

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики

Лاپин Алексей Сергеевич, группа БИВ-121

**РАЗРАБОТКА ВЕБ-ОРИЕНТИРОВАННОЙ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ
ДЛЯ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ ЛАБОРАТОРИИ**

Выпускная квалификационная работа
по направлению 09.03.01 Информатика и вычислительная техника
студента образовательной программы бакалавриата
«Информатика и вычислительная техника»

Студент

А.С. Лапин

подпись

Рецензент
к.т.н., ст. преподаватель факультета
Бизнеса и менеджмента НИУ ВШЭ
С.Г. Ефремов

Научный руководитель
ассистент МИЭМ НИУ ВШЭ
А.А. Дворников

Москва 2016 г.

ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

«УТВЕРЖДАЮ»

Академический руководитель образовательной
программы

Информатика и
вычислительная техника
Поталова

Т.А.Поталова

« 20 » декабря 2015 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы БИВ121 Лапину Алексею Сергеевичу

1. Тема работы

Разработка веб-ориентированной информационно-аналитической
системы для научно-исследовательской лаборатории

2. Требования к работе

Требования к составу и параметрам технических средств:

Серверные технологии:

- PHP 5.4

Клиентские технологии:

- JavaScript
- HTML
- CSS

Системы управления базами данных:

- MySQL

Требования к программной совместимости

Информационно-аналитическая система должна корректно отображаться
в следующих версиях браузеров:

- Google Chrome 47 и выше
- Opera 34 и выше
- Mozilla Firefox 43 и выше

Требования к условиям эксплуатации объекта

Программные модули информационно-аналитической системы должны
быть защищены от несанкционированного доступа к возможности их
изменения/редактирования.

3. Содержание работы

- 3.1. Анализ существующих технических решений в предметной области
- 3.2. Подготовка технического задания на объект разработки
- 3.3. Обоснование выбора методов решения поставленных задач
- 3.4. Обоснование выбора инструментальных и программных средств для решения задач ВКР
- 3.5. Разработка прикладного программного обеспечения объекта разработки
- 3.6. Создание пробной версии программного обеспечения объекта разработки
- 3.7. Экспериментальные исследования/тестирование объекта разработки

4. Сроки выполнения этапов работы

Проект ВКР представляется студентом в срок до «24» сентября 2016г.

Первый вариант ВКР представляется студентом в срок до «27» октября 2016г.

Итоговый вариант ВКР представляется студентом руководителю до загрузки работы в систему «Антиплагиат» в срок до «25» ноября 2016г.

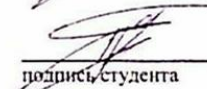
Задание выдано

«18» сентября 2015г.


подпись научного руководителя А.А. Дворников

Задание принято к
исполнению

«18» сентября 2015г.


подпись студента А.С. Лапин

ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

**График сдачи этапов
выпускной квалификационной работы бакалавра**

студента группы БИВ-121 Лапина Алексея Сергеевича

Тема работы

Разработка веб-ориентированной информационно-аналитической системы для
научно-исследовательской лаборатории

Дата представления проекта
ВКР

«15» февраля 2016 г.

_____ А.А. Дворников
подпись научного руководителя

Дата представления первого
варианта ВКР

«25» апреля 2016 г.

_____ А.А. Дворников
подпись научного руководителя

Дата представления итогового
варианта ВКР

«22» мая 2016 г.

_____ А.А. Дворников
подпись научного руководителя

Аннотация

Объектом разработки выпускной квалификационной работы является веб-ориентированная информационно-аналитическая система для межвузовской лаборатории инновационных проектов.

Целью работы является создание информационно-аналитической системы для размещения в сети Интернет, предназначенной для организации более продуктивной деятельности лаборатории, предоставления целевой аудитории информации по публикациями, проектам и курсам лаборатории, получения обратной связи от пользователей, а также для возможности online заказов проектов и подачи заявок на прохождение курсов.

В результате проведенной работы создан сервис на основе технологии PHP и MySQL, удовлетворяющий поставленной задаче.

Пояснительная записка к работе содержит 77 страниц, 25 иллюстраций, 10 таблиц, 14 источников информации и 2 приложения.

Annotation

The object of development for my graduation work is web-oriented information and analytical system for interuniversity lab of innovative projects.

The goal of the work is the creation of the web-based system to deploy in the Internet, which is aimed at organizing more productive activity of laboratory, give the target audience all information about publications, projects and laboratory courses, receiving user feedback and ordering projects online together with sending application for courses.

As a result, the service was created, based on PHP and MYSQL technologies, which completely fit the assigned task.

Explanatory note to this work contains 77 pages, 25 illustrations, 10 tables, 14 information sources and 2 applications.

Оглавление

Введение	9
1 Анализ современных методов разработки веб-систем	11
1.1 Методы разработки веб-систем	11
1.1.1 Онлайн-конструкторы	11
1.1.2 CMS	14
1.1.3 Разработка сайта с «нуля»	18
1.1.4 Использование фреймворка	19
1.2 Обоснование выбора метода решения поставленной задачи	21
1.3 Обоснование выбора языка программирования	22
1.4 Обоснование выбора фреймворка	24
2 Описание требований к объекту разработки	26
2.1 Введение	26
2.2 Требования к программе	26
2.2.1 Требования к функциональным характеристикам	26
2.2.2 Требования к надежности	26
2.2.3 Условия эксплуатации	27
2.2.4 Требования к информационной и программной совместимости	28
2.3 Требования к структуре и функционированию системы	28
3 Описание вспомогательных решений	30
3.1 HTML	30
3.2 CSS	31
3.3 JavaScript	32
3.4 AJAX	33
4 Практико-техническая часть	35
4.1 Разработка базы данных	35
4.1.1 Введение	35
4.1.2 Анализ предметной области	38
4.1.3 Обоснование выбора СУБД	39
4.1.4 Построение схемы базы данных и описание отношений	41
4.2 Описание системы	46

4.2.1 Модуль администрирования	48
4.2.2 Пользовательский интерфейс	57
Заключение.....	61
Список использованных источников	62
Приложение 1	63
Приложение 2	69

Введение

Повсеместное распространение Интернета изменило взгляд на информационную поддержку научно-исследовательской деятельности. Всё больше и больше научно-исследовательских лабораторий создают свои страницы, сайты, порталы. В настоящее время это не столь даже престижно, сколько удобно для всех, ведь наличие своего представительства в интернете – это отличная возможность для взаимодействия с единомышленниками и людьми, заинтересованными в продвижении научной деятельности. В режиме on-line может вестись общение с пользователями, обработка сообщений из формы обратной связи, публикация новостей лаборатории, сведений об имеющихся технических проектах, научных публикациях, сотрудниках лаборатории, а также может предоставляться та или иная сопутствующая информация, документы, описания, контакты. Стоит отметить, что на основании имеющихся в базе данных лаборатории сведений, при необходимости, можно собирать любые статистические данные, помогающие проанализировать деятельность лаборатории.

Из вышесказанного можно сделать вывод, что построение и внедрение информационно-аналитической системы для научно-исследовательской лаборатории **актуально** и обоснованно.

В данной работе поставлена именно эта задача – разработка информационно-аналитической системы для межвузовской лаборатории инновационных проектов. Необходимо осуществить полный цикл построения интерактивной веб-ориентированной системы на языке программирования php.

Непосредственному воплощению задания в строчках кода предшествует несколько этапов работы, которые были выполнены в рамках производственной практики:

- определение концепции разрабатываемого программного комплекса согласно поставленной задаче;
- определение функционала, модулей и компонентов;
- изучение инструментария веб-проектирования: программное обеспечение, его эффективное использование;

- изучение приёмов и методов языка программирования, с помощью которых возможна реализация системы;
- проектирование/выбор технических параметров: веб-платформы, версии языка программирования, структуры базы данных, взаимосвязи компонентов и др.

Создаваемый информационный ресурс должен иметь практическое применение. Поэтому на всём протяжении работы внимание обращается на удобство той или иной функции с точки зрения пользователя.

1 Анализ современных методов разработки веб-систем

1.1 Методы разработки веб-систем

Современные методы построения сайтов представляют из себя четыре основных направления разработки:

- использование онлайн-конструкторов, таких как Wix, uCoz, Jimbo и др.;
- использование бесплатных систем управления содержимым (CMS);
- самостоятельная разработка;
- использование фреймворка.

Каждый из этих подходов обладает своими достоинствами и недостатками, рассмотрим их подробнее.

1.1.1 Онлайн-конструкторы

Онлайн-конструктор сайтов является желанным инструментом для веб-энтузиастов и предпринимателей, помогая им с развитием малых и средних веб-проектов. Конструктор позволяет в кратчайшие сроки разработать любой тип сайта (визитки, интернет-магазины и т.д.), предоставляя пользователям широкий функционал без необходимости глубокого изучения языков программирования, инструментов и утилит веб-разработки.

На рынке существует огромное разнообразие онлайн-конструкторов. Наиболее распространенными среди них являются uCoz, Wix, A5 и Jimdo.

uCoz является одним из старейших конструкторов веб-сайтов. С момента его запуска в 2005 году количество сайтов, разработанных при помощи данного конструктора, перевалило отметку в два миллиона [1], что делает данный конструктор самым популярным не только на территории РФ, но и за рубежом. Самое значимое преимущество сервиса заключается в том, что он обладает наиболее широкими функциональными возможностями по сравнению с другими строителями веб-сайтов. Тысячи виджетов и гаджетов предоставляют пользователям огромный ассортимент дополнительных услуг, которые могут быть добавлены на сайт без каких-либо проблем. Также система предоставляет гибкие возможности по управлению группами пользователей и разграничению прав доступа

к ресурсам сайта. Для новичков предусмотрена удобная система модулей, помогающая легко подключать к сайту готовые элементы для создания блога, новостной ленты, гостевой книги, форума и даже интернет-магазина. Ключевым недостатком конструктора является наличие сторонней рекламы при использовании бесплатной версии, при этом содержание рекламных блоков никак не контролируется владельцами ресурса.

Wix в настоящее время насчитывает более 82 миллионов зарегистрированных пользователей. Среди пользователей данный ресурс имеет более 2 миллионов платных подписчиков. Рост популярности обусловлен, в первую очередь, тем, что Wix постоянно совершенствует свой конструктор, делая его более легким и привлекательным для людей. Разработчики данного ресурса придерживаются концепции добавления новых инструментов и конструкций на регулярной основе. Если у них нет какой-то конкретной функции, то в последующем, скорее всего, эта функция будет добавлена, если достаточное количество пользователей попросят об этом. В отличие от других подобных систем, данный конструктор выделяется простотой управления – редактирование содержимого происходит прямо на сайте, нет необходимости использовать отдельную панель управления [2]. Пользователю достаточно лишь разместить блоки в нужных местах, используя технологию drag-and-drop, а затем отредактировать их содержание. Недостатком также является наличие рекламы на сайте, которую, при желании, можно убрать за определенную плату.

A5 разработан на базе собственного движка, что позиционирует его только с лучшей стороны. Одним из преимуществ, упрощающих разработку сайта с применением данного конструктора, является возможность предпросмотра оформления, заинтересовавшего пользователя. Стоит упомянуть о такой функции, как отмена совершенного действия. Она запоминает до 14 последних операций, что без боязни позволяет экспериментировать с оформлением. Конструктор предлагает на выбор большой ассортимент виджетов: от банальных текстовых полей, галереи изображений и вставки HTML-кода до установки контейнеров, собирающих статистику посещаемости. В отличие от других систем, данный конструктор более лоялен в отношении внедрения рекламы на сайт. Рекламные блоки хорошо

вписываются в основной дизайн и не вызывают никакого раздражения при работе с сайтом. К другим положительным сторонам конструктора можно отнести возможность использования собственного доменного имени при использовании платного тарифа, а также наличие подробного обучающего материала, охватывающего все аспекты функционала конструктора. Недостатком же является неудобство работы с многостраничными сайтами.

Jimdo отличается простым и понятным интерфейсом, не вызывающим много вопросов у начинающих пользователей. Широкий выбор шаблонов предоставляет большие возможности создания сайтов с современным уникальным дизайном. Возможность работать напрямую с кодами HTML и CSS позволяет не только расширить функционал для продвинутого пользователя, но и облегчить знакомство новичков с фундаментальными основами построения веб-сайтов. Стоит отметить, что данный конструктор ориентирован в большей степени на представителей малого и среднего бизнеса. Бесплатный тарифный план предоставляет сильно ограниченный набор функциональных характеристик [3], что делает его не самым популярным среди обычных пользователей.

Несмотря на кажущиеся простоту и удобство использования конструктора веб-сайтов, данный метод разработки имеет существенные недостатки:

- нет доступа к исходному коду движка;
- ограниченный функционал, определенные сложности с внедрением внешних плагинов и скриптов;
- плохая поисковая оптимизация, затрудняющая продвижение сайта в сети;
- наличие сторонней рекламы, которую владельцы конструктора вывешивают на видном месте;
- нет возможности перенести сайт на другой хостинг.

Перечисленные недостатки делают использование конструктора для создания веб-ориентированной системы для научно-исследовательской лаборатории неприемлемым.

1.1.2 CMS

Система управления сайтом (англ. Content Management System, CMS) является программным обеспечением, направленным на поддержку и организацию совместного процесса по созданию, редактированию и управлению контентом (содержимым вашего сайта). Выражаясь простыми словами, можно сказать, что CMS есть не что иное, как движок сайта.

Подавляющее большинство сайтов на сегодняшний день разработаны при помощи той или иной системы управления. Благодаря наличию такой системы, владельцы сайта могут наполнять его любой информацией, публиковать новые страницы и новости, принимать платежи и заявки, не имея при этом каких-либо навыков в области веб-программирования.

CMS позволяет создавать и управлять текстовыми документами и мультимедийными файлами в удобном для пользователя виде. Для этих целей в системе предусмотрено наличие визуального редактора, который обладает большим набором функций, позволяющих форматировать текст, вставлять изображения, видео- и аудиофайлы, ссылки на внешние ресурсы и т.д. При этом содержимое редактора будет иметь точно такой же вид, как и после публикации записи.

Принцип работы любой CMS достаточно прост. После того как пользователь системы добавил контент на сайт, вся информация сохраняется либо в базе данных, либо в файлах. Когда посетитель открывает сайт, вся необходимая информация считывается из базы данных, а после этого запрошенная страничка формируется согласно предопределенному в системе шаблону (почти все популярные CMS предоставляют гибкую настройку шаблонов для сайта). Многие движки также поддерживают систему модулей. Это означает, что вы можете расширить функционал системы за счет подключения дополнительных модулей (плагинов), например, плагин чата или плагин обратной связи.

Среди великого множества доступных на рынке систем управления сайтом, особого внимания заслуживают следующие бесплатные CMS: WordPress, Joomla, и Drupal. Это наиболее распространенные и проверенные CMS. Их мы рассмотрим более подробно.

WordPress является программным обеспечением с открытым исходным кодом, которое позволяет пользователям создавать динамические веб-сайты и блоги. WordPress написан на языке программирования PHP и использует сервер базы данных MySQL для хранения данных. Впервые данная платформа была опубликована в мае 2003 года, а в октябре 2009 года исходные коды проекта были выложены в открытый доступ. В настоящее время позиционируется как самая популярная система для ведения блога в Интернете.

WordPress является очень расширяемой системой. Активное сообщество разработчиков, специализирующихся на проектировании сайтов с применением данного программного пакета, предоставляют дополнительные функциональные возможности в виде плагинов. К примеру, плагин Google XML Sitemap при установке автоматически генерирует карту для вашего сайта каждый день без каких-либо усилий с вашей стороны. Данный плагин улучшает индексацию блога поисковыми системами, путём дополнения карты и оповещения поисковиков о необходимости переиндексации страницы после каждого добавления новой записи или обновления имеющейся. Стоит отметить, что большинство всех плагинов являются бесплатными.

В целом, к плюсам данной системы можно отнести [4]:

- удобное управление пользователями, позволяющее изменять роли доступа пользователей (администратор, автор или редактор), менять их учетные данные, а также создавать или удалять пользователей из системы;
- простоту при установке и освоении системы, что может сыграть ключевую роль при выборе движка для новичков;
- неограниченный набор функциональных возможностей, множество тем и бесплатных плагинов;
- частый выход обновлений, исправляющих ошибки и уязвимости в системе;
- наличие различных инструментов для поисковой оптимизации сайта.

Однако, как и любая другая система, WordPress имеет ряд существенных недостатков:

- слабая оптимизация исходного кода приводит к высокой нагрузке на сервер;
- платформа не предназначена для сайтов с огромной посещаемостью;

- использование нескольких плагинов может сделать сайт «тяжелым» для загрузки и запуска;
- большинство плагинов могут вступать в конфликт друг с другом либо с используемой темой оформления.

Joomla является второй наиболее популярной системой управления сайтом после WordPress. Однако, данная система предназначена в большей мере для веб-разработчиков, имеющих опыт разработки функциональных и профессиональных веб-сайтов. У новичков же могут возникнуть определенные трудности в её изучении.

При начальной установке CMS включает минимальный набор инструментов, который может с легкостью дополняться в случае необходимости. Такая минимизация функциональных возможностей позволяет не загромождать панель администрирования ненужными элементами, экономит место на хостинге, а также снижает серверную нагрузку.

Каталог расширений включает различные языковые пакеты, которые устанавливаются при помощи штатных средств администрирования. Данная возможность позволяет отображать интерфейсы административной и фронтальной части на любом языке. Встроенная функция по отображению рекламных материалов упрощает добавление рекламных баннеров на страницы вашего сайта.

К дополнительным преимуществам относятся [5]:

- многоуровневая аутентификация пользователей;
- возможность создания веб-сайта любой сложности и содержания без привлечения дополнительных расширений или инструментов;
- наличие огромного количества готовых тем оформления, которые подходят совершенно под разные цели;
- возможность настройки существующих расширений либо самостоятельное написание модулей и компонентов.

Недостатки:

- слабая защита от взлома;
- сложная иерархия элементов в системе;
- возможные проблемы с продвижением сайта в поисковых системах;

- наличие незадействованных скриптов и расширений в коде движка приводит к медленной загрузке страниц.

Drupal представляет из себя многофункциональную и универсальную систему управления сайтом. Создавалась она специально для продвинутых пользователей, чтобы помочь им создавать мощные веб-сайты, способные обрабатывать большие объемы посетителей и сотни страниц контента. Drupal является настолько гибкой, что позволяет создавать как простые блоги, так и впечатляющие, интерактивные бизнес-сайты. Такая гибкость делает эту CMS отличным решением для растущего бизнеса с постоянно меняющимися требованиями к функциональности сайта.

Drupal доказала, что является одной из самых безопасных систем управления сайтом. Обеспечением безопасности CMS занимается специально обученная команда из 30-40 программистов. Обширное сообщество разработчиков, набранное Drupal за более чем 10 лет своего существования, помогает выявлять потенциальные пробелы в безопасности системы. Тот факт, что Drupal используется правительственными ведомствами и оборонными предприятиями, обязывает его более тщательно следить за безопасностью своей системы.

Drupal обладает еще большими возможностями по переводу вашего сайта на другие языки, предоставляя в распоряжение пользователя более 70 языков.

К несомненным плюсам данной системы относятся [6]:

- возможность сложной и глубокой разбивки страниц по категориям;
- гибкость архитектуры;
- практически неограниченные функциональные возможности;
- высокий уровень качества доступных модулей;
- встроенная организация поисковой оптимизации, поддерживающая различные виды человеко-понятных URL-адресов.

Недостатки также присутствуют, а именно:

- высокий порог вхождения;
- серьезная нагрузка на сервер из-за высоких требований к системе;
- сложность при установке обновлений и программных расширений.

Таким образом, подход с применением готовой CMS, по сравнению с использованием конструкторов, позволяет создать достаточно качественный и многофункциональный проект. В то же время, избыточный функционал любой CMS далеко не лучшим образом сказывается на производительности системы. А тот факт, что исходные коды многих популярных CMS выложены в общий доступ, ставит под угрозу взлома все сайты, построенные на их основе.

1.1.3 Разработка сайта с «нуля»

Написание самописного движка для системы является нелегкой задачей для любого разработчика. Однако, любой интернет-ресурс, разработанный с «нуля», позволит реализовать любые пожелания и индивидуальные потребности разработчика.

При использовании CMS не всегда получится найти сторонние плагины, целиком и полностью удовлетворяющие запросам пользователя. При попытке написания своего модуля для внедрения в существующую систему управления содержимым, могут возникнуть определенные трудности, связанные с незнанием внутреннего устройства CMS и принципов взаимодействия её компонент. При самостоятельной же разработке программист сам определяет архитектуру разрабатываемой системы и может добиться повышения производительности системы за счет устранения избыточных функций и связей. В противоположность этому, наличие излишних функциональных возможностей в CMS приводит к сильному спаду производительности на сервере.

Стоит отметить, что индивидуальные проекты в меньшей степени подвержены проблемам безопасности по причине того, что их код закрыт от посторонних глаз, вследствие чего обнаружить уязвимость на сайте будет намного сложнее. Также существенным преимуществом самостоятельного подхода к разработке веб-ориентированной системы является наличие простого и интуитивно понятного интерфейса, который ограничен только самыми необходимыми конкретному пользователю настройками.

Несмотря на то, что данный подход позволяет достичь более качественной разработки веб-приложения, он имеет главный недостаток: наличие больших затрат времени, а также трудоемкости работы. Без должных навыков в программировании и создании веб-сайтов велика вероятность получить на выходе «сырой» продукт, который будет уступать существующим техническим решениям по уровню качества и проработки.

1.1.4 Использование фреймворка

Задача фреймворка состоит в том, чтобы позволить дизайнерам и разработчикам сфокусироваться на создании уникальных функций для своих проектов, а не изобретать велосипед путем написания одинаковых вещей, общих для большого числа веб-сайтов и веб-приложений.

Фреймворк можно считать заготовленным шаблоном (структурой), управляющим большинством повторяющихся и общих для всех веб-сайтов особенностей. Как результат, в отличие от CMS, фреймворк скорее всего не будет иметь пользовательского интерфейса (хотя это и не всегда так, фреймворк Django, к примеру, включает в себя интерфейс администратора). Большая часть работы будет заключаться в написании кода и взаимодействии через него с различными частями фреймворка.

Многие опытные веб-разработчики предпочитают проектировать веб-системы, опираясь на функционал фреймворка. Существует огромное множество популярных и зарекомендовавших себя фреймворков (Yii2, Laravel, Ruby on Rails и т.д.). Мы не будем останавливаться на их подробном описании, однако приведем ключевые преимущества применения фреймворков, которые в той или иной степени относятся к каждому из них в отдельности.

Открытый исходный код. Большинство популярных фреймворков имеют открытый исходный код (или доступны для бесплатного использования). Также они распространяются под свободной лицензией, позволяющей создавать коммерческие продукты.

Документация и поддержка. Несмотря на то, что это может варьироваться (если используется популярный язык и этим фреймворком пользуется много разработчиков), можно ожидать, что фреймворк будет обладать хорошей документацией или поддержкой, а то и всем сразу. Стоит упомянуть, что понятие «хорошая поддержка» довольно субъективно. Обычно, платная поддержка всегда будет быстрее и точнее, но это также зависит от уровня распространенности фреймворка. К примеру, такие фреймворки как Yii2 и Ruby on Rails, собравшие вокруг себя огромное сообщество, известны своей доброжелательностью и хорошей поддержкой.

Эффективность. Можно расценивать как основную причину существования фреймворков. Они избавляют от необходимости писать огромное количество повторяющегося кода, который уже используется во множестве различных приложений. Это включает в себя, например, системы аутентификации пользователей и комментирования. В среднем (если разработчик обладает достаточным уровнем знаний определенного фреймворка) можно ожидать значительного снижения времени разработки проекта, в сравнении с написанием кода без фреймворка.

Безопасность. Обычно, фреймворк разработан и протестирован большим числом разработчиков. Крайне полезно, что большинство рисков для безопасности найдено и исправлено еще на этапе создания фреймворка или же оперативно редактируется после.

Интеграция. При разработке практически любого вида приложения (включая веб-сайт) возникает необходимость в хранении данных, и для этих целей, скорее всего, будет использоваться база данных. Помимо базы данных, существует много других инструментов, которые используются в веб-разработке. Многие фреймворки из-за этого позволяют осуществить легкую привязку к какому-то инструментарию и взаимодействовать с ним (например, порой «общение» с базой данных в некоторых фреймворках выполнено на достаточно абстрактном уровне, что сильно упрощает работу, исключая дополнительные усилия со стороны разработчика).

Зачастую требуется определенное время, чтобы изучить тот или иной фреймворк, однако один раз познакомившись с ними поближе, можно значительно ускорить время разработки.

1.2 Обоснование выбора метода решения поставленной задачи

В настоящее время сфера веб-разработок настолько обширна, что порой трудно сориентироваться и выбрать нужный вектор действий. Для нас же сейчас необходимо произвести оценку перечисленных выше методов создания нашей системы. В табл. 1 рассмотрены основные преимущества и недостатки каждого из них.

Таблица 1

Основные преимущества и недостатки методов разработки веб-системы

Метод	Преимущества	Недостатки
Использование онлайн-конструктора	<ul style="list-style-type: none"> • экономия времени • простота создания и редактирования сайта 	<ul style="list-style-type: none"> • сторонняя реклама на сайте • затрудненный перенос сайта на другой хостинг • ограничения по объему занимаемой памяти и количеству контента • плохая поисковая оптимизация • фиксированный набор функций
Использование бесплатной CMS	<ul style="list-style-type: none"> • доступность • готовый набор рабочих компонентов • документация, форумы поддержки 	<ul style="list-style-type: none"> • самостоятельная развертка • как правило, требуется доработка под свои нужды • отсутствие каких-либо гарантий (поставляются по принципу «AS-IS») • потенциальные проблемы в безопасности • устаревающие технические решения
Самостоятельная разработка	<ul style="list-style-type: none"> • полный учет целей и задач разработки • знание особенностей системы • планирование развития • возможность достижения максимальной производительности 	<ul style="list-style-type: none"> • большие временные затраты на разработку • требуется сопровождение и документирование • требуется большой опыт архитектурного проектирования систем

Метод	Преимущества	Недостатки
Использование фреймворка	<ul style="list-style-type: none"> • безопасность • высокая производительность • использование самых современных подходов и паттернов разработки • набор готовых, высококачественных компонентов • полный учет особенностей разрабатываемой системы • хорошая документация и форумы поддержки 	<ul style="list-style-type: none"> • есть порог входа: необходимо время на изучение и/или уверенное знание языка программирования, на котором написан фреймворк • «коробочный» функционал как правило отсутствует: разрабатывается либо с «нуля», либо на основе компонентов фреймворка

Таким образом, реализация нашей веб-ориентированной системы с применением функционала фреймворка является наиболее оправданным решением. Во-первых, в отличие от самописных решений, это позволит добиться простоты сопровождаемости проекта; во-вторых, решения на фреймворках работают значительно быстрее и способны выдерживать большую нагрузку, чем CMS и самописные системы, а также превосходят их по уровню безопасности.

1.3 Обоснование выбора языка программирования

В качестве серверного языка программирования для реализации нашей системы был выбран язык PHP.

PHP – распространенный, специально сконструированный для ведения веб-разработок язык программирования. В отличие от таких языков программирования, как Perl или C, в которых вывод HTML-кода осуществляется командами языка, PHP страница содержит HTML со встроенным в него кодом [7].

Практичность языка PHP для ведения веб-разработок обусловлена его следующими важными характеристиками:

- «движок» PHP устроен таким образом, что позволяет обрабатывать сценарии с достаточно высокой скоростью;
- поддержка широкого круга баз данных;

- поддержка большого количества аппаратных платформ и операционных систем, включая Linux, Windows, Mac OS X и многие другие;
- поддержка большинства современных веб-серверов (Apache, IIS и др.);
- гибкость по отношению к потребностям разработчика благодаря обилию различных библиотек и расширений;
- предоставление гибких и эффективных средств безопасности (ограничение максимального времени выполнения скрипта и использования памяти, надежные механизмы шифрования и многое другое), при правильной настройке которых можно добиться максимальной свободы действий и безопасности.

На рис.1 представлено процентное соотношение веб-сайтов, разработанных на том или ином языке программирования [8].

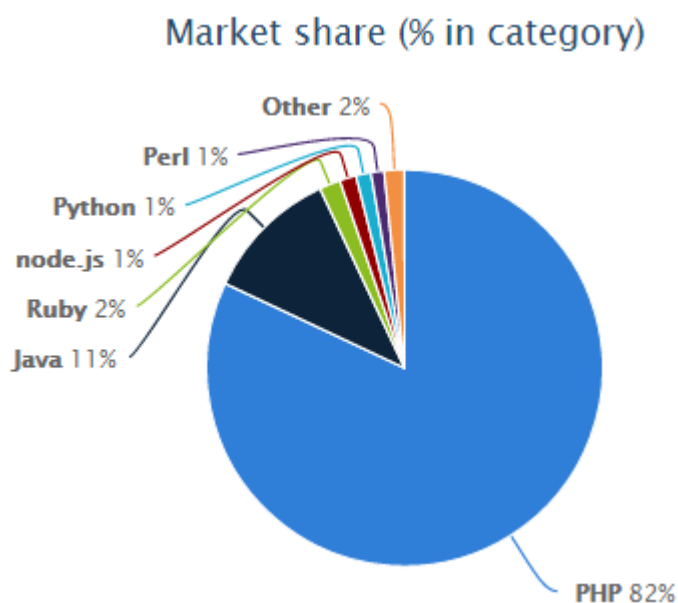


Рис. 1. Процентное соотношение ЯП, применяемых на веб-сайтах

Диаграмма, представленная выше, доказывает, что язык PHP является наиболее предпочтительным в среде веб-разработки. Такую популярность он заслужил во многом благодаря легкости в его освоении на различных этапах и рассмотренным выше особенностям.

1.4 Обоснование выбора фреймворка

Зачастую, выбор определенного фреймворка зависит от его популярности и простоты в использовании, а также от личных предпочтений разработчика. Любой фреймворк только предлагает достаточно низкоуровневые инструменты для разработки веб-приложений – большинство функционала разработчику придется реализовывать самостоятельно.

Для достижения поставленной в работе цели был выбран PHP-фреймворк Yii. Yii – это высокоэффективный фреймворк общего назначения, который используется для разработки веб-приложений любого масштаба. Как и большинство других популярных PHP-фреймворков, Yii использует широко применяемый в веб-программировании шаблон проектирования Model-View-Controller (MVC). Превосходство Yii перед другими фреймворками заключается в его эффективности, широких функциональных возможностях и качественной документации (включая обширное русскоязычное сообщество). Yii изначально был спроектирован таким образом, чтобы максимально соответствовать всем требованиям при разработке современных веб-приложений.

Yii – полностью объектно-ориентированный фреймворк, использующий все преимущества продвинутых функций PHP [9]. Арсенал данного фреймворка включает огромное множество различных компонентов и инструментов, помогающих уменьшить затрачиваемое на разработку время. Среди них стоит выделить:

- компонент Security, облегчающий организацию защиты приложения за счёт предоставления таких полезных методов как генерация безопасного хэша из пароля, проверка соответствия заданного пароля хранимому в хэше, а также генерация случайной строки и т.д.;
- готовые классы для авторизации и аутентификации пользователей;
- веб-инструмент для генерации кода Gii, позволяющий быстро создавать шаблоны для различных моделей, контроллеров, модулей и т.д.

Ключевым фактором выбора фреймворка Yii является то, что он превосходит другие популярные PHP-фреймворки (такие как Zend, Laravel, Symfony и CakePHP)

по показателям производительности. Результаты сравнения этих фреймворков представлены на рис.2 и в табл.2.

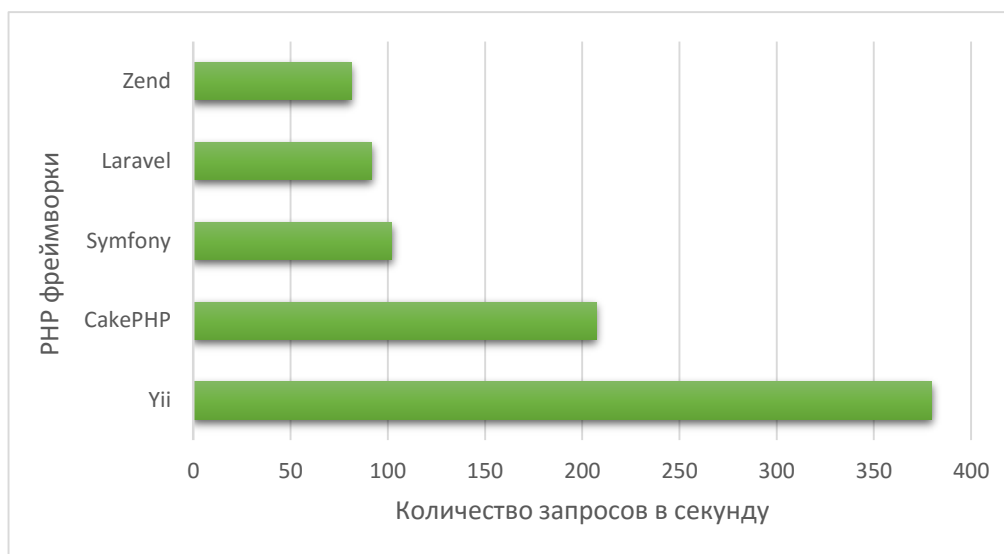


Рис. 2. Производительность популярных PHP-фреймворков

Таблица 2

Показатели производительности популярных PHP-фреймворков

Показатели производительности	PHP-фреймворки				
	Zend	Laravel	Symfony	CakePHP	Yii
Количество запросов в секунду	81,13	91,59	101,99	207,27	379,77
Пиковое потребление памяти	3,02	2,76	3,21	1,37	1,37

Все фреймворки тестировались на самой простой программе «Hello, World!». При этом, чтобы обеспечить более справедливое сравнение фреймворков, все их дополнительные функциональные возможности были отключены.

Как видно из результатов сравнения, фреймворк Yii с большим преимуществом опережает своих оппонентов. Приложение, написанное на фреймворке Yii, способно обрабатывать большее количество запросов в единицу времени, при этом пиковое использование памяти остается минимальным.

2 Описание требований к объекту разработки

2.1 Введение

Согласно заданным требованиям разрабатывается информационно-аналитическая система лаборатории, занимающейся научно-исследовательской деятельностью. Программный комплекс создается для функционирования в сети Интернет в качестве интерактивного инструмента взаимодействия лаборатории с единомышленниками и людьми, занимающимися продвижением научной деятельности.

2.2 Требования к программе

2.2.1 Требования к функциональным характеристикам

Первоначальная версия системы должна обеспечивать выполнение следующих функций:

- формирование информационных страниц;
- представление курсов повышения квалификации, а также проектов лаборатории;
- обратная связь для пользователей;
- приём заявок на прохождение обучения по выбранному направлению, если возможность приёма открыта для этого направления;
- приём заявок на приобретение проекта, если такая возможность открыта;
- администрирование: управление страницами, пользователями, заявками и т.д.

Хранение данных и работа с ними должны осуществляться с использованием базы данных.

2.2.2 Требования к надежности

Информационно-аналитическая система должна устойчиво функционировать в круглосуточном режиме, 7 суток в неделю и работать без останова по программным ошибкам. Должны быть исключены программные «тупики», способные привести к

отказу системы. В случае любой ошибки должны выдаваться соответствующие сообщения (инструкции) для пользователя системы.

Во избежание некорректного поведения программы, либо намеренных действий злоумышленника по выводу системы из строя при вводе входной информации необходимо осуществлять контроль её контроль на допустимость вводимых значений. Ввод данных в систему должен осуществляться только после того, как пользователь исправит ошибки ввода.

Системные файлы, программные модули и хранилище информации должны быть защищены от несанкционированного доступа. Административный модуль является закрытым для общего доступа. Любая его информация, функции и компоненты должны быть доступны только пользователям с административной учетной записью. Доступ к управлению системой должен осуществляться только после успешной авторизации по логину и паролю, выдаваемому администратором системы.

Параметры доступа к базе данных должны храниться в конфигурационном файле, заблокированном от чтения извне.

2.2.3 Условия эксплуатации

Программная система предназначена для работы на стороне веб-сервера, поэтому внешние условия эксплуатации для обеспечения бесперебойного функционирования (при установке на собственный сервер) должны соответствовать требованиям производителя аппаратуры. Помимо этого, правильная работа программы гарантируется только в случае соблюдения требований к составу и параметрам технических средств, на базе которых развёртывается информационная система, а также информационной и программной совместимости.

Обслуживание системы включает визуальный контроль правильности функционирования, изучение отчётов об ошибках, контроль скорости работы системы, оптимизацию базы данных, обновление на новые версии и осуществляется одним квалифицированным специалистом (администратором).

2.2.4 Требования к информационной и программной совместимости

Требования к программному обеспечению компьютеров конечных пользователей для организации работы с системой в варианте «клиент – сервер» можно сформулировать следующим образом:

- операционная система MS Windows 10/8/7;
- веб-браузеры Google Chrome 47 и выше, Opera 34 и выше, Mozilla Firefox 43 и выше.

Требования к программному обеспечению компьютера, на котором выполняется веб-сервер:

- операционная система семейства Unix;
- программа веб-сервер Apache версии 2.x;
- модуль языка программирования PHP версии не ниже 5.4 для веб-сервера.

Требования к программному обеспечению сервера базы данных:

- MySQL версии 5.x;
- операционная система: в соответствии с требованиями MySQL.

2.3 Требования к структуре и функционированию системы

Информационная система логически должна состоять из следующих компонентов:

- ядро, отвечающее за взаимодействие всех компонентов системы, выборку информации из базы данных, компоновку странички, выдаваемой пользователю;
- компонент «Новости», формирующий список новостей, а также подробную информацию по каждой новости в отдельности;
- компонент «Проекты», отвечающий за формирование перечня проектов, разрабатываемых лабораторией;
- компонент «Публикации», отвечающий за формирование подробного списка публикаций сотрудников лаборатории;

- компонент «Курсы», отвечающий за формирование перечня направлений обучения, предлагаемых лабораторией;
- компонент «Пользователи», предусматривающий формирование списка сотрудников с глобальной областью видимости, а также уточняющей информации по каждому из сотрудников;
- компонент «Обратная связь», обрабатывающий запросы от пользователей на получение дополнительной информации, отправку отзывов и т.п.;
- компонент администрирования.

Построение системы должно обеспечить автономное функционирование компонентов. Инициализация данных и параметров, общих для всех компонентов, должна осуществляться ядром системы. При этом функции и скрипты необходимо разделить логически и расположить по соответствующим файлам.

Взаимодействие модулей осуществляется через ядро системы, либо путём прямого вызова одного из компонентов модуля с заданными параметрами. Информационный обмен данными ведётся посредством базы данных. Все входные параметры модуль получает в результате диалога с пользователем, входные данные выбираются из БД. Выходные данные направляются либо в базу данных, либо в браузер пользователя.

3 Описание вспомогательных решений

В настоящий момент, разработка любой веб-ориентированной системы не может быть осуществлена в полной мере без использования фундаментальных, общепринятых технических решений. Ниже приведен обзор важных составляющих любой веб-разработки.

3.1 HTML

Язык HTML является разработкой британского ученого Тима Бернерса-Ли и берёт своё начало в стенах ЦЕРН (Европейского союза по ядерным исследованиям). В конце 1991 года появилось первое общедоступное описание языка HTML. Данный язык разметки создавался для обмена технической и научной документацией и был пригоден для людей с ограниченными знаниями в области верстки. HTML определял небольшой набор семантических и структурных элементов, с помощью которых можно было создать относительно простые, но красиво оформленные документы.

Кроме упрощения структуры документа, в HTML была добавлена поддержка гипертекста. Поддержка мультимедийных возможностей была добавлена позже. Изначально, язык HTML задумывался и создавался как средство форматирования и структурирования документов без их привязки к средствам отображения. В лучшем виде, текст, размеченный средствами HTML, должен был без структурных и стилистических искажений отображаться на оборудовании с различной технической оснащённостью. Однако сегодняшнее применение данного языка разметки очень далеко от его первоначальной задачи.

Использование языка HTML для разметки текста помогает добиться следующего:

- сделать текст курсивным, подчеркнутым или жирным;
- вставить специальные символы (математические символы, греческие и готические буквы, символы пунктуации, выходящие за рамки ASCII и т.п.);
- поменять кегль, начертание, гарнитуру и цвет шрифта;
- оформить текст в виде гиперссылки, ведущей на другую страницу или файл;
- нарисовать таблицу и т.д.

Немного позднее, когда появилась острая необходимость добавления интерактивности на веб-страницы, в язык HTML были добавлены:

- формы для ввода данных пользователем, которые после отправки подвергаются обработке на стороне сервера при помощи специальных серверных программ;
- возможность открытия мультимедийных файлов, выводимых как внешними приложениями, которые встраиваются в окно браузера, так и непосредственно самим браузером.

HTML5 – это пятая и, на сегодняшний день, самая новая версия HTML стандарта и языка разметки. HTML5 разрабатывался с целью улучшения языка разметки для работы с новыми приложениями и мультимедийными элементами. Семантические элементы, добавленные в новом стандарте, обеспечивают более осмысленный для поисковых механизмов код. Также отпала необходимость внедрения сторонних плагинов на сайт – современные браузеры, поддерживающие HTML5, включают в себя всё необходимое для воспроизведения и отображения мультимедийного контента непосредственно в HTML-разметке.

3.2 CSS

Каскадные таблицы стилей, или CSS, представляют собой формальный язык, который используется для описания представления документа, написанного на одном из языков разметки, таких как HTML или XHTML. Основной целью использования каскадных таблиц является разграничение структуры веб-сайта и его содержания. Применение CSS позволяет повысить доступность, гибкость и контроль за представлением контента.

Каскадные таблицы стилей были разработаны с целью создания единого подхода к обеспечению визуализации информации для веб-документов. За всё время своего существования, они вобрали в себя огромное множество стилистических возможностей для веб-разработчиков. Это дало дизайнерам больший контроль над внешним видом сайта, но одновременно с этим усложнились написание и поддержка HTML-кода.

Преимущества использования CSS:

- визуальное описание информации для всего веб-сайта можно поместить в один файл, что позволяет централизованно менять стилистическое отображение различных страниц сайта;
- код разметки документа значительно уменьшается в размере, так как нет необходимости включать в него какой-либо код, имеющий отношение к визуальному описанию документа;
- удобство сопровождения страниц, использующих CSS.

Последней, активно используемой спецификацией языка, является CSS3. По сравнению с предыдущими версиями, данная спецификация была разбита на модули с целью независимой разработки и развития каждого из них. Старая же спецификация была слишком большой и громоздкой, что сильно затрудняло её обновление и сопровождение.

Таблица стилей включает набор правил, каждое из которых состоит из одного или множества селекторов, а также набор свойств и их значений (рис.3).

```
селектор, селектор
{
    свойство: значение;
    свойство: значение;
    свойство: значение;
}
```

Рис. 3. Структура CSS документа

В качестве свойств может выступать практически любой параметр отображения: шрифты, цвета, позиционирование элемента, отступы и т.д.

3.3 JavaScript

JavaScript – сценарный язык программирования, который применяется с целью добавления интерактивности на веб-сайт. Скрипт может быть внедрен в любом месте непосредственно на веб-странице. Однако лучшей практикой является размещение тела скрипта в конце HTML-документа. Браузер считывает и обрабатывает HTML-страницу построчно и, когда встречает пользовательский скрипт, блокирует отрисовку страницы на время выполнения этого скрипта. Это приводит к

значительному снижению времени загрузки страницы. Поэтому более разумным подходом считается определение всех пользовательских скриптов после основной структуры HTML-документа.

Использование JavaScript на веб-страницах позволяет добиться следующего:

- возможность добавления различных эффектов анимации;
- возможность добавления и изменения элементов HTML;
- реагирование на передвижение курсора мыши и нажатия клавиш на клавиатуре;
- возможность проверки данных, введенных в поля формы, перед отправкой их на сервер, что позволяет уменьшить серверную нагрузку и ускорить работу веб-сайта.
- возможность загрузки определенной страницы в зависимости от версии браузера.

3.4 AJAX

AJAX описывает концепцию использования нескольких смежных технологий (в частности, JavaScript и XML) для построения пользовательских интерфейсов веб-приложения с высоким уровнем интерактивности. Применение данной технологии позволяет веб-приложениям передавать данные и получать данные с сервера в фоновом режиме (асинхронно) без какого-либо воздействия на текущую страницу. Путём логического разграничения слоя обмена данными и слоя представления, AJAX позволяет изменять содержимое веб-страниц динамически, без необходимости перезагрузки всей страницы.

AJAX включает в себя:

- HTML и CSS для отображения данных;
- DOM (объектная модель документа) для динамического отображения и взаимодействия с данными;
- JSON или XML, описывающие формат передаваемых данных;
- объект XMLHttpRequest для асинхронного взаимодействия;
- JavaScript для совместного использования перечисленных технологий.

Схематически принцип работы веб-приложения с применением технологии AJAX показан на рис.4.

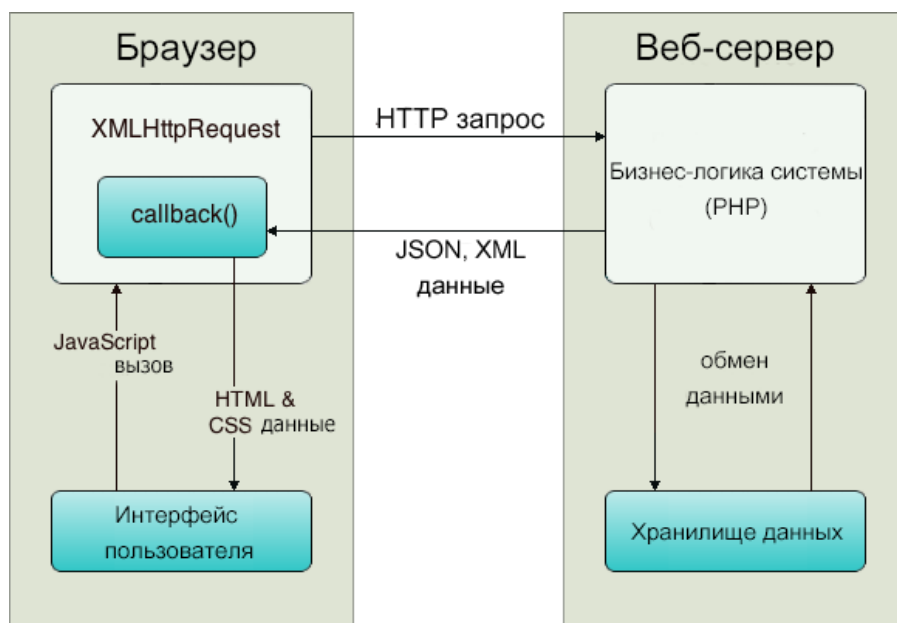


Рис. 4. Принцип работы веб-приложения с применением AJAX

4 Практико-техническая часть

4.1 Разработка базы данных

4.1.1 Введение

Один из основных принципов современной веб-системы состоит в том, что практически все её данные хранятся в базе данных, и пользователям предоставляется удобный интерфейс для работы с системой. Администратору же системы для обновления не требуется вручную редактировать файлы на сервере – для этого создаются интерактивные веб-формы.

Исходя из многолетнего опыта использования баз данных, можно выделить общий набор присущих им рабочих характеристик:

- полнота;
- правильная организация;
- актуальность;
- удобство использования.

Перед созданием новой базы данных необходимо сначала определиться с тем, какой тип и какая структура данных будут в ней использоваться. В противном случае велика вероятность получить не то, что требовалось, так как информация, которая содержится в базе данных, должна иметь структурированный характер.

Сейчас используются три основные модели представления данных в базе данных [10]: иерархическая, сетевая и реляционная. Рассмотрим вкратце каждую из них.

Иерархическая модель. Иерархическая структура представлена совокупностью элементов, связанных определенными правилами. Графически информация в иерархической базе представляется в виде древовидной структуры, состоящей из объектов различных уровней (рис.5).

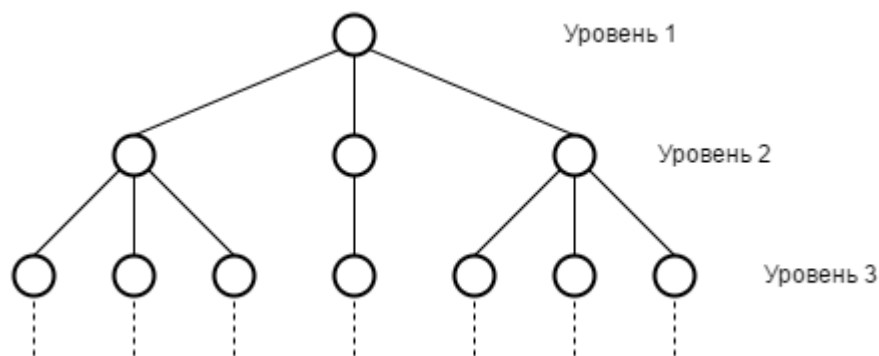


Рис. 5. Иерархическая структура

Элементы такого дерева называются узлами и представляют собой совокупность атрибутов, описывающих объекты.

К достоинствам иерархической модели построения баз данных можно отнести эффективное использование памяти ЭВМ и быстрое выполнение операций над данными. Основными недостатками являются громоздкость модели при работе с данными со сложными логическими связями, а также невозможность реализовать отношения «многие-ко-многим».

Сетевая модель. В отличие от иерархической структуры, каждый элемент сетевой структуры может быть связан с любым другим элементом (рис.6).

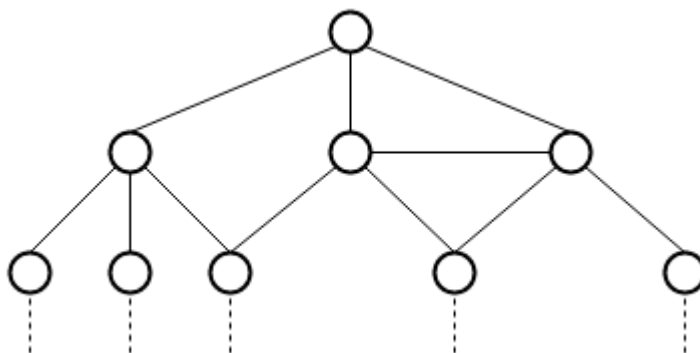


Рис. 6. Сетевая структура

Достоинствами сетевой модели данных являются эффективная реализация по показателям затрат памяти и скорости и предоставление больших, по сравнению с иерархической моделью, возможностей по организации связей между элементами. К недостаткам же можно отнести слабый контроль целостности данных ввиду создания произвольных связей между элементами, а также высокую сложность и жесткость схемы базы данных, построенной на основе сетевой модели.

Реляционная модель. Реляционная модель представления данных является наиболее популярной на сегодняшний день. Все данные в такой модели представляются в виде двумерных таблиц (рис.7). Любая таблица в реляционной базе данных состоит из строк и столбцов, называемых также кортежами и атрибутами соответственно.

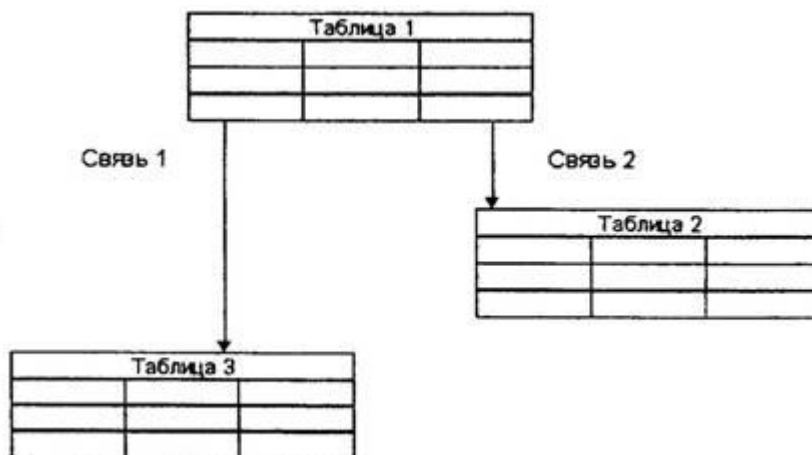


Рис. 7. Схема реляционной модели данных

В реляционной модели можно выделить несколько ключевых правил, которые обеспечивают целостность хранимых данных:

- каждый кортеж в таблице является уникальным, то есть содержит одно или несколько значений атрибутов, которые позволяют однозначно идентифицировать его среди остальных;
- каждое значение в ячейках таблицы должно быть атомарным;
- значения в пределах одного атрибута должны принадлежать к одному и тому же типу;
- наименования таблиц и атрибутов в базе данных должны быть уникальными;
- последовательность кортежей и атрибутов в таблице несущественна.

Для построения базы данных была выбрана реляционная модель. К её достоинствам можно отнести простоту представления данных (благодаря привычной для большинства пользователей табличной форме), минимальную избыточность данных при нормализации отношений, высокий контроль целостности данных, а

также гибкую систему их защиты. Плюсом также является то, что реляционная модель позволяет менять структуру базы данных, не затрагивая при этом само приложение.

4.1.2 Анализ предметной области

База данных представляет собой информационную модель предметной области. Поэтому, для создания базы данных необходимо первым делом проанализировать предметную область и создать её модель.

В соответствии с предметной областью система строится с учётом следующих особенностей:

- каждый сотрудник лаборатории может быть прикреплен в качестве автора к неограниченному количеству публикаций и проектов;
- каждый сотрудник может иметь несколько достижений, причём каждое достижение закреплено за одним сотрудником;
- за каждой новостью или публикацией может быть закреплено несколько новостных тем или ключевых слов соответственно;
- к каждой новости может быть прикреплено несколько документов, при этом отдельно взятый документ имеет привязку только к одной новости;
- каждому курсу, проводимому лабораторией, или проекту может соответствовать несколько заявок в разделе «обратная связь»;
- «сообщения» и «галерея» являются отдельными структурными элементами и не имеют привязки к другим элементам.

Для того, чтобы построить ER-модель, необходимо выделить основные сущности предметной области: новости, новостные категории, публикации, проекты, курсы, ключевые слова к публикациям, сотрудники, достижения сотрудников, заявки на проекты, заявки на курсы, а также сообщения из формы обратной связи и галерея. Исходя из выявленных сущностей, построим ER-диаграмму (рис.8):

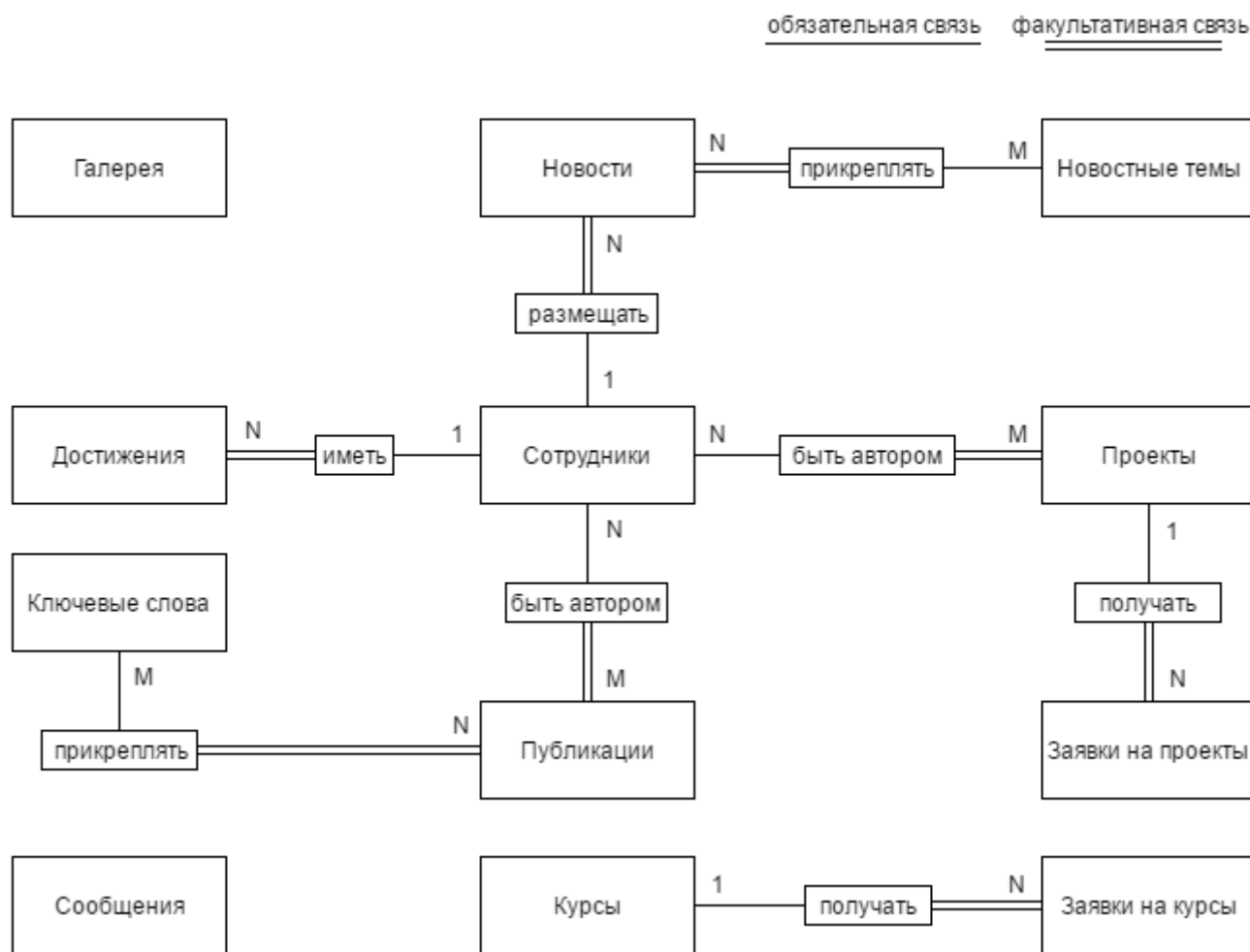


Рис. 8. ER-диаграмма предметной области «Научно-исследовательская лаборатория»

4.1.3 Обоснование выбора СУБД

СУБД представляет собой программное обеспечение, позволяющее создавать, определять и манипулировать базой данных. Это инструмент, который используется для выполнения каких-либо операций над данными в базе данных. СУБД обеспечивает надежную защиту и безопасность базы данных, поддерживает целостность данных при работе нескольких пользователей.

Рассмотрим основные свободно распространяемые СУБД [11]:

SQLite – мощная система управления, встраиваемая в приложение, которое её использует. Все данные система хранит в одном файле, что повышает её переносимость на другие машины. Все обращения при работе с данной СУБД происходят напрямую к файловой системе, что делает эту систему крайне быстрой и эффективной. Однако, данная СУБД существенно ограничивает операции записи до одного процесса записи в заданный промежуток времени, что сильно сказывается на

производительности системы. Существенным недостатком также является и то, что данная СУБД не предназначена для работы с многопользовательскими приложениям.

MySQL позиционируется как наиболее приспособленная для использования в веб-среде СУБД. Благодаря наличию большого количества информации в интернете, данная система достаточно проста в установке и использовании. Следует отметить, что высокая популярность MySQL привела к появлению различных сторонних инструментов (плагинов и расширений), которые в значительной мере облегчают работу с данной СУБД. Существует ряд причин, которые выделяют MySQL среди остальных СУБД:

- широкие функциональные возможности по обеспечению безопасности;
- работа с большими объемами данных;
- хорошая масштабируемость;
- значительный прирост производительности системы в связи с упрощением некоторых стандартов.

Несмотря на некоторые ограничения функциональности, вызванные неполным соответствием системы стандартам языка SQL, большинство веб-приложений спокойно работает с MySQL. Будучи легкой и масштабируемой, система MySQL проверена временем.

PostgreSQL – передовая система управления базой данных, главной целью которой является максимальное соответствие стандартам языка SQL. Возможности PostgreSQL очень легко расширить за счет написания хранимых процедур. Эти функции помогают упростить использование сложных и часто повторяемых операций с базой данных. Так же, как и MySQL, PostgreSQL может похвастаться наличием большого количества дополнений, несмотря на всё богатство встроенных функций. Однако, сложность настройки и не самые лучшие показатели производительности при выполнении простых операций чтения, делают эту систему не самой популярной.

Таким образом, наиболее оптимальным решением будет выбор СУБД MySQL. В дополнение к перечисленным преимуществам стоит также добавить, что данная СУБД обладает наилучшей скоростью обработки данных объемом до 500 тыс.

записей, а также поддерживается большинством компаний по предоставлению хостинга.

4.1.4 Построение схемы базы данных и описание отношений

Для того, чтобы создать базу данных, необходимо построить её схему. Преобразование ER-диаграммы в схему базы данных осуществляется путем замены выявленных сущностей и связей между ними реляционными отношениями (таблицами) базы данных.

Степень связи один-ко-многим (1:N) между отношениями реализуется путем добавления внешнего ключа в то отношение, к которому эта множественная связь осуществляется. При этом, внешний ключ обязательно должен соответствовать первичному или уникальному ключу родительского отношения.

Связи быть автором и прикреплять между отношениями принадлежат к типу многие-ко-многим (N:M). Для такого типа связи вводятся вспомогательные отношения, которые содержат комбинации первичных ключей соответствующих исходных отношений.

С учетом этих особенностей построим окончательную схему проектируемой базы данных (рис.9).

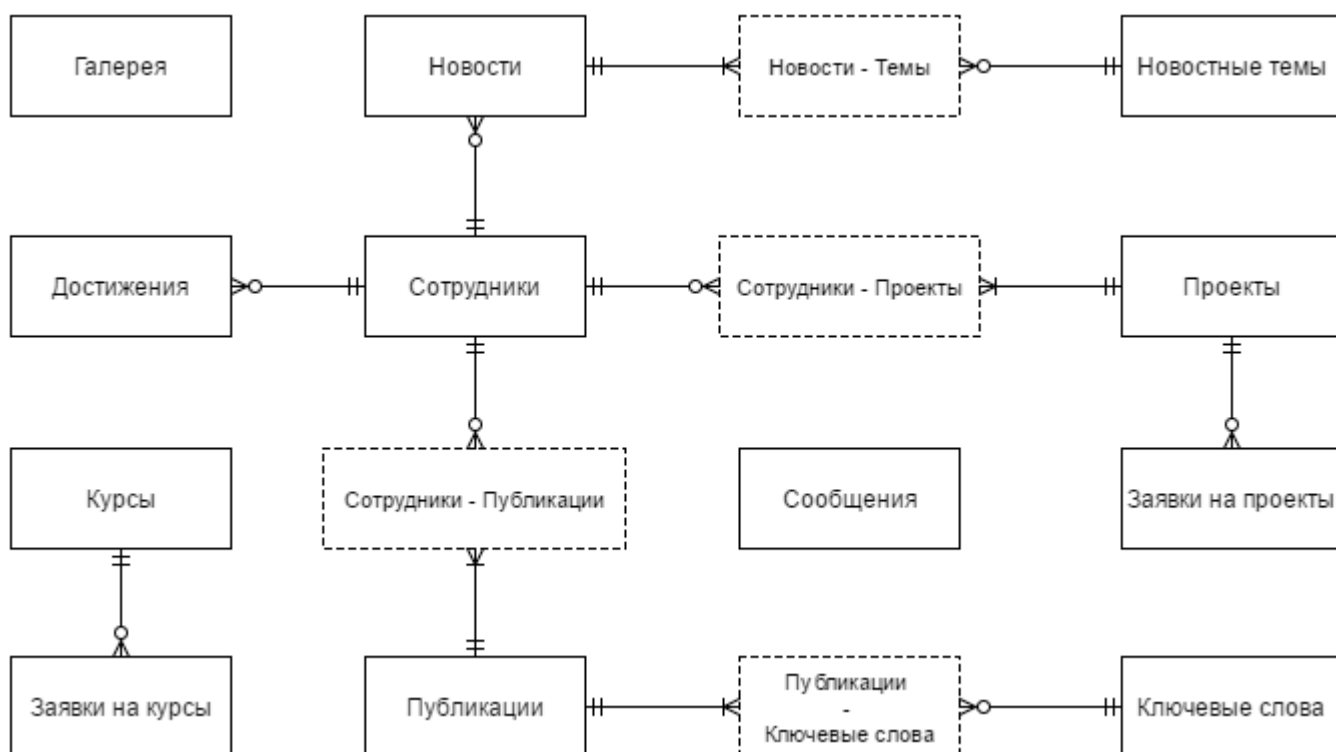


Рис. 9. Схема реляционной базы данных лаборатории

Ниже приведено описание некоторых полученных отношений (табл.3 – 10):

Таблица 3

Отношение «Новости»

Поле	Тип	Доп. атрибуты	Примечание
id	int(11)	auto_increment, unsigned	уникальный идентификатор новости
post_date	datetime		дата добавления новости
post_author_id	int(11)	unsigned	внешний ключ, указывающий на автора новости
post_title	varchar(255)		заголовок новости
post_announce	text		краткое описание новости
post_desc	text		полное описание новости
post_type	tinyint(1)	unsigned	тип новости: обычная/важная/черновик
created_at	int(11)		дата внесения записи в базу
updated_at	int(11)		дата обновления записи

Таблица 4

Отношение «Сотрудники»

Поле	Тип	Доп. атрибуты	Примечание
id	int(11)	auto_increment, unsigned	уникальный идентификатор пользователя
user_login	varchar(45)		логин, уникальное значение

user_password	varchar(64)		пароль
role	varchar(64)		роль доступа
auth_key	varchar(32)		ключ аутентификации
user_name	varchar(45)		имя
user_surname	varchar(45)		фамилия
user_patronymic	varchar(45)		отчество
user_email	varchar(255)		email
user_phone	varchar(30)		телефон
user_biography	text		биография
user_resume	varchar(255)		путь к файлу резюме пользователя
user_photo	varchar(255)		путь к фотографии пользователя
user_acdegree	varchar(45)		ученая степень
user_acrank	varchar(45)		ученое звание
visibility	tinyint(1)	unsigned	видимость: видимый/скрытый

В данных пользователей (табл.4) интерес представляют поля role, auth_key и visibility. Ключ аутентификации (auth_key) автоматически генерируется системой при внесении нового пользователя в базу и используется для проверки подлинности этого пользователя. Поле role осуществляет привязку пользователя к определенной группе (в нашем случае, администраторы или сотрудники), обладающей теми или иными правами доступа. Поле visibility определяет, будет отображаться пользователь в общем списке на сайте или нет.

Таблица 5

Отношение «Публикации»

Поле	Тип	Доп. атрибуты	Примечание
id	int(11)	auto_increment, unsigned	уникальный идентификатор публикации
project_id	int(11)	unsigned	внешний ключ, указывающий на привязанный проект
public_date	date		дата публикации
public_type	tinyint(1)		тип: статья/глава книги/книга
public_title	varchar(255)		название публикации
public_annotation	text		аннотация
public_info	text		уточняющая информация
public_lang	varchar(20)		язык публикации
public_file	varchar(255)		путь к файлу с полным текстом публикации

Поле `project_id` позволяет осуществить привязку публикации к проекту (если такой имеется), по результатам которого была написана публикации. Данная привязка используется для отображения всех имеющихся публикации на странице проекта.

Таблица 6

Отношение «Проекты»

Поле	Тип	Доп. атрибуты	Примечание
<code>id</code>	<code>int(11)</code>	<code>auto_increment, unsigned</code>	уникальный идентификатор проекта
<code>project_title</code>	<code>varchar(255)</code>		название проекта
<code>project_announce</code>	<code>text</code>		краткое описание проекта
<code>project_desc</code>	<code>text</code>		полное описание проекта
<code>project_price</code>	<code>decimal(7,2)</code>		предполагаемая цена
<code>project_status</code>	<code>tinyint(1)</code>	<code>unsigned</code>	статус проекта: в разработке/в производстве/устарел
<code>created_at</code>	<code>int(11)</code>		дата внесения записи в базу
<code>updated_at</code>	<code>int(11)</code>		дата обновления записи

Поле `project_status` позволяет задать статус проекта, в зависимости от которого будет решаться, отображать форму заказа по данному проекту или же нет.

Отношение «Курсы» имеет идентичную с табл.6 структуру, поэтому описание этой таблицы приводиться не будет.

Таблица 7

Отношение «Достижения сотрудника»

Поле	Тип	Доп. атрибуты	Примечание
<code>id</code>	<code>int(11)</code>	<code>auto_increment, unsigned</code>	уникальный идентификатор достижения
<code>user_id</code>	<code>int(11)</code>	<code>unsigned</code>	внешний ключ, указывающий на автора достижения
<code>achieve_date</code>	<code>date</code>		дата достижения
<code>achieve_desc</code>	<code>text</code>		описание достижения

Отношения «Заявки по проектам» и «Заявки по курсам» имеют схожую структуру. В качестве примера будет приведена структура отношения «Заявки по курсам» (табл.8).

Таблица 8

Отношение «Заявки по курсам»

Поле	Тип	Доп. атрибуты	Примечание
id	int(11)	auto_increment, unsigned	уникальный идентификатор заявки
request_course_id	int(11)	unsigned	внешний ключ, указывающий на запрошенный курс
request_date	datetime		дата подачи заявки
user_fio	varchar(255)		ФИО пользователя
user_email	varchar(64)		email пользователя
user_phone	varchar(30)		телефон пользователя
addtext	text		добавочная информация
request_status	tinyint(1)	unsigned	статус заявки: обработана/не обработана

Таблица 9

Вспомогательное отношение «Новости – Новостные темы»

Поле	Тип	Доп. атрибуты	Примечание
pt_id	int(11)	auto_increment, unsigned	уникальный идентификатор записи в таблице
post_id	int(11)	unsigned	внешний ключ к новости
tag_id	int(11)	unsigned	внешний ключ к новостной теме

Структура остальных вспомогательных отношений абсолютно идентична той, что указана в табл.9. Отличие состоит только в наименовании полей и внешних ключах, которые указывает на соответствующие отношения, связь которых описывается в вспомогательном отношении.

Таблица 10

Отношение «Сообщения из формы обратной связи»

Поле	Тип	Доп. атрибуты	Примечание
id	int(11)	auto_increment, unsigned	уникальный идентификатор сообщения
message_date	datetime		дата сообщения
message_topic	varchar(255)		тема сообщения
user_name	varchar(45)		имя пользователя
user_email	varchar(64)		email пользователя
user_phone	varchar(30)		телефон пользователя
message_text	text		полный текст сообщения
message_status	tinyint(1)	unsigned	статус: прочитано/не прочитано

В Приложении 1 приведен SQL-код, описывающий создание представленных отношений в базе данных.

4.2 Описание системы

Вся структура приложения спроектирована согласно популярному паттерну проектирования Модель-Представление-Контроллер (рис.10). Преимуществом такого подхода является четкое разделение логики приложения на различные компоненты таким образом, что изменение одного из них оказывает минимальное воздействие на все остальные [12].

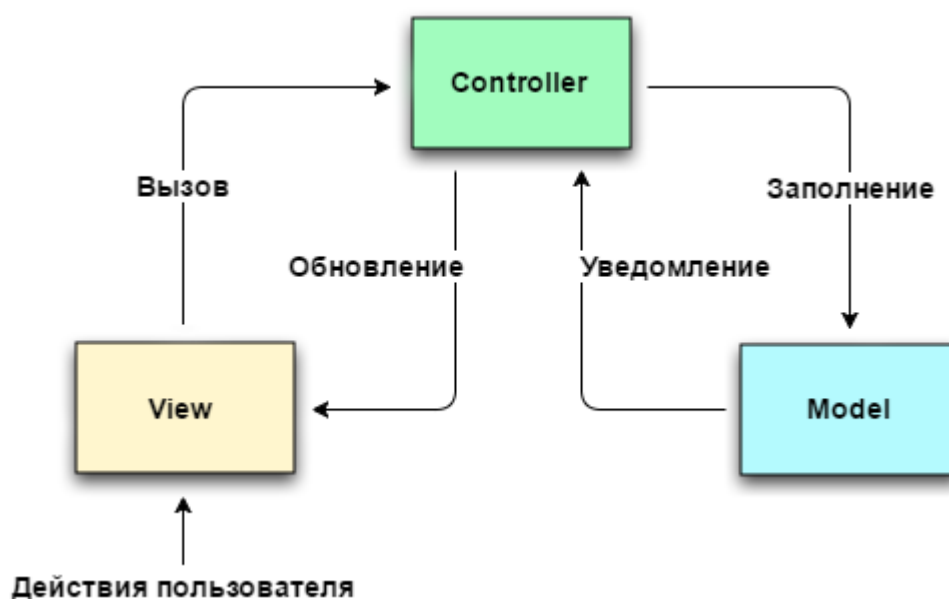


Рис. 10. Паттерн проектирования MVC

Модель реализует определенную бизнес-логику для домена данных приложения. Задачей модели является приём данных из контроллера, валидация данных согласно заданным в модели правилам, и использование этих данных для отправки запроса в базу данных.

Представление описывает способ представления данных в пользовательском интерфейсе приложения. В основном, для отображения представление использует данные, полученные из модели.

Контроллер служит для управления запросами пользователей, работы с моделью, а также выбора представления для отображения пользовательского интерфейса. Контроллер отвечает на действия пользователей, обрабатывает

вводимые ими данные и загружает эти данные в модель, после чего генерирует необходимое представление.

На рис.11 показано, каким образом приложение обрабатывает запрос пользователя.

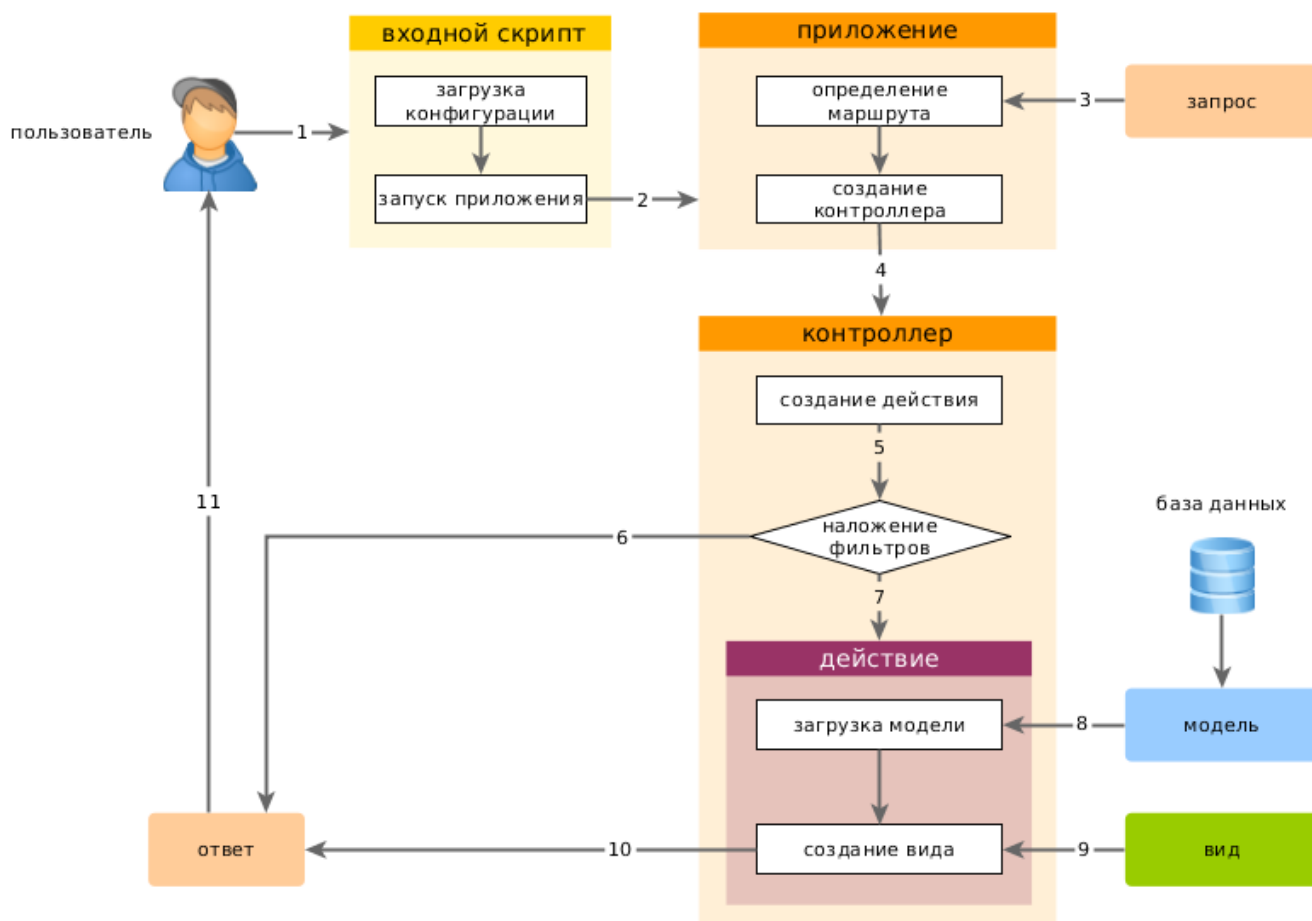


Рис. 11. Реакция приложения на запрос пользователя

Каждый раз, когда пользователь заходит в систему, входной скрипт `index.php` загружает файл конфигурации `web.php`, после чего создает экземпляр приложения для дальнейшей обработки запроса. Приложение определяет маршрут, на который необходимо перевести запрос, при помощи встроенного в приложение компонента «запрос». Далее для выполнения запроса приложение создает экземпляр контроллера, который, в свою очередь, создает действие и накладывает на него необходимые фильтры (такие как фильтр доступа, фильтр по типу запроса и т.д.). После того, как все фильтры будут успешно пройдены, действие контроллера загружает модель данных из базы данных и генерирует представление, заполняя его данными из

модели. Сгенерированное представление передается как компонент приложения «ответ». Компонент «ответ» отправляет результат работы приложения в браузер пользователя.

4.2.1 Модуль администрирования

Структурная организация административного модуля представлена на рис.12.

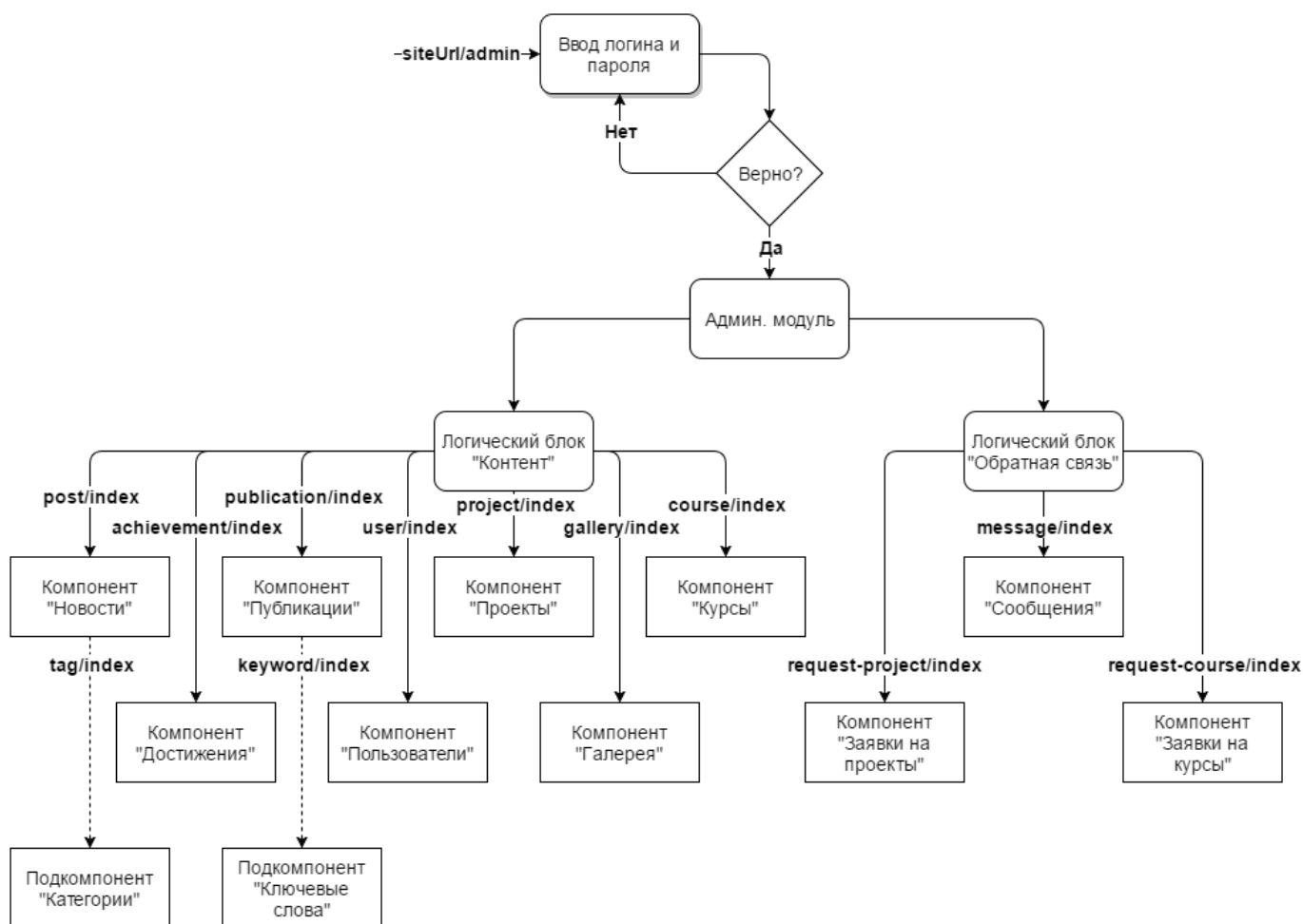
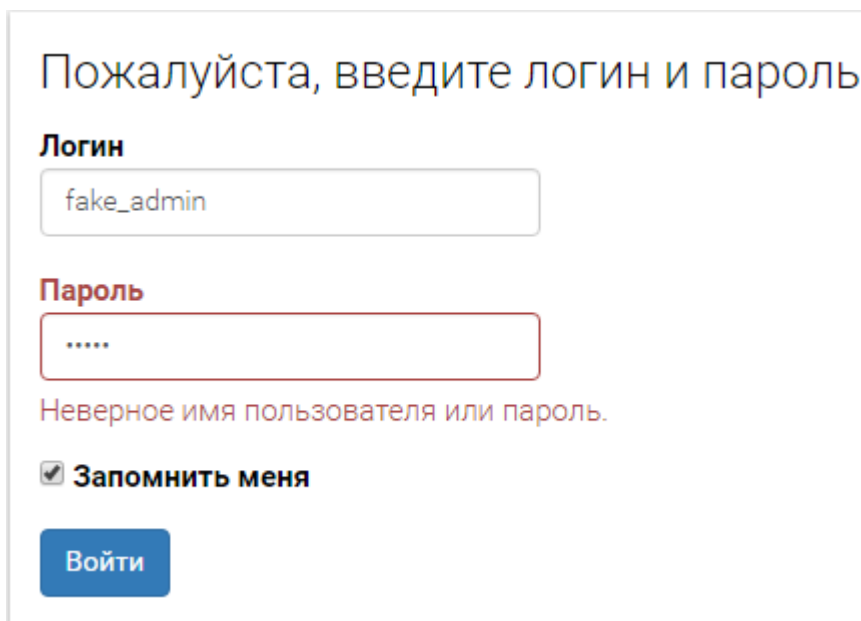


Рис. 12. Структурная организация модуля администрирования

Переход в административную панель осуществляется по маршруту `siteUrl/admin`. `siteUrl` является указателем на входной скрипт приложения, `admin` в данном случае представляет идентификатор соответствующего модуля.

Для входа в панель администратора необходимо авторизоваться в системе. После того, как пользователь отправит учетные данные на сервер, они проходят валидацию на соответствие логина и пароля имеющимся сведениям в базе данных.

Если пользователя с такими учетными данными не удалось найти в базе данных, возвращается сообщение о некорректно введенных данных (рис.13).



Пожалуйста, введите логин и пароль

Логин

Пароль

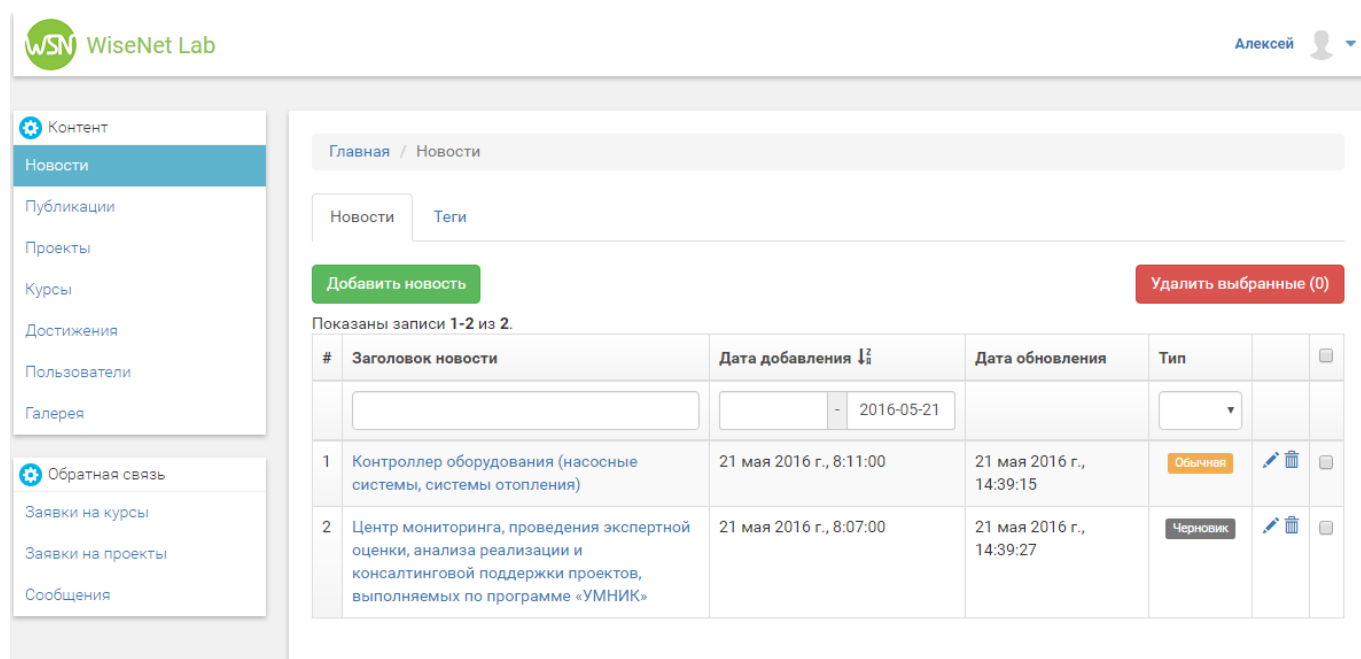
Неверное имя пользователя или пароль.

☒ **Запомнить меня**

Войти

Рис. 13. Сообщение об ошибке авторизации

В случае же успешной авторизации осуществляется подключение компонентов модуля и происходит перенаправление пользователя в административный интерфейс (рис.14).



WSN WiseNet Lab

Алексей

Главная / Новости

Новости Теги

Добавить новость Удалить выбранные (0)

Показаны записи 1-2 из 2.

#	Заголовок новости	Дата добавления	Дата обновления	Тип	
	<input type="text"/>	<input type="text"/> - 2016-05-21		<input type="text"/>	
1	Контроллер оборудования (насосные системы, системы отопления)	21 мая 2016 г., 8:11:00	21 мая 2016 г., 14:39:15	Обычная	<input type="text"/> <input type="text"/> <input type="text"/>
2	Центр мониторинга, проведения экспертной оценки, анализа реализации и консалтинговой поддержки проектов, выполняемых по программе «УМНИК»	21 мая 2016 г., 8:07:00	21 мая 2016 г., 14:39:27	Черновик	<input type="text"/> <input type="text"/> <input type="text"/>

Контент

- Новости
- Публикации
- Проекты
- Курсы
- Достижения
- Пользователи
- Галерея

Обратная связь

- Заявки на курсы
- Заявки на проекты
- Сообщения

Рис. 14. Административный интерфейс

Как было указано на рис.12, за отображение различных компонентов модуля отвечает конструкция вида `ModuleID/ControllerID/ActionID`, где `ModuleID` – идентификатор нашего модуля администрирования, `ControllerID` – идентификатор контроллера компонента, `ActionID` – идентификатор действия, ответственного за отображение главной страницы компонента. Действие `index()` ответственно за создание поисковой модели компонента, которая загружает записи из базы данных, и передачу загруженных данных в главное представление `index.php`. За отображение записей отвечает встроенный в Yii2 виджет `GridView`, для которого необходимо задать следующие параметры:

- `$dataProvider` – набор данных, полученных от контроллера;
- `$filterModel` – поисковая модель для организации гибкой системы фильтрации данных;
- `$gridColumns` – отвечает за формирование ячеек таблицы и формат их отображения.

За отображение кнопок редактирования и удаления записей отвечает колонка `ActionColumn`. За редактирование и удаление записей отвечают действия `update($id)` и `delete($id)` контроллера компонента соответственно, на вход которым подается идентификатор `$id` выбранной записи.

Также для каждого компонента предусмотрена возможность множественного удаления записей. На рис.14, представленном выше, изображена главная страница компонента «Новости». Главные страницы остальных компонентов имеют схожую структуру и отличаются только набором полей в таблице.

Рассмотрим вкратце каждый компонент в отдельности.

Компонент «Новости»

Компонент логически разделен на две части: новостные записи и новостные категории.

Функционал компонента представлен следующими особенностями:

- возможность добавления различных новостных категорий, при этом для большего удобства добавление может осуществляться в реальном времени, без необходимости использования подкомпонента «Новостные категории»;

- возможность прикрепления новости к какому-то конкретному проекту, для того чтобы пользователь мог отследить динамику развития проекта при просмотре странички проекта на сайте;
- гибкое редактирование содержимого новости с применением визуального редактора с возможностью добавления списков, изображений, видео и др. (рис.15);
- возможность добавления документов при добавлении новой новости и их открепления при редактировании новости;
- возможность выбора статуса новости (обычная/важная/черновик);
- изменение даты добавления.

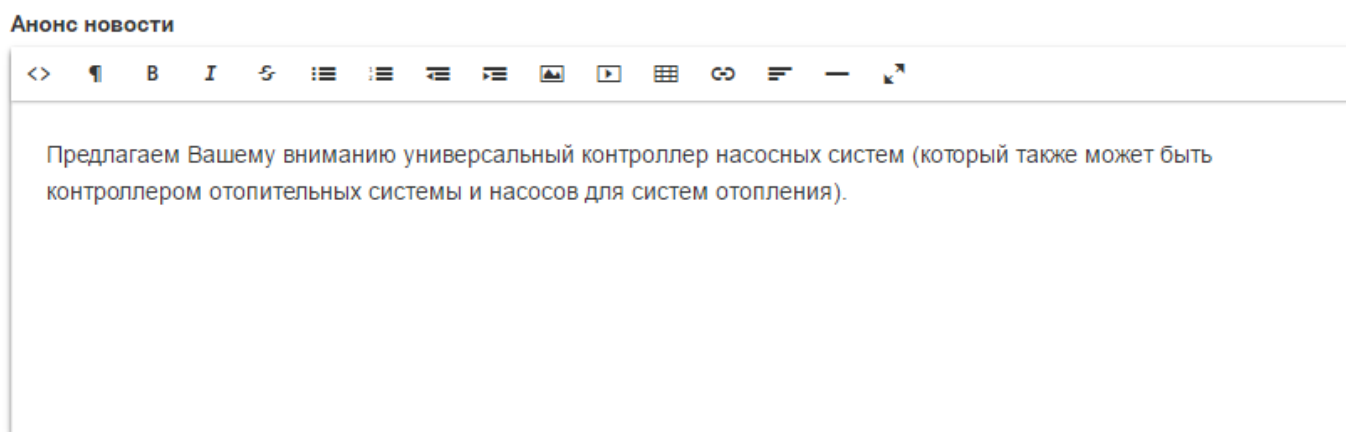


Рис. 15. Визуальный редактор контента

При этом каждое поле проходит обязательную валидацию на соответствие заданным в модели компонента требованиям. В соответствии с этими требованиями, все поля, за исключением привязки новости к проекту и добавления документов должны быть обязательно заполнены, в противном случае администратор получит предупреждение (рис.16).

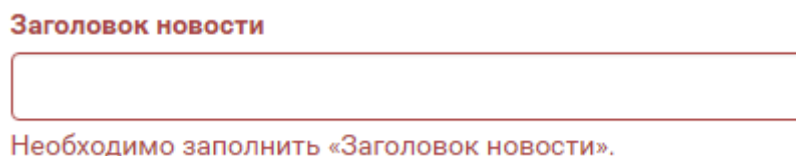


Рис. 16. Предупреждение о пропущенном поле

Также исключена вероятность дублирования категорий при добавлении/редактировании новостей. В методе `afterSave()` модели компонента,

который срабатывает после сохранения модели в базе данных, обеспечивается проверка на наличие повторений загруженного в модель массива категорий и удаление повторяющихся категорий. После обработки все категории сохраняются в базе данных и осуществляется привязка новости к категориям через вспомогательное отношение «Новости – Новостные категории».

Выбор статуса новости влияет на характер её отображения на веб-сайте: обычные новости отображаются в общей ленте новостей, важные новости помещаются в блок «важное», черновые варианты новостей на веб-сайте не отображаются.

Компонент «Публикации»

Компонент также логически разделен на две части: публикации и ключевые слова.

Компонент предоставляет следующие возможности по управлению публикациями:

- возможность установки типа публикации (статья/глава книги/книга);
- возможность добавления ключевых слов;
- возможность привязки и удаления авторов публикации;
- возможность привязки публикации к конкретному проекту;
- возможность добавления документа к публикации (например, её полный текст);
- возможность экспорта таблицы с перечнем публикаций в различные форматы файлов (PDF, Excel и др.).

Все поля в форме редактирования публикаций, за исключением привязки к проекту, являются обязательными для заполнения. Также исключено дублирование ключевых слов за счет удаления повторяющихся.

Если при обновлении публикации был добавлен новый файл, то в методе `beforeSave()` модели компонента осуществляется удаление старого файла с сервера, сохранение нового файла на сервер, после чего модель сохраняется в базе данных, перезаписывая поле `public_file` (путь к файлу публикации).

Компоненты «Проекты» и «Курсы»

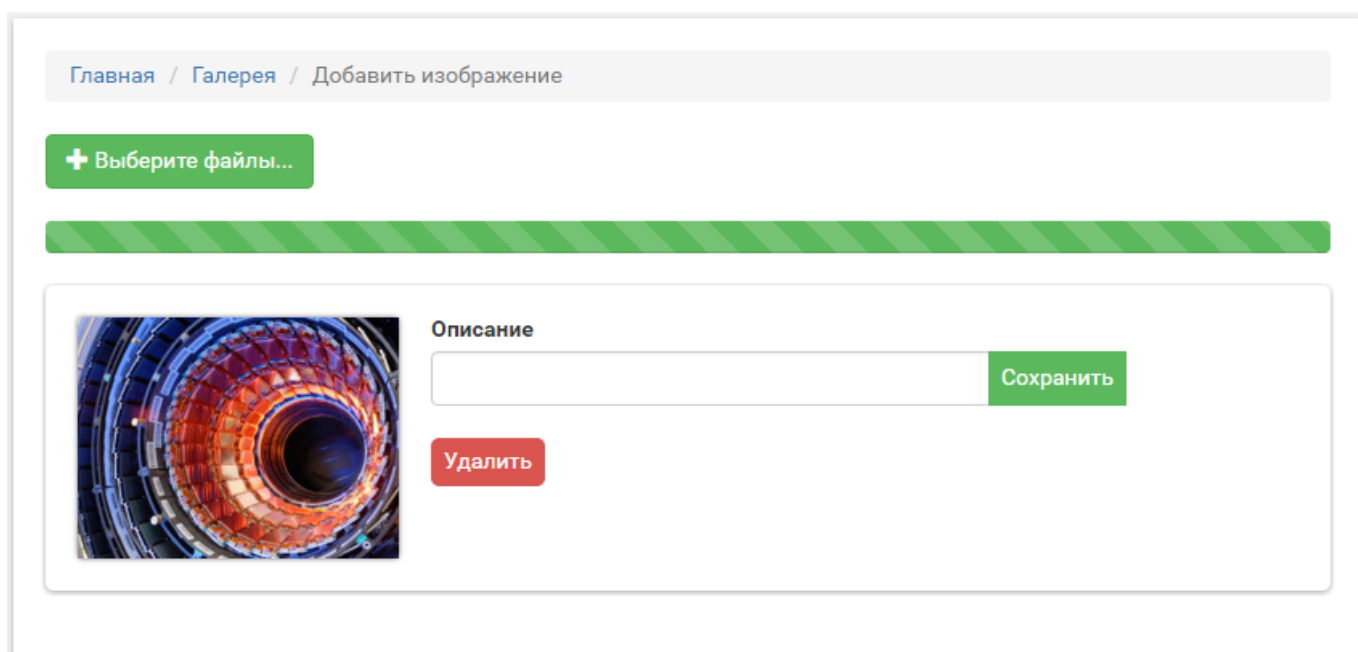
Данные компоненты имеют схожий функционал и предоставляют следующие возможности по управлению ими:

- редактирование содержимого с помощью текстового редактора;
- установка статуса проекта (в разработке/в производстве/устарел) / курса (регистрация открыта/регистрация закрыта/неактивен);
- привязка и удаление авторов (только для проекта);
- установка цены проекта/курса (для проекта опционально).

Если проект находится в производстве, то пользователю будет доступна форма заказа на страничке проекта, в противном случае форма отображаться не будет. Форма регистрации на курс будет доступна только в том случае, если регистрация на него открыта администратором. Неактивные курсы не отображаются в общем списке на странице веб-сайта.

Компонент «Галерея»

Компонент предназначен для быстрой загрузки изображений на сервер для их последующего отображения на веб-сайте. На рис.17 показан интерфейс загрузки изображения.



The screenshot shows a web interface for uploading an image. At the top, there is a breadcrumb navigation bar with links: Главная / Галерея / Добавить изображение. Below this is a green button with a plus icon and the text '+ Выберите файлы...'. A thick green diagonal-striped bar separates the upload section from the image preview section. The preview section contains a square image of a colorful, circular, textured object. To the right of the image is a text input field labeled 'Описание'. Below the input field are two buttons: a green 'Сохранить' (Save) button and a red 'Удалить' (Delete) button.

Рис. 17. Интерфейс загрузки изображения

Виджет загрузки изображений поддерживает также множественный выбор. После того, как администратор выбрал изображения, они посылаются AJAX-запросом на сервер. Перед сохранением в базу данных, каждое изображение проходит проверку на соответствие заданным в системе требованиям:

- изображение не должно превышать размер 5 Мбайт;
- если первое условие выполнено, осуществляется проверка размеров изображения (максимально допустимым считаются размеры 800 пкс в ширину и 600 пкс в высоту).

Если размеры изображения не соответствуют этим требованиям, то оно ужимается до приемлемых размеров (для этих целей используется стандартная библиотека языка PHP `imagick.dll`). В целях оптимизации загрузки веб-сайта для изображения также создается его уменьшенная копия. После предварительной обработки изображение и его уменьшенная копия сохраняются на сервере, а в базу заносится уникальное имя. В модели компонента предусмотрены методы, которые возвращают путь к файлу на сервере:

- методы `getImagePath()` и `getThumbPath()` возвращают абсолютные пути к изображению и его уменьшенной копии соответственно;
- методы `getImageLink()` и `getThumbLink()` возвращают их относительные пути.

После сохранения модели изображения в базу, сервер посылает ответ браузеру с передачей в него данных модели, после чего загруженное изображение высвечивается в интерфейсе загрузки. При желании, администратор может тут же изменить описание изображения, либо вовсе его удалить (рис.17).

Компонент «Пользователи»

Компонент используется для просмотра имеющихся в системе пользователей, их редактирования/удаления и добавления новых. На рис.18 представлен интерфейс компонента.

При добавлении пользователя необходимо указать его логин и пароль, роль доступа и статус отображения, а также ФИО и email. Остальные поля являются опциональными. При редактировании предоставляется возможность изменения

пароля пользователя: для этого необходимо указать его текущий пароль, а также задать новый (рис.19).

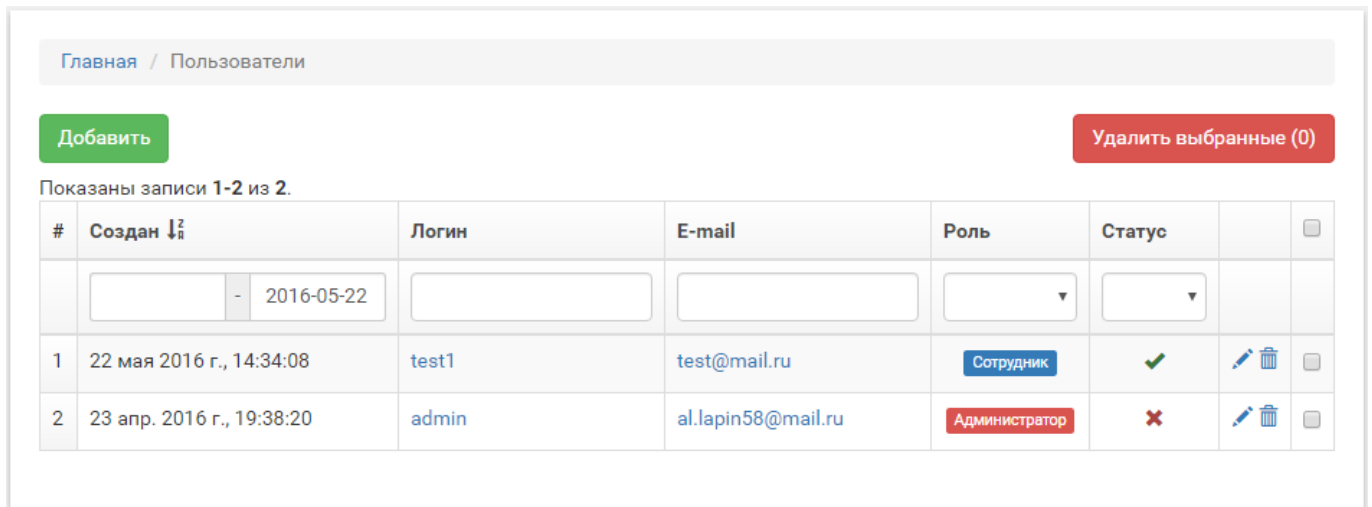


Рис. 18. Интерфейс компонента «Пользователи»

Пользователи со статусом скрытый на веб-сайте не отображаются.

The screenshot shows the 'Change Password' form for the 'admin' user. The breadcrumb navigation bar shows 'Главная / Пользователи / admin / Редактирование'. There are two tabs: 'Основное' (Main) and 'Изменение пароля' (Change Password). The form has three input fields: 'Текущий пароль' (Current password), 'Новый пароль' (New password), and 'Повторите пароль' (Repeat password). A 'Сохранить' (Save) button is at the bottom.

Главная / Пользователи / admin / Редактирование

Основное | Изменение пароля

Текущий пароль

Новый пароль

Повторите пароль

Сохранить

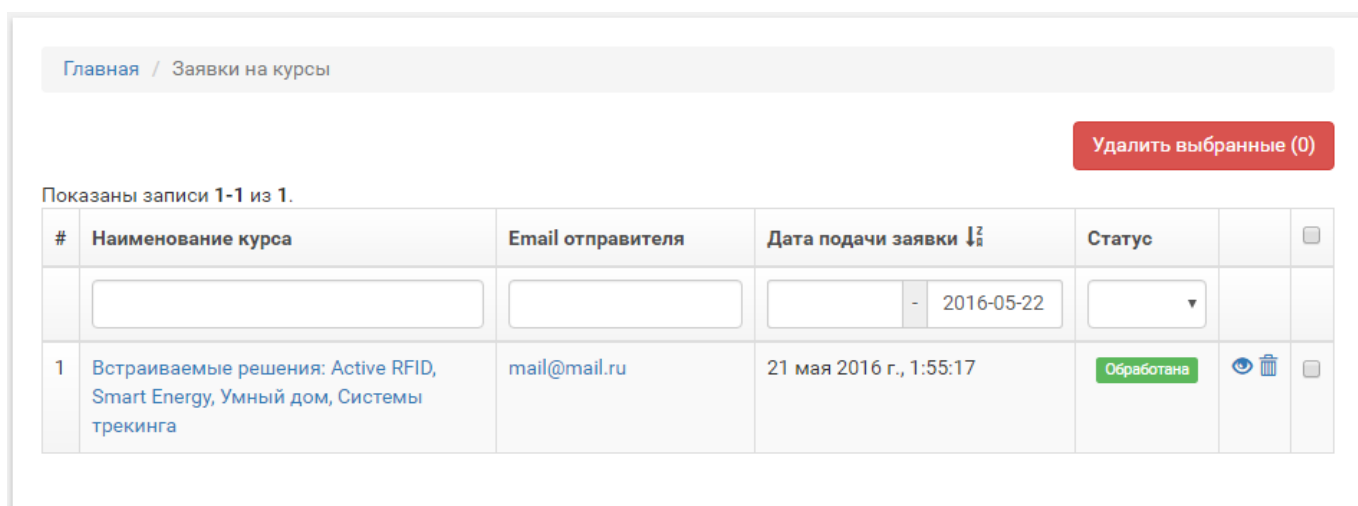
Рис. 19. Изменение пароля пользователя

В модели компонента предусмотрен метод `validatePassword()`, отвечающий за проверку пароля из базы данных и пароля, введенного администратором, путем сопоставления их хэшей. В случае несовпадения хэшей выводится соответствующее предупреждение.

Компонент «Достижения»

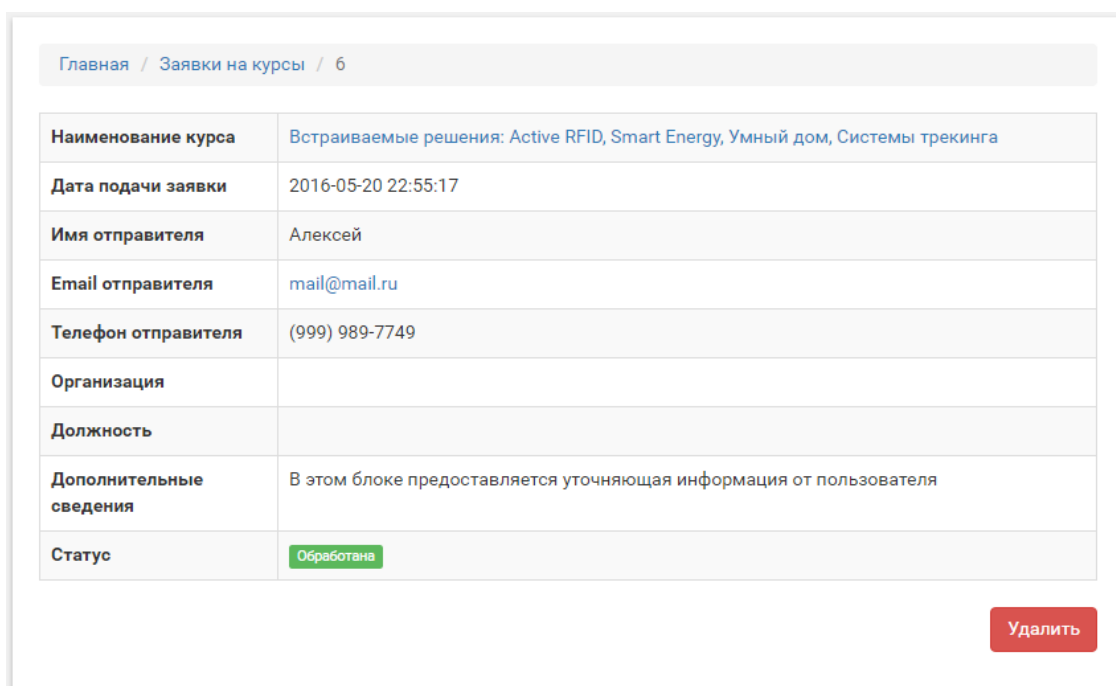
Компонент предоставляет функционал по управлению достижениями сотрудников. Для внесения достижения в базу необходимо указать идентификатор сотрудника, год и месяц достижения, а также его краткое описание.

Логический блок «обратная связь» представлен тремя компонентами: заявки по проектам, заявки по курсам, сообщения из формы обратной связи. Общий вид отображения всех записей и просмотра отдельной записи каждого из компонентов представлен на рис.20 и рис.21 соответственно.



Главная / Заявки на курсы						
Удалить выбранные (0)						
Показаны записи 1-1 из 1.						
#	Наименование курса	Email отправителя	Дата подачи заявки	Статус		
1	Встраиваемые решения: Active RFID, Smart Energy, Умный дом, Системы трекинга	mail@mail.ru	21 мая 2016 г., 1:55:17	Обработана		

Рис. 20. Страница отображения записей из блока обратной связи на примере компонента «Заявки на курсы»



Наименование курса	Встраиваемые решения: Active RFID, Smart Energy, Умный дом, Системы трекинга
Дата подачи заявки	2016-05-20 22:55:17
Имя отправителя	Алексей
Email отправителя	mail@mail.ru
Телефон отправителя	(999) 989-7749
Организация	
Должность	
Дополнительные сведения	В этом блоке предоставляется уточняющая информация от пользователя
Статус	Обработана

Удалить

Рис. 21. Окно подробного просмотра заявки

Для заявок по курсам и заявок по проектам также предусмотрена возможность присвоения заявке статуса обработана. Счетчик всех необработанных заявок и непрочитанных сообщений отображается на главной странице административного интерфейса напротив каждой соответствующей секции.

В Приложении 2 приведен программный код некоторых основных блоков, описывающих поведение того или иного компонента.

4.2.2 Пользовательский интерфейс

Организация интерфейса пользователя представлена на рис.22.

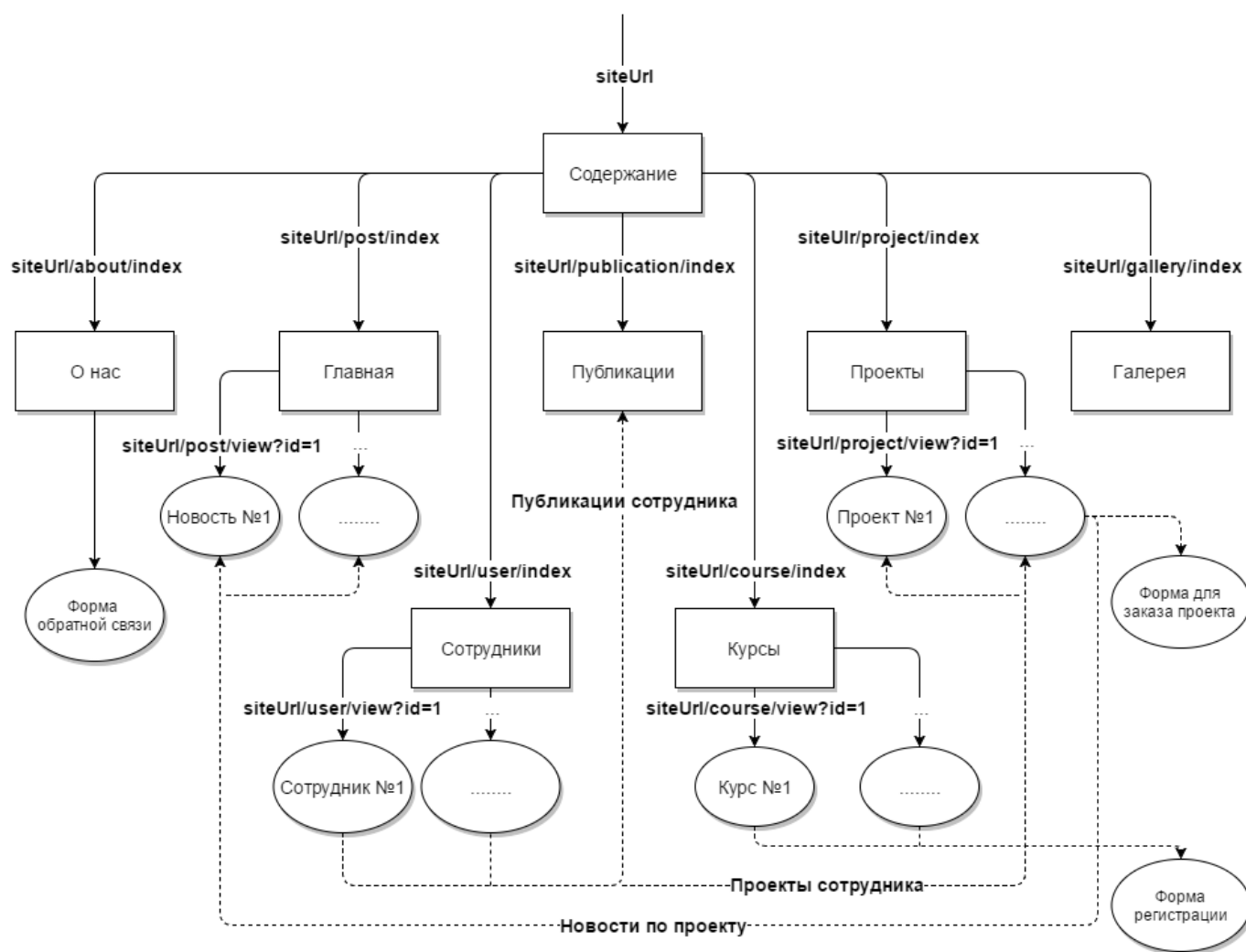


Рис. 22. Структурная организация веб-сайта

За отображение различных компонентов отвечает URL конструкция типа `siteUrl/ControllerID/ActionID`. Путь `siteUrl` указывает на входной скрипт приложения, который выполняет его инициализацию. `ControllerID` указывает на идентификатор

контроллера, ответственного за работу компонента. ActionID представляет идентификатор действия контроллера, которое выполняет строго определенную функцию. Так действие index предназначено для отображения полного списка записей по каждому из компонентов, в то время как действие view отвечает за формирование единичной записи (например, подробный просмотр новости). Для действия view в GET запросе также передается идентификатор записи \$id, по которому осуществляется инициализация модели для данной записи.

Информационная система состоит из набора взаимосвязанных веб-страниц с информационным и интерактивным материалом. Для некоторых компонентов также реализована взаимосвязь друг с другом. Так, например, можно просмотреть все проекты или публикации конкретного сотрудника, взаимосвязь проекта с новостями поможет получить более подробную информацию о его развитии.

Общая структура информационной системы показана на рис.23.

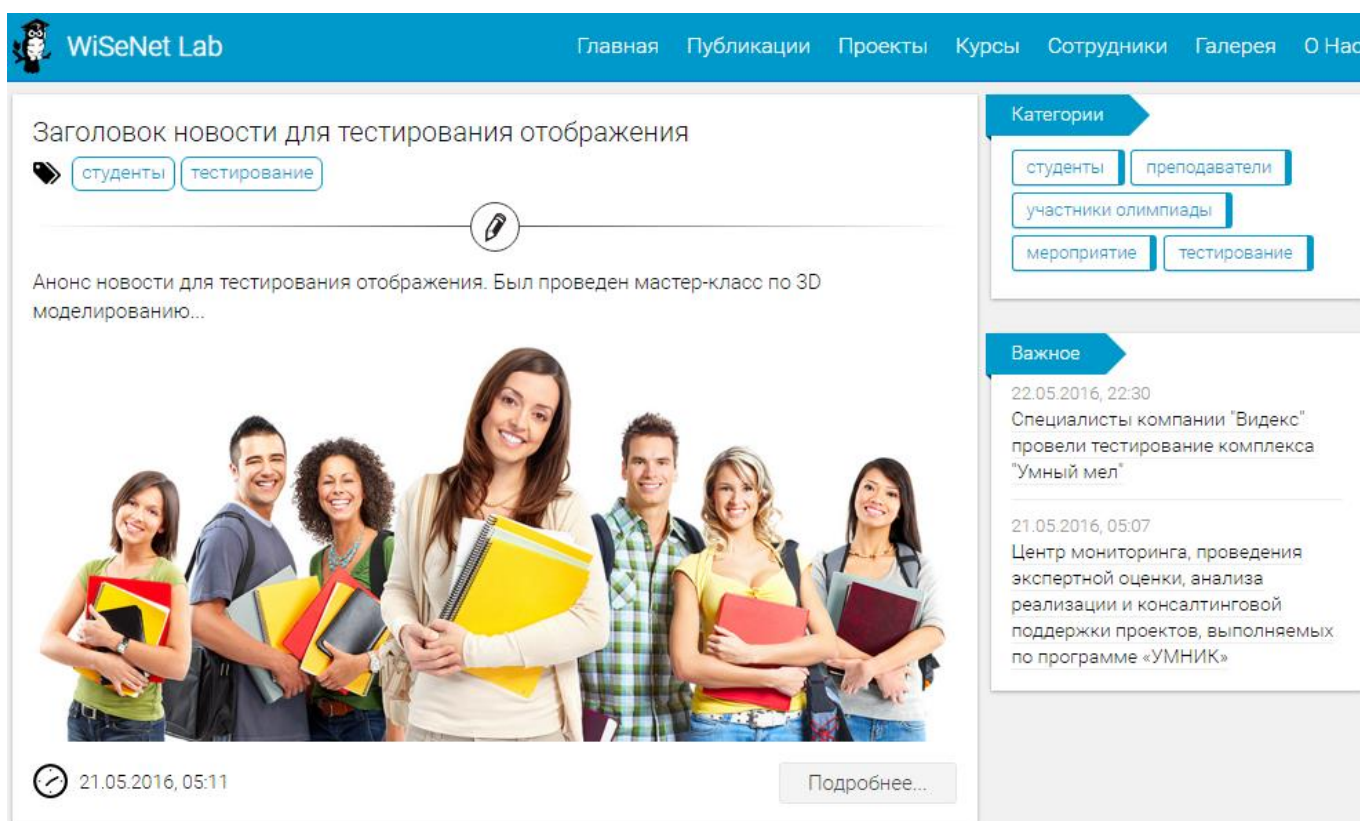


Рис. 23. Общая структура веб-сайта

Структура веб-сайта поделена на два блока. Левый блок применяется для отображения записей из различных компонентов веб-сайта. Правый блок является

фиксированным и отображает дополнительную информацию, которая варьируется в зависимости от отображаемого компонента. Так, для новостной ленты отображаются категории новостей (каждая категория является ссылкой на поиск новостей из этой категории), на страничке проекта в качестве дополнительной информации будут приведены новости и публикации по проекту, на страничке сотрудника будут отображаться публикации и проекты, к которым он причастен. Блок «важное» и «последние новости» являются общими при отображении всех компонентов.

На рис. 24 приведен обобщенный алгоритм работы форм для заявок и обратной связи.

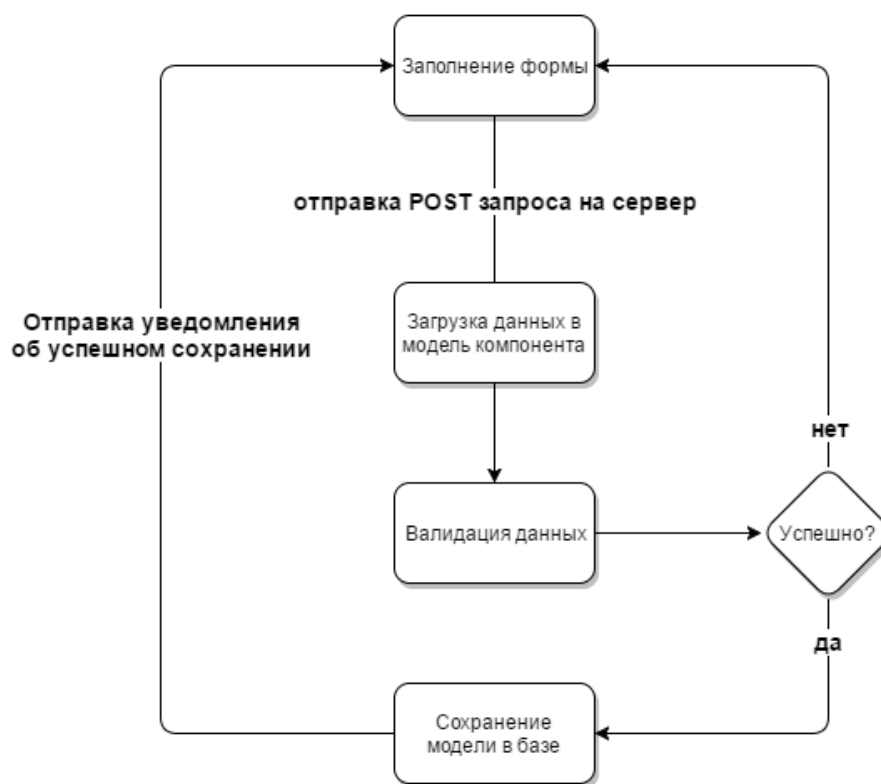



Рис. 24. Обобщенный алгоритм работы форм

Пример уведомления пользователя об успешной отправке формы представлен на рис.25.



Анонс проекта для тестирования отображение

Полное описание проекта для тестирования отображения


 Спасибо, Ваша заявка принята!

Рис. 25. Уведомление пользователя об успешной отправке формы

Заключение

В результате выполнения выпускной квалификационной работы была разработана информационно-аналитическая система для межвузовской лаборатории инновационных проектов. На момент написания работы ресурс находился в стадии внедрения.

В процессе выполнения работы были выполнены следующие задачи:

- проведен сравнительный обзор существующих современных методов разработки веб-ориентированных систем;
- выбран наиболее оптимальный метод разработки системы;
- приведено обоснование выбора языка программирования PHP и фреймворка Yii2 для достижения поставленной цели;
- описаны требования к разрабатываемому ресурсу;
- приведено описание вспомогательных решений, использующихся в веб-разработке;
- изучены методы построения баз данных;
- как итог, была разработана веб-система, включающая пользовательский интерфейс и административную панель для управления компонентами веб-системы (новости, публикации и проекты сотрудников лаборатории, курсы, фотогалерея, сотрудники).

Проект имеет большие возможности для дальнейшего развития. Предполагается усовершенствовать и оптимизировать имеющиеся и внедрить новые модули (рассылки, возможность регистрации и т.д.).

Наличие своего представительства в интернете наилучшим образом отобразится на высоком уровне организации деятельности лаборатории, на её имидже. Предполагается также, что разработанная система будет способствовать налаживанию связей с другими научными группами или организациями.

Список использованных источников

1. Ucoz.ru [Электронный ресурс]. URL: <http://www.ucoz.ru/tour/> (дата обращения 20.04.2016).
2. Wix.com [Электронный ресурс]. URL: <http://ru.wix.com/about/features> (дата обращения 20.04.2015).
3. Jimdo.com [Электронный ресурс]. URL: <http://ru.jimdo.com/цены/> (дата обращения 20.04.2015).
4. Williams B., Richard O., Tadlock J. Professional WordPress Plugin Development. – Wrox Press Ltd., 2011.
5. Marriott J., Waring E. Official Joomla! Book. – Addison-Wesley Professional, 2010.
6. Hunter L. The Drupal overview //Community Documentation. – 2008.
7. Achour M. et al. PHP manual. – 2006.
8. Wappalyzer.com [Электронный ресурс]. URL: <https://wappalyzer.com/categories/programming-languages> (дата обращения 11.05.2016).
9. Safronov M., Winesett J. Web Application Development with Yii 2 and PHP. – Packt Publishing Ltd, 2014.
10. Бегг К. и др. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. – 2000.
11. TEZER O. S. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems //Digital Ocean [online]. – 2014.
12. Curteanu M. Using the Model-View-Controller for Creating Applications for Project Management //Open Source Science Journal. – 2010. – Т. 2. – №. 4. – С. 150-66.
13. Колисниченко Д. Н. PHP и MySQL. Разработка Web-приложений. 4-е изд. – БХВ-Петербург, 2013.
14. Маклафлин Б., Поллайс Г., Уэст Д. Объектно-ориентированный анализ и проектирование //СПб.: Питер. – 2013.

SQL-скрипт создания таблиц в базе данных

1. Новости

```
CREATE TABLE IF NOT EXISTS `laboratorydb`.`post` (  
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,  
  `post_date` DATETIME NOT NULL,  
  `post_author_id` INT(11) UNSIGNED NOT NULL,  
  `post_project_id` INT(11) UNSIGNED NULL DEFAULT NULL,  
  `post_title` VARCHAR(255) NOT NULL,  
  `post_title_en` VARCHAR(255) NULL DEFAULT NULL,  
  `post_announce` TEXT NOT NULL,  
  `post_announce_en` TEXT NULL DEFAULT NULL,  
  `post_desc` TEXT NOT NULL,  
  `post_desc_en` TEXT NULL DEFAULT NULL,  
  `post_type` TINYINT(1) UNSIGNED NOT NULL DEFAULT '1',  
  `post_views` INT(11) UNSIGNED NOT NULL DEFAULT '0',  
  `created_at` INT(11) NOT NULL,  
  `updated_at` INT(11) NOT NULL,  
  PRIMARY KEY (`id`),  
  INDEX `fk_post_author` (`post_author_id` ASC),  
  INDEX `fk_project_id` (`post_project_id` ASC),  
  CONSTRAINT `fk_post_author`  
    FOREIGN KEY (`post_author_id`)  
    REFERENCES `laboratorydb`.`user` (`id`)  
    ON UPDATE CASCADE,  
  CONSTRAINT `fk_project_id`  
    FOREIGN KEY (`post_project_id`)  
    REFERENCES `laboratorydb`.`project` (`id`))  
    ON DELETE SET NULL  
  ENGINE = InnoDB  
  DEFAULT CHARACTER SET = utf8
```

2. Публикации

```
CREATE TABLE IF NOT EXISTS `laboratorydb`.`publication` (  
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,  
  `project_id` INT(11) UNSIGNED NULL DEFAULT NULL,  
  `public_date` DATE NOT NULL,  
  `public_type` TINYINT(1) UNSIGNED NOT NULL,  
  `public_title` VARCHAR(255) NOT NULL,  
  `public_annotation` TEXT NOT NULL,  
  `public_info` TEXT NOT NULL,  
  `public_lang` VARCHAR(20) NOT NULL,  
  `public_file` VARCHAR(255) NOT NULL,  
  PRIMARY KEY (`id`),  
  INDEX `fk_publication_project` (`project_id` ASC),  
  CONSTRAINT `fk_publication_project`  
    FOREIGN KEY (`project_id`)  
    REFERENCES `laboratorydb`.`project` (`id`)  
    ON DELETE SET NULL  
    ON UPDATE CASCADE)  
  ENGINE = InnoDB  
  DEFAULT CHARACTER SET = utf8
```

3. Проекты

```
CREATE TABLE IF NOT EXISTS `laboratorydb`.`project` (  
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,  
  `project_title` VARCHAR(255) NOT NULL,  
  `project_title_en` VARCHAR(255) NULL DEFAULT NULL,  
  `project_announce` TEXT NOT NULL,  
  `project_announce_en` TEXT NULL DEFAULT NULL,  
  `project_desc` TEXT NOT NULL,  
  `project_desc_en` TEXT NULL DEFAULT NULL,  
  `project_price` DECIMAL(7,2) NULL DEFAULT NULL,  
  `project_status` TINYINT(1) UNSIGNED NOT NULL,  
  `created_at` INT(11) NOT NULL,  
  `updated_at` INT(11) NOT NULL,  
  PRIMARY KEY (`id`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8
```

4. Курсы

```
CREATE TABLE IF NOT EXISTS `laboratorydb`.`course` (  
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,  
  `course_title` VARCHAR(255) NOT NULL,  
  `course_title_en` VARCHAR(255) NULL DEFAULT NULL,  
  `course_announce` TEXT NOT NULL,  
  `course_announce_en` TEXT NULL DEFAULT NULL,  
  `course_desc` TEXT NOT NULL,  
  `course_desc_en` TEXT NULL DEFAULT NULL,  
  `course_status` TINYINT(1) NOT NULL DEFAULT '0',  
  `course_price` DECIMAL(7,2) NOT NULL,  
  `created_at` INT(11) NOT NULL,  
  `updated_at` INT(11) NOT NULL,  
  PRIMARY KEY (`id`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8
```

5. Сотрудники

```
CREATE TABLE IF NOT EXISTS `laboratorydb`.`user` (  
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,  
  `created_at` INT(11) NOT NULL,  
  `updated_at` INT(11) NOT NULL,  
  `user_login` VARCHAR(45) NOT NULL,  
  `user_password` VARCHAR(64) NOT NULL,  
  `role` VARCHAR(64) NULL DEFAULT NULL,  
  `auth_key` VARCHAR(32) NULL DEFAULT NULL,  
  `user_name` VARCHAR(45) NOT NULL,  
  `user_surname` VARCHAR(45) NOT NULL,  
  `user_patronymic` VARCHAR(45) NOT NULL,  
  `user_email` VARCHAR(255) NOT NULL,  
  `user_phone` VARCHAR(30) NULL DEFAULT NULL,  
  `user_biography` TEXT NULL DEFAULT NULL,  
  `user_biography_en` TEXT NULL DEFAULT NULL,  
  `user_resume` VARCHAR(255) NULL DEFAULT NULL,  
  `user_photo` VARCHAR(255) NOT NULL DEFAULT '/user.png',  
  `user_acdegree` VARCHAR(45) NULL DEFAULT NULL,  
  `user_acrank` VARCHAR(45) NULL DEFAULT NULL,  
  `visibility` TINYINT(1) UNSIGNED NOT NULL DEFAULT '1',  
  PRIMARY KEY (`id`),
```



```

        UNIQUE INDEX `user_login_UNIQUE` (`user_login` ASC),
        UNIQUE INDEX `ix_user_email` (`user_email` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

6. Достижения

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`achievement` (
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `user_id` INT(11) UNSIGNED NOT NULL,
  `achieve_date` DATE NOT NULL,
  `achieve_desc` TEXT NOT NULL,
  `achieve_desc_en` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_achievement_user` (`user_id` ASC),
  CONSTRAINT `fk_achievement_user`
    FOREIGN KEY (`user_id`)
      REFERENCES `laboratorydb`.`user` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

7. Галерея

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`gallery` (
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `image_desc` VARCHAR(255) NULL DEFAULT NULL,
  `image_desc_en` VARCHAR(255) NULL DEFAULT NULL,
  `image_name` VARCHAR(255) NULL DEFAULT NULL,
  `created_at` INT(11) NOT NULL,
  `updated_at` INT(11) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

8. Новостные темы

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`tag` (
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `tag_name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `tag_name_UNIQUE` (`tag_name` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

9. Ключевые слова

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`keyword` (
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `keyword_name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `keyword_UNIQUE` (`keyword_name` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

10. Заявки на курсы

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`request_course` (
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `request_course_id` INT(11) UNSIGNED NOT NULL,
  `request_date` DATETIME NOT NULL,

```

```

`user_fio` VARCHAR(255) NOT NULL,
`user_email` VARCHAR(64) NOT NULL,
`user_phone` VARCHAR(30) NOT NULL,
`user_company` VARCHAR(100) NULL DEFAULT NULL,
`user_rank` VARCHAR(45) NULL DEFAULT NULL,
`request_status` TINYINT(1) UNSIGNED NOT NULL DEFAULT '0',
`addtext` TEXT NULL DEFAULT NULL,
PRIMARY KEY (`id`),
INDEX `fk_request_course` (`request_course_id` ASC),
CONSTRAINT `fk_request_course`
  FOREIGN KEY (`request_course_id`)
  REFERENCES `laboratorydb`.`course` (`id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

11. Заявки на проекты

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`request_project` (
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `request_project_id` INT(11) UNSIGNED NOT NULL,
  `request_date` DATETIME NOT NULL,
  `user_fio` VARCHAR(255) NOT NULL,
  `user_email` VARCHAR(64) NOT NULL,
  `user_phone` VARCHAR(30) NOT NULL,
  `request_quantity` INT(11) UNSIGNED NOT NULL DEFAULT '1',
  `request_status` TINYINT(1) UNSIGNED NOT NULL DEFAULT '0',
  `addtext` TEXT NULL DEFAULT NULL,
PRIMARY KEY (`id`),
INDEX `fk_request_project` (`request_project_id` ASC),
CONSTRAINT `fk_request_project`
  FOREIGN KEY (`request_project_id`)
  REFERENCES `laboratorydb`.`project` (`id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

12. Сообщения

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`message` (
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `message_date` DATETIME NOT NULL,
  `message_topic` VARCHAR(255) NOT NULL,
  `user_name` VARCHAR(45) NOT NULL,
  `user_email` VARCHAR(64) NOT NULL,
  `user_phone` VARCHAR(30) NULL DEFAULT NULL,
  `message_text` TEXT NOT NULL,
  `message_status` TINYINT(1) UNSIGNED NOT NULL DEFAULT '0',
PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

13. Новости – Темы

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`post_tag` (
  `pt_id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `post_id` INT(11) UNSIGNED NOT NULL,
  `tag_id` INT(11) UNSIGNED NOT NULL,

```

```

PRIMARY KEY (`pt_id`),
UNIQUE INDEX `fk_pt_UNIQUE` (`post_id` ASC, `tag_id` ASC),
INDEX `fk_pt_tag` (`tag_id` ASC),
CONSTRAINT `fk_pt_post`
  FOREIGN KEY (`post_id`)
  REFERENCES `laboratorydb`.`post` (`id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
CONSTRAINT `fk_pt_tag`
  FOREIGN KEY (`tag_id`)
  REFERENCES `laboratorydb`.`tag` (`id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

14. Публикации – Ключевые слова

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`publication_keyword` (
  `pk_id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `public_id` INT(11) UNSIGNED NOT NULL,
  `keyword_id` INT(11) UNSIGNED NOT NULL,
  PRIMARY KEY (`pk_id`),
  UNIQUE INDEX `fk_pk_UNIQUE` (`public_id` ASC, `keyword_id` ASC),
  INDEX `fk_pk_keyword` (`keyword_id` ASC),
  CONSTRAINT `fk_pk_keyword`
    FOREIGN KEY (`keyword_id`)
    REFERENCES `laboratorydb`.`keyword` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_pk_publication`
    FOREIGN KEY (`public_id`)
    REFERENCES `laboratorydb`.`publication` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

15. Сотрудники – Проекты

```

CREATE TABLE IF NOT EXISTS `laboratorydb`.`user_project` (
  `upro_id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `user_id` INT(11) UNSIGNED NOT NULL,
  `project_id` INT(11) UNSIGNED NOT NULL,
  PRIMARY KEY (`upro_id`),
  UNIQUE INDEX `fk_upro_UNIQUE` (`user_id` ASC, `project_id` ASC),
  INDEX `fk_upro_project` (`project_id` ASC),
  CONSTRAINT `fk_upro_project`
    FOREIGN KEY (`project_id`)
    REFERENCES `laboratorydb`.`project` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_upro_user`
    FOREIGN KEY (`user_id`)
    REFERENCES `laboratorydb`.`user` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8

```

16. Сотрудники – Публикации

```
CREATE TABLE IF NOT EXISTS `laboratorydb`.`user_publication` (  
  `upub_id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,  
  `user_id` INT(11) UNSIGNED NOT NULL,  
  `public_id` INT(11) UNSIGNED NOT NULL,  
  PRIMARY KEY (`upub_id`),  
  UNIQUE INDEX `fk_upub_UNIQUE` (`user_id` ASC, `public_id` ASC),  
  INDEX `fk_upub_publication` (`public_id` ASC),  
  CONSTRAINT `fk_upub_publication`  
    FOREIGN KEY (`public_id`)  
    REFERENCES `laboratorydb`.`publication` (`id`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
  CONSTRAINT `fk_upub_user`  
    FOREIGN KEY (`user_id`)  
    REFERENCES `laboratorydb`.`user` (`id`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8
```

Пример программного кода компонентов модуля администрирования

1. Класс инициализации модуля администрирования (/admin/Module.php)

```
class Module extends \yii\base\Module
{
    public $controllerNamespace = 'app\modules\admin\controllers';

    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'rules' => [
                    [
                        'allow' => true,
                        'roles' => ['admin'],
                    ],
                ],
            ],
        ];
    }

    public function init()
    {
        parent::init();
        $this->layout = 'main';
        $this->defaultRoute = 'post/index';
    }
}
```

2. Пример действий index(), view(), create(), update(), delete() на примере контроллера компонента «Новости» (/admin/controllers/PostController.php)

```
public function actionIndex()
{
    $searchModel = new PostSearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    $dataProvider->pagination->pageSize = 25;

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}

/**
 * Displays a single Post model.
 * @param string $id
 * @return mixed
 */
public function actionView($id)
{
    return $this->render('view', [
```

```

        'model' => $this->findModel($id),
    ]);
}

/**
 * Creates a new Post model.
 * If creation is successful, the browser will be redirected to the 'view'
page.
 * @return mixed
 */
public function createAction()
{
    $model = new Post();
    $uploadModel = new UploadForm();

    if (Yii::$app->request->isPost) {
        $model->load(Yii::$app->request->post());
        $model->post_author_id = Yii::$app->user->id;
        $uploadModel->files = UploadedFile::getInstances($uploadModel,
'files');

        if ($model->validate() && $uploadModel->validate()) {
            $model->save();
            $fileInfo = $uploadModel->upload();
            if (!empty($fileInfo)) {
                foreach ($fileInfo as $fileItem) {
                    $ps = new PostStorage();
                    $ps->post_id = $model->id;
                    $ps->file_name = $fileItem[0];
                    $ps->storage_path = $fileItem[1];
                    $ps->save();
                }
            }
            return $this->redirect(['index']);
        }
    } else {
        return $this->render('create', [
            'model' => $model,
            'uploadModel' => $uploadModel,
        ]);
    }
}

/**
 * Updates an existing Post model.
 * If update is successful, the browser will be redirected to the 'view'
page.
 * @param string $id
 * @return mixed
 */
public function actionUpdate($id)
{
    $model = $this->findModel($id);
    $model->provideTagNames();
    $uploadModel = new UploadForm();

    if (Yii::$app->request->isPost) {

```

```

        $model->load(Yii::$app->request->post());
        $model->filesToBeDeleted = Yii::$app->request->post('filesToBeDeleted');
        $uploadModel->files = UploadedFile::getInstances($uploadModel, 'files');

        if ($model->validate() && $uploadModel->validate()) {
            $model->save();
            $fileInfo = $uploadModel->upload();
            if (!empty($fileInfo)) {
                foreach ($fileInfo as $fileItem) {
                    $ps = new PostStorage();
                    $ps->post_id = $model->id;
                    $ps->file_name = $fileItem[0];
                    $ps->storage_path = $fileItem[1];
                    $ps->save();
                }
            }
            return $this->redirect(['index']);
        } else {
            return $this->render('update', [
                'model' => $model,
                'uploadModel' => $uploadModel,
            ]);
        }
    }

    /**
     * Deletes an existing Post model.
     * If deletion is successful, the browser will be redirected to the 'index'
page.
     * @param string $id
     * @return mixed
     */
    public function actionDelete($id)
    {
        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }
    protected function findModel($id)
    {
        if (($model = Post::findOne($id)) !== null) {
            return $model;
        } else {
            throw new NotFoundHttpException('The requested page does not
exist.');
```

3. Отображение главной страницы компонента на примере компонента «Новости» (/admin/views/post/index.php)

```

<div class="post-index">

    <ul class="nav nav-tabs">
```

```

        <li class="active"><a href="<?= Url::toRoute('post/index')
?>">Новости</a></li>
        <li><a href="<?= Url::toRoute('tag/index') ?>">Теги</a></li>
    </ul>

    <div class="content-panel">
        <p class="top_opts">
            <?= Html::a('Добавить новость', ['create'], ['class' => 'btn btn-
success']) ?>
            <?= Html::a('Удалить выбранные (' . '<span>0</span>' . ')', false,
[
                'class' => 'btn btn-danger multi-delete',
                'data-url' => Url::to(['multi-delete']) ?>
        </p>
        <?= GridView::widget([
            'dataProvider' => $dataProvider,
            'filterModel' => $searchModel,
            'layout' => "{summary}\n{items}\n<div class='text-
right'>{pager}</div>",
            'tableOptions' => ['class' => 'table table-bordered table-striped'],
            'columns' => [
                ['class' => 'yii\grid\SerialColumn'],

                [
                    'attribute' => 'post_title',
                    'format' => 'raw',
                    'value' => function ($model, $key, $index, $column) {
                        return Html::a($model->post_title, ['/post/view', 'id'
=> $model->id]);
                    },
                ],

                [
                    'attribute' => 'post_date',
                    'filter' => DatePicker::widget([
                        'model' => $searchModel,
                        'attribute' => 'date_from',
                        'attribute2' => 'date_to',
                        'type' => DatePicker::TYPE_RANGE,
                        'separator' => '-',
                        'pluginOptions' => ['format' => 'yyyy-mm-dd']
                    ]),
                    'format' => 'datetime',
                ],

                [
                    'attribute' => 'updated_at',
                    'format' => 'datetime',
                ],

                [
                    'class' => SetColumn::className(),
                    'filter' => Post::getStatusesArray(),
                    'attribute' => 'post_type',
                    'name' => 'statusName',
                    'cssClasses' => [

```



```

        Post::STATUS_DRAFT => 'label label-default',
        Post::STATUS_COMMON => 'label label-warning',
        Post::STATUS_TOP => 'label label-success',
    ],
],

[
    'class' => 'yii\grid\ActionColumn',
    'template' => '{update} {delete}',
    'contentOptions' => ['style' => 'white-space: nowrap; text-align: center;'],
],

['class' => 'yii\grid\CheckboxColumn'],
],

'pager' => [
    'maxButtonCount' => 5,
],

]); ?>
</div>
</div>

```

4. КОМПОНЕНТ «Новости» (модель /models/Post.php): метод afterSave()

```

public function afterSave($insert, $changedAttributes)
{
    PostTag::deleteAll(['post_id' => $this->id]);
    $values = [];
    if (!empty($this->tagNames)) {
        foreach ($this->tagNames as $tag) {
            $existQuery = Tag::find()->where(['tag_name' => $tag]);
            if (!$existQuery->exists()) {
                $tagModel = new Tag();
                $tagModel->tag_name = $tag;
                $tagModel->save();
                $values[] = [$this->id, $tagModel->id];
            } else {
                $existModel = $existQuery->one();
                if (!in_array([$this->id, $existModel->id], $values)) {
                    $values[] = [$this->id, $existModel->id];
                }
            }
        }
        Yii::$app->db->createCommand()->batchInsert(PostTag::tableName(),
            ['post_id', 'tag_id'], $values)->execute();
    }

    if (!empty($this->filesToBeDeleted)) {
        foreach ($this->filesToBeDeleted as $fileToBeDeleted) {
            $query = PostStorage::findOne(['id' => (int) $fileToBeDeleted,
                'post_id' => $this->id]);
            $filePath = Yii::getAlias("@webroot")
                . ArrayHelper::getValue($query, 'storage_path');
            if (file_exists($filePath)) {
                unlink($filePath);
            }
        }
    }
}

```

```

        }
        $query->delete();
    }
    unset($this->filesToBeDeleted);
}

parent::afterSave($insert, $changedAttributes); // TODO: Change the
autogenerated stub
}

```

5. Компонент «Галерея» (модель /models/Gallery.php): метод обработки изображений

```

protected function processUploadedImage(UploadedFile $imageInstance) {
    $basePath = Yii::getAlias('@webroot') . '/images/gallery/';
    do {
        $imageName = $this->generateImageName();
    } while (file_exists($basePath . $imageName));
    $imageFullPath = $basePath . $imageName;
    $imageThumbPath = $basePath . 'thumbs/' . $imageName;
    $image = new \Imagick($imageInstance->tempName);
    if ($image->getImageWidth() > Yii::$app->params['imageMaxWidth'] ||
    $image->getImageHeight() > Yii::$app->params['imageMaxHeight']) {
        $image->thumbnailImage(800, 600, true);
    }
    $image->writeImage($imageFullPath);
    $image->cropThumbnailImage(200, 150);
    $image->writeImage($imageThumbPath);
    $this->image_name = $imageName;
}

public function generateImageName() {
    $imageName = uniqid(time()) . '.jpg';
    return $imageName;
}

public function getImagePath() {
    return Yii::getAlias('@webroot') . '/images/gallery/' . $this-
>image_name;
}

public function getThumbPath() {
    return Yii::getAlias('@webroot') . '/images/gallery/thumbs/' . $this-
>image_name;
}

public function getImageLink() {
    return '/images/gallery/' . $this->image_name;
}

public function getThumbLink()
{
    return '/images/gallery/thumbs/' . $this->image_name;
}
}

```

6. Компонент «Публикации» (модель /models/Publication.php): методы beforeSave() и afterSave()

```

public function beforeSave($insert)

```

```

    {
        if (parent::beforeSave($insert)) {
            if ($insert) {
                if ($this->attach) {
                    $attachPath = '/documents/publications/' . uniqid(time()) .
                '.' . $this->attach->extension;
                    $this->attach->saveAs(Yii::getAlias("@webroot")
                $attachPath);
                    $this->public_file = $attachPath;
                }
            } else {
                if ($this->attach) {
                    $checkFile = Yii::getAlias("@webroot") . $this->public_file;
                    if (file_exists($checkFile)) {
                        unlink($checkFile);
                    }
                    $attachPath = '/documents/publications/' . uniqid(time()) .
                '.' . $this->attach->extension;
                    $this->attach->saveAs(Yii::getAlias("@webroot")
                $attachPath);
                    $this->public_file = $attachPath;
                }
            }
            $this->public_date = (new \DateTime($this->year . '-' . $this->month
            . '-' . '01'))->format('Y-m-d');
            return true;
        }
        return false;
    }

    public function afterSave($insert, $changedAttributes)
    {
        UserPublication::deleteAll(['public_id' => $this->id]);
        PublicationKeyword::deleteAll(['public_id' => $this->id]);
        if (!empty($this->authorIds)) {
            $authorIdsValues = [];
            foreach ($this->authorIds as $authorId) {
                $authorIdsValues[] = [(int) $authorId, $this->id];
            }
            Yii::$app->db->createCommand()-
        >batchInsert(UserPublication::tableName(), ['user_id', 'public_id'],
        $authorIdsValues)->execute();
        }
        if (!empty($this->keywordNames)) {
            $keywordValues = [];
            foreach ($this->keywordNames as $keyword) {
                $existQuery = Keyword::find()->where(['keyword_name' =>
        $keyword]);
                if (!$existQuery->exists()) {
                    $keywordModel = new Keyword();
                    $keywordModel->keyword_name = $keyword;
                    $keywordModel->save();
                    $keywordValues[] = [$this->id, $keywordModel->id];
                } else {
                    $existModel = $existQuery->one();

```

```

        if (!in_array([$this->id, $existModel->id],
$keywordValues)) {
            $keywordValues[] = [$this->id, $existModel->id];
        }
    }
    Yii::$app->db->createCommand()-
>batchInsert(PublicationKeyword::tableName(), ['public_id', 'keyword_id'],
$keywordValues)->execute();
    }
    parent::afterSave($insert, $changedAttributes); // TODO: Change the
autogenerated stub
}

```

7. Модель формы смены пароля (/admin/models/PasswordChangeForm.php)

```

class PasswordChangeForm extends Model
{
    public $currentPassword;
    public $newPassword;
    public $newPasswordRepeat;

    /**
     * @var User
     */
    private $_user;

    /**
     * @param User $user
     * @param array $config
     */
    public function __construct(User $user, $config = [])
    {
        $this->_user = $user;
        parent::__construct($config);
    }

    public function rules()
    {
        return [
            [['currentPassword', 'newPassword', 'newPasswordRepeat'],
'required'],
            [['currentPassword', 'validatePassword'],
['newPassword', 'string', 'min' => 6],
['newPasswordRepeat', 'compare', 'compareAttribute' =>
'newPassword'],
        ];
    }

    public function attributeLabels()
    {
        return [
            'newPassword' => 'Новый пароль',
            'newPasswordRepeat' => 'Повторите пароль',
            'currentPassword' => 'Текущий пароль',
        ];
    }
}

```

```

/**
 * @param string $attribute
 * @param array $params
 */
public function validatePassword($attribute, $params)
{
    if (!$this->hasErrors()) {
        if (!$this->_user->validatePassword($this->$attribute)) {
            $this->addError($attribute, 'Неверный текущий пароль');
        }
    }
}

/**
 * @return boolean
 */
public function changePassword()
{
    if ($this->validate()) {
        $user = $this->_user;
        $user->setPassword($this->newPassword);
        return $user->save();
    } else {
        return false;
    }
}
}

```