пыли и загрязнений.

- Работы по ремонту электрооборудования должны выполняться специально обученным обслуживающим персоналом (в том числе представителями сторонних организаций);
- электрооборудование, имеющее контакты для подключения заземления, должно быть заземлено, а помещения, где размещаются рабочие места с ПЭВМ (компьютерами), должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации оборудования;
- все крышки и защитные панели должны находиться на своих местах (при отсутствии крышки или защитной панели эксплуатация электрооборудования не допускается);
- при работе с электрооборудованием не допускать попадания влаги на поверхность электрооборудования, а также запрещается работать на электрооборудовании влажными руками;
- вентиляционные отверстия электрооборудования не должны быть перекрыты находящимися вплотную стенами, мебелью, посторонними предметами;
- выдергивание штепсельной вилки электроприбора необходимо осуществлять за корпус штепсельной вилки, при необходимости придерживая другой рукой корпус штепсельной розетки;
- подключение и отключение разъемов компьютеров и оргтехники должно производиться при отключенном питании (за исключением подключения и отключения USB-устройств);

- установка и удаление бумаги осуществляется в лотки (установленные места) печатающих и копирующих устройств;
- выемка, установка, замена картриджей в печатающих и копирующих устройствах, а также выемка застрявшей бумаги должны осуществляться при отключенном электрооборудовании;
- удаление пыли с электрооборудования должно производиться в отключенном от электрической цепи состоянии;
- перед использованием электроприборов необходимо проверить надёжность крепления электророзетки, свериться с номиналом используемого напряжения;
- в помещениях, в которых используется напряжение двух и более номиналов, на всех штепсельных розетках должны быть надписи с указанием номинального напряжения;
- корпуса штепсельных розеток и выключателей не должны содержать трещин, оплавлений и других дефектов, способных снизить защитные свойства или нарушить надёжность контакта;
- недопустимо использовать штепсельные разъёмы в случае существенного нагревания штепсельной розетки или вилки электроприбора при эксплуатации;
- кабели (шнуры) электропитания не должны содержать повреждений изоляции, сильных изгибов и скручиваний;
- неэлектротехническому персоналу, выполняющему работы, при которых может возникнуть опасность поражения электрическим током, присваивается группа I по электробезопасности.

Перечень должностей и профессий, требующих присвоения персоналу I группы по электробезопасности, определяет руководитель Потребителя. Персоналу, усвоившему требования по электробезопасности, относящиеся к его производственной деятельности, присваивается группа I с оформлением в журнале установленной формы; удостоверение не выдается.

Присвоение группы I производится путем проведения инструктажа, который, как правило, должен завершаться проверкой знаний в форме устного опроса и (при необходимости) проверкой приобретенных навыков безопасных способов работы или оказания первой помощи при поражении электрическим током. Присвоение I группы по электробезопасности проводит работник из числа электротехнического персонала данного Потребителя с группой по электробезопасности не ниже III.

Присвоение I группы по электробезопасности проводится с периодичностью не реже 1 раза в год.

Заключение

В результате выполнения данной работы был спроектирован, разработан и протестирован модуль для Moodle, позволяющий создавать отчёты по успеваемости студентов.

В данной работе я продемонстрировал, как можно устроить автоматизацию организационной деятельности в системе Moodle:

- были проанализированы бизнес-процессы сотрудников деканата, результатом чего была поставлена цель для автоматизации;
- предложена концепция приложения, проводящего автоматизацию;
- был разработан модуль по представленной концепции, позволяющий решать поставленные задачи;

В дальнейшем модуль может быть расширен путём добавления новых функций к уже существующим структурам данных и логики.

Список литературы

1. PHP 5. Полное руководство, 3-е издание/ Д. Коггзолл. – Вильямс, 2008. – 752 с.
2. Moodle Add-ons/ G. Hentick, M. Raadt – Learning Technology Services, 2013. – 227 с.
3. Moodle Gradebook/ R. Barrington– Packt Publishing, 2013. – 118 с.
4. Moodle 1.9 Extension Development/ J. Moore, M. Churchward – Packt Publishing, 2010. – 302 с.
5. Moodle: Extending your Moodle site with Community Addons/ J Mr G Henrick – Learning Technology Services, 2013. – 187 с.
6. Moodle Administration / A. Büchner – Packt Publishing, 2011. – 379 с.
7. Moodle Developer Documentation – Plugins Development: [Электронный ресурс]. – (https://docs.moodle.org/dev/Plugins). Проверено 01.06.2015.
8. Moodle Developer Documentation – Core APIs: [Электронный ресурс]. – (https://docs.moodle.org/dev/Core_APIs). Проверено 01.06.2015.
9. Moodle 2.9 Documentation: [Электронный ресурс]. – (https://docs.moodle.org/29/en/Main_page). Проверено 01.06.2015.
10. Moodle development courses: [Электронный ресурс]. – (http://dev.moodle.org/). Проверено 01.06.2015.
11. Priya Ramakrishnan - Developing your first Moodle plugin - iMoot 2013: [Электронный ресурс]. – (http://www.youtube.com/watch?v=UICqCXseDm4). Проверено 01.06.2015.
12. Moodle tutorials: [Электронный ресурс]. – (http://moodle-tutorials.blogspot.ru/). Проверено 01.06.2015.

Приложение

Код модуля для автоматизации организационной деятельности:

```
<?php

require_once('../../../config.php');
require_once($CFG->libdir . '/gradelib.php');
require_once($CFG->dirroot . '/grade/lib.php');
require_once($CFG->dirroot . '/grade/report/customgrader/lib.php');
require_once($CFG->dirroot .
'/grade/report/customgrader/classes/customgraderreport_form.php');
require_once($CFG->libdir . '/coursecatlib.php');

$sortitemid = optional_param('sortitemid', 0, PARAM_ALPHANUM);
// sort by which grade item
$action = optional_param('action', 0, PARAM_ALPHAEXT);
$target = optional_param('target', 0, PARAM_ALPHANUM);
$toggle = optional_param('toggle', NULL, PARAM_INT);
$toggle_type = optional_param('toggle_type', 0,
PARAM_ALPHANUM);


$context = context_system::instance();
$PAGE->set_context($context);
$PAGE->set_url(new
moodle_url('/grade/report/customgrader/index.php'));
$PAGE->requires->css('/grade/report/customgrader/styles.css');

require_login();
require_capability('gradereport/customgrader:view', $context);
require_capability('moodle/grade:viewall', $context);

// Handle toggle change request.
if (!is_null($toggle) && !empty($toggle_type)) {
   set_user_preferences(array('grade_report_show' . $toggle_type =>
$toggle));
}

// First make sure we have proper final grades - this must be done
before constructing of the grade tree.
//grade_regrade_final_grades($courseid);

// Perform actions.
```

```
        if (!empty($target) && !empty($action) && confirm_sesskey()) {
            grade_report_grader::do_process_action($target, $action);
        }
        $reportname = get_string('pluginname', 'gradereport_customgrader');

        // Print header
        print_grade_page_head(1, 'report', 'customgrader', $reportname, false);


        $cohorts = array();
        $allcohorts = $DB->get_records('cohort', null, 'name');
        $context = null;
        foreach ($allcohorts as $c) {
            if (!empty($c->component)) {
                // external cohorts can not be modified
                continue;
            }
            $context = context::instance_by_id($c->contextid);
            if (!has_capability('moodle/cohort:assign', $context)) {
                continue;
            }

            if (empty($c->idnumber)) {
                $cohorts[$c->id] = format_string($c->name);
            } else {
                $cohorts[$c->id] = format_string($c->name) . ' [' . $c->idnumber .
']';
            }
        }
        unset($allcohorts);

        $grade_categories = [];
        $all_grade_categories = grade_category::fetch_all([]);
        foreach ($all_grade_categories as $grade_category) {
            if ($grade_category->parent && !in_array($grade_category-
>fullname, array_values($grade_categories))) {
                $grade_categories[$grade_category->id] = $grade_category-
>fullname;
            }
        }

        $id = optional_param('id', 0, PARAM_INT);
        $mform = new customgraderreport_form(null, array(
            'categoryid' => 0, //$id,
            'parent' => 0,
```

```
            'context' => $context,
            'itemid' => 0,//$itemid,
            'cohorts' => $cohorts,
            'grade_categories' => $grade_categories,
        ));
        $mform->display();


        if (isset($_POST['course_category'], $_POST['cohort'],
$_POST['grade_category'])) {

            $course_category_id = (int)$_POST['course_category'];
            $cohort_id = (int)$_POST['cohort'];
            $grade_category_id = (int)$_POST['grade_category'];

            $options = array();
            $options['recursive'] = true;
            $courses = coursecat::get($course_category_id)-
>get_courses($options);


            $grade_category_name = '';
            foreach ($all_grade_categories as $grade_category) {
                if ($grade_category->id == $grade_category_id) {
                    $grade_category_name = $grade_category->fullname;
                }
            }
            $grade_categories_with_same_name = [];
            foreach ($all_grade_categories as $grade_category) {
                if ($grade_category->fullname == $grade_category_name) {
                    $grade_categories_with_same_name[] = $grade_category->id;
                }
            }

            $user_ids =
grade_report_customgrader::get_cohort_user_ids($cohort_id);

            if (!empty($courses)) {

                $reports = [];

                foreach ($courses as $thiscourse) {
                    $courseid = $thiscourse->id;
                    $context = context_course::instance($courseid);
                    if (has_capability('moodle/grade:viewall', $context)) {
```

```
                    if (has_capability('gradereport/customgrader:view', $context))
{

                        grade_regrade_final_grades($courseid);

                        $gpr = new grade_plugin_return(array('type' => 'report',
'plugin' => 'customgrader', 'courseid' => $courseid));

                        // Initialise the custom grader report object that produces
the table
                        $report = new grade_report_customgrader($courseid, $gpr,
$context, 0, $sortitemid, $grade_categories_with_same_name, $thiscourse-
>shortname);

                        $grades = $report->get_final_grades($courseid, $user_ids);

                        $users = $report->load_users($user_ids);

                        $reports[] = $report;
                    }
                }
            }

            if ($reports) {
                $reporthtml =
grade_report_customgrader::get_all_reports_table($reports);
                echo $reporthtml;

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


    <?php

    require_once($CFG->dirroot . '/grade/report/lib.php');
    require_once($CFG->libdir . '/tablelib.php');

    /**
     * Class providing an API for the custom grader report building and
displaying.
     * @uses grade_report
     * @package gradereport_customgrader
     */
```

```
class grade_report_customgrader extends grade_report {


    public $course_grades;

    /**
     * The final grades.
     * @var array $grades
     */
    public $grades;

    /**
     * Array of errors for bulk grades updating.
     * @var array $gradeserror
     */
    public $gradeserror = array();

//// SQL-RELATED

    /**
     * The id of the grade_item by which this report will be sorted.
     * @var int $sortitemid
     */
    public $sortitemid;

    /**
     * Sortorder used in the SQL selections.
     * @var int $sortorder
     */
    public $sortorder;

    /**
     * An SQL fragment affecting the search for users.
     * @var string $userselect
     */
    public $userselect;

    /**
     * The bound params for $userselect
     * @var array $userselectparams
     */
    public $userselectparams = array();

    /**
     * List of collapsed categories from user preference
```

```
     * @var array $collapsed
     */
    public $collapsed;

    /**
     * A count of the rows, used for css classes.
     * @var int $rowcount
     */
    public $rowcount = 0;

    /**
     * Capability check caching
     * @var boolean $canviewhidden
     */
    public $canviewhidden;

    var $preferencespage=false;

    /**
     * Length at which feedback will be truncated (to the nearest word)
and an ellipsis be added.
     * TODO replace this by a report preference
     * @var int $feedback_trunc_length
     */
    protected $feedback_trunc_length = 50;

    /**
     * @var array
     */
    public $grade_category_ids;

    /**
     * @var string
     */
    public $course_name;

    /**
     * Constructor. Sets local copies of user preferences and initialises
grade_tree.
     * @param int $courseid
     * @param object $gpr grade plugin return tracking object
     * @param string $context
     * @param int $page The current page being viewed (when report is
paged)
```

```
     * @param int $sortitemid The id of the grade_item by which to sort
the table
     */
    public function __construct($courseid, $gpr, $context, $page = null,
$sortitemid = null, $grade_category_ids, $course_name) {
        global $CFG;
        parent::__construct($courseid, $gpr, $context, $page);

        $this->grade_category_ids = $grade_category_ids;
        $this->course_name = $course_name;
        $this->canviewhidden =
has_capability('moodle/grade:viewhidden', context_course::instance($this-
>course->id));

        // load collapsed settings for this report
        if ($collapsed =
get_user_preferences('grade_report_customgrader_collapsed_categories')) {
            $this->collapsed = unserialize($collapsed);
        } else {
            $this->collapsed = array('aggregatesonly' => array(),
'gradesonly' => array());
        }

        if (empty($CFG->enableoutcomes)) {
            $nooutcomes = false;
        } else {
            $nooutcomes =
get_user_preferences('grade_report_shownooutcomes');
        }

        // if user report preference set or site report setting set use it,
otherwise use course or site setting
        $switch = $this->get_pref('aggregationposition');
        if ($switch == '') {
            $switch = grade_get_setting($this->courseid,
'aggregationposition', $CFG->grade_aggregationposition);
        }

        // Grab the grade_tree for this course
        $this->gtree = new grade_tree($this->courseid, true, $switch,
$this->collapsed, $nooutcomes);

        $this->sortitemid = $sortitemid;

        // base url for sorting by first/last name
```

```
            $this->baseurl = new moodle_url('index.php', array('id' => $this-
>courseid));

            $studentsperpage = $this->get_students_per_page();
            if (!empty($this->page) && !empty($studentsperpage)) {
                $this->baseurl->params(array('perpage' => $studentsperpage,
'page' => $this->page));
            }

            $this->pbarurl = new
moodle_url('/grade/report/customgrader/index.php', array('id' => $this-
>courseid));

            $this->setup_groups();

            $this->setup_sortitemid();
        }

        /**
         * Processes the data sent by the form (grades and feedbacks).
         * Caller is responsible for all access control checks
         * @param array $data form submission (with magic quotes)
         * @return array empty array if success, array of warnings if
something fails.
         */
        public function process_data($data) {
            global $DB;
            $warnings = array();

            $separategroups = false;
            $mygroups = array();
            if ($this->groupmode == SEPARATEGROUPS and
!has_capability('moodle/site:accessallgroups', $this->context)) {
                $separategroups = true;
                $mygroups = groups_get_user_groups($this->course->id);
                $mygroups = $mygroups[0]; // ignore groupings
                // reorder the groups fro better perf below
                $current = array_search($this->currentgroup, $mygroups);
                if ($current !== false) {
                    unset($mygroups[$current]);
                    array_unshift($mygroups, $this->currentgroup);
                }
            }
```

```
// always initialize all arrays
$queue = array();
$this->load_users();
$this->load_final_grades();

// Were any changes made?
$changedgrades = false;

foreach ($data as $varname => $students) {

    $needsupdate = false;

    // skip, not a grade nor feedback
    if (strpos($varname, 'grade') === 0) {
        $datatype = 'grade';
    } else if (strpos($varname, 'feedback') === 0) {
        $datatype = 'feedback';
    } else {
        continue;
    }

    foreach ($students as $userid => $items) {
        $userid = clean_param($userid, PARAM_INT);
        foreach ($items as $itemid => $postedvalue) {
            $itemid = clean_param($itemid, PARAM_INT);

    // Was change requested?
    $oldvalue = $this->grades[$userid][$itemid];
    if ($datatype === 'grade') {
        // If there was no grade and there still isn't
        if (is_null($oldvalue->finalgrade) && $postedvalue == -1) {
            // -1 means no grade
            continue;
        }

        // If the grade item uses a custom scale
        if (!empty($oldvalue->grade_item->scaleid)) {

            if ((int) $oldvalue->finalgrade === (int) $postedvalue) {
                continue;
            }
        } else {
            // The grade item uses a numeric scale
```

```
                    // Format the finalgrade from the DB so that it matches the
grade from the client
                    if ($postedvalue === format_float($oldvalue->finalgrade,
$oldvalue->grade_item->get_decimals())) {
                        continue;
                    }
                }

                $changedgrades = true;

            } else if ($datatype === 'feedback') {
                if ($oldvalue->feedback === $postedvalue) {
                    continue;
                }
            }

            if (!$gradeitem = grade_item::fetch(array('id' => $itemid,
'courseid' => $this->courseid))) { // we must verify course id here!
                print_error('invalidgradeitemid');
            }

            // Pre-process grade
            if ($datatype == 'grade') {
                $feedback = false;
                $feedbackformat = false;
                if ($gradeitem->gradetype == GRADE_TYPE_SCALE) {
                    if ($postedvalue == -1) { // -1 means no grade
                        $finalgrade = null;
                    } else {
                        $finalgrade = $postedvalue;
                    }
                } else {
                    $finalgrade = unformat_float($postedvalue);
                }

                $errorstr = '';
                // Warn if the grade is out of bounds.
                if (is_null($finalgrade)) {
                    // ok
                } else {
                    $bounded = $gradeitem->bounded_grade($finalgrade);
                    if ($bounded > $finalgrade) {
                        $errorstr = 'lessthanmin';
                    } else if ($bounded < $finalgrade) {
                        $errorstr = 'morethanmax';
```

```
                    }
                }
                if ($errorstr) {
                    $user = $DB->get_record('user', array('id' => $userid), 'id,
firstname, lastname');
                    $gradestr = new stdClass();
                    $gradestr->username = fullname($user);
                    $gradestr->itemname = $gradeitem->get_name();
                    $warnings[] = get_string($errorstr, 'grades', $gradestr);
                }

            } else if ($datatype == 'feedback') {
                $finalgrade = false;
                $trimmed = trim($postedvalue);
                if (empty($trimmed)) {
                    $feedback = null;
                } else {
                    $feedback = $postedvalue;
                }
            }

            // group access control
            if ($separategroups) {
                // note: we can not use $this->currentgroup because it would
fail badly
                //       when having two browser windows each with different
group
                $sharinggroup = false;
                foreach ($mygroups as $groupid) {
                    if (groups_is_member($groupid, $userid)) {
                        $sharinggroup = true;
                        break;
                    }
                }
                if (!$sharinggroup) {
                    // either group membership changed or somebody is
hacking grades of other group
                    $warnings[] = get_string('errorsavegrade', 'grades');
                    continue;
                }
            }

            $gradeitem->update_final_grade($userid, $finalgrade,
'gradebook', $feedback, FORMAT_MOODLE);
```

```
                // We can update feedback without reloading the grade item as it
doesn't affect grade calculations
                if ($datatype === 'feedback') {
                    $this->grades[$userid][$itemid]->feedback = $feedback;
                }
            }
                }
            }

            if ($changedgrades) {
                // If a final grade was overriden reload grades so dependent
grades like course total will be correct
                $this->grades = null;
            }

            return $warnings;
        }


        /**
         * Setting the sort order, this depends on last state
         * all this should be in the new table class that we might need to use
         * for displaying grades.
         */
        private function setup_sortitemid() {

            global $SESSION;

            if (!isset($SESSION->gradeuserreport)) {
                $SESSION->gradeuserreport = new stdClass();
            }

            if ($this->sortitemid) {
                if (!isset($SESSION->gradeuserreport->sort)) {
                    if ($this->sortitemid == 'firstname' || $this->sortitemid ==
'lastname') {
                        $this->sortorder = $SESSION->gradeuserreport->sort =
'ASC';
                    } else {
                        $this->sortorder = $SESSION->gradeuserreport->sort =
'DESC';
                    }
                } else {
                    // this is the first sort, i.e. by last name
                    if (!isset($SESSION->gradeuserreport->sortitemid)) {
```

```
                    if ($this->sortitemid == 'firstname' || $this->sortitemid ==
'lastname') {
                        $this->sortorder = $SESSION->gradeuserreport->sort =
'ASC';
                    } else {
                        $this->sortorder = $SESSION->gradeuserreport->sort =
'DESC';
                    }
                } else if ($SESSION->gradeuserreport->sortitemid == $this-
>sortitemid) {
                    // same as last sort
                    if ($SESSION->gradeuserreport->sort == 'ASC') {
                        $this->sortorder = $SESSION->gradeuserreport->sort =
'DESC';
                    } else {
                        $this->sortorder = $SESSION->gradeuserreport->sort =
'ASC';
                    }
                } else {
                    if ($this->sortitemid == 'firstname' || $this->sortitemid ==
'lastname') {
                        $this->sortorder = $SESSION->gradeuserreport->sort =
'ASC';
                    } else {
                        $this->sortorder = $SESSION->gradeuserreport->sort =
'DESC';
                    }
                }
            }
            $SESSION->gradeuserreport->sortitemid = $this->sortitemid;
        } else {
            // not requesting sort, use last setting (for paging)

            if (isset($SESSION->gradeuserreport->sortitemid)) {
                $this->sortitemid = $SESSION->gradeuserreport->sortitemid;
            } else {
                $this->sortitemid = 'lastname';
            }

            if (isset($SESSION->gradeuserreport->sort)) {
                $this->sortorder = $SESSION->gradeuserreport->sort;
            } else {
                $this->sortorder = 'ASC';
            }
        }
```

```
    }


    public static function get_cohort_user_ids($cohort_id)
    {
        global $DB;
        $params = [
            'cohortid' => $cohort_id,
        ];

        $sql = "SELECT u.id
                FROM {user} u
                 JOIN {cohort_members} cm ON (cm.userid = u.id AND
cm.cohortid = :cohortid)";
        $users = $DB->get_records_sql($sql, $params);

        if (empty($users)) {
            return array();
        }

        $user_ids = [];
        foreach ($users as $user_row) {
            $user_ids[] = $user_row->id;
        }

        return $user_ids;
    }


    /**
     * pulls out the userids of the users to be display, and sorts them
     */
    public function load_users($userids) {
        global $CFG, $DB;

        if (!empty($this->users)) {
            return;
        }

        $sort = "u.lastname $this->sortorder, u.firstname $this->sortorder";

        $userwheresql = ' AND u.id IN(' . implode(', ', $userids) . ')';

        $sql = "SELECT u.*
                FROM {user} u
```

```
                    WHERE u.deleted = 0
                      $userwheresql
                    ORDER BY $sort";
            $this->users = $DB->get_records_sql($sql);

            if (empty($this->users)) {
                $this->userselect = '';
                $this->users = array();
                $this->userselect_params = array();
            } else {
                list($usql, $uparams) = $DB->get_in_or_equal(array_keys($this->users), SQL_PARAMS_NAMED, 'usid0');
                $this->userselect = "AND g.userid $usql";
                $this->userselect_params = $uparams;
            }
            return $this->users;
        }

        /**
         * we supply the userids in this query, and get all the grades
         * pulls out all the grades, this does not need to worry about paging
         */
        public function load_final_grades() {
            global $CFG, $DB;

            if (!empty($this->grades)) {
                return;
            }

            if (empty($this->users)) {
                return;
            }

            // please note that we must fetch all grade_grades fields if we want to construct grade_grade object from it!
            $params = array_merge(array('courseid' => $this->courseid), $this->userselect_params);
            $sql = "SELECT g.*
                      FROM {grade_items} gi,
                           {grade_grades} g
                     WHERE g.itemid = gi.id AND gi.courseid = :courseid {$this->userselect}";

            $userids = array_keys($this->users);
```

```
            if ($grades = $DB->get_records_sql($sql, $params)) {
                foreach ($grades as $graderec) {
                    if (in_array($graderec->userid, $userids) and
array_key_exists($graderec->itemid, $this->gtree->get_items())) { // some
items may not be present!!
                        $this->grades[$graderec->userid][$graderec->itemid] =
new grade_grade($graderec, false);
                        $this->grades[$graderec->userid][$graderec->itemid]-
>grade_item = $this->gtree->get_item($graderec->itemid); // db caching
                    }
                }
            }

            // prefil grades that do not exist yet
            foreach ($userids as $userid) {
                foreach ($this->gtree->get_items() as $itemid => $unused) {
                    if (!isset($this->grades[$userid][$itemid])) {
                        $this->grades[$userid][$itemid] = new grade_grade();
                        $this->grades[$userid][$itemid]->itemid = $itemid;
                        $this->grades[$userid][$itemid]->userid = $userid;
                        $this->grades[$userid][$itemid]->grade_item = $this-
>gtree->get_item($itemid); // db caching
                    }
                }
            }
        }


        public function get_final_grades($course_id, $userids) {
            global $CFG, $DB;

            if (!empty($this->grades)) {
                return;
            }
            if (empty($userids)) {
                return;
            }

            $userselect = ' AND g.userid IN(' . implode(', ', $userids) . ')';

            // please note that we must fetch all grade_grades fields if we want
to construct grade_grade object from it!
            $params = array('courseid' => $course_id);
            $sql = "SELECT g.*
```

```
                    FROM {grade_items} gi,
                        {grade_grades} g
                    WHERE g.itemid = gi.id AND gi.courseid = :courseid
{$userselect}";

            if ($grades = $DB->get_records_sql($sql, $params)) {
                foreach ($grades as $graderec) {
                    if (in_array($graderec->userid, $userids) and
array_key_exists($graderec->itemid, $this->gtree->get_items())) { // some
items may not be present!!
                        $this->grades[$graderec->userid][$graderec->itemid] =
new grade_grade($graderec, false);
                        $this->grades[$graderec->userid][$graderec->itemid]-
>grade_item = $this->gtree->get_item($graderec->itemid); // db caching
                    }
                }
            }

            // prefil grades that do not exist yet
            foreach ($userids as $userid) {
                foreach ($this->gtree->get_items() as $itemid => $unused) {
                    if (!isset($this->grades[$userid][$itemid])) {
                        $this->grades[$userid][$itemid] = new grade_grade();
                        $this->grades[$userid][$itemid]->itemid = $itemid;
                        $this->grades[$userid][$itemid]->userid = $userid;
                        $this->grades[$userid][$itemid]->grade_item = $this-
>gtree->get_item($itemid); // db caching
                    }
                }
            }
            return $this->grades;
        }



        /**
         * Gets html toggle
         * @deprecated since Moodle 2.4 as it appears not to be used any
more.
         */
        public function get_toggles_html() {
            throw new coding_exception('get_toggles_html() can not be used
any more');
            }
```

```
        /**
         * Prints html toggle
        * @deprecated since 2.4 as it appears not to be used any more.
         * @param unknown $type
         */
        public function print_toggle($type) {
            throw new coding_exception('print_toggle() can not be used any
more');
            }


        public function get_left_rows() {
            global $CFG, $USER, $OUTPUT;

            $rows = array();

            $colspan = 1;
            if (has_capability('gradereport/' . $CFG->grade_profilereport .
':view', $this->context)) {
                $colspan++;
            }

            $headerrow = new html_table_row();
            $headerrow->attributes['class'] = 'heading';

            $studentheader = new html_table_cell();
            $studentheader->attributes['class'] = 'header';
            $studentheader->scope = 'col';
            $studentheader->header = true;
            $studentheader->id = 'studentheader';
            if (has_capability('gradereport/' . $CFG->grade_profilereport .
':view', $this->context)) {
                $studentheader->colspan = 1;
            }

            $headerrow->cells[] = $studentheader;

            $rows[] = $headerrow;

            $rows = $this->get_left_icons_row($rows, $colspan);

            $rowclasses = array('even', 'odd');

            $suspendedstring = null;
            foreach ($this->users as $userid => $user) {
```

```
                $userrow = new html_table_row();
                $userrow->id = 'fixed_user_' . $userid;
                $userrow->attributes['class'] = 'r' . $this->rowcount++ . ' ' .
$rowclasses[$this->rowcount % 2];

                $usercell = new html_table_cell();
                $usercell->attributes['class'] = 'user';
                $usercell->attributes['style'] = 'border-right:1px;';

                $usercell->header = true;
                $usercell->scope = 'row';

                $usercell->text .= html_writer::link(new
moodle_url('/user/view.php', array('id' => $user->id, 'course' => $this-
>course->id)), fullname($user));

                if (!empty($user->suspendedenrolment)) {
                    $usercell->attributes['class'] .= ' usersuspended';

                    //may be lots of suspended users so only get the string once
                    if (empty($suspendedstring)) {
                        $suspendedstring = get_string('userenrolmentsuspended',
'grades');
                    }
                    $usercell->text .= html_writer::empty_tag('img', array('src' =>
$OUTPUT->pix_url('i/enrolmentsuspended'), 'title' => $suspendedstring, 'alt'
=> $suspendedstring, 'class' => 'usersuspendedicon'));
                }

                $userrow->cells[] = $usercell;

                $rows[] = $userrow;
            }

            return $rows;
        }



        /**
         * Builds and returns the rows that will make up the right part of the
custom grader report
         * @return array Array of html_table_row objects
         */
        public function get_right_rows() {
```

```
        global $CFG, $USER, $OUTPUT, $DB, $PAGE;

        $rows = array();
        $this->rowcount = 0;
        $numrows = count($this->gtree->get_levels());
        $numusers = count($this->users);
        $gradetabindex = 1;
        $columnstounset = array();
        $strgrade = $this->get_lang_string('grade');
        $strfeedback = $this->get_lang_string("feedback");
        $arrows = $this->get_sort_arrows();


        $headingrow = new html_table_row();
        $headingrow->attributes['class'] = 'heading_name_row';

        $categorycell = new html_table_cell();
        $categorycell->attributes['class'] = 'category 1';
        $categorycell->colspan = 1;
        $categorycell->text = $this->course_name;
        $categorycell->header = true;
        $categorycell->scope = 'col';

        $headingrow->cells[] = $categorycell;
        $rows[] = $headingrow;

        // Preload scale objects for items with a scaleid and initialize tab
indices
        $scaleslist = array();
        $tabindices = array();

        foreach ($this->gtree->get_items() as $itemid => $item) {
            $scale = null;
            if (!empty($item->scaleid)) {
                $scaleslist[] = $item->scaleid;
            }
            $tabindices[$item->id]['grade'] = $gradetabindex;
            $tabindices[$item->id]['feedback'] = $gradetabindex +
$numusers;
            $gradetabindex += $numusers * 2;
        }

        $rowclasses = array('even', 'odd');

        foreach ($this->users as $userid => $user) {
```

```
                if ($this->canviewhidden) {
                    $altered = array();
                    $unknown = array();
                } else {
                    $hidingaffected = grade_grade::get_hiding_affected($this-
>grades[$userid], $this->gtree->get_items());
                    $altered = $hidingaffected['altered'];
                    $unknown = $hidingaffected['unknown'];
                    unset($hidingaffected);
                }


                $itemrow = new html_table_row();
                $itemrow->id = 'user_' . $userid;
                $itemrow->attributes['class'] = $rowclasses[$this->rowcount %
2];


                $itemcell = new html_table_cell();
                $itemcell->id = 'u' . $userid . 'i' . $this->courseid;

                $grades_cnt = 0;
                $grades_sum = 0;
                $gradedisplaytype = GRADE_DISPLAY_TYPE_DEFAULT;
                foreach ($this->gtree->items as $itemid => $unused) {
                    $item = & $this->gtree->items[$itemid];

                    if (!isset($this->grades[$userid]) || !in_array($item-
>categoryid, $this->grade_category_ids)) {
                        break;
                    }

                    $grade = $this->grades[$userid][$item->id];
                    $gradedisplaytype = $item->get_displaytype();
                    if (in_array($itemid, $unknown)) {
                        $gradeval = null;
                    } else if (array_key_exists($itemid, $altered)) {
                        $gradeval = $altered[$itemid];
                    } else {
                        $gradeval = $grade->finalgrade;
                    }

                    if ($gradeval != null) {
                        $grades_cnt++;
```

```
                    $grades_sum += $gradeval;
                }
            }
            $avg_grade = ($grades_cnt == 0) ? null : $grades_sum /
$grades_cnt;
            $grade_formatted = grade_format_gradevalue($avg_grade,
$item, true, $gradedisplaytype, null);
            $url = new moodle_url('/grade/report/' . $CFG-
>grade_profilereport . '/index.php', array('userid' => $user->id, 'id' => $this-
>course->id));
            $link = html_writer::tag('a', $grade_formatted, array('class' =>
"gradevalue", 'href' => $url, 'target' => '_blank'));
            $itemcell->text = html_writer::tag('span', $link, array('class' =>
"gradevalue"));

            $itemrow->cells[] = $itemcell;
            $rows[] = $itemrow;
        }

        return $rows;
    }

    /**
     * Depending on the style of report (fixedstudents vs traditional one-
table),
     * arranges the rows of data in one or two tables, and returns the
output of
     * these tables in HTML
     * @return string HTML
     */
    public function get_grade_table() {
        global $OUTPUT;

        $leftrows = $this->get_left_rows();
        $rightrows = $this->get_right_rows();

        $html = '';

        $fulltable = new html_table();
        $fulltable->attributes['class'] = 'gradestable flexible boxaligncenter
generaltable';
        $fulltable->id = 'user-grades';

        // Extract rows from each side (left and right) and collate them into
one row each
```

```
            foreach ($leftrows as $key => $row) {
                $row->cells = array_merge($row->cells, $rightrows[$key]-
>cells);
                $fulltable->data[] = $row;
            }
            $html .= html_writer::table($fulltable);

            return $OUTPUT->container($html, 'gradeparent');
        }


        public static function get_all_reports_table($reports) {
            global $OUTPUT;

            $leftrows = $reports[0]->get_left_rows();
            $reports_rightrows = [];
            /** @var $report grade_report_customgrader */
            foreach ($reports as $report) {
                $reports_rightrows[] = $report->get_right_rows();
            }
            $html = '';

            $fulltable = new html_table();
            $fulltable->attributes['class'] = 'gradestable flexible boxaligncenter
generaltable';
            $fulltable->id = 'user-grades';

            // Extract rows from each side (left and right) and collate them into
one row each
            foreach ($leftrows as $key => $row) {
                foreach ($reports_rightrows as $rightrows) {
                    $row->cells = array_merge($row->cells, $rightrows[$key]-
>cells);
                }
                $fulltable->data[] = $row;
            }
            $html .= html_writer::table($fulltable);

            return $OUTPUT->container($html, 'gradeparent');
        }


        /**
         * Builds and return the row of icons for the left side of the report.
         * It only has one cell that says "Controls"
```

```
     * @param array $rows The Array of rows for the left part of the
report
     * @param int $colspan The number of columns this cell has to span
     * @return array Array of rows for the left part of the report
     */
    public function get_left_icons_row($rows = array(), $colspan = 1) {
        global $USER;

        return $rows;
    }

    /**
     * Builds and return the header for the row of ranges, for the left part
of the custom grader report.
     * @param array $rows The Array of rows for the left part of the
report
     * @param int $colspan The number of columns this cell has to span
     * @return array Array of rows for the left part of the report
     */
    public function get_left_range_row($rows = array(), $colspan = 1) {
        global $CFG, $USER;

        if ($this->get_pref('showranges')) {
            $rangerow = new html_table_row();
            $rangerow->attributes['class'] = 'range r' . $this->rowcount++;
            $rangecell = new html_table_cell();
            $rangecell->attributes['class'] = 'header range';
            $rangecell->colspan = $colspan;
            $rangecell->header = true;
            $rangecell->scope = 'row';
            $rangecell->text = $this->get_lang_string('range', 'grades');
            $rangerow->cells[] = $rangecell;
            $rows[] = $rangerow;
        }

        return $rows;
    }

    /**
     * Builds and return the headers for the rows of averages, for the left
part of the custom grader report.
     * @param array $rows The Array of rows for the left part of the
report
     * @param int $colspan The number of columns this cell has to span
```

```
         * @param bool $groupavg If true, returns the row for group
averages, otherwise for overall averages
         * @return array Array of rows for the left part of the report
         */
        public function get_left_avg_row($rows = array(), $colspan = 1,
$groupavg = false) {
            if (!$this->canviewhidden) {
                // totals might be affected by hiding, if user can not see hidden
grades the aggregations might be altered
                // better not show them at all if user can not see all hideen grades
                return $rows;
            }

            $showaverages = $this->get_pref('showaverages');
            $showaveragesgroup = $this->currentgroup && $showaverages;
            $straveragegroup = get_string('groupavg', 'grades');

            if ($groupavg) {
                if ($showaveragesgroup) {
                    $groupavgrow = new html_table_row();
                    $groupavgrow->attributes['class'] = 'groupavg r' . $this-
>rowcount++;
                    $groupavgcell = new html_table_cell();
                    $groupavgcell->attributes['class'] = 'header range';
                    $groupavgcell->colspan = $colspan;
                    $groupavgcell->header = true;
                    $groupavgcell->scope = 'row';
                    $groupavgcell->text = $straveragegroup;
                    $groupavgrow->cells[] = $groupavgcell;
                    $rows[] = $groupavgrow;
                }
            } else {
                $straverage = get_string('overallaverage', 'grades');

                if ($showaverages) {
                    $avgrow = new html_table_row();
                    $avgrow->attributes['class'] = 'avg r' . $this->rowcount++;
                    $avgcell = new html_table_cell();
                    $avgcell->attributes['class'] = 'header range';
                    $avgcell->colspan = $colspan;
                    $avgcell->header = true;
                    $avgcell->scope = 'row';
                    $avgcell->text = $straverage;
                    $avgrow->cells[] = $avgcell;
                    $rows[] = $avgrow;
```

```
            }
        }

        return $rows;
    }

    /**
     * Builds and return the row of icons when editing is on, for the right part of the custom grader report.
     * @param array $rows The Array of rows for the right part of the report
     * @return array Array of rows for the right part of the report
     */
    public function get_right_icons_row($rows = array()) {
        global $USER;

        return $rows;
    }

    /**
     * Builds and return the row of ranges for the right part of the custom grader report.
     * @param array $rows The Array of rows for the right part of the report
     * @return array Array of rows for the right part of the report
     */
    public function get_right_range_row($rows = array()) {
        global $OUTPUT;

        if ($this->get_pref('showranges')) {
            $rangesdisplaytype = $this->get_pref('rangesdisplaytype');
            $rangesdecimalpoints = $this->get_pref('rangesdecimalpoints');
            $rangerow = new html_table_row();
            $rangerow->attributes['class'] = 'heading range';

            foreach ($this->gtree->items as $itemid => $unused) {
                $item = & $this->gtree->items[$itemid];
                $itemcell = new html_table_cell();
                $itemcell->header = true;
                $itemcell->attributes['class'] .= ' header range';

                $hidden = '';
                if ($item->is_hidden()) {
                    $hidden = ' hidden ';
                }
```

```
                    $formattedrange = $item-
>get_formatted_range($rangesdisplaytype, $rangesdecimalpoints);

                    $itemcell->text = $OUTPUT->container($formattedrange,
'rangevalues' . $hidden);
                    $rangerow->cells[] = $itemcell;
                }
                $rows[] = $rangerow;
            }
            return $rows;
        }

        /**
         * Builds and return the row of averages for the right part of the
custom grader report.
         * @param array $rows Whether to return only group averages or all
averages.
         * @param bool $grouponly Whether to return only group averages
or all averages.
         * @return array Array of rows for the right part of the report
         */
        public function get_right_avg_row($rows = array(), $grouponly =
false) {
            global $CFG, $USER, $DB, $OUTPUT;

            if (!$this->canviewhidden) {
                // totals might be affected by hiding, if user can not see hidden
grades the aggregations might be altered
                // better not show them at all if user can not see all hidden
grades
                return $rows;
            }

            $showaverages = $this->get_pref('showaverages');
            $showaveragesgroup = $this->currentgroup && $showaverages;

            $averagesdisplaytype = $this->get_pref('averagesdisplaytype');
            $averagesdecimalpoints = $this-
>get_pref('averagesdecimalpoints');
            $meanselection = $this->get_pref('meanselection');
            $shownumberofgrades = $this->get_pref('shownumberofgrades');

            $avghtml = '';
            $avgcssclass = 'avg';
```

```
            if ($grouponly) {
                $straverage = get_string('groupavg', 'grades');
                $showaverages = $this->currentgroup && $this-
>get_pref('showaverages');
                $groupsql = $this->groupsql;
                $groupwheresql = $this->groupwheresql;
                $groupwheresqlparams = $this->groupwheresql_params;
                $avgcssclass = 'groupavg';
            } else {
                $straverage = get_string('overallaverage', 'grades');
                $showaverages = $this->get_pref('showaverages');
                $groupsql = "";
                $groupwheresql = "";
                $groupwheresqlparams = array();
            }

            if ($shownumberofgrades) {
                $straverage .= ' (' . get_string('submissions', 'grades') . ') ';
            }

            $totalcount = $this->get_numusers($grouponly);

            // Limit to users with a gradeable role.
            list($gradebookrolessql, $gradebookrolesparams) = $DB-
>get_in_or_equal(explode(',', $this->gradebookroles),
SQL_PARAMS_NAMED, 'grbr0');

            // Limit to users with an active enrollment.
            list($enrolledsql, $enrolledparams) = get_enrolled_sql($this-
>context);

            // We want to query both the current context and parent contexts.
            list($relatedctxsql, $relatedctxparams) = $DB-
>get_in_or_equal($this->context->get_parent_context_ids(true),
SQL_PARAMS_NAMED, 'relatedctx');

            $params = array_merge(array('courseid' => $this->courseid),
$gradebookrolesparams, $enrolledparams, $groupwheresqlparams,
$relatedctxparams);

            // Find sums of all grade items in course.
            $sql = "SELECT g.itemid, SUM(g.finalgrade) AS sum
                        FROM {grade_items} gi
                        JOIN {grade_grades} g ON g.itemid = gi.id
```

```
                    JOIN {user} u ON u.id = g.userid
                    JOIN ($enrolledsql) je ON je.id = u.id
                    JOIN (
                            SELECT DISTINCT ra.userid
                              FROM {role_assignments} ra
                             WHERE ra.roleid $gradebookrolessql
                               AND ra.contextid $relatedctxsql
                        ) rainner ON rainner.userid = u.id
                     $groupsql
                    WHERE gi.courseid = :courseid
                      AND u.deleted = 0
                      AND g.finalgrade IS NOT NULL
                      $groupwheresql
                    GROUP BY g.itemid";
            $sumarray = array();
            if ($sums = $DB->get_records_sql($sql, $params)) {
               foreach ($sums as $itemid => $csum) {
                  $sumarray[$itemid] = $csum->sum;
               }
            }

            // MDL-10875 Empty grades must be evaluated as grademin, NOT
always 0
            // This query returns a count of ungraded grades (NULL finalgrade
OR no matching record in grade_grades table)
            $sql = "SELECT gi.id, COUNT(DISTINCT u.id) AS count
                      FROM {grade_items} gi
                      CROSS JOIN {user} u
                      JOIN ($enrolledsql) je
                          ON je.id = u.id
                      JOIN {role_assignments} ra
                          ON ra.userid = u.id
                      LEFT OUTER JOIN {grade_grades} g
                          ON (g.itemid = gi.id AND g.userid = u.id AND
g.finalgrade IS NOT NULL)
                      $groupsql
                     WHERE gi.courseid = :courseid
                          AND ra.roleid $gradebookrolessql
                          AND ra.contextid $relatedctxsql
                          AND u.deleted = 0
                          AND g.id IS NULL
                          $groupwheresql
                    GROUP BY gi.id";

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

```
            $avgrow = new html_table_row();
            $avgrow->attributes['class'] = 'avg';

            foreach ($this->gtree->items as $itemid => $unused) {
                $item = & $this->gtree->items[$itemid];

                if ($item->needsupdate) {
                    $avgcell = new html_table_cell();
                    $avgcell->text = $OUTPUT->container(get_string('error'),
'gradingerror');
                    $avgrow->cells[] = $avgcell;
                    continue;
                }

                if (!isset($sumarray[$item->id])) {
                    $sumarray[$item->id] = 0;
                }

                if (empty($ungradedcounts[$itemid])) {
                    $ungradedcount = 0;
                } else {
                    $ungradedcount = $ungradedcounts[$itemid]->count;
                }

                if ($meanselection == GRADE_REPORT_MEAN_GRADED)
{
                    $meancount = $totalcount - $ungradedcount;
                } else { // Bump up the sum by the number of ungraded items *
grademin
                    $sumarray[$item->id] += $ungradedcount * $item-
>grademin;
                    $meancount = $totalcount;
                }

                $decimalpoints = $item->get_decimals();

                // Determine which display type to use for this average
                if ($averagesdisplaytype ==
GRADE_REPORT_PREFERENCE_INHERIT) { // no ==0 here, please
resave the report and user preferences
                    $displaytype = $item->get_displaytype();
                } else {
                    $displaytype = $averagesdisplaytype;
                }
```

```
                // Override grade_item setting if a display preference (not
inherit) was set for the averages
                if ($averagesdecimalpoints ==
GRADE_REPORT_PREFERENCE_INHERIT) {
                    $decimalpoints = $item->get_decimals();
                } else {
                    $decimalpoints = $averagesdecimalpoints;
                }

                if (!isset($sumarray[$item->id]) || $meancount == 0) {
                    $avgcell = new html_table_cell();
                    $avgcell->text = '-';
                    $avgrow->cells[] = $avgcell;
                } else {
                    $sum = $sumarray[$item->id];
                    $avgradeval = $sum / $meancount;
                    $gradehtml = grade_format_gradevalue($avgradeval, $item,
true, $displaytype, $decimalpoints);

                    $numberofgrades = '';
                    if ($shownumberofgrades) {
                        $numberofgrades = " ($meancount)";
                    }

                    $avgcell = new html_table_cell();
                    $avgcell->text = $gradehtml . $numberofgrades;
                    $avgrow->cells[] = $avgcell;
                }
            }
            $rows[] = $avgrow;
            return $rows;
        }

        /**
         * Given a grade_category, grade_item or grade_grade, this function
         * figures out the state of the object and builds then returns a div
         * with the icons needed for the grader report.
         *
         * @param array $object
         * @return string HTML
         */
        protected function get_icons($element) {
            global $CFG, $USER, $OUTPUT;
```

```
            // Init all icons
            $editicon = '';

            if ($element['type'] != 'categoryitem' && $element['type'] !=
'courseitem') {
                $editicon = $this->gtree->get_edit_icon($element, $this->gpr);
            }

            $editcalculationicon = '';
            $showhideicon = '';
            $lockunlockicon = '';

            if (has_capability('moodle/grade:manage', $this->context)) {
                if ($this->get_pref('showcalculations')) {
                    $editcalculationicon = $this->gtree-
>get_calculation_icon($element, $this->gpr);
                }

                if ($this->get_pref('showeyecons')) {
                    $showhideicon = $this->gtree->get_hiding_icon($element,
$this->gpr);
                }

                if ($this->get_pref('showlocks')) {
                    $lockunlockicon = $this->gtree->get_locking_icon($element,
$this->gpr);
                }

            }

            $gradeanalysisicon = '';
            if ($this->get_pref('showanalysisicon') && $element['type'] ==
'grade') {
                $gradeanalysisicon .= $this->gtree-
>get_grade_analysis_icon($element['object']);
            }
            return;
            //return $OUTPUT-
>container($editicon.$editcalculationicon.$showhideicon.$lockunlockicon.$g
radeanalysisicon, 'grade_icons');
        }

        /**
         * Given a category element returns collapsing +/- icon if available
         * @param object $object
```

```
     * @return string HTML
     */
    protected function get_collapsing_icon($element) {
        global $OUTPUT;

        $icon = '';
        // If object is a category, display expand/contract icon
        if ($element['type'] == 'category') {
            // Load language strings
            $strswitchminus = $this->get_lang_string('aggregatesonly',
'grades');
            $strswitchplus = $this->get_lang_string('gradesonly', 'grades');
            $strswitchwhole = $this->get_lang_string('fullmode', 'grades');

            $url = new moodle_url($this->gpr->get_return_url(null,
array('target' => $element['eid'], 'sesskey' => sesskey())));

            if (in_array($element['object']->id, $this-
>collapsed['aggregatesonly'])) {
                $url->param('action', 'switch_plus');
                $icon = $OUTPUT->action_icon($url, new
pix_icon('t/switch_plus', $strswitchplus));

            } else if (in_array($element['object']->id, $this-
>collapsed['gradesonly'])) {
                $url->param('action', 'switch_whole');
                $icon = $OUTPUT->action_icon($url, new
pix_icon('t/switch_whole', $strswitchwhole));

            } else {
                $url->param('action', 'switch_minus');
                $icon = $OUTPUT->action_icon($url, new
pix_icon('t/switch_minus', $strswitchminus));
            }
        }
        $icon = '';
        return $icon;
    }

    public function process_action($target, $action) {
        return self::do_process_action($target, $action);
    }

    /**
     * Processes a single action against a category, grade_item or grade.
```

```
             * @param string $target eid ({type}{id}, e.g. c4 for category4)
             * @param string $action Which action to take (edit, delete etc...)
             * @return
             */
            public static function do_process_action($target, $action) {
                // TODO: this code should be in some grade_tree static method
                $targettype = substr($target, 0, 1);
                $targetid = substr($target, 1);
                // TODO: end

                if ($collapsed =
get_user_preferences('grade_report_grader_collapsed_categories')) {
                    $collapsed = unserialize($collapsed);
                } else {
                    $collapsed = array('aggregatesonly' => array(), 'gradesonly' =>
array());
                }

                switch ($action) {
                    case 'switch_minus': // Add category to array of aggregatesonly
                        if (!in_array($targetid, $collapsed['aggregatesonly'])) {
                            $collapsed['aggregatesonly'][] = $targetid;

set_user_preference('grade_report_grader_collapsed_categories',
serialize($collapsed));
                        }
                        break;

                    case 'switch_plus': // Remove category from array of
aggregatesonly, and add it to array of gradesonly
                        $key = array_search($targetid, $collapsed['aggregatesonly']);
                        if ($key !== false) {
                            unset($collapsed['aggregatesonly'][$key]);
                        }
                        if (!in_array($targetid, $collapsed['gradesonly'])) {
                            $collapsed['gradesonly'][] = $targetid;
                        }

set_user_preference('grade_report_customgrader_collapsed_categories',
serialize($collapsed));
                        break;
                    case 'switch_whole': // Remove the category from the array of
collapsed cats
                        $key = array_search($targetid, $collapsed['gradesonly']);
                        if ($key !== false) {
```

```
                    unset($collapsed['gradesonly'][$key]);

set_user_preference('grade_report_customgrader_collapsed_categories',
serialize($collapsed));
                }

                break;
            default:
                break;
        }

        return true;
    }

    /**
     * Returns whether or not to display fixed students column.
     * Includes a browser check, because IE6 doesn't support the
scrollbar.
     *
     * @return bool
     */
    public function is_fixed_students() {
        global $USER, $CFG;
        return empty($USER->screenreader) && $CFG-
>grade_report_fixedstudents &&
                (check_browser_version('MSIE', '7.0') ||
                check_browser_version('Firefox', '2.0') ||
                check_browser_version('Gecko', '2006010100') ||
                check_browser_version('Camino', '1.0') ||
                check_browser_version('Opera', '6.0') ||
                check_browser_version('Chrome', '6') ||
                check_browser_version('Safari', '300'));
    }

    /**
     * Refactored function for generating HTML of sorting links with
matching arrows.
     * Returns an array with 'studentname' and 'idnumber' as keys, with
HTML ready
     * to inject into a table header cell.
     * @param array $extrafields Array of extra fields being displayed,
such as
     *   user idnumber
     * @return array An associative array of HTML sorting links+arrows
     */
```

```
public function get_sort_arrows(array $extrafields = array()) {
    global $OUTPUT;
    $arrows = array();

    $strsortasc = $this->get_lang_string('sortasc', 'grades');
    $strsortdesc = $this->get_lang_string('sortdesc', 'grades');
    $strfirstname = $this->get_lang_string('firstname');
    $strlastname = $this->get_lang_string('lastname');

    $firstlink = $strfirstname;
    $lastlink = $strlastname;

    $arrows['studentname'] = $lastlink;

    if ($this->sortitemid === 'lastname') {
        if ($this->sortorder == 'ASC') {
            $arrows['studentname'] ;
        } else {
            $arrows['studentname'] ;
        }
    }

    $arrows['studentname'] .= ' ' . $firstlink;

    if ($this->sortitemid === 'firstname') {
        if ($this->sortorder == 'ASC') {
            $arrows['studentname'] ;
        } else {
            $arrows['studentname'] ;
        }
    }

    foreach ($extrafields as $field) {
        $fieldlink =  get_user_field_name($field);
        $arrows[$field] = $fieldlink;

        if ($field == $this->sortitemid) {
            if ($this->sortorder == 'ASC') {
                $arrows[$field] ;
            } else {
                $arrows[$field] ;
            }
        }
    }
```

```
            return $arrows;
        }

        /**
         * Returns the maximum number of students to be displayed on each
page
         *
         * Takes into account the 'studentsperpage' user preference and the
'max_input_vars'
         * PHP setting. Too many fields is only a problem when submitting
grades but
         * we respect 'max_input_vars' even when viewing grades to prevent
students disappearing
         * when toggling editing on and off.
         *
         * @return int The maximum number of students to display per page
         */
        public function get_students_per_page() {
            global $USER;
            static $studentsperpage = null;

            if ($studentsperpage === null) {
                $originalstudentsperpage = $studentsperpage = $this-
>get_pref('studentsperpage');

                // Will this number of students result in more fields that we are
allowed?
                $maxinputvars = ini_get('max_input_vars');
                if ($maxinputvars !== false) {
                    $fieldspergradeitem = 0; // The number of fields output per
grade item for each student

                    if ($this->get_pref('quickgrading')) {
                        // One grade field
                        $fieldspergradeitem++;
                    }
                    if ($this->get_pref('showquickfeedback')) {
                        // One feedback field
                        $fieldspergradeitem++;
                    }

                    $fieldsperstudent = $fieldspergradeitem * count($this->gtree-
>get_items());
                    $fieldsrequired = $studentsperpage * $fieldsperstudent;
                    if ($fieldsrequired > $maxinputvars) {
```

```
                    $studentsperpage = floor($maxinputvars /
$fieldsperstudent);
                    if ($studentsperpage < 1) {
                        // Make sure students per page doesn't fall below 1
                        // PHP max_input_vars could potentially be reached with
1 student
                        // if there are >500 grade items and quickgrading and
showquickfeedback are on
                        $studentsperpage = 1;
                    }

                    $a = new stdClass();
                    $a->originalstudentsperpage = $originalstudentsperpage;
                    $a->studentsperpage = $studentsperpage;
                    $a->maxinputvars = $maxinputvars;
                    debugging(get_string('studentsperpagereduced', 'grades',
$a));
                }
            }
        }
        return $studentsperpage;
    }
}


<?php

defined('MOODLE_INTERNAL') || die;

require_once($CFG->libdir.'/formslib.php');
require_once($CFG->libdir.'/coursecatlib.php');

/**
 * Edit category form.
 *
 * @package core_course
 * @copyright 2002 onwards Martin Dougiamas
(http://dougiamas.com)
 * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */
class customgraderreport_form extends moodleform {

    /**
     * The form definition.
     */
```

```
        public function definition() {
            global $CFG, $DB;
            $mform = $this->_form;
            $parent = $this->_customdata['parent'];
            $cohorts = $this->_customdata['cohorts'];
            $grade_categories = $this->_customdata['grade_categories'];

            // Get list of categories to use as parents, with site as the first one.
            $options = array();
            if (has_capability('moodle/category:manage',
context_system::instance()) || $parent == 0) {
                $options[0] = get_string('top', 'gradereport_customgrader');
            }
            $options +=
coursecat::make_categories_list('moodle/category:manage');

            $mform->addElement('select', 'course_category',
get_string('category', 'gradereport_customgrader'), $options);

            $mform->addElement('select', 'cohort', get_string('cohort',
'gradereport_customgrader'), $cohorts);

            $mform->addElement('select', 'grade_category',
get_string('grade_category', 'gradereport_customgrader'), $grade_categories);

            $this->add_action_buttons(false, get_string('submit',
'gradereport_customgrader'));
        }

        /**
         * Returns the description editor options.
         * @return array
         */
        public function get_description_editor_options() {
            global $CFG;
            $context = $this->_customdata['context'];
            $itemid = $this->_customdata['itemid'];
            return array(
                'maxfiles'  => EDITOR_UNLIMITED_FILES,
                'maxbytes'  => $CFG->maxbytes,
                'trusttext' => true,
                'context'   => $context,
                'subdirs'   => file_area_contains_subdirs($context, 'coursecat',
'description', $itemid),
            );
```

```
        }

        /**
         * Validates the data submit for this form.
         *
         * @param array $data An array of key,value data pairs.
         * @param array $files Any files that may have been submit as well.
         * @return array An array of errors.
         */
        public function validation($data, $files) {
            global $DB;
            $errors = parent::validation($data, $files);
            if (!empty($data['idnumber'])) {
                if ($existing = $DB->get_record('course_categories',
array('idnumber' => $data['idnumber']))) {
                    if (!$data['id'] || $existing->id != $data['id']) {
                        $errors['idnumber'] = get_string('categoryidnumbertaken',
'error');
                    }
                }
            }
            return $errors;
        }
    }

    <?php

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

    $capabilities = array(
        'gradereport/customgrader:view' => array(
            'riskbitmask' => RISK_PERSONAL,
            'captype' => 'read',
            'contextlevel' => CONTEXT_COURSE,
            'archetypes' => array(
                'teacher' => CAP_ALLOW,
                'editingteacher' => CAP_ALLOW,
                'manager' => CAP_ALLOW
            )
        )
    );
```

Заключительный лист работы

Дипломная работа выполнена мною самостоятельно. Использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

Список использованной литературы (источников): 12 наименований.

Работа выполнена на 38 листах.

Приложения к работе на 40 листах.

Один экземпляр сдан на кафедру.

Подпись_____/_____
(фамилия, инициалы)

Дата «_____» _____ 20___г.