МИНОБРНАУКИ РОССИИ
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»
im. В.И. Ульянова (Ленина)

Факультет Компьютерных технологий и информатики
Направление Информатика и вычислительная техника
Кафедра Математического обеспечения и применения ЭВМ

К защите допустить:
Заведующий кафедрой: к.т.н. Кринкин Кирилл Владимирович

ВЫПУСКНАЯ
КВАЛИФИКАЦИОННАЯ РАБОТА
БАКАЛАВРА

Тема: Система управления контентом для научного коллектива

Студент

Руководитель асп. каф. САПР (уч. степень, уч. звание)

Консультанты к.э.н., доцент (уч. степень, уч. звание)
k.п.н., доцент (уч. степень, уч. звание)

Подпись

Фамилия И.О.

Обманов И. И.

Руководитель асп. каф. САПР (уч. степень, уч. звание)

Красильников А. В.

Ичкитидзе Ю. Р.

Яценко И. В.

Подпись

Подпись

Подпись

Фамилия И.О.

Фамилия И.О.

Фамилия И.О.

Санкт-Петербург
2016
ЗАДАНИЕ
на выпускную квалификационную работу бакалавра

Студент

Обманов Илья Игоревич

Группа № 0304

1. Тема работы
Система управления контентом для научного коллектива.

2. Исходные данные (технические требования)
Документ с требованиями к функциональности веб-ресурса, требования к аппаратным ресурсам веб-сервера, требования к программным ресурсам

3. Содержание работы
Анализ требований к веб-ресурсу, выбор средств разработки, проектирование ядра и модулей веб-ресурса, прототипирование пользовательского интерфейса, разработка программного ядра веб-ресурса, разработка модулей веб-ресурса, разработка пользовательского интерфейса веб-ресурса, развертывание решения на веб-сервере

4. Перечень отчетных материалов
Пояснительная записка, исходные коды веб-ресурса на языке Java.

5. Дополнительные разделы
Технико-экономическое обоснование ВКР

Дата выдачи задания «__» 2016 г.
Дата представления работы к защите «__» 2016 г.

Руководитель асп. каф. САПР
(уч. степень, уч. звание) __________________
Фамилия И.О. ____________________________

Студент ________________________________
Фамилия И.О. ____________________________

«» 2016 г.
Календарный план выполнения выпускной квалификационной работы

Студент Обманов И.И. Группа № 0304

Тема ВКР Система управления контентом для научного коллектива

<table>
<thead>
<tr>
<th>№ п/п</th>
<th>Наименование работ</th>
<th>Срок выполнения</th>
<th>Отметка о выполнении</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Обзор литературы, посвященной разработке веб-ресурсов</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Анализ требований и формирование критериев выбора технологий</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Изучение современных средств разработки веб-ресурсов, выбор стека технологий</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Проектирование ядра и модулей веб-ресурса, прототипирование интерфейса</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Разработка ядра, модулей и интерфейса веб-ресурса</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Технико-экономическое обоснование</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Оформление пояснительной записки</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Предзащита ВКР</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Руководитель асп. каф. САПР (уч. степень, уч. звание) подписа

Красильников А.В. Фамилия И.О.

Студент

Обманов И.И. Фамилия И.О.

«» 2016 г
СОДЕРЖАНИЕ

Список терминов и сокращений........................................................................................................6
ВВЕДЕНИЕ...........................................................................................................................................7
1 ПРОГРАММНЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ВЕБ-РЕСУРСОВ. ВЫБОР СТЕКА ТЕХНОЛОГИЙ.................................................................8
  1.1 Формирование критериев выбора технологий .................................................................8
  1.2 Современные платформы для веб-разработки ...........................................................9
    1.2.1 Фреймворк Django ......................................................................................................9
    1.2.2 Фреймворк Ruby on Rails .........................................................................................10
    1.2.3 Платформа на основе языка РНР .............................................................................11
    1.2.4 Платформа ASP.NET 5 ...........................................................................................11
    1.2.5 Стек java технологий .............................................................................................12
  1.3 Обоснование использования стека java технологий для разработки веб-ресурса .....13
  1.4 Технологии хранения данных .........................................................................................14
    1.4.1 Классические реляционные СУБД .........................................................................14
    1.4.2 Документо-ориентированные СУБД ....................................................................15
    1.4.3 Технология объектно-реляционного отображения Hibernate ................................17
  1.5 Используемые на стороне клиента технологии .............................................................18
  1.6 Выводы ....................................................................................................................................19
2 ПРОЕКТИРОВАНИЕ ВЕБ-РЕСУРСА .......................................................................................20
  2.1 Проектирование программного ядра системы ..............................................................21
  2.2 Проектирование модулей ресурса ...................................................................................22
    2.2.1 Структура модуля Авторы .......................................................................................23
    2.2.2 Структура модуля Публикации ..............................................................................24
    2.2.3 Структура модуля Новости .....................................................................................24
    2.2.4 Структура модуля События ....................................................................................25
  2.3 Прототипирование пользовательского интерфейса .......................................................26
  2.4 Выводы ....................................................................................................................................29
3 РАЗРАБОТКА ВЕБ-РЕСУРСА .................................................................................................30
  3.1 Разработка модулей ресурса ...........................................................................................31
    3.1.1 Разработка модуля Авторы .....................................................................................32
    3.1.2 Разработка модуля Публикации ............................................................................37
    3.1.3 Разработка модуля Новости ..................................................................................42
    3.1.4 Разработка модуля События ................................................................................45
  3.2 Выводы ....................................................................................................................................47
4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ВКР ...........................................................48
  4.1 Детализированный план-график выполнения работ ....................................................48
  4.2 Оценка величины заработной платы и социальных отчислений .................................49
4.3 Материальные затраты ...........................................................................................................50
4.4 Полная себестоимость проекта ..........................................................................................53
4.5 Выводы ...............................................................................................................................53
ЗАКЛЮЧЕНИЕ ..................................................................................................................................54
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ ..............................................................................55
ПРИЛОЖЕНИЕ А – ИСХОДНЫЕ КОДЫ ..................................................................................57
Список терминов и сокращений

<table>
<thead>
<tr>
<th>Условное обозначение</th>
<th>Полное значение</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJAX</td>
<td>Asynchronous JavaScript and XML (Асинхронный JavaScript и XML)</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface (Интерфейс программирования приложений)</td>
</tr>
<tr>
<td>CMS</td>
<td>Content Management System (Система управления содержимым)</td>
</tr>
<tr>
<td>DTO</td>
<td>Data Transfer Object (Объект передачи данных)</td>
</tr>
<tr>
<td>GUID</td>
<td>Globally Unique Identifier (Гlobально уникальный идентификатор)</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment (Интегрированная среда разработки)</td>
</tr>
<tr>
<td>MIME</td>
<td>Multipurpose Internet Mail Extensions (Многоцелевые расширения интернет-почты)</td>
</tr>
<tr>
<td>MVC</td>
<td>Model-View-Controller (Модель-Вид-Контроллер)</td>
</tr>
<tr>
<td>MVT</td>
<td>Model-View-Template (Модель-Вид-Шаблон)</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language (Язык структурированных запросов)</td>
</tr>
<tr>
<td>VCS</td>
<td>Version Control System (Система управления версиями)</td>
</tr>
<tr>
<td>БД</td>
<td>База данных</td>
</tr>
<tr>
<td>ПО</td>
<td>Программное обеспечение</td>
</tr>
<tr>
<td>СУБД</td>
<td>Система управления базами данных</td>
</tr>
</tbody>
</table>
ВВЕДЕНИЕ

Деятельность современного научного коллектива, независимо от области исследований, требует работы с большим количеством различной информации, обмена ею, ведения записей об исследованиях, коммуникации участников коллектива и т.д. Выполнение всех этих задач критически важно для нормального функционирования коллектива и его продуктивной работы. Не являясь основными задачами, они, тем не менее, требуют времени на свое выполнение. Соответственно, минимизация затрат времени на подобные рутинные задачи является важным этапом оптимизации деятельности коллектива.

Все эти задачи могут решать веб-ресурсы, работа с которыми возможна удаленно как с ПК, так и с любого современного мобильного устройства. Существует множество специализированных средств обмена информацией, ее хранения, управления коллективом. Однако использование большого количества разрозненных средств затрудняет работу. Актуальной задачей является создание платформы, объединяющей все необходимые возможности и интегрирующей их друг с другом.

Целью данной работы является повышение эффективности деятельности малых научных коллективов за счет создания веб-ресурса для организации взаимодействия между членами коллектива и хранения общих данных, а также получение компетенций ПК-1, ПК-6, ПК-9. Для достижения поставленной цели необходимо решить следующие задачи:

1) Выбор программной платформы для реализации веб-ресурса;
2) Проектирование архитектуры веб-ресурса и создание прототипа его интерфейса;
3) Разработка веб-ресурса на основе спроектированной архитектуры.
При разработке любого программного продукта важно выбрать наиболее подходящую платформу и вспомогательные инструменты. Критерии их выбора базируются на требованиях к продукту во всех аспектах его работы. В данном разделе производится анализ требований к разрабатываемому веб-ресурсу, формируются критерии выбора технологий и определяется используемый при разработке стек технологий.

1.1 Формирование критериев выбора технологий

Основными критериями выбора серверного стека технологий являются:

1) Скорость разработки. В современном мире одним из решающих факторов успешности продукта является возможность быстрого выхода на рынок или внедрения.

2) Расширяемость. Гибкость платформы и простота добавления новых возможностей важна, поскольку напрямую влияет на скорость разработки.

3) Поддержка платформы разработчиками. Данный критерий критически важен, поскольку если обнаруживаемые ошибки и уязвимости не будут исправляться, безопасность работы пользователей будет под угрозой.

4) Простота размещения ресурса в сети Интернет. Данный фактор влияет как на скорость внедрения продукта, так и на скорость доставки обновлений и исправлений ошибок и уязвимостей.

При выборе стека технологий рассматриваются наиболее известные и актуальные в данное время веб-фреймворки. Готовые системы управления контентом (CMS) исключены из рассмотрения, так как ни одна из них не решает всех поставленных задач, расширяемость ограничена, а некоторые возможности в принципе не могут быть реализованы.
1.2 Современные платформы для веб-разработки

Рассмотрим основные существующие на данный момент веб-фреймворки и выявим их достоинства и недостатки.

1.2.1 Фреймворк Django


Веб-приложение на Django строится из одного или нескольких приложений, которые отделены друг от друга и могут независимо подключаться. Это одно из существенных архитектурных отличий этого фреймворка от других. Также, в отличие от других фреймворков, обработчики запросов в Django конфигурируются явно, а не выводятся автоматически из структуры моделей контроллеров.


Рисунок 1.1 – Архитектура приложения на Django
Основные преимущества Django:
– встроенный интерфейс администратора, с уже имеющимся переводом на многие языки;
– расширяемая система шаблонов с поддержкой наследования;
– авторизация и аутентификация, подключение внешних модулей аутентификации;
– встроенная автоматическая документация по тегам шаблонов и моделям данных, доступная через административное приложение.

Основные недостатки Django:
– сложность замены ряда компонентов на свои при появлении такой необходимости;
– сложность расширения ядра.

1.2.2 Фреймворк Ruby on Rails

Ruby on Rails – фреймворк, написанный на языке программирования Ruby. Является открытым программным обеспечением и распространяется под лицензией MIT. Ruby on Rails предоставляет средства для создания веб-приложений по принципу MVC, а также обеспечивает интеграцию с веб-сервером и сервером базы данных. Архитектура приложения на Ruby on Rails представлена на рисунке 1.2.

Рисунок 1.2 – Архитектура приложения на Ruby on Rails
Достоинства:
– низкий процент дублирования кода;
– оптимальная конфигурация поумолчанию;
– быстрота разработки;
– большое количество плагинов;
– качественная документация.

Недостатки:
– «борьба с платформой» в случае необходимости отступить от рекомендуемых принципов разработки на данном фреймворке;
– высокий порог вхождения.

1.2.3 Платформа на основе языка PHP
На основе языка PHP создано большое количество платформ, предоставляющих различные модели создания веб-ресурсов. Однако все эти платформы исключаются из рассмотрения по причине того, что сам язык PHP подвергается серьезной критике из-за отсутствия поддержки многих современных концепций, несогласованности синтаксиса функций и проблем безопасности [4].

1.2.4 Платформа ASP.NET 5
ASP.NET 5 [5]– платформа для создания веб-ресурсов и веб-сервисов от компании Microsoft. Данная технология позволяет разработчикам писать код, используя практически любые языки программирования, входящие в стек технологий .NET Framework (C#, F#, Visual Basic.NET, JScript .NET, IronPython и т.д.).

Достоинства:
– компилируемый статически типизированный язык на серверной стороне;
– простота тестирования;
– возможность совмещения модели MVC и сервисов WebAPI в одном веб-приложении;
большое количество подключаемых и легко заменяемых средств для решения различных задач.

Недостатки:
– технология находится в стадии раннего доступа.

1.2.5 Стек java-технологий

Servlet это простой Java интерфейс, который взаимодействует с клиентами посредством принципа запрос-ответ. Java код исполняется на серверной java машине.

Веб-ресурсы на платформе JavaServlet строятся по принципу MVC (рисунок 1.3). Кроме того, на базе данной платформы доступна технология создания веб-сервисов WebAPI.

Рисунок 1.3 – Архитектура приложения на JavaServlet

JSP (JavaServerPages) – технология, позволяющая веб-разработчикам создавать содержимое, которое имеет как статические, так и динамические компоненты. Страница JSP содержит текст двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов HTML, SVG, WML, или XML, и JSP-элементы, которые конструируют динамическое содержимое. Кроме этого могут использоваться библиотеки JSP-тегов, а также EL (ExpressionLanguage), для внедрения Java-кода в статичное содержимое JSP-страниц.
Также при разработке на JSPудобно использовать JSTL (JavaServer Pages Standard Tag Library), для возможности разбора XML, создания циклов, условной обработки.

При формировании страницы JSP подаются входные данные из сервлета, чтобы отобразить интерфейс клиенту. С помощью JSP и библиотеки тегов JSTL появляется возможность эти данные обрабатывать с большей гибкостью, чем при использовании разметки простых страниц html.

Достоинства:
– Эффективность. Не надо порождать нового процесса.
– Единство среды разработки программного обеспечения.
– Надежность и безопасность (реализованы на JAVA).
– Платформенно-независимы. Легкая интеграция на другие сервера, главное, чтобы на них присутствовала Java машина.
– Множество инструментов мониторинга и отладки.
– Бесплатность.

Недостатки:
– Слабое разделение уровня представления и бизнес-логики.
– Возможны конфликты при параллельной обработке запросов.

1.3 Обоснование использования стека java технологий для разработки веб-ресурса

На основании ранее сформулированных критериев выберем платформу для разработки. Сформулированным критерием наиболее удовлетворяет стек java технологий. Присутствует поддержка всех основных современных технологий. Платформа является модульной и легко расширяемой. Важен тот факт, что язык Java статически типизирован, что облегчает обнаружение различных ошибок, повышая тем самым скорость разработки. Кроме того, начало разработки на Java Servlet требует относительно небольшое количество навыков и низкий порог вхождения. Размещение ресурса в сети Интернет возможно на базе практически любой
современной операционной системы и платформы. На сервере требуется минимальные условия, это чтобы на нем была JavaRuntimeEnvironment (среда для выполнения java), а также была возможность запустить сервер с javамашиной.

1.4 Технологии хранения данных

Известны два основных подхода, которые используются в настоящее время для организации баз данных и работы с ними. Это классические реляционные СУБД, которые уже долгое время де-факто являются стандартом и на сегодняшний день используются в подавляющем большинстве проектов, связанных с хранением данных, и NoSQL СУБД, появившиеся сравнительно недавно и стремительно развивающиеся в настоящий момент.

1.4.1 Классические реляционные СУБД

Реляционная модель данных впервые была предложена британским учёным, сотрудником компании IBM Эдгаром Франком Коддом в 1970 году в статье «A Relational Model of Data for Large Shared Data Banks» [6].

К. Дейт в своей книге[7] определил три составные части реляционной модели данных:

– структурная;
– манипуляционная;
– целостная.

Структурная часть модели определяет, что единственной структурой данных является нормализованное n-арное отношение. Отношения удобно представлять в форме «таблиц», где каждая строка есть кортеж, а каждый столбец – атрибут, определенный на некотором домене. Домен можно рассматривать как множество допустимых значений данного атрибута. Несколько атрибутов одного отношения и атрибуты разных отношений могут быть определены на одном и том же домене. Данный подход к понятию отношения дает привычную для разработчиков и пользователей
форму представления, в которой реляционная база данных представляет собой конечный набор таблиц.

Манипуляционная часть модели определяет два фундаментальных механизма манипулирования данными – реляционная алгебра и реляционное исчисление. Основной функцией манипуляционной части реляционной модели является обеспечение меры реляционности любого конкретного языка реляционных БД.

Целостная часть модели определяет требования целостности сущностей и целостности ссылок. Первое требование состоит в том, что любой кортеж любого отношения отличим от любого другого кортежа этого отношения, т.е. другими словами, любое отношение должно обладать первичным ключом. Требование целостности по ссылкам, или требование внешнего ключа состоит в том, что для каждого значения внешнего ключа, появляющегося в ссылающемся отношении, в отношении, на которое ведет ссылка, должен найтись кортеж с таким же значением первичного ключа, либо значение внешнего ключа должно быть неопределенным (т.е. ни на что не указывать).

Достоинства реляционной модели:
– простота и доступность для понимания разработчиком. Единственной используемой информационной конструкцией является «таблица»;
– строгие правила проектирования, базирующиеся на математическом аппарате реляционной алгебры и реляционного исчисления;
– для организации запросов и написания прикладного ПО нет необходимости знать конкретную организацию БД во внешней памяти.

1.4.2 Документо-ориентированные СУБД
Однако стоит заметить, что рассмотренная реляционная модель имеет ряд недостатков:
– далеко не всегда предметная область может быть представлена в виде «таблиц»;
в результате логического проектирования появляется множество «таблиц». Это приводит к трудности понимания структуры данных;

– относительно низкая скорость доступа к данным.

В таких случаях выгодно использовать модели, не обладающие описанными недостатками, например, NoSQL. NoSQL («not only SQL», «не только SQL») – концепция, которая включает в себя ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL.

Документо-ориентированные базы данных и СУБД для работы с ними не подразумевают внутренних связей. Они не основываются на одной модели, а каждая база данных в зависимости от целей использует различные модели. Нормализация, необходимая в реляционной модели для устранения логической избыточности, неприменима для NoSQL подхода, в котором, напротив, проводится максимальная денормализация.

Однако и эта модель имеет существенный недостатки, документо-ориентированные СУБД не имеют контроля ссылочной информации, а также отсутствует транзакционная логика.

Также в отличие от реляционных БД, хранилища, ориентированные на использование в «облаке», имеют гораздо меньше общих стандартов. Хоть концептуально они и не отличаются, они все имеют разные API, интерфейсы запросов и свою специфику.

Другое преимущество реляционных БД заключается в том, что они вынуждают пройти через процесс разработки модели данных. Если вы хорошо спроектировали модель, то база данных будет содержать логическую структуру, которая полностью отражает структуру хранимых данных, однако расходится со структурой приложения. Таким образом, данные становятся независимы от приложения. Это значит, что другое приложение сможет использовать те же самые данные и логика
приложения может быть изменена без каких-либо изменений в модели базы.

Делая выводы из вышесказанного было принято решение воспользоваться классической реляционной СУБД. Это облегчает задачу реализации веб-приложения, так как не нужно будет контролировать программным кодом ссылочную целостность и реализовывать транзакционную логику.

Была выбрана MySQL как СУБД. Она позволяет нам реализовывать все потребности и в отличие от более сложных СУБД (такие как Microsoft SQL Server), обеспечивает высокую скорость обмена данными.

1.4.3 Технология объектно-реляционного отображения Hibernate

Hibernate – библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения (object-relational mapping – ORM). Она представляет собой свободное программное обеспечение с открытым исходным кодом (open source), распространяемое на условиях GNU Lesser General Public License. Данная библиотека предоставляет легкий в использовании каркас (фреймворк) для отображения объектно-ориентированной модели данных в традиционные реляционные базы данных.

Hibernate позволяет еще больше увеличить скорость разработки. Эта библиотека даст инструменты, которые освободят нас от программирования для обеспечения хранения данных. С помощью Hibernate мы избавимся от необходимости разработки большого числа задач, так как в этой библиотеке проблемы с помещением, обновлением и получением данных уже решены.

Так же Hibernate хорошо вписывается в концепцию ранее выбранного стека java-технологий.

Для разработки с помощью Hibernate есть все инструменты:
– много документации по этой технологии;
– легкая интеграция в систему разработки;
– удобная система аннотаций, которая упрощает mapping.

1.5 Используемые на стороне клиента технологии

Рассмотрим технологии, которые будут обеспечивать взаимодействие с пользователем на стороне браузера.

1) Bootstrap [15]. При создании интерфейса веб-ресурса используется интерфейсный фреймворк Bootstrap. Он позволяет создавать адаптивный интерфейс, который автоматически масштабируется под нужное разрешение, что удобно при разработке ресурса, используемого как с компьютеров, так и с мобильных устройств. Имеется множество элементов управления и стилей для оформления страниц. Также имеется большое количество дополнений, которые при использовании вместе с Bootstrap используют предоставляемые им стили, образуя единообразный внешний вид.

2) jQuery [16]. Для упрощения работы с контентом страниц на стороне клиентов используется библиотека jQuery, позволяющая манипулировать элементами страницы. Также она предоставляет средства выполнения асинхронных запросов (AJAX), в которых браузер фоново обменивается данными с сервером, позволяя обновлять не всю страницу, а загружать только требуемые данные, сокращая тем самым объемы передающихся по сети данных.

3) FullCalendar. Плагин для работы с календарем и событиями. Построен на jQuery, имеет много средств взаимодействия с данными этого календаря. В качестве источника события на вход будет подаваться Gson с выбранными событиями из базы данных.

4) bootstrap-wysiwyg. Средство для работы с форматированием больших текстов. Предоставляет широкий функционал редактирования текста, так же позволяет вставлять изображения. Данные сохраняются в html разметке в базе данных.
1.6 Выводы

В данном разделе были рассмотрены существующие технологии создания веб-ресурсов, хранения данных, и средства, работающие на стороне пользователя. Были проанализированы их возможности, сходства и различия. В результате проведенного анализа был сформирован и обоснован стек технологий для реализации проекта.

В качестве веб-фреймворка выбран стек Java-технологий, так как это решение удовлетворяет всем потребностям к разрабатываемой системе. При разработке навыбранном варианте мы получаем относительно большую скорость разработки, также получаем возможность запускать систему на разных платформах, и получаем возможность расширения функционала.

Для хранения данных была выбрана классическая реляционная СУБД MySQL. Так же инструментом объектно-реализационного отображения был выбран Hibernate, так как он хорошо вписывается в структуру ранее выбранных технологий, а также дает значительный прирост в скорости разработки.

Выбраны средства, используемые на стороне клиента, которые помогают без дополнительных затрат ресурсов воспользоваться уже разработанной системой взаимодействия с пользовательским интерфейсом.
2 ПРОЕКТИРОВАНИЕ ВЕБ-РЕСУРСА

Мартин Фаулер в работе[11] определил концепцию слоев как одну из общеупотребимых моделей, используемых разработчиками программного обеспечения для разделения сложных систем на более простые части. Автор описывает архитектуру, состоящую из трех основных слоев:

— представление охватывает все относящееся к работе пользователя с системой;
— домен описывает основные функции приложения, его предметную область и бизнес-логику;
— источник данных отвечает за работу с базой данных, обмен сообщениями с другими приложениями и т.д.

В основе организации слоев лежит следующий принцип: слои нижнего уровня «не знают» про слои верхнего уровня и не требуют их наличия для своей работы. В свою очередь, слои верхнего уровня взаимодействуют только с ближайшим нижестоящим слоем.

При проектировании разрабатываемого веб-ресурса применим следующую модель слоев:

— инфраструктурный слой отвечает за такие служебные возможности, как интерфейсы протоколирования и внедрения зависимостей, а также содержит методы расширения [18] классов окружения для использования во всех частях проекта;
— слой «ядро» содержит модель предметной области, ключевые классы результатов выполнения операций, валидации, вычисления прав пользователя и т.д.;
— слой доступа к данным включает в себя DAO-классы, валидаторы, а также менеджеры, являющиеся публичным API всей бизнес-логики веб-ресурса;
—слой представления содержит веб-сервисы для выполнения команд пользователя, обработчики запросов страниц веб-ресурса, шаблоны страниц, скрипты и т.п.

В описанной модели слои «ядро» и «сервисы» являются подслоями слоя «домен» в модели, описанной Фаулером, а слой «источник данных» отсутствует, поскольку его обязанности перекладываются на клиент выбранной СУБД.

Общая концепция создаваемого веб-ресурса такова: все запросы данных и команды выполняются через методы hibernate, а он, в свою очередь, использует методы ODBC драйвера для запросов из базы данных, а действия JSP отвечают только за возврат пользователю шаблонов страниц. Принцип взаимодействия клиента и сервера показан нарисунке 2.1.

Рисунок 2.1–Взаимодействие клиента и сервера

2.1 Проектирование программного ядра системы

Программное ядро системы содержит средства, которые лежат в основе реализации и используются всеми модулями системы.

Список объектов, созданных для реализации системы:
1) Статья. Объект создан для хранения и доступа к информации о работах научного коллектива.
2) Параметры статьи. Неосновные параметры, объект создан для возможности динамического добавления параметров разного типа, не меняя структуру данных объекта «статья».

3) Авторы. Объект создан для хранения информации о сотрудниках научного коллектива, а также для системы аутентификации.

4) События. Объект, в котором хранится информация о мероприятиях, которые необходимо включить в календарь.

5) Новости. Объект, который содержит в себе длинный текст, с возможностью вставки изображения в него.

**2.2 Проектирование модулей ресурса**

Разрабатываемый веб-ресурс состоит из нескольких модулей, отвечающих за разные части функциональности. Структура модулей имеет вид дерева, показанного на рисунке 2.2.

![Рисунок 2.2 – Дерево модулей веб-ресурса](image)

Ключевым модулем является модуль Авторы, отвечающий за работу с пользователями, группами пользователей и разрешениями. Все остальные модули зависят от него.

В ходе проектирования была создана модель базы данных, которая представлена на рисунке 2.3.
2.2.1 Структура модуля Авторы

Рассмотрим, какая информация об авторе будет храниться. Существует набор ключевых полей, хранящихся в самой учетной записи пользователя. Это те поля, к которым доступ требуется наиболее часто или логика их использования предполагает их хранение именно в учетной записи. Часть этих полей предназначена для использования в процессах регистрации, аутентификации и авторизации:

– уникальный идентификатор пользователя;
– имя аккаунта, в том числе нормализованное;
– электронный почтовый ящик, к которому привязан аккаунт, в том числе нормализованный, а также информация о том, подтвержден ли он;
– хеш-сумма пароля для обеспечения аутентификации;
– штамп безопасности для обеспечения авторизации и ее отмены.

Под нормализацией понимается приведение к верхнему регистру. Она служит для облегчения контроля повторного использования одних и тех же имен аккаунтов или e-mail.
Другая часть полей содержит основную публичную информацию об аккаунте и его владельце и указывается им самим:

– персональная информация о личности владельца: ФИО, дата рождения;
– аватар (хранится путь к файлу в хранилище).

Существуют и другие служебные поля, однако они добавляются и используются рассматриваемыми далее подсистемами модуля и будут описаны при описании соответствующих подсистем.

2.2.2 Структура модуля Публикации

Данный модуль отвечает за управление публикациями. Его задачи:

– создание, изменение, удаление публикаций;
– поиск по публикациям;
– выборка публикаций по автору.

Публикация хранит следующую информацию:

– заголовок и описание;
– дата публикации;
– идентификатор добавившего публикацию;
– список ссылок на прикрепленные файлы;
– список идентификаторов аккаунтов авторов.

Структура модуля приведена на рисунке 2.4.

![Рисунок 2.4 – Структура модуля Публикации](image)

2.2.3 Структура модуля Новости

Данный модуль отвечает за систему новостей. Требования к модулю:

– возможность создания, редактирования, удаления записей в блоге;
– запись состоит из названия и текста записи;
— к записи можно прикрепить файл(ы);
— запись может иметь изображение в контенте текста.

С учетом требований, структурамодуля принимает вид, показанный нарисунке 2.5.

<table>
<thead>
<tr>
<th>BlogEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>+PublisherId</td>
</tr>
<tr>
<td>+Title</td>
</tr>
<tr>
<td>+Content</td>
</tr>
<tr>
<td>+PicturePath</td>
</tr>
<tr>
<td>+Published: DateTime</td>
</tr>
<tr>
<td>+FilePaths: HashSet&lt;string&gt;</td>
</tr>
</tbody>
</table>

Рисунок 2.5 — Структура модуля Новости

### 2.2.4 Структура модуля События

Данный модуль реализует систему календарей. Единственной сущностью, добавляемой этим модулем, является «событие». Событие имеет:

— дату и время начала и конца;
— название и описание;
— флаг «весь день» (время не учитывается, дни занимаются полностью);
— флаг «повторять» и правило повторения (может включать частоту, интервал повторения, день недели, количество повторений и т.д.)

С учетом требований, структура модуля принимает вид, показанный нарисунке 2.6.

<table>
<thead>
<tr>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>+StartTime : DateTime</td>
</tr>
<tr>
<td>+EndTime : DateTime</td>
</tr>
<tr>
<td>+Title</td>
</tr>
<tr>
<td>+Description</td>
</tr>
<tr>
<td>+AllDay : bool</td>
</tr>
<tr>
<td>+Repeat : bool</td>
</tr>
<tr>
<td>+RepeatRule : string</td>
</tr>
</tbody>
</table>

Рисунок 2.6 — Структура модуля События
2.3 Прототипирование пользовательского интерфейса

При прототипировании пользовательского интерфейса воспользуемся подходом «wireframing». Wireframe — это низкодетализированное представление дизайна [20], которое показывает:
– основные группы содержимого;
– информационную структуру;
– описание взаимодействия пользователя с интерфейсом и его примерную визуализацию.

При использовании этого подхода нужно создать целое представление конечного дизайна и не пропустить ни одного важного элемента, при этом, не углубляясь в детали.

Создадим wireframe для макета страниц веб-ресурса. Рисунок 2.7 показывает то, как страницы выглядят в браузере, и какой макет страниц используется.

Рисунок 2.7 – Макет страниц веб-ресурса

Содержимое страниц веб-ресурса располагается в рамках представленного макета. Рассмотрим некоторые основные страницы
веб-ресурса. На рисунке 2.8 представлен макет публичного профиля пользователя.

На рисунке 2.9 представлен макет страницы со списком публикаций.
На рисунке 2.10 представлен макет страницы списка записей в блоге.

Рисунок 2.10 – Макет страницы списка записей в блоге

На рисунке 2.11 представлен макет страницы календаря.

Рисунок 2.11 – Макет страницы календаря
2.4 Выводы

В данном разделе была описана архитектура веб-ресурса, показана модель взаимодействия с пользователем. Описаны функции и структура ядра системы. С учетом требований к функциональности веб-ресурса выделены модули создаваемой системы, для каждого из модулей определены принципы их работы, а также представлены диаграммы, показывающие структуру данных модулей. Представлены прототипы некоторых страниц пользовательского интерфейса, созданные с помощью подхода wireframing.
3 РАЗРАБОТКА ВЕБ-РЕСУРСА

При разработке данного программного продукта было принято решение использовать JavaServlet, JSP и Hibernate в качестве стека технологий.

Для упрощения работы с формированием JSP, а также, чтобы не порождать одни и те же сущности, было принято решение создавать на одну сущность один класс сервлета, который выполняется по-разному, в зависимости от текущего действия. Например, на странице списка публикаций мы находимся в сервлете articals, без каких-либо условий, и с представлением страницы списка публикаций. Если мы начинаем редактировать какую-либо статью, то посылается GET запрос к этому же сервлету articals, но с параметром action = edit и с параметром id = ‘id редактируемого элемента’, при этом вызывается представление страницы публикации. При сохранении этого элемента посылается post запрос, и если параметр post id не пустой, то запись в БД обновляется, далее переходим обратно на страницу списка публикаций.

Если же мы хотим добавить элемент, то со страницы списка публикаций также посылается GET запрос только уже с параметром action = add, и без параметра id. Соответственно если мы сохраняем эту запись, то посылаем тот же post запрос к сервлету articals, что и при редактировании, но теперь параметр id у нас пустой, поэтому мы создаем новую запись в таблице articals, далее переходим обратно на страницу списка публикаций.

Если же мы хотим удалить элемент, то посылается просто GET запрос с параметром action = delete, и возвращается представление списка публикации.

Так же у сервлета news есть еще одно действие это view, оно похоже на действие edit, только без возможности редактирования. Описанное взаимодействие представлено на рисунке 3.1.
3.1 Разработка модулей ресурса

Каждый объект представлен в виде 2 классов и одного интерфейса. Рассмотрим, каждую часть объекта:

— Непосредственно сам класс. В нем описано с какой таблицей связан объект, атрибуты класса и соответствия их с колонками таблицы, связь этого объекта с другими, так же стандартные методы get и set атрибутов класса, так же объявлено вычисление Hash объекта и сравнение объектов.

— Интерфейс класса. В нем объявлены общие методы, над всеми объектами этого типа. Объявлены методы добавление записи в таблицу, обновление записи, удаление записи, получение всех записей, получение записи по id.
–Реализация интерфейса. Данный класс реализует объявленные методы в интерфейсе, при переходе на другую систему хранение данных, нужно будет просто добавить еще одну реализацию объявленных методов в интерфейсе класса.

Реализация объекта представлена на рисунке 3.2.

Рисунок 3.2 – Схема реализации объекта

3.1.1 Разработка модуля Авторы

Разработка модуля Авторы является одной из основных частей разработки этого веб-приложения, потому что почти все сущности ссылаются на сущность Авторы.

Структура таблицы авторы представлена на рисунке 3.3.

Рисунок 3.3 – Структура таблицы Авторы

Также таблица авторы связана с другими, связь представлена на рисунке 3.4.
Для этих связей были использованы аннотации hibernate, которые позволили быстро описать связь объектов. Описание таких связей представлено на рисунке 3.5.

```java
@ManyToMany(fetch = FetchType.EAGER, mappedBy = "authors")
private Set<Articals> articles = new HashSet<>();
@ManyToMany(fetch = FetchType.EAGER, mappedBy = "authors")
private Set<News> news = new HashSet<>();
```

Логин и хэш пароля хранится в той же таблицы что и авторы, для быстрого доступа к ним.

При сохранении приложенного изображения, в данном случае аватара, происходит переименования файла, чтобы избежать конфликтов имен. Новое имя это сгенерированный GUID.

Так же создана система действий, которая описана выше. На рисунке 3.6 представлена эта система для объекта автор.
Действия объявлены в классе сервлета как атрибуты, и получаются из GET-запроса, после чего выполняется то или иной действие, в зависимости от переданного параметра action. Реализация схемы работы сервлета представлена на рисунке 3.7.
На Рисунке 3.7 представлена реализация схемы работы сервлета.

Рисунок 3.8 представлен внешний вид профиля пользователя на широком экране. А на рисунке 3.9 внешний вид профиля пользователя на мобильном устройстве.

Рисунок 3.8 – Профиль пользователя на широком экране
Рисунок 3.9 – Профиль пользователя на мобильном устройстве

На рисунке 3.10 представлен внешний вид списка авторов на широком экране.

Рисунок 3.10 – Список авторов на широком экране

На рисунке 3.11 представлен список авторов на мобильном устройстве.
Рисунок 3.11 – Список авторов на мобильном устройстве

Страницы веб-ресурса адаптируются к размеру экрана, вследствие этого веб-ресурс удобно использовать как на больших мониторах ПК, так и на мобильных устройствах.

3.1.2 Разработка модуля Публикации

Реализация данного модуля мало отличается от рассмотренного ранее модуля и не имеет каких-либо особенностей. Одна публикация представляется одной записью в БД. К публикации может быть прикреплен файл.

Схема сервлета artical точно такая же, как и для authors, только с тем отличием, что при изменении и при создании новой публикации передается весь список авторов и список авторов, которые уже указаны в статье, для формировании списка выбора авторов. Реализация такого решения представлена на рисунке 3.12.
Рисунок 3.12 – Реализация схемы сервлета articals

Также в модуле Публикации была добавлена возможность формирования разных полей с типом список значений.

Структура таблицы articals, а также ее связь с другими таблицами, представлена на рисунке 3.13.
Рисунок 3.13 – Структура таблицы articles и связь её с другими таблицами

На рисунке 3.14 показана страница добавления публикации на широком экране, на рисунке 3.15 страница добавления публикаций на мобильном устройстве, а на рисунке 3.16 представлена страница со списком публикаций на широком экране, на рисунке 3.17 страница со списком публикаций на мобильном устройстве.
Рисунок 3.14 – Страница добавления публикации на широком экране

Рисунок 3.15 – Страница добавления публикации на мобильном устройстве
Добавление уже зарегистрированных авторов происходит по имени аккаунта. При этом, если начать вводить имя, динамически подгружается список пользователей с подходящими именами, тем самым добавление авторов облегчается.
3.1.3 Разработка модуля Новости

Схема сервлета представлена на рисунке 3.18.

Рисунок 3.18 – Схема сервлета news

Эта схема соответствует предыдущим, с одним исключением появился еще один action – view. Он похож на действие edit, только без возможности редактирования, и представление данных выглядит более развернуто и наглядно.

Для того чтобы реализовать редактирование длинных текстов была использована компонента bootstrap-wysiwyg, которая позволяет интерактивно создавать длинные тексты, помещать в их контекст изображения, и пользоваться широким спектром редактирование текста. После сохранения внесённого текста весь html этого элемента переносится в отдельное поле, невидимое для клиента, чтобы можно было легко на стороне события POST в сервлете newsэто htmlтекст разобрать. Этот процесс обрабатывается кодом на javascript, который представлен на рисунке 3.19.
newso.jsp:

```java
newso.jsp:

…
<div id="editor" contenteditable="true">…

<textarea style="display: none;" id="editorCopy" name="body"><c:out value="${news.text}" /></textarea>

…
<script>
$(document).ready(function() {
  $('#editor').html($('#editorCopy').val());
  $('#postSubmit').click(function () {
    $('#editorCopy').val($('#editor').html());
  });
});
</script>

Рисунок 3.19 – Процесс передачи html кода

Далее весь этот html код сохраняется в таблице news, чтобы в дальнейшем при просмотре этого события, выводился весь текст в нужном форматировании и изображение в контексте.

Схема таблицы news в базе данных, а также ее связь, представлена на рисунке 3.20.

Рисунок 3.20 – Схема таблицы news, а также ее связь с таблицей author

На рисунке 3.21 представлена страница создания записи, на рисунке 3.22 – список созданных записей, на рисунке 3.23–страница просмотра записи.

43
Рисунок 3.21 – Создание записи

Рисунок 3.22 – Список всех записей

Рисунок 3.23 – Страница просмотра записи
3.1.4 Разработка модуля События

При разработке данного модуля был использован сторонний компонент [24], реализующий представление календаря и позволяющий управлять всеми аспектами его работы. Для его использования достаточно предоставить один метод, который при запросе событий за некоторый промежуток времени возвращает их список.

Для подачи данных в календарь был создан отдельный сервлет, реализация, которого представлена на рисунке 3.24.

```
@Servlet("/events/CalendarServlet")
public class ForGetEvents extends HttpServlet {
  private EventDao eventDao;
  public ForGetEvents() {
    eventDao = new EventDaoExampleImpl();
  }

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
  throws ServletException, IOException {
    List<Event> AllEvent = eventDao.getAll();
    System.out.println("EVENTSALL");
    resp.setContentType("application/json");
    resp.setCharacterEncoding("UTF-8");
    PrintWriter out = resp.getWriter();
    out.write(new Gson().toJson(AllEvent));
  }
}
```

Рисунок 3.24 – Сервлет для подачи данных в календарь

В интерфейсе данный модуль представлен всего одной веб-страницей, на которой сосредоточены все средства по управлению календарем. На рисунке 3.25 показан вид календаря.
Рисунок 3.25 – Календарь пользователя

На Рисунок 3.16 представлен вид окна добавления события, который вызывается нажатием на свободное от событий поле дня, также если стоит пометка весь день, то время начала и конца игнорируются, события продолжается весь день.
3.2 Выводы

В данном разделе показаны детали реализации веб-ресурса. Рассмотрены аспекты функционирования модулей. Показан внешний вид компонентов и страниц веб-ресурса. Внешний вид страниц веб-ресурса спроектирован на основе ранее показанных шаблонов, созданных с использованием подхода wireframing.

Рассмотренные модули обладают всеми базовыми возможностями, необходимыми для работы с ними.
4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ВКР

Одним из важнейших факторов, влияющих на появление того или иного продукта в мире, является экономическая целесообразность этого продукта. В рамках технико-экономического обоснования данной работы рассматриваются следующие вопросы:

– составление детализированного план-графика выполнения работ;
– оценка величины основной и дополнительной заработной платы, а также социальных отчислений участников разработки;
– материальные затраты на проект;
– полная себестоимость проекта.

4.1 Детализированный план-график выполнения работ

Детализированный план-график выполнения работ (Таблица 4.1) позволяет рассмотреть процесс выполнения работы на временной шкале.

<table>
<thead>
<tr>
<th>№ п/п</th>
<th>Наименование работ</th>
<th>Исполнитель</th>
<th>Срок выполнения, дней</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Составление требований к функциональности веб-ресурса</td>
<td>Руководитель</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Обзор литературы, посвященной разработке веб-ресурсов</td>
<td>Инженер</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Анализ требований и формирование критериев выбора технологий</td>
<td>Инженер</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Изучение современных средств разработки веб-ресурсов, выбор стека технологий</td>
<td>Инженер</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Проектирование ядра и модулей веб-ресурса, прототипирование интерфейса</td>
<td>Инженер</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>Разработка ядра, модулей и интерфейса веб-ресурса</td>
<td>Инженер</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>Автоматизация тестирования и развертывания на веб-сервере</td>
<td>Инженер</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>Технико-экономическое обоснование</td>
<td>Инженер</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>Оформление пояснительной записки</td>
<td>Инженер</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>Приемка результатов работы</td>
<td>Руководитель</td>
<td>2</td>
</tr>
</tbody>
</table>

По данным
Таблица 4.1 видим, что время, затраченное инженером и руководителем, составляет 53 и 5 дней соответственно.

4.2 Оценка величины заработной платы и социальных отчислений

Разрабатывая веб-ресурс, необходимо учитывать все затраты, направленные на его разработку. К таким затратам относятся затраты на основную и дополнительную оплату человеческого труда, а также налоговые отчисления.

Для того, чтобы рассчитать затраты на основную оплату человеческого труда, необходимо установить заработную плату всех исполнителей. Для инженера установим заработную плату размером 15000 рублей, для руководителя – 25000 рублей.

Дневная ставка рассчитывается по формуле (4.1):

$$3РД = \frac{3П_{месц}}{N},$$

где $N$ – количество рабочих дней в месяце;

ЗП_{месц} – заработная плата в месяц.

С учетом того, что в месяце 21 рабочий день, дневная ставка инженера равна 714 руб./день, руководителя – 1190 руб./день.

Расходы на основную заработную плату исполнителей определяются по формуле (4.2) и показаны в Таблица 4.2:

$$Z_{осн.пл} = \sum_{i=1}^{k} T_i \cdot C_i,$$

где $Z_{осн.пл}$ – расходы на основную заработную плату исполнителей (руб.);

$k$ – количество исполнителей;

$T_i$ – время, затраченное $i$-м исполнителем на проведение исследования (дни);

$C_i$ – ставка $i$-го исполнителя (руб./день).

<table>
<thead>
<tr>
<th>Исполнитель</th>
<th>Оплата, руб./день</th>
<th>Количество</th>
<th>Основная оплата за</th>
</tr>
</thead>
</table>

Таблица 4.2 – Затраты на основную оплату труда
Расходы на дополнительную заработную плату рассчитываются по формуле (4.3):

\[ Z_{доп.з/пл} = Z_{осн.з/пл} \cdot \frac{H_{доп}}{100}, \]  

где \( Z_{осн.з/пл} \) – расходы на основную заработную плату исполнителей (руб.);
\( H_{доп} \) – норматив дополнительной заработной платы.

При \( H_{доп} \), равном 14%, дополнительная заработная плата составляет 6131 рубль.

Отчисления на страховые взносы на обязательное социальное, пенсионное и медицинское страхование с основной и дополнительной заработной платы исполнителей определяются по формуле (4.4):

\[ Z_{соц} = (Z_{осн.з/пл} + Z_{доп.з/пл}) \cdot \frac{H_{соц}}{100}, \]  

где \( Z_{соц} \) – отчисления на социальные нужды с заработной платы(руб.);
\( Z_{осн.з/пл} \) – расходы на основную заработную плату исполнителей(руб.);
\( Z_{доп.з/пл} \) – расходы на дополнительную заработную плату исполнителей (руб.);
\( H_{соц} \) – норматив отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование (%).

Поскольку \( H_{соц} \) составляет 30% , отчисления на страховые взносы на обязательное социальное, пенсионное и медицинское страхование составляют 14977 рублей.

Таким образом, полная оплата человеческого труда, с учетом отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование, составляет 64900 рублей.
4.3 Материальные затраты

Для разработки веб-ресурса необходимо организовать рабочие места инженера и руководителя, а также платформу для размещения системы непрерывной интеграции и веб-сервера. Необходимые компоненты представлены в Таблица 4.3.

Таблица 4.3 – Необходимые компоненты

<table>
<thead>
<tr>
<th>№ п/п</th>
<th>Наименование</th>
<th>Количество</th>
<th>Цена, руб./шт</th>
<th>Итого, руб.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Персональный компьютер</td>
<td>2</td>
<td>24000</td>
<td>48000</td>
</tr>
<tr>
<td>2</td>
<td>Сервер</td>
<td>1</td>
<td>18000</td>
<td>18000</td>
</tr>
<tr>
<td></td>
<td><strong>Итого:</strong></td>
<td></td>
<td></td>
<td><strong>66000</strong></td>
</tr>
</tbody>
</table>

В услуги сторонних организаций входит распечатка пояснительной записи. Поскольку максимальный объем пояснительной записи составляет 80 страниц, тариф составляет 3 руб./страница, а ставка НДС равна 18%, затраты на оплату услуг сторонних организаций, рассчитанные по формуле (4.5), составляют 196,8 рублей.

\[ Z_{оусо} = A_{стр} \cdot T_{стр} - A_{стр} \cdot T_{стр} \cdot НДС, \]  
где \( Z_{оусо} \) – затраты на оплату услуг сторонних организаций (руб.);

\( A_{стр} \) – количество страниц для распечатки (шт.);

\( T_{стр} \) – тариф на оплату услуг (руб./стр.);

НДС – ставка НДС (%).

Затраты на содержание и эксплуатацию оборудования представляют из себя затраты на электроэнергию. Общие затраты на содержание и эксплуатацию оборудования рассчитываются по формуле (4.6):

\[ Z_{эо} = \sum_{i=1}^{m} C_i^{мч} \cdot t_i^{м}, \]  
где \( Z_{эо} \) – затраты на содержание и эксплуатацию оборудования (руб.);

\( C_i^{мч} \) – расчетная себестоимость одного машино-часа работы оборудования на \( i \)-й технологической операции (руб./м-ч);
$t_i^{м}$ – количество машино-часов, затрачиваемых на выполнение $i$-й технологической операции (м-ч).

Введем формулу (4.7) для расчета затрат на электроэнергию:

$$Z_{эои} = W h_i \cdot t_i \cdot T,$$

где $Z_{эои}$ – затраты на содержание и эксплуатацию средства (руб.);

$W h_i$ – мощность средства (кВт·ч);

$t_i$ – время, проведенное за использованием средства (ч);

$T$ – тариф на электроэнергию.

Тариф на электроэнергию по состоянию на март 2015 года составляет 3.53 руб/кВт·ч. Результаты расчета затрат на содержание и эксплуатацию оборудования сведены в Таблица 4.4.

Таблица 4.4 – Расходы на содержание и эксплуатацию оборудования

<table>
<thead>
<tr>
<th>№ п/п</th>
<th>Наименование</th>
<th>Мощность, кВт·ч</th>
<th>Время использования средства, ч</th>
<th>Затраты, руб.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Персональный компьютер</td>
<td>0,6</td>
<td>464</td>
<td>983</td>
</tr>
<tr>
<td>2</td>
<td>Сервер</td>
<td>0,45</td>
<td>264</td>
<td>419</td>
</tr>
<tr>
<td></td>
<td><strong>Итого:</strong></td>
<td></td>
<td></td>
<td><strong>1402</strong></td>
</tr>
</tbody>
</table>

Амортизационные отчисления для каждого из средств, определяемые по формуле (4.8), показаны в Таблица 4.5.

$$A_i = \frac{\Pi_{п.н.т} \cdot H_{аf}}{100},$$

где $A_i$ – амортизационные отчисления за год по $i$-му основному средству (руб.);

$\Pi_{п.н.т}$ – первоначальная стоимость $i$-го основного средства (руб.);

$H_{аf}$ – годовая норма амортизации $i$-го основного средства (%).

Таблица 4.5 – Амортизационные отчисления по основным средствам

<table>
<thead>
<tr>
<th>№ п/п</th>
<th>Наименование</th>
<th>Срок использования, лет</th>
<th>Годовая норма амортизации, %</th>
<th>Цена, руб.</th>
<th>Амортизационные отчисления, руб.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Персональный компьютер</td>
<td>5</td>
<td>20</td>
<td>24000</td>
<td>4800</td>
</tr>
<tr>
<td>2</td>
<td>Сервер</td>
<td>5</td>
<td>20</td>
<td>18000</td>
<td>3600</td>
</tr>
<tr>
<td></td>
<td><strong>Итого:</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>8400</strong></td>
</tr>
</tbody>
</table>
Величина амортизационных отчислений по каждому основному средству, используемому при работе над данным проектом, определяется по формуле (4.9): 

\[ A_{i_{ВКР}} = A_i \cdot \frac{T_{i_{ВКР}}}{365}, \]  
(4.9)

где \( A_{i_{ВКР}} \) — амортизационные отчисления по \( i \)-му основному средству, используемому при работе над данным проектом (руб.);

\( A_i \) — амортизационные отчисления по \( i \)-му основному средству за год (руб.);

\( T_{i_{ВКР}} \) — время, в течение которого было использовано средство (дней).

Таблица 4.6 показывает амортизационные отчисления за время работы над проектом.

<table>
<thead>
<tr>
<th>№ п/п</th>
<th>Наименование</th>
<th>Амортизационные отчисления, руб./день</th>
<th>Время использования, дней</th>
<th>Амортизационные отчисления, руб.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Персональный компьютер</td>
<td>13,15</td>
<td>58</td>
<td>763</td>
</tr>
<tr>
<td>2</td>
<td>Сервер</td>
<td>9,86</td>
<td>33</td>
<td>325</td>
</tr>
</tbody>
</table>

Итого: 1088

4.4 Полная себестоимость проекта

Таблица 4.7 показывает совокупную величину затрат на создание проекта.

<table>
<thead>
<tr>
<th>№ п/п</th>
<th>Наименование затрат</th>
<th>Затраты, руб.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Расходы на оплату труда</td>
<td>49923</td>
</tr>
<tr>
<td>2</td>
<td>Отчисления на социальные нужды</td>
<td>14977</td>
</tr>
<tr>
<td>4</td>
<td>Затраты на работы, выполняемые сторонними организациями</td>
<td>196.8</td>
</tr>
<tr>
<td>5</td>
<td>Расходы на содержание и эксплуатацию оборудования</td>
<td>1402</td>
</tr>
<tr>
<td>6</td>
<td>Амортизационные отчисления</td>
<td>1088</td>
</tr>
<tr>
<td>7</td>
<td>Накладные расходы</td>
<td>19969</td>
</tr>
</tbody>
</table>

Итого: 87555,8

4.5 Выводы

В данном разделе были рассчитаны экономические затраты на создание веб-ресурса. Была подсчитана величина заработной
платы инженера, реализующего проект, затраты на нее составляют 49923 рублей, что является самой большой затратой из всех. От этой суммы зависят затраты на социальные нужды и на накладные расходы. Общие затраты за более чем два месяца работы составляют 87555,8 рублей.
ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы получены следующие результаты:

– Изучен ряд современных технологий для разработки веб-ресурсов. В процессе работы была выбрана технология JavaServlet, как наиболее актуальная и удовлетворяющая всем поставленным требованиям.

– На основе спроектированной архитектуры реализован веб-ресурс с использованием выбранной технологии JavaServlet. Реализовано универсальное ядро системы и набор модулей, таких как: Авторы, Публикации, События, Новости. За счет гибкой архитектуры ядра системы любой новый модуль может быть легко разработан и интегрирован в веб-ресурс.

Разработанная платформа позволила обеспечить единое информационное пространство в работе научного коллектива УНЦ «КТИ» и может быть адаптирована к работе других научных коллективов.

В перспективе платформа будет поддерживаться и расширяться за счет внедрения новых возможностей, таких как:

– Модуль управления проектами;
– Модуль интеграции с системами индексирования научных работ: РИНЦ, Scopus, WebofScience;
– Модуль организации удаленных лабораторных работ и проведения компьютерного тестирования;
– Модуль оповещения участников коллектива о важных событиях через SMS;
– Приложения для популярных мобильных платформ: iOS, Android.
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


3) Уоллс К. Spring в действии. Москва: ДМК Пресс, 2013.


7) Дейт К.Д. Введение в системы баз данных. Москва: Вильямс, 2005.


13) Вайрфреймы, прототипы и мокапы [Электронный ресурс] // Проекторат: [сайт]. [2015]. URL: http://projectorat.ru/wireframes-


ПРИЛОЖЕНИЕ А—ИСХОДНЫЕ КОДЫ

Модель «Автор»

Author.java:

```java
@Entity
@Table(name = "author")
public class Author {
    @Id
    @Column(name = "ID")
    @GeneratedValue
    private int id;

    @Column(name = "FirstName")
    private String firstName;

    @Column(name = "Patronymic")
    private String midName;

    @Column(name = "SurName")
    private String lastName;

    @Column(name = "eMail")
    private String eMail;

    @Column(name = "image")
    private String image;

    @Column(name = "login")
    private String login;

    @Column(name = "pass")
    private String password;

    @ManyToMany(fetch = FetchType.EAGER, mappedBy = "authors")
    private Set<Articals> articles = new HashSet<>();
    @ManyToMany(fetch = FetchType.EAGER, mappedBy = "authors")
    private Set<News> news = new HashSet<>();
    @ManyToMany(fetch = FetchType.EAGER, mappedBy = "authors")
    private Set<DocApp> docApps = new HashSet<>();

    public Author() {
    }
    public Author(int id, String firstName, String midName, String lastName, String eMail, String image) {
        this.id = id;
        this.firstName = firstName;
    }
```
this.firstName = firstName;
this.midName = midName;
this.lastName = lastName;
this.eMail = eMail;
this.image = image;
}

public Author(String firstName, String midName, String lastName, String eMail, String image) {
    this.firstName = firstName;
    this.midName = midName;
    this.lastName = lastName;
    this.eMail = eMail;
    this.image = image;
}

public Set<Articals> getArticles() {
    return this.articles;
}

public void setArticles(Set<Articals> articles) {
    this.articles = articles;
}

public Set<DocApp> getDocApp() {
    return this.docApps;
}

public void setDocApp(Set<DocApp> DocApps) {
    this.docApps = DocApps;
}

public Set<News> getNews() {
    return this.news;
}

public void setNews(Set<News> News) {
    this.news = News;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getLogin() {
    return login;
}
public void setLogin(String Login) {
    this.login = Login;
}

public String getPassword() {
    return password;
}

public void setPassword(String Password) {
    this.password = Password;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getMidName() {
    return midName;
}

public void setMidName(String midName) {
    this.midName = midName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String geteMail() {
    return eMail;
}

public void seteMail(String eMail) {
    this.eMail = eMail;
}

public String getImage() {
    return image;
}

public void setImage(String image) {
    this.image = image;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Author author = (Author) o;
    if (getId() != author.getId()) return false;
    return true;
}
if (!getFirstName().equals(author.getFirstName())) return false;
if (getMidName() != null ? !getMidName().equals(author.getMidName()) : author.getMidName() != null) return false;
if (!getLastName().equals(author.getLastName())) return false;
if (geteMail() != null ? !geteMail().equals(author.geteMail()) : author.geteMail() != null) return false;
return getImage() != null ? getImage().equals(author.getImage()) : author.getImage() == null;
}

@Override
public int hashCode() {
    int result = getId();
    result = 31 * result + getFirstName().hashCode();
    result = 31 * result + (getMidName() != null ? getMidName().hashCode() : 0);
    result = 31 * result + getLastName().hashCode();
    result = 31 * result + (geteMail() != null ? geteMail().hashCode() : 0);
    result = 31 * result + (getImage() != null ? getImage().hashCode() : 0);
    return result;
}

@Override
public String toString() {
    return "Author{" +
            "id=" + id +
            ", firstName=" + firstName + \\
            "", midName=" + midName + \\
            ", lastName=" + lastName + \\
            ", eMail=" + eMail + \\
            ", IMAGE=" + image + \\
            "}";
}

AuthorDao.java:

public interface AuthorDao {

    void add(Author author);

    void delete(Author author);

    void update(Author author);

    List<Author> getAll();

    Author getById(int id);

    List<Author> getByLogin(String login);
}

AuthorDaoExampleImpl.java:

public class AuthorDaoExampleImpl implements AuthorDao{
    private static final Random r = new Random();
}
private static final Map<Integer, Author> AUTHOR_MAP = new HashMap<Integer, Author>();
public static final String IMAGE = "img/placeholders/avatar.jpg";

public AuthorDaoExampleImpl() {
}
	public void add(Author author) {
		Session session = HibernateUtil.getSessionFactory().getCurrentSession();
	//author.setId(r.nextInt(Integer.MAX_VALUE));
	author.setImage(IMAGE);
	//AUTHOR_MAP.put(author.getId(), author);
	session.beginTransaction();
	session.save(author);
	session.getTransaction().commit();
}
	public void delete(Author author) {
		Session session = HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		session.delete(author);
		session.flush();
		session.getTransaction().commit();
}
	public void update(Author author) {
		Session session = HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		//author = (Author) session.merge(author);
		session.update(author);
		session.getTransaction().commit();
}
	public List<Author> getAll() {
		Session session = HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		Query q = session.createQuery("from Author");
		List<Author> AllAuthors = q.list();
		session.getTransaction().commit();
		return AllAuthors;
}
	public Author getById(int id) {
		Session session = HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		Query q = session.createQuery("from Author A where A.id= :author_id");
		q.setParameter("author_id", id);
		List<Author> Aut = q.list();
		session.getTransaction().commit();
		return Aut.get(0);
}
public List<Author> getByLogin(String login) {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    Query q = session.createQuery("from Author A where A.login= :login");
    q.setParameter("login", login);
    List<Author> Aut = q.list();
    session.getTransaction().commit();
    return Aut;
}

AuthorServlet.java:

@WebServlet("/authors")
@MultipartConfig(fileSizeThreshold=1024*1024,
    maxFileSize=1024*1024*5, maxRequestSize=1024*1024*5*5)
public class AuthorServlet extends HttpServlet {
    private static final String EDIT_ACTION = "edit";
    private static final String DELETE_ACTION = "delete";
    private static final String ADD_ACTION = "add";
    private static final String ACTION = "action";
    private static final String AUTHORS = "/frame.jsp";
    private static final String AUTHOR = "/author.jsp";
    private static final String SAVE_DIR = "img";
    private AuthorDao authorDao;

    public AuthorServlet() {
        authorDao = new AuthorDaoExampleImpl();
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String action = req.getParameter(ACTION);
        HttpSession session = req.getSession();
        req.setAttribute("authors", authorDao.getAll());
        if (session.getAttribute("userInfo") != null) {
            Author AutIn = authorDao.getById((Integer) session.getAttribute("userInfo"));
            req.setAttribute("AutIN", true);
            req.setAttribute("userin", AutIn);
        } else { req.setAttribute("AutIN", false); } 
        if (action != null) {
            if (action.equals(ADD_ACTION)) {
                RequestDispatcher view = req.getRequestDispatcher(AUTHOR);
                view.forward(req, resp);
                return;
            }
            String id = req.getParameter("id");
            Author author = authorDao.getById(Integer.valueOf(id));
        }
    }
}
switch (action) {
    case EDIT_ACTION:
        req.setAttribute("author", author);
        RequestDispatcher view = req.getRequestDispatcher(AUTHOR);
        view.forward(req, resp);
        return;
    case DELETE_ACTION:
        authorDao.delete(author);
        break;
}
RequestDispatcher view = req.getRequestDispatcher(AUTHORS);
view.forward(req, resp);

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse resp) throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    resp.setCharacterEncoding("UTF-8");
    String stringId = request.getParameter("id");
    Author author = new Author();
    String appPath = request.getServletContext().getRealPath("/");
    String savePath = appPath + SAVE_DIR;
    String fileName = "";
    author.setFirstName(request.getParameter("firstName"));
    author.setLastName(request.getParameter("lastName"));
    author.setMidName(request.getParameter("midName"));
    author.seteMail(request.getParameter("eMail"));
    if (stringId == null || stringId.isEmpty()) {
        for (Part part : request.getParts()) {
            fileName = extractFileName(part);
            //System.out.print(savePath + File.separator + fileName);
            part.write(savePath + File.separator + fileName);
            break;
        }
        author.setImage(SAVE_DIR + File.separator + fileName);
        authorDao.add(author);
    } else {
        Author oldauthor = authorDao.getById(Integer.valueOf(stringId));
        Set<Articals> OldArt = oldauthor.getArticles();
        author.setId(Integer.valueOf(stringId));
        author.setArticles(OldArt);
        author.setLogin(oldauthor.getLogin());
        author.setPassword(oldauthor.getPassword());
        for (Part part : request.getParts()) {
            authorDao.add(author);
        }
    }
}
fileName = extractFileName(part);
UUID uuid = UUID.randomUUID();
String extension = "";

int i = fileName.lastIndexOf('鹭');
if (i > 0) {
    extension = fileName.substring(i+1);
}
fileName = ""+uuid+".鹭"+extension;
System.out.print(oldauhor.getImage());
System.out.print(fileName);

if (oldauhor.getImage()==null || !oldauhor.getImage().equals(SAVE_DIR+File.separator+fileName) && !fileName.equals(""))
    {part.write(savePath + File.separator + fileName);
        author.setImage(SAVE_DIR+File.separator+fileName);
    }else
    {author.setImage(oldauhor.getImage());}
break;

authorDao.update(author);
}
doGet(request, resp);

private String extractFileName(Part part) {
    String contentDisp = part.getHeader("content-disposition");
    String[] items = contentDisp.split(";");
    for (String s : items) {
        if (s.trim().startsWith("filename")) {
            return s.substring(s.indexOf("=") + 2, s.length()-1);
        }
    }
    return "";
}

Модель «Публикация»

Articals.java:
@Entity
@Table(name = "articles")
public class Articals {
    @Id
    @Column(name = "ID")
    @GeneratedValue
    private int id;

    @Column(name = "FileName")
    private String filename;
@Column(name = "TextArticals")
private String textArticals;

@Column(name = "Description")
private String description;

@Column(name = "DateCreate")
private String dateCreate;

@ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
@JoinTable(name = "articles_author",
    joinColumns = {@JoinColumn(name = "IDArt", referencedColumnName = "id")},
    inverseJoinColumns = {@JoinColumn(name = "IDAuthor", referencedColumnName = "id")})
private Set<Author> authors = new HashSet<>(0);

@ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
@JoinTable(name = "articles_articalprop",
    joinColumns = {@JoinColumn(name = "IDArt", referencedColumnName = "id")},
    inverseJoinColumns = {@JoinColumn(name = "IDArtProp", referencedColumnName = "id")})
private Set<ArticalsProp> articalsProps = new HashSet<>(0);

public Articals() {
}

public Articals(int ID, String TextArticals, String Description, String DateCreate) {
    this.id = ID;
    this.textArticals = TextArticals;
    this.description = Description;
    this.dateCreate = DateCreate;
}

public int getId() {
    return id;
}

public void setId(int ID) {
    this.id = ID;
}

public String getFileName() {
    return filename;
}

public void setFileName(String FileName) {
    this.filename = FileName;
}
public String getTextArticals() {
    return textArticals;
}

public void setTextArticals(String TextArt) {
    this.textArticals = TextArt;
}

public String getDescription() {
    return description;
}

public void setDescription(String Description) {
    this.description = Description;
}

public String getDateCreate() {
    return dateCreate;
}

public void setDateCreate(String DateCreate) {
    this.dateCreate = DateCreate;
}

public Set<ArticalsProp> getArticalsProps() {
    return articalsProps;
}

public void setArticalsProps(Set<ArticalsProp> articalsProp) {
    this.articalsProps = articalsProp;
}

public Set<Author> getAuthors() {
    return authors;
}

public void setAuthors(Set<Author> art) {
    this.authors = art;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Articals articals = (Articals) o;
    return getId() == articals.getId();
}

Articals articals = (Articals) o;
return getId() == articals.getId();
@Override
public String toString() {
    return "Articals{
        id=" + id + ", TextArticals=" + textArticals + '\'' +
        ", Description=" + description + '\'' +
        ", DateCreate=" + dateCreate + '\'' +
    '}';
}

ArticalsDao.java:

public interface ArticalsDao {
    void add(Articals articals);
    void delete(Articals articals);
    void update(Articals articals);
    List<Articals> getAll();
    Articals getById(int id);
}

ArticalsDaoExampleImpl.java:

public class ArticalsDaoExampleImpl implements ArticalsDao {
    public ArticalsDaoExampleImpl() {

    }

    public void add(Articals articals) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        session.save(articals);
        session.getTransaction().commit();
    }

    public void delete(Articals articals) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        session.delete(articals);
    }
}
session.flush();
session.getTransaction().commit();
}

public void update(Articals articals) {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    session.update(articals);
    session.getTransaction().commit();
}

public List<Articals> getAll() {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    Query q = session.createQuery("from Articals");
    List<Articals> AllArticals = q.list();
    session.getTransaction().commit();
    return AllArticals;
}

public Articals getById(int id) {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    Query q = session.createQuery("from Articals A where A.id= :articals_id");
    q.setParameter("articals_id", id);
    List<Articals> Art = q.list();
    session.getTransaction().commit();
    return Art.get(0);
}

AricalsServlet.java:

@WebServlet("/articals")
public class ArticalsServlet extends HttpServlet {
    private ArticalsDao articalsDao;
    private AuthorDao authorDao;
    private ArticalsPropDao articalsPropDao;
    private static final String EDIT_ACTION = "edit";
    private static final String DELETE_ACTION = "delete";
}
private static final String ADD_ACTION = "add";
private static final String ACTION = "action";
private static final String AUTHORS = "/articals.jsp";
private static final String ARTICAL = "/artical.jsp";

public ArticalsServlet() {
    articalsDao = new ArticalsDaoExampleImpl();
    authorDao = new AuthorDaoExampleImpl();
    articalsPropDao = new ArticalsPropDaoExampleImpl();
}

@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String action = req.getParameter(ACTION);
    HttpSession session = req.getSession();
    req.setAttribute("authors", authorDao.getAll());
    if (session.getAttribute("userInfo") != null) {
        Author AutIn = authorDao.getById((Integer) session.getAttribute("userInfo"));
        req.setAttribute("AutIN",true);
        req.setAttribute("userin", AutIn);
    }else{req.setAttribute("AutIN",false);}
    if (action != null) {
        if (action.equals(ADD_ACTION)) {
            List<Author> authorList =authorDao.getAll();
            List<ArticalsProp> articalsProps = articalsPropDao.getAllPopDesc();
            List<ArticalsProp> articalsPropList = articalsPropDao.getAllValue("Вид публикации");
            req.setAttribute("AllDescsValue",articalsPropList);
            req.setAttribute("AllDescs",articalsProps);
            req.setAttribute("authorsall", authorList);
            RequestDispatcher view = req.getRequestDispatcher(ARTICAL);
            view.forward(req, resp);
            return;
        }
        String id = req.getParameter("id");
        Articals artical = articalsDao.getById(Integer.valueOf(id));
        switch (action) {
            case EDIT_ACTION:
                List<Author> authorList =authorDao.getAll();
                List<Author> inauthor = new ArrayList<>(artical.getAuthors());
                authorList.removeAll(inauthor);
                List<ArticalsProp> articalsProps = articalsPropDao.getAllPopDesc();
                System.out.print("GetALLPopDesc");
                List<ArticalsProp> propList = articalsPropDao.getAllValue(articalsProps);
                System.out.print("getAllValue");
                List<ArticalsProp> inprop = new ArrayList<>(artical.getArticalsProps());
                propList.removeAll(inprop);
                req.setAttribute("artical", artical);
                req.setAttribute("authorsall", authorList);
                req.setAttribute("authorsin", inauthor);
                break;
            case DELETE_ACTION:
                break;
        }
    }
}

String id = req.getParameter("id");
Articals artical = articalsDao.getById(Integer.valueOf(id));
switch (action) {
    case EDIT_ACTION:
        List<Author> authorList =authorDao.getAll();
        List<Author> inauthor = new ArrayList<>(artical.getAuthors());
        authorList.removeAll(inauthor);
        List<ArticalsProp> articalsProps = articalsPropDao.getAllPopDesc();
        System.out.print("GetALLPopDesc");
        List<ArticalsProp> propList = articalsPropDao.getAllValue(articalsProps);
        System.out.print("getAllValue");
        List<ArticalsProp> inprop = new ArrayList<>(artical.getArticalsProps());
        propList.removeAll(inprop);
        req.setAttribute("artical", artical);
        req.setAttribute("authorsall", authorList);
        req.setAttribute("authorsin", inauthor);
req.setAttribute("AllDescsValue", propList);
req.setAttribute("propin", inprop);
req.setAttribute("AllDescs", articalsProps);
RequestDispatcher view = req.getRequestDispatcher(ARTICAL);
System.out.print("PrintStr");
view.forward(req, resp);
return;

case DELETE_ACTION:
    articalsDao.delete(artical);
    break;
}
req.setAttribute("articals", articalsDao.getAll());
RequestDispatcher view = req.getRequestDispatcher(AUTHORS);
view.forward(req, resp);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse resp) throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    resp.setCharacterEncoding("UTF-8");
    Articals articals = new Articals();
    articals.setTextArticals(request.getParameter("textArticals"));
    articals.setDescription(request.getParameter("description"));
    articals.setDateCreate(request.getParameter("dateCreate"));
    String[] arrayidauthor[] = request.getParameterValues("authorssel");
    Set<Author> newAuthor = new HashSet<>();
    System.out.print("StartOUTPUT");
    Set<ArticalsProp> articalsPropHashSet = new HashSet<>();
    List<FileItem> multiparts = null;
    Boolean havefile = false;
    try {
        multiparts = new ServletFileUpload(new DiskFileItemFactory()).parseRequest(request);
        String inputName = null;
        for(FileItem item : multiparts){
            if(item.isFormField()){ // Check regular field.
                inputName = (String)item.getFieldName();
                if(inputName.equalsIgnoreCase("authorssel")){
                    String username = (String)item.getString();
                    System.out.println("UserName is:"+username);
                    newAuthor.add(authorDao.getById(Integer.valueOf(username)));
                }
            }
            if(inputName.equalsIgnoreCase("authorssel")){
                String s = (String) item.getString();
                articalsPropHashSet.add(articalsPropDao.getById(Integer.valueOf(s)));
            }
            if(inputName.equalsIgnoreCase("textArticals")){
                String s = (String) item.getString();
            }
        }
    }
}
articals.setTextArticals(s);
}
if (inputName.equalsIgnoreCase("description")) {
    String s = (String) item.getString();
    articals.setDescription(s);
}
if (inputName.equalsIgnoreCase("dateCreate")) {
    String s = (String) item.getString();
    articals.setDateCreate(request.getParameter("s"));
}
}
}
}
try {
}
}
}
} catch (FileUploadException e) {
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

//System.out.print(arrayidauthor);
//for(String s: arrayidauthor) {
//    newAuthor.add(authorDao.getById(Integer.valueOf(s)));
//}
if (!newAuthor.isEmpty()) {
    articals.setAuthors(newAuthor);
} else {

// String idProp[] = request.getParameterValues("prop1");
//for(String s: idProp) {
//    articalsPropHashSet.add(articalsPropDao.getById(Integer.valueOf(s)));
//}
if (!articalsPropHashSet.isEmpty()) {
    articals.setArticalsProps(articalsPropHashSet);
} else {

String stringId = request.getParameter("id");
if (stringId == null || stringId.isEmpty()) {
    if (!havefile) {articals.setFileName("");}
    articalsDao.add(articals);
} else {
    articals.setId(Integer.valueOf(stringId));
    articalsDao.update(articals);
}
doGet(request, resp);
}
AricalsProp.java:

```java
@Entity
@Table(name = "articalprop")
public class AricalsProp {
    @Id
    @Column(name = "ID")
    @GeneratedValue
    private int id;

    @Column(name = "Description")
    private String description;

    @Column(name = "ValueOf")
    private String valueOf;

    @Column(name = "Usege", columnDefinition = "TINYINT")
    @Type(type = "org.hibernate.type.NumericBooleanType")
    private Boolean usege;

    @ManyToMany(fetch = FetchType.EAGER, mappedBy = "articalsProps")
    private Set<Aricals> articles = new HashSet<>();

    public AricalsProp() {
    }

    public AricalsProp(int ID, String Description, boolean Usege) {
        this.id = ID;
        this.description = Description;
        this.usege = Usege;
    }

    public Set<Aricals> getArticles() {
        return this.articles;
    }

    public void setArticles(Set<Aricals> articles) {
        this.articles = articles;
    }

    public int getId() {
        return id;
    }

    public void setId(int ID) {
        this.id = ID;
    }

    public String getDescription() {
        return description;
    }

    public String getDescription() {
        return description;
    }

    public String getDescription() {
        return description;
    }
```

73
public void setDescription(String Description) {
    this.description = Description;
}

public String getValueOf() {
    return valueOf;
}

public void setValueOf(String ValueOf) {
    this.valueOf = ValueOf;
}

public void setUsege(Boolean Usege) {
    this.usege = Usege;
}

public Boolean getUsege() {
    return usege;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    ArticalsProp articalsProp = (ArticalsProp) o;

    return getId() == articalsProp.getId();
}

@Override
public int hashCode() {
    int result = getId();
    result = 31 * result + getUsege().hashCode();
    result = 31 * result + (getDescription() != null ? getDescription().hashCode() : 0);
    return result;
}

}

ArticalsPropDao.java:

public interface ArticalsPropDao {
    void add(ArticalsProp articalsProp);
    void delete(ArticalsProp articalsProp);
    void update(ArticalsProp articalsProp);
    List<ArticalsProp> getAll();
    List<ArticalsProp> getAllPopDesc();
}
List<ArticalsProp> getAllValue(String option);

ArticalsProp getById(int id);

ArticalsPropDaoExampleImpl.java:

public class ArticalsPropDaoExampleImpl implements ArticalsPropDao {
    public ArticalsPropDaoExampleImpl() {
    }

    public void add(ArticalsProp articalsProp) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        session.save(articalsProp);
        session.getTransaction().commit();
    }

    public void delete(ArticalsProp articalsProp) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        session.delete(articalsProp);
        session.flush();
        session.getTransaction().commit();
    }

    public void update(ArticalsProp articalsProp) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        session.update(articalsProp);
        session.getTransaction().commit();
    }

    public List<ArticalsProp> getAllPopDesc() {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        Query q = session.createQuery("from ArticalsProp AP GROUP BY AP.description");
        List<ArticalsProp> AllArticals = q.list();
        session.getTransaction().commit();
        return AllArticals;
    }

    public List<ArticalsProp> getAllValue(String option) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        Query q = session.createQuery("from ArticalsProp AP WHERE AP.description = :option").setString("option", option);
        List<ArticalsProp> AllArticals = q.list();
        session.getTransaction().commit();
        return AllArticals;
    }

    public List<ArticalsProp> getAll() {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    }
session.beginTransaction();
Query q = session.createQuery("from ArticalsProp");
List<ArticalsProp> AllArticals = q.list();
session.getTransaction().commit();
return AllArticals;
}
public ArticalsProp getById(int id) {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    Query q = session.createQuery("from ArticalsProp A where A.id = :articals_id");
    q.setParameter("articals_id", id);
    List<ArticalsProp> Aut = q.list();
    session.getTransaction().commit();
    return Aut.get(0);
}

Модель «События»

Event.java:

@Entity
@Table(name = "event")
public class Event {
    @Id
    @Column(name = "ID")
    @GeneratedValue
    private int id;

    @Column(name = "Name")
    private String title;

    @Column(name = "DateStart", columnDefinition="DATETIME")
    private Date start;

    @Column(name = "DateEnd", columnDefinition="DATETIME")
    private Date end;

    public Event() {
    }

    public Event(int ID, String Header, Date Start, Date Image) {
        this.id = ID;
        this.title = Header;
        this.start = Start;
        this.end = Image;
    }

    public int getId() {
        return id;
    }

    public void setId(int ID) {
    }
}
    this.id = ID;
}

public String getTitle() {
    return title;
}

public void setTitle(String Header) {
    this.title = Header;
}

public Date getStart() {
    return start;
}

public void setStart(Date DateCreate) {
    this.start = DateCreate;
}

public Date getEnd() {
    return end;
}

public void setEnd(Date DateEnd) {
    this.end = DateEnd;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Event news = (Event) o;
    return getId() == news.getId();
}

@Override
public int hashCode() {
    int result = getId();
    result = 31 * result + (getTitle() != null ? getTitle().hashCode() : 0);
    result = 31 * result + getStart().hashCode();
    return result;
}

}

EventDao.java:

public interface EventDao {

}
void add(Event event);
void delete(Event event);
void update(Event event);
List<Event> getAll();
Event getById(int id);
}

EventDaoExampleImpl.java:

public class EventDaoExampleImpl implements EventDao {
    public EventDaoExampleImpl() {
    }

    public void add(Event event) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        session.save(event);
        session.getTransaction().commit();
    }

    public void delete(Event event) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        session.delete(event);
        session.flush();
        session.getTransaction().commit();
    }

    public void update(Event event) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        session.update(event);
        session.getTransaction().commit();
    }

    public List<Event> getAll() {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        Query q = session.createQuery("from Event");
        List<Event> AllNews = q.list();
        session.getTransaction().commit();
        return AllNews;
    }

    public Event getById(int id) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        Query q = session.createQuery("from Event");
        List<Event> AllNews = q.list();
        session.getTransaction().commit();
        return AllNews;
    }
}
System.out.println("Начали");
Query q = session.createQuery("from Event where id= :news_id");
q.setParameter("news_id", id);
List<Event> newses = q.list();
System.out.println("Получили");
session.getTransaction().commit();
return newses.get(0);}}

Events.java:

@WebServlet("/events")
public class Events extends HttpServlet {
    private static final String EDIT_ACTION = "edit";
    private static final String DELETE_ACTION = "delete";
    private static final String ADD_ACTION = "add";
    private static final String ACTION = "action";
    private static final String AUTHORS = "/events.jsp";
    private static final String AUTHOR = "/event.jsp";
    private AuthorDao authorDao;
    private EventDao eventDao;

    public Events() {
        authorDao = new AuthorDaoExampleImpl();
        eventDao = new EventDaoExampleImpl();
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        HttpSession session = req.getSession();
        req.setAttribute("event", authorDao.getAll());
        if (session.getAttribute("userInfo") != null) {
            Author AutIn = authorDao.getById((Integer) session.getAttribute("userInfo"));
            req.setAttribute("AutIN", true);
            req.setAttribute("userin", AutIn);
        } else {req.setAttribute("AutIN", false);}
        RequestDispatcher view = req.getRequestDispatcher(AUTHORS);
        view.forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("INDOPOST");
        request.setCharacterEncoding("UTF-8");
        resp.setCharacterEncoding("UTF-8");
        Event event = new Event();
        DateFormat format = new SimpleDateFormat("dd.MM.yyyy HH:mm");
        String stringId = request.getParameter("eventid");
try {
    Date dateStart = format.parse(request.getParameter("start_date"));
    Date dateEnd = format.parse(request.getParameter("end_date"));
    event.setStart(dateStart);
    event.setEnd(dateEnd);
    System.out.print("WHERE");
    System.out.print(dateStart);
    event.setTitle(request.getParameter("nameEvent"));
    if (stringId == null || stringId.isEmpty()) {
        eventDao.add(event);
    } else {
        String act = request.getParameter("act");
        if (act == null) {
            //no button has been selected
        } else if (act.equals("delete")) {
            event.setId(Integer.valueOf(stringId));
            eventDao.delete(event);
        } else if (act.equals("update")) {
            event.setId(Integer.valueOf(stringId));
            eventDao.update(event);
        } else {
            //someone has altered the HTML and sent a different value!
        }
    }
}
'

} catch (ParseException e) {
    System.out.print("ErrorConvertData");
    e.printStackTrace();
}

ForGetEvents.java:

@WebServlet("/events/CalendarJsonServlet")
public class ForGetEvents extends HttpServlet {

    private EventDao eventDao;

    doGet(request, resp);

}
public ForGetEvents() {
    eventDao = new EventDaoExampleImpl();
}

@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    List<Event> AllEvent = eventDao.getAll();
    System.out.print("EVENTSALL");
    System.out.print(AllEvent.get(0).getStart());
    resp.setContentType("application/json");
    resp.setCharacterEncoding("UTF-8");
    PrintWriter out = resp.getWriter();
    out.write(new Gson().toJson(AllEvent));
}

Модель «Новость»

News.java:

@Entity
@Table(name = "news")
public class News {
    @Id
    @Column(name = "ID")
    @GeneratedValue
    private int id;

    @Column(name = "Header")
    private String header;

    @Column(name = "TextNews")
    private String text;

    @Column(name = "DateAdd", columnDefinition="DATE")
    private Date dateCreate;

    @Column(name = "Image")
    private String image;

    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinTable(name = "author_news",
                joinColumns = {@JoinColumn(name = "IDNews", referencedColumnName = "id")},
                inverseJoinColumns = {@JoinColumn(name = "IDAut", referencedColumnName = "id")})
    private Set<Author> authors = new HashSet<>(0);
public News() {
}

public News(int ID, String Header, String Text, Date DateCreate, String Image) {
    this.id = ID;
    this.header = Header;
    this.text = Text;
    this.dateCreate = DateCreate;
    this.image = Image;
}

public Set<Author> getAuthors() {
    return authors;
}

public void setAuthors(Set<Author> art) {
    this.authors = art;
}

public int getId() {
    return id;
}

public void setId(int ID) {
    this.id = ID;
}

public String getText() {
    return text;
}

public void setText(String Text) {
    this.text = Text;
}

public String getHeader() {
    return header;
}

public void setHeader(String Header) {
    this.header = Header;
}

public Date getDateCreate() {
    return dateCreate;
}

public void setDateCreate(Date DateCreate) {
    this.dateCreate = DateCreate;
}

public String getImage() {
    return image;
}
public void setImage(String Image) {
    this.image = Image;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;

    News news = (News) o;

    return getId() == news.getId();
}

@Override
public int hashCode() {
    int result = getId();
    result = 31 * result + getText().hashCode();
    result = 31 * result + (getHeader() != null ? getHeader().hashCode() : 0);
    result = 31 * result + getDateCreate().hashCode();
    return result;
}

NewsDao.java:

public interface NewsDao {
    void add(News news);
    void delete(News news);
    void update(News news);
    List<News> getAll();
    News getById(int id);
}

NewsDaoExampleImpl.java:

public class NewsDaoExampleImpl implements NewsDao {
    public NewsDaoExampleImpl() {
    }

    public void add(News news) {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        
    }
}
public void delete(News news) {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    session.delete(news);
    session.flush();
    session.getTransaction().commit();
}

public void update(News news) {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    session.update(news);
    session.getTransaction().commit();
}

public List<News> getAll() {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    Query q = session.createQuery("from News");
    List<News> AllNews = q.list();
    session.getTransaction().commit();
    return AllNews;
}

public News getById(int id) {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    System.out.print("Начали");
    Query q = session.createQuery("from News where id= :news_id");
    q.setParameter("news_id", id);
    List<News> newses = q.list();
    System.out.print("Получили");
    session.getTransaction().commit();
    return newses.get(0);
}

NewsServlet.java:

@WebServlet("/news")
public class NewsServlet extends HttpServlet {
    //private ArticalsDao articalsDao;
    private AuthorDao authorDao;
    private NewsDao newsDao;
    private static final String EDIT_ACTION = "edit";
}
private static final String DELETE_ACTION = "delete";
private static final String ADD_ACTION = "add";
private static final String VIEW_ACTION = "view";
private static final String ACTION = "action";
private static final String AUTHORS = "/news.jsp";
private static final String ARTICAL = "/newso.jsp";
private static final String NewsView = "/newsview.jsp";

public NewsServlet() {
    authorDao = new AuthorDaoExampleImpl();
    newsDao = new NewsDaoExampleImpl();
}

@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String action = req.getParameter(ACTION);
    HttpSession session = req.getSession();
    req.setAttribute("authors", authorDao.getAll());
    if (session.getAttribute("userInfo") != null) {
        Author AutIn = authorDao.getById((Integer) session.getAttribute("userInfo"));
        req.setAttribute("AutIN",true);
        req.setAttribute("userin",AutIn);
    }else{req.setAttribute("AutIN",false);}
    if (action != null) {
        if (action.equals(ADD_ACTION)) {
            RequestDispatcher view = req.getRequestDispatcher(ARTICAL);
            view.forward(req, resp);
            return;
        }
        String id = req.getParameter("id");
        DateFormat format = new SimpleDateFormat("dd.MM.yyyy");
        News news = newsDao.getById(Integer.valueOf(id));
        switch (action) {
            
        }
    }
}

String id = req.getParameter("id");
DateFormat format = new SimpleDateFormat("dd.MM.yyyy");
News news = newsDao.getById(Integer.valueOf(id));
switch (action) {

}
case EDIT_ACTION:

    RequestDispatcher view = req.getRequestDispatcher(ARTICAL);
    view.forward(req, resp);
    return;

case VIEW_ACTION:

    req.setAttribute("DateCreate", format.format(news.getDateCreate()));
    req.setAttribute("news", news);
    RequestDispatcher views = req.getRequestDispatcher(NewsView);
    views.forward(req, resp);
    return;

case DELETE_ACTION:

    newsDao.delete(news);
    break;
}

req.setAttribute("news", newsDao.getAll());
RequestDispatcher view = req.getRequestDispatcher(AUTHORS);
view.forward(req, resp);

}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse resp) throws ServletException, IOException {

    request.setCharacterEncoding("UTF-8");
    resp.setCharacterEncoding("UTF-8");

    String stringId = request.getParameter("id");
    if (stringId == null || stringId.isEmpty()) {
        HttpSession session = request.getSession();
        Author AutIn = authorDao.getById((Integer) session.getAttribute("userInfo"));
    }
News news = new News();
//news.setDateCreate();
news.setHeader(request.getParameter("description"));
news.setText(request.getParameter("body"));

Date date = new Date();
news.setDateCreate(date);
Set<Author> authorSet = new HashSet<>();
authorSet.add(AutIn);
news.setAuthors(authorSet);
newsDao.add(news);
}
]
else {

News news = newsDao.getById(Integer.valueOf(stringId));
//news.setDateCreate();
news.setHeader(request.getParameter("description"));
news.setText(request.getParameter("body"));
//news.setId(Integer.valueOf(stringId));
newsDao.update(news);
}
doGet(request, resp);
}