РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ «КИМ ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ»

Жуков А.О.,
Волкова Т.И., к. п. н., доцент
г. Бирск, ФГБОУ ВПО БирГСПА

В каждой области жизнедеятельности человека свою нишу заняли информационные системы и программные продукты со списком качеств и требований, отражающих специфику соответствующей предметной сферы. В образовательных учреждениях активно внедряются автоматизированные распределенные информационные системы для организации оперативного контроля знаний.

Особую актуальность приобретает задача разработки веб-приложений, направленных на повышение уровня информатизации и автоматизации управления системой оценки качества образования в школе. Основное направление решения этой задачи связано с созданием единого банка контрольно-измерительных материалов (КИМ) школы. Такие информационные системы на уровне образовательных учреждений в настоящее время практически отсутствуют, что и определило выбор темы разрабатываемого проекта. Практическое значение исследования направлено на повышение уровня информатизации и автоматизации учебных процессов в образовательном учреждении, а так же упрощение взаимодействия преподавателей между собой благодаря введению автоматизированной инструментальной системы.

Социально-экономические преобразования в стране повлекли за собой изменения в сфере российского образования. Одним из оснований обновления образования можно считать компетентностный подход. Предметные ***образовательные компетенции*** характеризуют уровень развития личности обучаемого, связанный с качественным освоением содержания по предмету, что определяется требованиями и нормами к подготовке выпускника школы и ВУЗа. Предметные образовательные компетенции отражены в ***контрольно – измерительных материалах (КИМ),*** которые выступают одним из средств формирования предметных знаний и навыков обучаемого.

Контрольно-измерительные материалы являются неотъемлемой частью любого предмета и служат для выявления успехов и достижений ученика в учебном процессе. Успешность ученика может определяться по различным направлениям. Эти направления определяют цели контроля и связаны, по существу, с диагностируемыми качествами

успешности ученика. В КИМ содержание предмета представлено материалом различных содержательных линий и тем. Темы могут иметь различное значение в учебном процессе: основной материал, ознакомительный, дополнительный и т.д.

Введение КИМ в нормативную и практическую составляющую образования позволяет решать проблему, когда ученики могут хорошо владеть набором теоретических знаний, но испытывают значительные трудности в деятельности, требующей использования этих знаний для решения конкретных жизненных задач или проблемных ситуаций.

При разработке и проектировании информационной системы необходимо придерживаться следующих общепринятых требований и стандартов:

- централизация данных в единой базе;
- близкий к реальному времени режим работы;
- сохранение общей модели управления для различных образовательных учреждений;
- поддержка территориально-распределенных структур;
- работа на широком круге аппаратно-программных платформ и СУБД.

Прототипирование программного продукта «КИМ-ОБРАЗ»

Разрабатываемый программный продукт «КИМ-ОБРАЗ» должен решать следующие задачи:

- Централизованное хранение и удобный доступ к КИМ;
- Упрощение взаимодействия преподавателей между собой путем легкого обмена авторскими материалами;
- Автоматизация генерации билетов для проведения контроля по определенным темам.

А также удовлетворять следующим ***критериям:***

- простое развертывание программного продукта на сервере
- удобное управление системой в целом
- безопасность и целостность данных

Централизованное хранение банка КИМ

Для централизованного хранения контрольно измерительных материалов необходимо использовать реляционную базу данных. Существует множество баз и СУБД, но наиболее распространенной является MySQL в виду ее гибкости, стабильности и быстродействия, а также открытости программного обеспечения, что является большим плюсом при проектировании программных продуктов для образовательных учреждений.

В БД необходимо хранить сами вопросы (задания), темы и предметы к которым они относятся. Для этого необходимо будет создать несколько таблиц с соответствующими связями. Также было бы актуально создать отдельную таблицу для записи всех параметров настроек, которые задает администратор системы. К этим параметрам мы относим логин и

пароль главного пользователя (администратора), настройки интерфейса системы, приветственный текст главной страницы и др.

Проектирование интерфейса. Формы, навигация и диалоговые окна.

Структурно-функциональная схема информационной системы приведена на рис.1. В программном продукте необходимо логически разделить доступ для администратора, который управляет системой (вносит изменения в продукт, добавляет и редактирует задания) и пользователя (генерирует билеты, используя централизованную базу КИМ).

Рис.1 Структурно-функциональная схема системы

Для ***администратора*** необходимо предусмотреть страницы (формы): добавление/редактирования списка предметов, тем и заданий. Для форматирования текста задания важно добавить удобную панель.

Все действия ***пользователя*** можно разбить на 4 условных шага:

- выбор предмета
- наполнение билета заданиями (выбор тем и уровней сложности)
- дополнительные опции (количество вариантов и шаблон для печати)
- сохранение результата

Для каждого шага создадим окно с соответствующими полями и формами.

Генерация билета с заданиями

После прохождения 4-х шагов необходимо сгенерировать для пользователя билет с заданиями, либо несколько билетов, если на третьем шаге было указано количество вариантов более 2х. Для получения заданий из базы будем использовать SQL-запросы, а затем, используя «хитрый» алгоритм сортировки и выборки, сгенерируем нужный нам результат. Останется только собрать все вместе по шаблону и выдать билет пользователю.

Структура кода

Существует множество концепций написания и организации программного кода. Для веб-приложений наиболее распространена MVC модель, поэтому будем использовать именно ее. MVC (Model-View-Controller) предполагает разделение программного кода на 3 основные группы: классы и методы для получения данных из таблиц БД, визуальные шаблоны представления программы, классы для связи пользователя с системой. При написании приложения будем придерживаться этой концепции.

Использование связки PHP + MySQL на веб-сервере Apache

Проект выполнен на языке программирования **PHP**, который установлен на web-сервере ***Apache***. Для хранения данных используется база данных MySQL.

Основная задача веб-сервера - это ожидание запросов от клиентов и отправка ответов. Взаимодействие с клиентами происходит по протоколу HTTP. Клиент (веб-браузер) запрашивает ресурс. Сервер связывает запрос с файлом или направляет запрос программе, которая генерирует необходимые данные. После этого сервер отсылает ответ обратно клиенту. *(рис.2)*

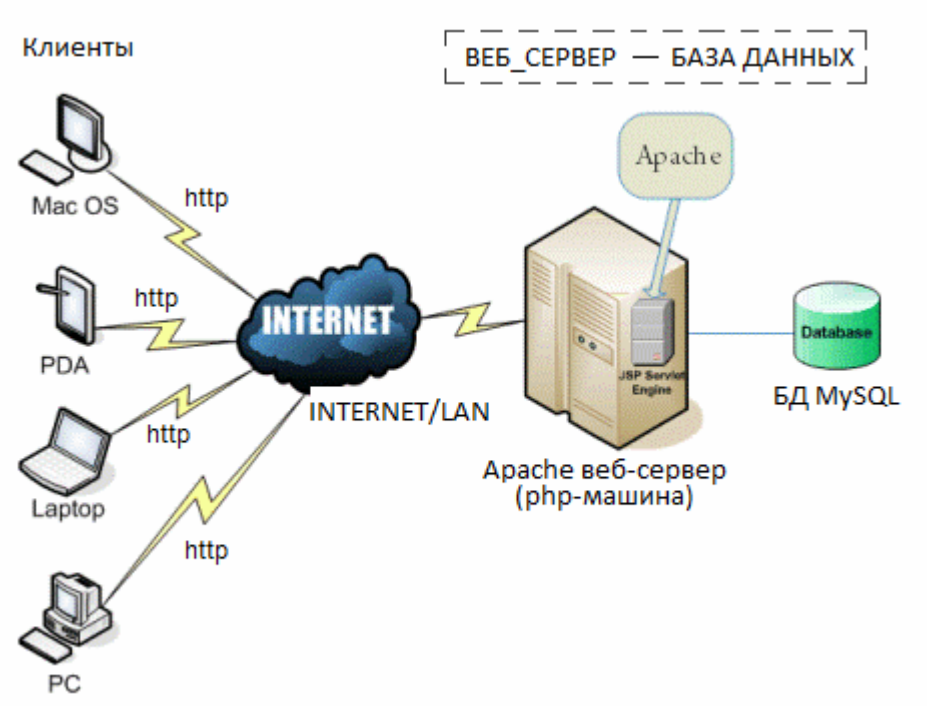

Рис 2. веб-сервер Apache и клиенты

Для предоставления данных из ***MySQL*** базы данных используется web-сервер Apache, который поддерживает язык ***PHP***, поэтому сам проект написан с помощью этого языка. Проект реализован в соответствии с популярной концепцией ***MVC***.

MVC модель взаимодействия

Model-view-controller (MVC, ***«Модель-представление-поведение»***, «Модель-представление-контроллер») — схема использования нескольких шаблонов проектирования, с помощью которых модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные. Данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области.

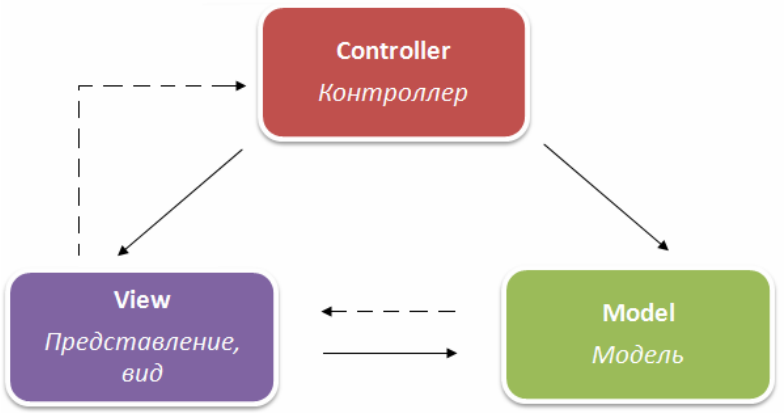

Рис 3. Концепция Model-View-Controller

Концепция Model-View-Controller. *(рис.2)* Сплошными линиями показаны прямые связи (вызовы методов, присвоение значений полей), прерывистыми линиями показаны косвенные связи (сообщения через события).

Концепция **MVC** позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

- ***Модель*** (англ. Model). Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.
- ***Представление***, вид (англ. View). Отвечает за отображение информации (визуализация). Часто в качестве представления выступает форма (окно) с графическими элементами.
- ***Контроллер*** (англ. Controller). Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

Так же в проекте выделены модели, контроллеры и представления, которые хранятся в отдельных файлах ***php*** на сервере.

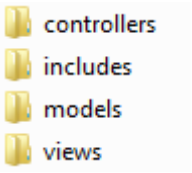

Рис 4. Структура каталогов проекта

Все запросы браузера направляются на файл index.php, который узнает какой именно контроллер нужно грузить. Подгруженный контроллер использует модели для получения данных из БД, и, обрабатывая данные, грузит их в какое-либо представление. Например, в контроллере *main_page* функция *subjects_show_all* берет данные из модели *subjects* (список всех тем и вопросов), а затем отправляет эти данные в представление *subjects*. В модели происходит выборка нужных данных из БД с помощью *SQL-запроса*, а в представлении полученные данные получают «оболочку» с помощью HTML-тэгов. Таким образом, контроллер отдает браузеру пользователя конечный результат в виде готовой HTML-страницы, в которой содержится нужная информация.

В проекте используются 4 контроллера: error_page_403, error_page_403, main_page, admin_panel. Первые два контроллера предназначены для страниц ошибок соответственно 403-ей и 404-ой. Например, если на сервере не будет запрашиваемой страницы, то пользователь получит результат выполнения контроллера 404. 403 – в случае запроса страницы или каталога, к которому у пользователя нет доступа. Контроллер ***main_page*** – главный контроллер, который содержит функции, отвечающие за работу всего проекта. Контроллер ***admin_panel*** — содержит функции для добавления и редактирования предметов, списка тем и вопросов.

Для того, чтобы пользователь увидел результат работы какой-либо функции в контроллере используется технология ***AJAX***, суть которой состоит в подгрузке данных без перезагрузки всей страницы. Эта технология представлена в виде подключаемой библиотеки ***XAJAX***.

Пример работы такой системы: пользователь кликает по кнопке, на обработчике которой висит функция XAJAX. Эта функция генерирует запрос к серверу, который направляется на контроллер ***main_page***. А контроллер, понимая от какой функции поступил запрос, возвращает тот или иной результат в виде HTML.

Таким образом реализована концепция ***MVC***.

Модели в проекте служат для получения данных из БД, а также реализуют некоторые функции по работе с сущностями, хранимыми в БД. Всего моделей 4: subjects, themes, questions, settings. Модели subjects и themes обеспечивает получение данных о предметах и

темах. Модели questions и settings – соответственно о настройках, заданиях и дополнительных параметрах.

Представления лежат в папке ***views*** на сервере *(рис. 5)*.

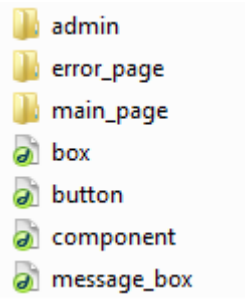

Рис 5. Содержание каталога «представление»

Структура БД. Проектирование и обзор таблиц

База данных состоит из 5 таблиц: subjects, themes, questions, settings и patterns. *(рис. 6)* 3 таблицы связаны по ключу, 2 не имеют связей.

Таблица subjects содержит информацию о существующих предметах, из которых можно выбрать задания. В таблице settings хранятся общие настройки: настройки оформления, текст приветствия, пароль администратора. Названия тем хранятся в таблице themes, а вопросы - в questions. Каждому предмету соответствуют некоторые темы, которые необходимо выбрать, чтобы получить список вопросов (таблица questions).

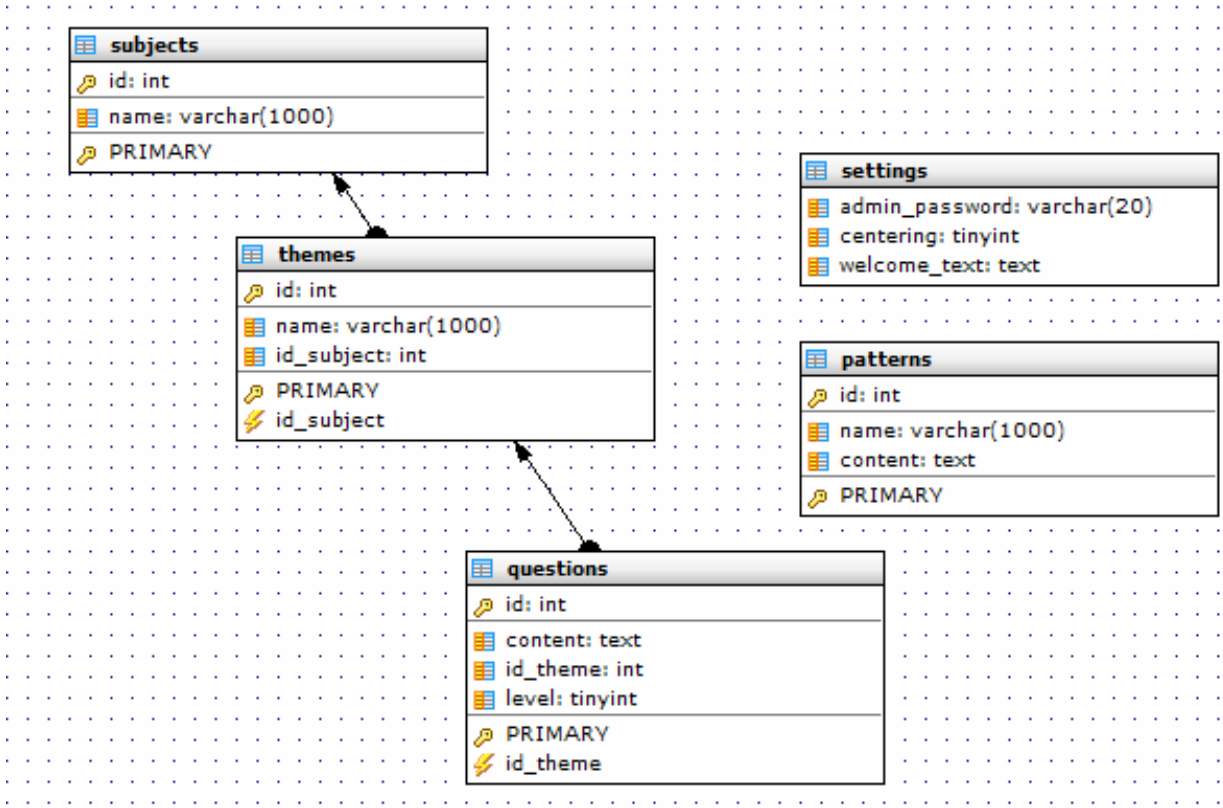

Рис 6. Структура таблиц БД

SQL-запросы к БД. Выборка вопросов из базы

Для работы с базой заданий были написаны несколько основных функций, содержащие SQL-запросы. Программный код функций располагается в моделях, так как согласно концепции MVC именно они осуществляют SQL-запросы к БД. Рассмотрим их вид и модификации.

Есть три типа функций:

1. Получение списка заданий/тем/предметов и добавление данных в массив.
2. Добавление нового задания/темы/предмета в базу.
3. Внесение изменений в существующее задание/название темы.

Функция получения всех заданий с темой имеющей id№ (id_theme)

```
public static function getAllWithIdTheme($id_theme)
{
	$res = mysql_query("SELECT * FROM questions WHERE id_theme=$id_theme ORDER BY id DESC");
	$array = array();
	while ($mas = mysql_fetch_array($res))
	{
		$array [] = $mas;
	}
	return $array;
}
```

Функция содержит SQL-запрос:

*mysql_query("SELECT * FROM questions WHERE id_theme=$id_theme ORDER BY id DESC");*

То есть получаем все записи из таблицы QUESTIONS, где номер темы равен переданному параметру. Полученные данные возвращаем как массив данных $array. Данную операцию выполняем каждый раз, когда необходимо получить все задания для последующей выборки при создании уникального билета.

Ее модификацией является ***функция получения заданий*** с указанием ***темы*** вопроса ***и уровня сложности***.

```
public static function getAllWithIdThemeAndLevel($id_theme,$level)
	{
		$res = mysql_query("SELECT * FROM questions WHERE id_theme=$id_theme and
level=$level ORDER BY id DESC");
		$array = array();
		while ($mas = mysql_fetch_array($res))
		{
			$array [] = $mas;
		}
		return $array;
	}
```

Соответственно SQL-запрос имеет вид:

*mysql_query("SELECT * FROM questions WHERE id_theme=$id_theme and level=$level ORDER BY id DESC");*

Для ***добавления нового вопроса*** в базу данных выполняется sql-запрос вида:

mysql_query("INSERT INTO questions (content,id_theme,level) VALUES ('$content',$id_theme,$level)");

В качестве параметров передаем: содержание задания (форматированный html-код), id темы и уровень сложности. Номер вопроса (id_question) определяется автоматически и вычисляется по правилу *i++*. Все данные записываем в таблицу ***questions***.

Удаление вопроса из БД осуществляется выполнением функции

```
public static function delete($id)
{
	mysql_query("DELETE FROM questions WHERE id=$id LIMIT 1");
}
```

Функция удаляет вопрос с номером (id), который мы передаем в параметрах.

Чтобы ***изменить задание*** выполняется функция update. В параметрах передаем id вопроса, тело (содержание) задания, уровень сложности. Эти параметры используются при выполнении sql-запроса:

mysql_query("UPDATE questions SET content='$content',level=$level WHERE id=$id LIMIT 1");

Получить список всех предметов можно с помощью ***функции getAll***.

```
public static function getAll()
	{
		$res = mysql_query("SELECT * FROM subjects ORDER BY id DESC");
		$array = array();
		while ($mas = mysql_fetch_array($res))
		{
			$array [] = $mas;
		}
		return $array;
	}
```

Для добавления предмета **add**

```
public static function add($name)
	{
		mysql_query("INSERT INTO subjects (name) VALUES ('$name')");
	}
```

Функция ***add*** принимает единственный параметр – **название предмета**. Id предмета выставляется автоматически.

Алгоритм генерации уникальных билетов

Особенностью программного продукта является алгоритм, позволяющий генерировать билеты с максимальной уникальностью. Это значит, что задания в разных вариантах (билетах) повторяются минимум раз из возможных.

Данный алгоритм не использует функции типа ***random*** для получения случайного задания. Для этой задачи создается общий массив, в который заносятся все задания, удовлетворяющие запросу. Далее создаются массивы вопросов для каждого билета, в которые происходит единичная выборка заданий.

Алгоритм работы рассмотрен ниже:

Пример запроса:

Задание №1	Тема **1**	Ур. сложности **1**
Задание №2	Тема **1**	Ур. сложности **2**
Задание №3	Тема **2**	Ур. сложности **1**
Задание №4	Тема **1**	Ур. сложности **1**

Выбираем одинаковые запросы по теме и уровню сложности и *получаем* ***общий массив заданий***.

Тема 1	Ур. сложности **1**	B1	B2	B3	B4	B5	B6	B7			
Тема 1	Ур. сложности **2**	B1	B2	B3	B4	B5					
Тема 2	Ур. сложности **1**	B1	B2	B3	B4	B5	B6	B7	B8		

Когда общий массив заданий получен, приступаем к выборке заданий. Начинаем с I варианта: как только вопрос приписывается к билету (варианту), мы удаляем его из общего массива заданий, чтобы он не был повторно использован в другом билете.

Вариант 1	Задание **№1**	**Тема1 У1**	**B1** Т1У1
	Задание **№2**	**Тема1 У2**	**B1** Т1У2
	Задание **№3**	**Тема2 У1**	**B1** Т2У1
	Задание **№4**	**Тема1 У1**	**B2** Т1У1
Вариант 2	Задание **№1**	**Тема1 У1**	**B3** Т1У1
	Задание **№2**	**Тема1 У2**	**B2** Т1У2
	Задание **№3**	**Тема2 У1**	**B2** Т2У1
	Задание **№4**	**Тема1 У1**	**B4** Т1У1

Таким образом мы можем получить максимальное количество уникальных вариантов билетов.

Описание разработанного продукта

Структура страниц программного продукта

В процессе работы с программным продуктом пользователь линейно перемещается по двум веткам: Генерирование билета (4 шага) и Панели администрирования с ее подразделами. *(Рис.7)* Обе ветки прозрачно связаны сквозными ссылками внизу каждой страницы.

Общий вид структуры программы представлен на схеме ниже:

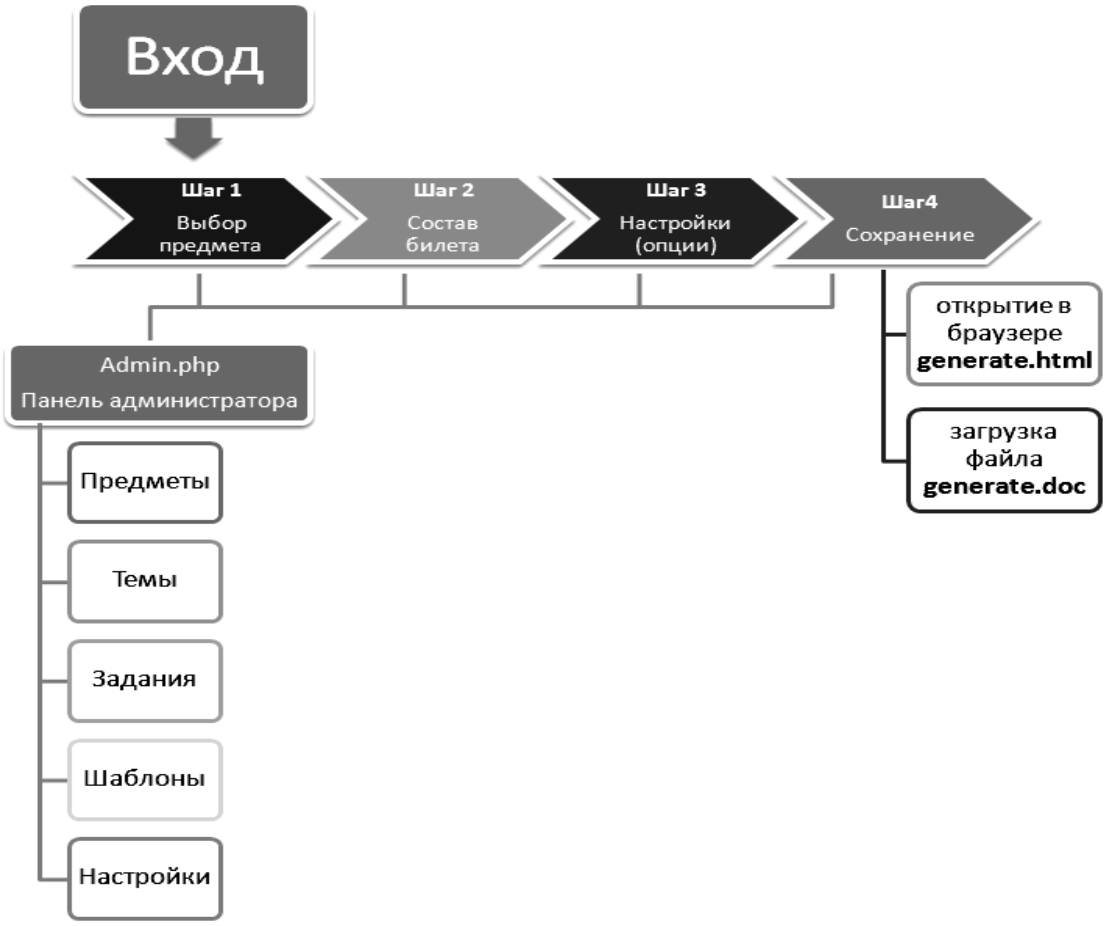

Рис 7. Структура страниц программного продукта

«4 шага создания билета»

Процесс формирования билетов с заданиями состоит из **4-х** шагов (рис.8):

Рис 8. «4 шага создания билета»

Выбор предмета. На этом этапе пользователю предлагается выбрать предмет, по которому он хочет сформировать билеты (рис 9.):.

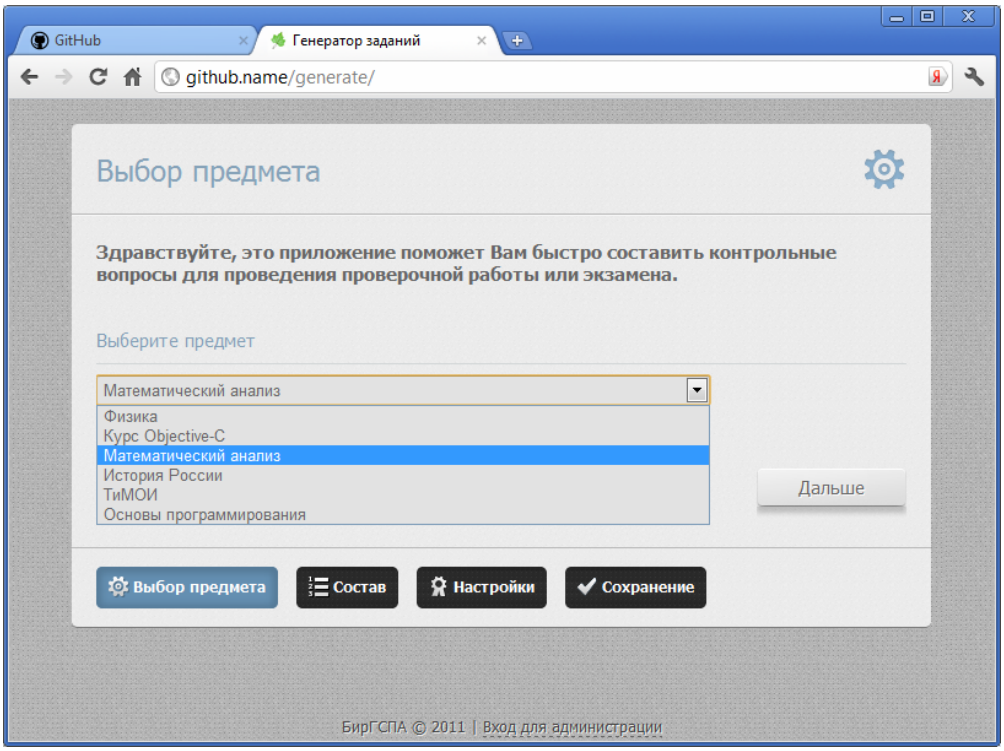

Рис 9. Начальная страница (Выбор предмета)

Список предметов формируется автоматически исходя из заполненной базы с темами и заданиями.

На втором шаге необходимо определить ***состав билета (рис.10)***. Здесь можно выбрать тему каждого задания, уровень сложности и общее количество вопросов в билете.

Если вопросы с выбранными критериями не будут найдены, система сообщит об ошибке и выведет номер задания, в который необходимо внести изменения.

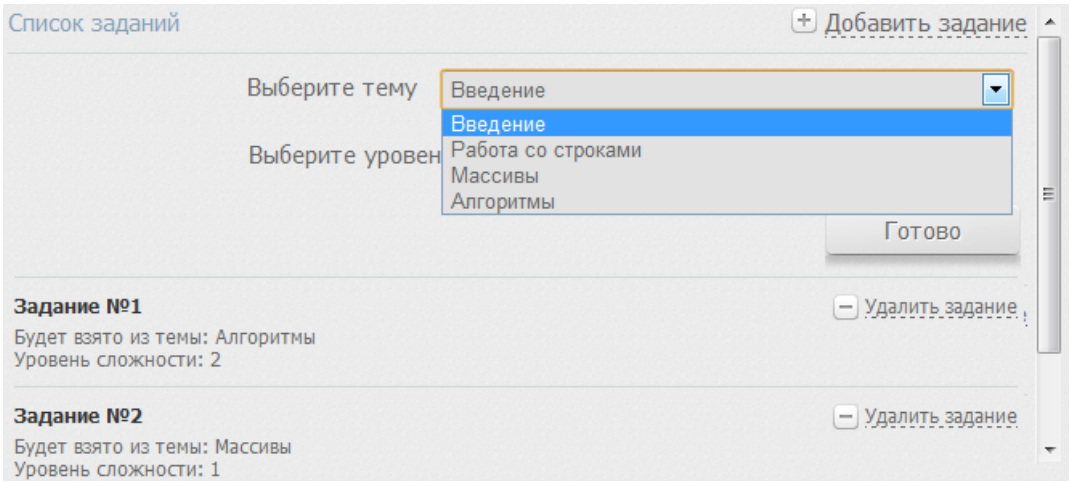

Рис 10. Состав билета

После формирования состава билета, можно приступить к третьему шагу – ***опциям (настройкам) заданий***. Необходимо выбрать шаблон и количество генерируемых вариантов заданий. Опционально можно указать фамилию преподавателя, которая будет выведена в конце каждого билета.

4й шаг – «сохранение». Пользователю необходимо выбрать, в каком виде он хочет получить результат: открыть в новом окне браузера, либо сохранить в формате Word (рис.11).

При выборе первого варианта откроется новая вкладка со списком уникальных заданий по каждой теме, при выборе сохранения данных в формате Word билеты будут конвертированы в .doc-формат.

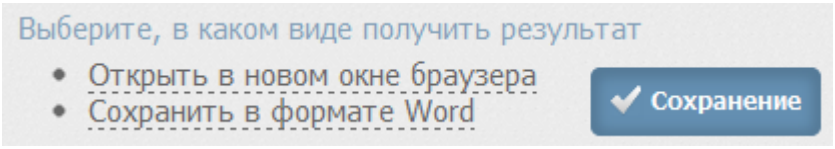

Рис 11. Сохранение и экспорт в .doc

Обзор панели администрирования

Панель администратора предназначена для удобного заполнения базы данных с заданиями. Панель предусматривает редактирование списка предметов, добавление/удаление тем заданий, добавление вопросов по каждой теме с указанием уровня сложности. Администратор может редактировать внешний вид шаблонов генерируемых билетов, а так же модифицировать внешний вид программного продукта. Для входа в панель необходимо перейти по ссылке в нижней части страницы и указать логин и пароль. Пароль можно изменить в любое время в разделе общие настройки панели администрирования.

Панель администрирования состоит из ***5 основных разделов***:

Предметы – раздел предназначен для редактирования общего списка предметов. Возможен вариант «быстрого редактирования» названия предмета без перезагрузки страницы.

Темы – позволяет добавлять/удалять темы по каждому предмету.

Задания – основной раздел, который необходим для заполнения базы данных с заданиями. Позволяет добавлять вопросы, редактировать внешний вид задания, а так же прикреплять изображения и формулы к вопросам. Для удобства поле ввода текста оснащено визуальным редактором в виде JavaScript библиотеки.

Шаблоны – в этом разделе администратор может модифицировать шаблоны билетов. Администратор может создать большое количество шаблонов на каждый случай: экзамен, контрольная, проверочная, промежуточный контроль знаний.

Настройки – позволяют редактировать текст приветствия на главной странице, а так же изменить пароль для входа в панель администрирования.

Информационная система «КИМ-ОБРАЗ» прошла опытно-экспериментальную проверку на физико-математическом факультете БирГСПА. Экспериментальная работа продемонстрировала базовые возможности информационной системы: генерирование уникальных билетов с заданиями по выбранным темам и уровню сложности, определенному преподавателем при работе с системой, а также возможность формирования базы заданий в удобном для пользователя виде. Программный продукт позволяет работать как локально на одном ПК, так и в многопользовательском режиме по локальной сети или сети Интернет. В дальнейшем планируется использовать ее в масштабе всего вуза для комплексной экспертизы качества контрольно-измерительных материалов, хранения и регламентированного использования контрольно-измерительных материалов для мониторинга и оценки качества образовательных достижений студентов, уровня сформированности ключевых компетенций в соответствии с требованиями стандартов третьего поколения. Заложенная в системе функциональность позволяет использовать ее в качестве инструмента для создания единого автоматизированного банка КИМ любого образовательного учреждения.

Литература

1. PHP5 Полное руководство/ Д.С. Зольников. – М.:НТ Пресс, 2007. -256с.
2. Панкова, Е.В. Разработка веб-приложения для среднего профессионального учебного заведения: эффективность алгоритмов при проектировании [Текст] / Панкова Е.В., Косинова С. А. // Научные и технические библиотеки. – 2008. – №3. – С. 41-44
3. Режепп, А. Типичные ошибки при создании корпоративных web-порталов [Текст] / А. Режепп, Ю. Степанов, О. Павлова // Мир Internet. – 2001. – №2. – С. 70-73