ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

СОЗДАНИЕ ИНТЕРНЕТ-ПОРТАЛА ДЛЯ ОБЕСПЕЧЕНИЯ ПРОВЕДЕНИЯ НАУЧНЫХ МЕРОПРИЯТИЙ В НИУ «БЕЛГУ»

Выпускная квалификационная работа
обучающегося по направлению подготовки 09.03.02 Информационные системы и технологии
очной формы обучения, группы 07001408
Евтеева Вадима Эдуардовича

Научный руководитель
к.ф.-м.н., доцент
Мигаль Л.В.

БЕЛГОРОД 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Роль науки в развитии общества без преувеличения является определяющей. Также наука – это основной элемент, на котором строится деятельность университета. Сегодня университеты, в частности НИУ «БелГУ», имеют возможность решать исследовательские, конструкторские, технологические задачи. К формам организации научной деятельности можно отнести: научную конференцию, семинар, форум, конгресс, совещание и т.д.

Научные конференции позволяют предоставлять результаты исследований научной деятельности, развивая тем самым науку. В ходе проведения научных конференций участники делятся идеями, узнают новые достижения в сфере науки и инноваций.

Научные конференции прекрасно дополняют основную научную деятельность. Они позволяют делиться опытом, общаться на научные темы. А это, в свою очередь, влияет на качество научного процесса.

На научных конференциях участники могут познакомиться с актуальность различных исследований, с новостями из мира науки, научными трендами. Конференции позволяют взглянуть не только на работу других людей, но и посмотреть со стороны на свои исследования: подвергнуться конструктивной критике со стороны принимающих в конференции участие специалистов, которые могут направить к новым идеям или указать на неточности в исследованиях.

Кроме того, участие в научных конференциях помогает в улучшении личностных качеств, таких как: уверенность, способность выступать на публике, ораторство.

Исходя из вышесказанного, сложно не согласиться в полезности и актуальности научных конференций. Они предоставляют множество преимуществ. При этом, хорошо организованная конференция не несет за собой недостатков.

Стоит отметить, что в современном мире, самым популярным ресурсом получения информации является интернет. Терабайты данных в глобальной сети позволяют получить знания практически обо всех открытиях, практически обо всем, что изобрело человечество. Также интернет является и одним из самых доступных ресурсов в современном обществе.

Данные преимущества интернета способны облегчить и организацию научных мероприятий, оповестить о новых конференциях, помочь зарегистрироваться на мероприятиях потенциальным участникам.

Во многих университетах имеются специальные порталы научных мероприятий, которые позволяют автоматизировать часть работы организаторов мероприятия, что ведет к повышению качества мероприятий, упрощению процедуры регистрации участников конференций. Для реализации таких порталов следует использовать серверные технологии, реляционные базы данных, методологии проектирования (DFD/IDEF0), алгоритмы хеширования (SHA, MD5 и др.) для обеспечения безопасности, и другие технологии.

Целью данной выпускной квалификационной работы является усовершенствование процесса организации научных мероприятий в НИУ «БелГУ» путем создания портала для обеспечения проведения научных мероприятий. Для достижения цели следует выполнить ряд следующих задач:

– провести анализ предметной области;

– рассмотреть структуру существующих порталов научных мероприятий;

– разработать диаграммы потоков данных;

– спроектировать и реализовать базу данных;

– создать портал для обеспечения проведения научных мероприятий;

– провести тестирование созданного портала.

Выполнение данных задач повысит качество проведения организации научных мероприятий, проводимых в НИУ «БелГУ».

1 Аналитическая часть

1.1 Общая информация о научных конференциях

Научная конференция – это форма организации научной деятельности, при которой исследователи представляют и обсуждают свои работы. Обычно заранее (в информационном письме либо стендовом объявлении) сообщается о теме, времени и месте проведения конференции. Затем начинается сбор тезисов докладов и иногда организационных взносов. По своему статусу научная конференция занимает промежуточное положение между научным семинаром и конгрессом [1].

Этапы проведения научных мероприятий:

– регистрация участников с раздачей программы конференции (с указанием очередности выступлений);

– открытие и пленарное заседание с выступлением организаторов конференции;

– работа по секциям или круглым столам с заслушиванием докладов и последующим обсуждением;

– кофе-брейк в середине работы конференции и фуршет или банкет по её окончании;

– культурные программы (экскурсии) для иногородних гостей;

– публикация сборника научных трудов и сертификата участника. Часто сборник выдается участникам конференции при регистрации.

Виды научных конференций [2]:

– научно-теоретическая конференция;

– научно-практическая конференция;

– научно-техническая конференция;

– научно-исследовательская конференция.

Научно-техническая конференция – это конференция, на которой обсуждаются теоретические подходы к решению различных научным проблем и вопросов, постоянно возникающих в ходе исследований или экспериментов.

Научно-практическая конференция – это такая конференция, на которой осуществляется обмен опытом и знаниями по, различного рода, практическим и прикладным задачам.

Научно-техническая конференция – это конференция, на которой осуществляется обмен опытом и знаниями по различным техническим и технологическим вопросам.

Научно-исследовательская конференция – это конференция, на которой осуществляется обмен опытом и знаниями по различным научно-исследовательским вопросам.

В зависимости от охваченной территории выделяют следующие типы научных конференций:

– локальные (школьные, факультетские, внутривузовские, межвузовские);

– региональные, областные;

– всероссийские;

– всероссийские с международным участием;

– международные.

Также конференции можно классифицировать по тематике:

– узкоспециализированные, т.е. посвященные какой-либо отдельной тематике;

– конференции широкой тематики, охватывающие общенаучные вопросы.

Существует несколько форм конференций [3]:

– очные (участник приезжает на саму конференцию и принимает непосредственное участие);

– заочные (участник отправляет свои тезисы и заявку на почту оргкомитета конференции, конференция проходит без его непосредственного участия).

– Internet-конференции (конференция проводится на сайте конференции или организации, чаще всего на форуме, в виде коллективного обсуждения)

1.2 Организация научных мероприятий в НИУ «БелГУ»

Инициирование проведения научных мероприятий осуществляется структурными подразделениями НИУ «БелГУ» и проводятся в соответствии с тематическим планом научно-исследовательской работы.

Конференция, в основном, проводится в несколько этапов [4]:

– планирование;

– первый организационный этап;

– информационный этап;

– второй организационный этап;

– конференция.

При планировании конференции продумывается форма конференции и определяется ее название. Конференция может быть посвящена какому-нибудь юбилейному событию (разовые конференции) или проводиться через определенные промежутки времени, что позволяет специалистам поддерживать и расширять контакты с разными группами исследователей, знакомиться с современными разработками в сфере науки и инноваций.

На момент первого организационного этапа назначаются и распределяются рабочие группы, оргкомитет, определяется председатель и сопредседатель оргкомитета конференции. Чётко определяется тематика, секции конференции и т.п.

На информационном этапе чаще всего о конференциях сообщается в информационном письме или на стендовом объявлении. Как бы ни выглядело информационное сообщение, в нем обязательно должны быть указаны:

– название конференции и эмблема, дата и место проведения;

– организаторы конференции, тематические направления, контактные номера телефонов и факсов, адреса (почтовые и электронные) и адрес сайта конференции, где доступна более подробная информация;

– информация о том, где и как можно зарегистрироваться для участия в конференции, а также что и в какие сроки нужно представить;

– рабочий язык конференции, а также информация о том, где и как будут опубликованы материалы конференции.

На втором организационном этапе поступают заявки, тезисы и оргвзносы от будущих участников конференции, задаются вопросы, появляются проблемы, разворачивается большой круг задач, которые должны быть решены к началу конференции. Проводится отбор участников из числа людей, подавших заявку. Чаще всего, на этом этапе рассылается второе информационное письмо отобранным участникам.

На этапе конференции проходит заселение участников, открытие конференции (часто сопровождается небольшим концертом) слушанье докладов, просмотр стендовых докладов. Во время конференции обычно устраиваются различные экскурсии в НИИ, университеты, музеи, на предприятия, обзорные экскурсии по городу. Если конференция длится несколько дней, то, скорее всего, один из дней посвящен научным или научно-популярным лекциям для участников. Между длинными докладами устраиваются кофе-брейки на 15-20 минут. После конференции жюри обсуждают работы, представленные на ней, и присуждают места лучшим докладчикам. Проходит награждение победителей (а зачастую и всех участников). Чаще всего на награждении вручаются и сборники работ, в которых присутствуют отправленные участником тезисы. После конференции

практически всегда проходит фуршет или банкет со всеми участниками и оргкомитетом.

Правила участия в конференции определяются оргкомитетом данной конференции и оговариваются в информационном письме. Поэтому каких-то определенных строгих правил нет. Чаще всего, в зависимости от вида конференции, участником ее может стать школьник или студент, а также молодой ученый в возрасте до 35 лет. Участник должен заполнить заявку на участие по форме, которая прилагается в информационном письме либо на сайте конференции. После этого участник должен написать и отправить тезисы своей работы (краткое описание проделанной работы) на электронную почту конференции или организации, которая ее проводит. Для написания тезисов обычно предписаны строгие правила, это необходимо для того, чтобы не возникло проблем при печати сборника тезисов. Если конференция международная или всероссийская с международным участием, то обычно просят выслать в дополнение к русскому варианту тезисов еще и английский сокращенный вариант. После того как тезисы отправлены, остается только ждать ответа от оргкомитета. В случае положительного ответа, участник вправе попросить прислать ему пригласительное письмо, чтобы получить деньги на поездку от командирующей стороны (школы, вуза). Пригласительное письмо должно содержать в себе ФИО участника, название конференции, дату и место ее проведения, название работы участника, контакты оргкомитета (секретаря) и подпись ответственного лица.

1.3 Анализ существующих порталов

В настоящее время существует большое количество порталов, разработанных с целью способствовать информированию общества о предстоящих или прошедших научных мероприятиях. В данном пункте выпускной квалификационной работы приведен анализ представленной информации на соответствующих порталах.

На рисунке 1.1 предаствлена страница, на которой изображен список проводимых научных мероприятий портала konferencii.ru [5].

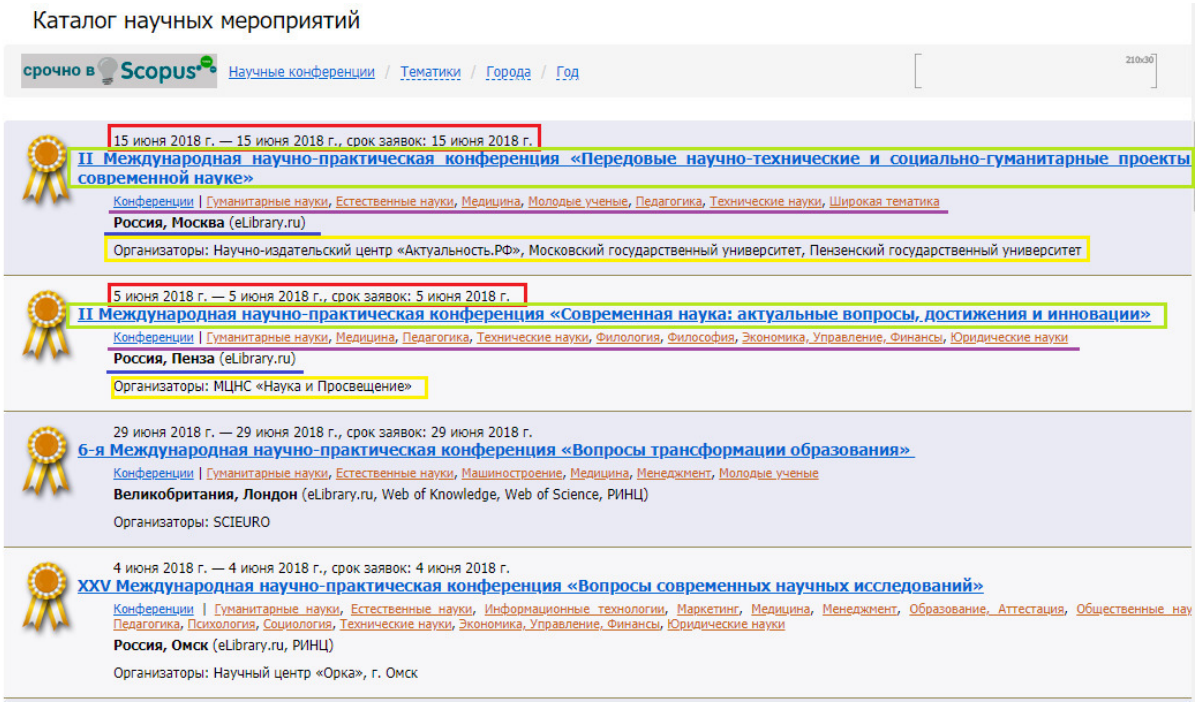


Рисунок 1.1 – Меню научных мероприятий портала konferencii.ru

Из рисунка видно, что основной информацией на данной странице являются поля:

– дата проведения конференции;

– срок принятия заявок;

– название конференции;

– теги;

– место проведения конференции;

– организаторы конференции.

На рисунке 1.2 изображена страница выбранного мероприятия портала konferencii.ru. Как видно, страница состоит из семи блоков:

– название мероприятия;

– место проведения мероприятия;

– форма участия;

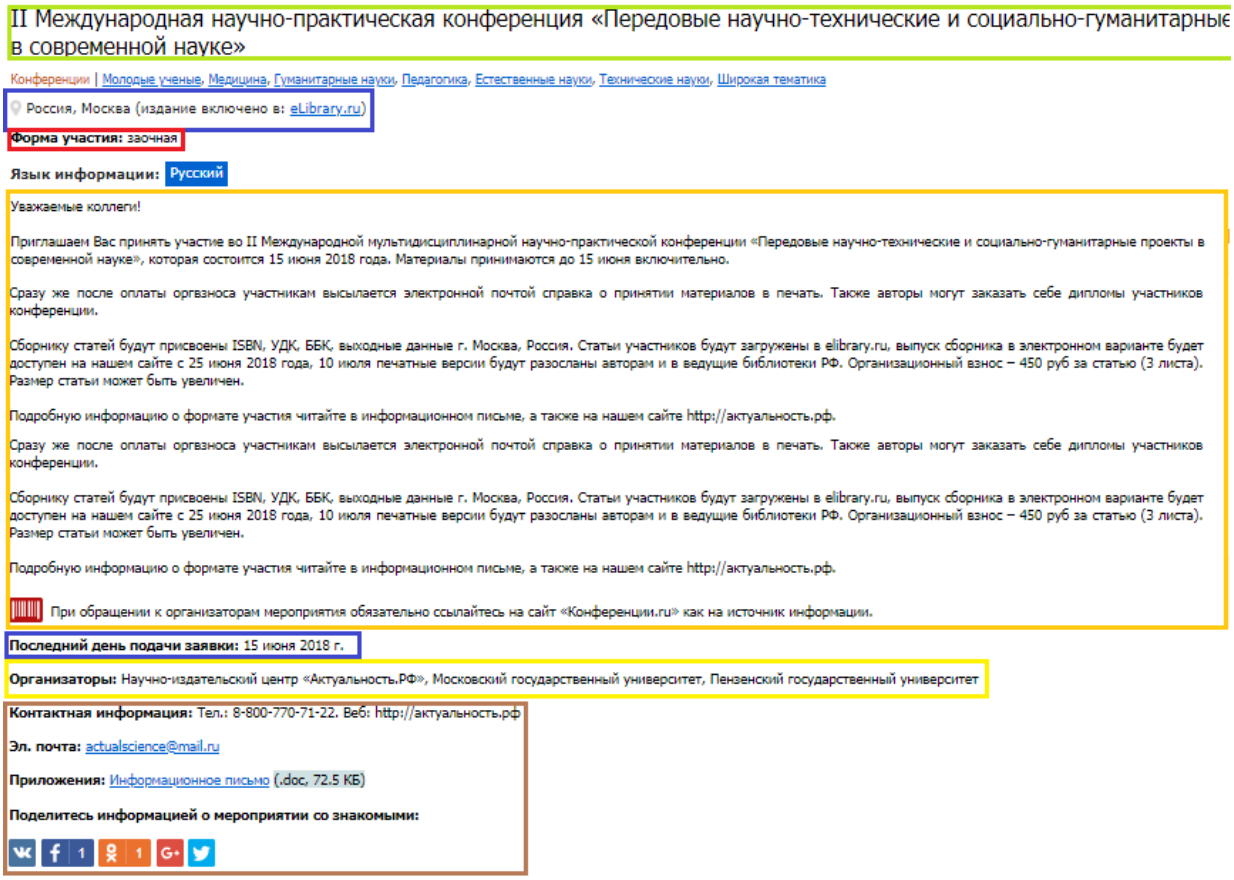
– текстовый информационный блок;

– крайняя дата подачи заявки;

– организаторы;

– контактная информация.

Исходя из анализа блоков страниц, можно прийти к выводу, что данные страницы различаются лишь несколькими блоками и оформлением. Информация на странице с конференциями дана более тезисно и кратко, а информация о выбранном мероприятии представлена в более развернутом виде.


II Международная научно-практическая конференция «Передовые научно-технические и социально-гуманитарные в современной науке»

Конференции | Молодые ученые, Медицина, Гуманитарные науки, Педагогика, Естественные науки, Технические науки, Широкая тематика

Россия, Москва (издание включено в: eLibrary.ru)

Форма участия: заочная

Язык информации: Русский

Уважаемые коллеги!

Приглашаем Вас принять участие во II Международной мультидисциплинарной научно-практической конференции «Передовые научно-технические и социально-гуманитарные проекты в современной науке», которая состоится 15 июня 2018 года. Материалы принимаются до 15 июня включительно.

Сразу же после оплаты оргвзноса участникам высылается электронной почтой справка о принятии материалов в печать. Также авторы могут заказать себе дипломы участников конференции.

Сборнику статей будут присвоены ISBN, УДК, ББК, выходные данные г. Москва, Россия. Статьи участников будут загружены в elibrary.ru, выпуск сборника в электронном варианте будет доступен на нашем сайте с 25 июня 2018 года, 10 июля печатные версии будут разосланы авторам и в ведущие библиотеки РФ. Организационный взнос – 450 руб за статью (3 листа). Размер статьи может быть увеличен.

Подробную информацию о формате участия читайте в информационном письме, а также на нашем сайте http://актуальность.рф.

Сразу же после оплаты оргвзноса участникам высылается электронной почтой справка о принятии материалов в печать. Также авторы могут заказать себе дипломы участников конференции.

Сборнику статей будут присвоены ISBN, УДК, ББК, выходные данные г. Москва, Россия. Статьи участников будут загружены в elibrary.ru, выпуск сборника в электронном варианте будет доступен на нашем сайте с 25 июня 2018 года, 10 июля печатные версии будут разосланы авторам и в ведущие библиотеки РФ. Организационный взнос – 450 руб за статью (3 листа). Размер статьи может быть увеличен.

Подробную информацию о формате участия читайте в информационном письме, а также на нашем сайте http://актуальность.рф.

При обращении к организаторам мероприятия обязательно ссылайтесь на сайт «Конференции.ru» как на источник информации.

Последний день подачи заявки: 15 июня 2018 г.

Организаторы: Научно-издательский центр «Актуальность.РФ», Московский государственный университет, Пензенский государственный университет

Контактная информация: Тел.: 8-800-770-71-22. Веб: http://актуальность.рф

Эл. почта: actualscience@mail.ru

Приложения: Информационное письмо (.doc, 72.5 КБ)

Поделитесь информацией о мероприятии со знакомыми:

Рисунок 1.2 – Страница конференции портала konferencii.ru

На рисунке 1.3 представлена страница, на которой изображены научные мероприятия портала science-community.org [6].

Ключевыми пунктами здесь являются:

– название конференции;

– сроки проведения;

– дата окончания подачи заявки;

– место проведения.

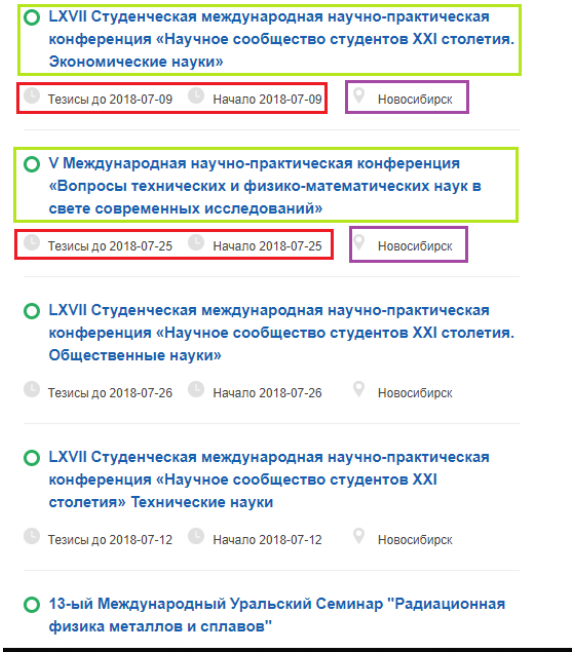


Рисунок 1.3 – Меню научных мероприятий портала science-community.org

При переходе по ссылке мероприятия, предстает страница данного мероприятия, которая продемонстрирована на рисунке 1.4.

Исходя из рисунка видно, что основными блоками страницы являются блоки:

– названия конференции;

– места проведения;

– организаторов;

– обратной связи;

– области наук;

– блока с текстовой информацией.

Из данного анализа видно, что блоки на странице самого мероприятия являются расширенной версией, по сравнению с блоками на странице, отображающей все мероприятия.

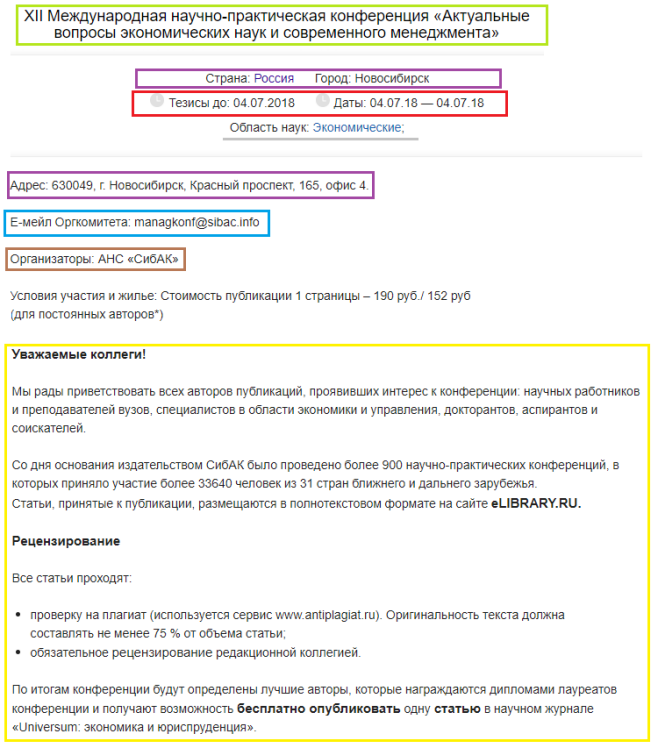


Рисунок 1.4 – Страница конференции портала science-community.org

Если объединить полученную информацию, то на странице мероприятия на обеих порталах присутствует:

– название мероприятия;

– сроки проведения;

– крайняя дата подачи заявки;

– организаторы;

– место проведения;

– описательная часть (текстовый блок);

– контактная информация;

– область науки.

Существует также страница, посвященная научным мероприятиям, на портале НИУ «БелГУ», которая представлена на рисунке 1.5.

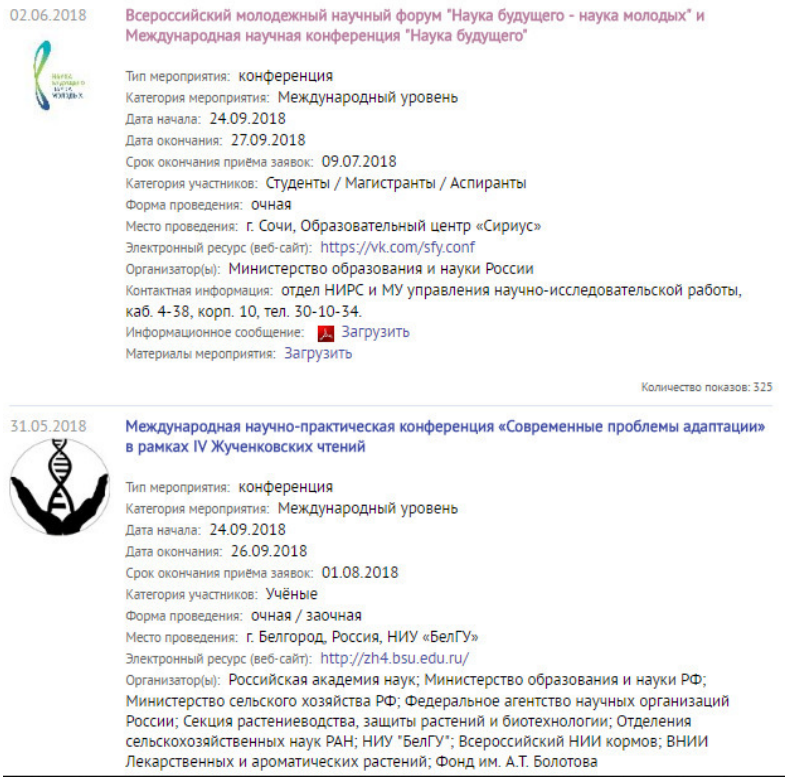


Рисунок 1.5 – Страница научных мероприятий НИУ «БелГУ»

Данная страница содержит поля, аналогичные предыдущим ресурсам, но она не является достаточно автономной. Страница представлена новостной лентой. Также отсутствует возможность онлайн регистрации на мероприятие. Данные недостатки будут устранены при создании портала в рамках данной работы.

Выводы по первому разделу

В данном разделе был проведен анализ предметной области и рассмотрены существующие решения. В результате этого были определены: примерный перечень элементов, которые должен содержать ресурс; структура интернет-портала; информационное обеспечение порталов научной тематики.

2 Проектирование программного продукта

2.1 Структурный анализ потоков данных

На данном этапе рассмотрены потоки данных, которые представлены в программном продукте. Описание потоков проводится с помощью аннотации DFD.

DFD (data flow diagrams) – диаграммы потоков данных – методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ. Диаграмма потоков данных – один из основных инструментов структурного анализа и проектирования информационных систем. Нотация DFD прекрасно подходит для моделирования информационной системы с точки зрения хранения, обработки и передачи данных [7].

Нотация DFD предполагает использование следующих блоков:

– внешняя сущность – представляет собой объект, не входящий в состав системы, но являющийся для нее источником информации, получателем или же и тем, и другим;

– процесс – функция, которая осуществляет обработку данных;

– хранилище данных – хранилище, в которое осуществляется занесение информации после обработки данных или же ее получение.

Данный блоки связываются между собой с помощью потоков данных, представляемых стрелками. Потоки данных отображают входную и выходную информацию, демонстрируя перемещение данных между составляющими системы.

DFD может выполняться в двух нотациях: Йордана, или же в нотации Гейна-Сарсона. Данная работа выполнена в нотации Гейна-Сарсона, которая является наиболее распространенной на данный момент.

На рисунке 2.1 демонстрируется главная диаграмма потоков данных (DFD).

Рисунок 2.1 – Главная диаграмма потоков данных

Из диаграммы видно, что внешней сущностью является пользователь, который взаимодействует с порталом для обеспечения проведения научных мероприятий.

От пользователя поступает личная информация, представленная потоком данных «информация о пользователе», а также заявка на участие, содержащая в себе мероприятие, в котором пользователь желает принять участие, доклад и т.д., представленная потоком данных «заявка на участие».

В ответ пользователь получает информацию о результате проверки, а также подтверждение или отклонение поданной им заявки, представленный потоком данных «ответ на заявку».

На рисунке 2.2 рассматривается декомпозиция главной диаграммы.

Рисунок 2.2 – Декомпозиция главной диаграммы потоков данных

Из данной диаграммы видно, что информация, предоставляемая пользователем о себе, передается в функцию «зарегистрировать пользователя». При регистрации, личные данные пользователя заносятся в базу данных, в таблицу, хранящую список пользователей, а также их личные данные. Таблица представлена хранилищем данных «список пользователей».

В функции «авторизовать пользователя» происходит авторизация пользователя на портале. Функция получает данные для авторизации и личные данные пользователя, представленные одноименными потоками. При авторизации образуется поток «данные авторизованного пользователя». Этот поток поступает в функцию «подать заявку на участие» вместе с потоком «заявка на участие». Данная функция производит запись оформленной заявки в базу данных.

Новые заявки выгружаются в функцию «обработать заявку». Данная функция взаимодействует с сущностью «рецензент», предоставляя ему новые заявки на участие, а в ответ получая результат проверки. Далее, функция производит обновление заявки на участие в хранилище данных «заявки на участие в конференциях», по средствам потока данных «проверенная заявка».

Поток данных «проверенные заявки» отправляются в функцию «составить список участников». Запись о списке участников производится в хранилище данных «списки участников конференций».

Сформированные списки участников попадают в функцию «выслать ответ на заявку», которая передает пользователю ответ на заявку.

2.2 Проектирование базы данных

Прежде всего, следует определиться с понятием базы данных.

База данных называется упорядоченная совокупность взаимосвязанных сведений по свойствам определенной группы объектов (статей, научных материалов, новостей, нормативных актов и иных подобных материалов), систематизированных таким образом, чтобы эти сведения могли быть найдены и обработаны, например, с помощью электронной вычислительной машины (ЭВМ) [8].

Одним из типов баз данных является реляционная база данных. Она строится на отношении таблиц между собой [9]. Разработка как раз такой базы данных и ведется в данном проекте.

Схема базы данных – это её структура, описываемая в схематическом виде, на формальном языке.

Схема реляционной базы данных состоит из таблиц (сущностей) и полей (характеристик сущностей). В каждой таблице должен присутствовать первичный ключ, который является идентификатором записи, по которому другие таблицы смогут связаться с данной. Внешний ключ также определяет связь с другой таблицей.

В разделе 1 были выявлены основные поля, которые должны присутствовать в базе данных. Остается определиться с сущностями, их полями и связью между ними. Схема базы данных изображена на рисунке 2.3.

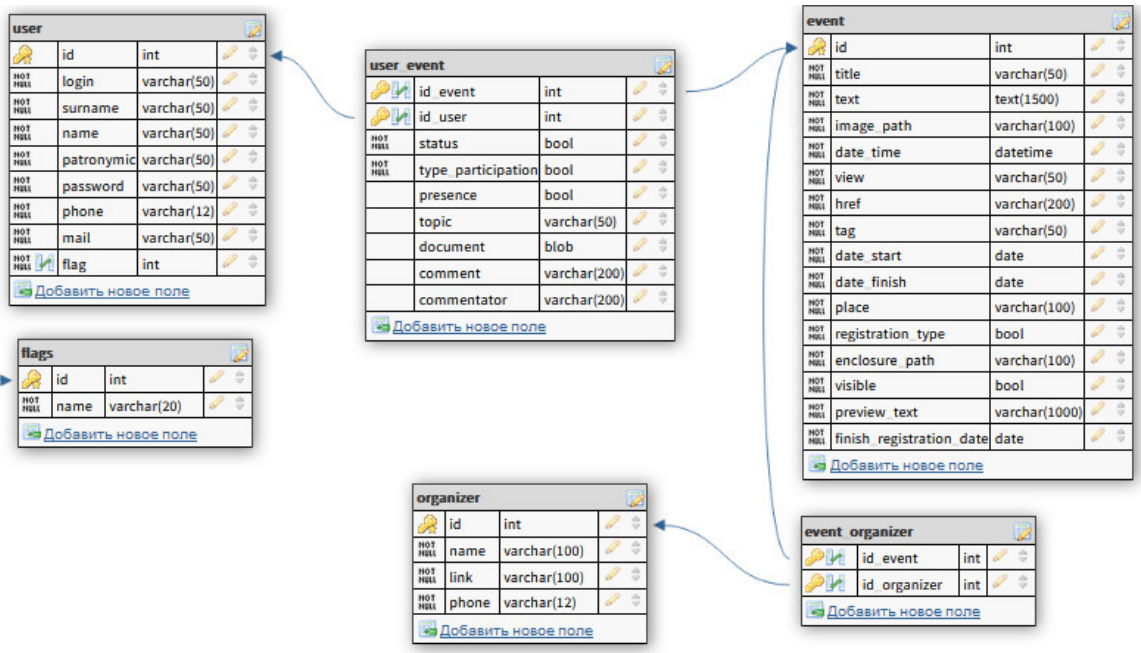


Рисунок 2.3 – Схема базы данных

Для начала следует разобрать таблицу научного мероприятия «event». Данная таблица содержит следующие поля:

– id – идентификатор мероприятия;

– title – заголовок мероприятия;

– text – текст мероприятия;

– image_path – путь к главной картинке мероприятия;

– date_time – дата и время поста;

– view – количество просмотров поста;

– href – ссылка на страницу поста;

– tag – направление научного мероприятия;

– data_start – дата начала мероприятия;

– data_finish – дата окончания мероприятия;

– place – место проведения мероприятия;

– registration_type – тип регистрации (требуется ли регистрация);

– enclosure_path – путь к файлу-вложению;

– visible – видимость поста;

– preview_text – текст, который будет отображаться в списке мероприятий;

– finish_registration_date – дата окончания регистрации.

Таблица организаторов «organizer» содержит следующие поля:

– id – идентификатор таблицы;

– name – наименование организатора;

– link – ссылка на сайт организатора;

– phone – телефон организатора.

Таблицы event и organizer связаны между собой соотношением «многое-ко-многому». Данную связь обеспечивает таблица event_organizer. Эта таблица содержит в себе поля id_event и id_organizer. Вышеназванные поля являются одновременно и первичными, и внешними, что и позволяет отразить связь «многое-ко-многому».

Таблица «user» содержит в себе следующие поля:

– id – идентификатор таблицы;

– login – логин пользователя;

– name – имя пользователя;

– surname – фамилия пользователя;

– patronymic – отчество пользователя;

– password – пароль;

– phone – номер телефона;

– mail – адрес электронной почты;

– flag – флаг пользователя (определяет его роль).

Поле flag является внешним ключом, который связан с таблицей flags. В данной таблице есть два поля: id и название. Эта таблица определяет права пользователей: администратор, рецензент или обычный пользователь.

Поля user и event связаны между собой соотношением «многое-ко-многому», которое осуществляется с помощью таблицы «user_event».

Данная таблица имеет два первичных ключа «id_event» и «id_user», которые являются одновременно и внешними. Кроме этого, таблицы имеют следующие поля:

– status – определяет допущен пользователь или нет;

– type_participation – тип принятия участия (очное или заочное);

– presence – определяет, является ли пользователь выступающим или слушателем;

– topic – тема доклада;

– document – сам доклад;

– comment – комментарий рецензента;

– commentator – ФИО рецензента.

Кроме того, существует таблица «content». В ней будут храниться данные для постов: новостей и статей (рисунок 2.4). Таблица содержит следующие поля:

– id – идентификатор;

– title – заголовок контента;

– text – основной текст;

– image_path – путь к картинке записи;

– date_time – дата и время поста;

– view – количество просмотров поста;

– tag – направление;

– type – тип поста (новость, главная новость или статья);

– author – автор статьи;

– visible – видимость поста;

– preview_text – текст для страницы, отображающей весь контент.

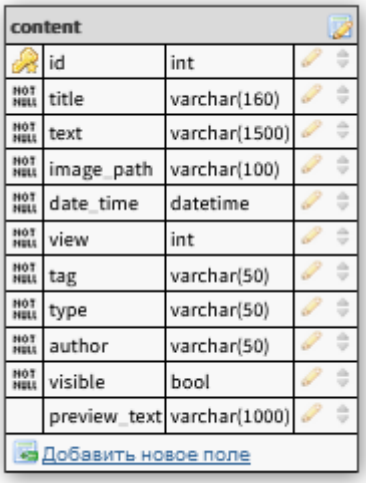


Рисунок 2.4 – Таблица «content»

Выводы по второму разделу

В данном разделе были разработаны диаграммы потоков данных и база данных. Это позволило определить логику клиент-серверного приложения и его приблизительную структуру. Полученные модели позволят перейти к следующему этапу, а именно – написание программного кода.

3 Реализация портала научных мероприятий НИУ «БелГУ»

3.1 Выбор средств реализации портала

Создание web-ресурса проводится в несколько этапов:

– проектирование внешнего вида ресурса;

– верстка страниц ресурса;

– создание базы данных ресурса;

– создание ядра проекта;

– осуществление связи между ядром и страницами web-ресурса.

Для проектирования внешнего вида ресурсов больше всего подходит программа Adobe Photoshop. Данное программное приложение позволяет создавать текстовые и графические элементы, применять к ним различные графические инструменты, по слоям располагать эти элементы.

Для осуществления верстки страниц ресурса больше всего подходит язык HTML и CSS. HTML – язык гипертекстовой разметки [10]. Он позволяет добавить необходимые элементы на страницу ресурса, а также объединять их в группы. Как правило, к HTML подключается CSS (каскадная таблица стилей), которая позволяет описать внешний вид сайта [11]. Кроме того, к HTML можно также подключить и скрипты (программные модули на языке JavaScript), которые позволят выполнить необходимые команды, сделать web-ресурс более динамическим [12].

Создание базы данных проводится с помощью языка SQL. Чтобы создать базу данных, первоначально следует выбрать систему управления базой данных. Выбор был сделан в пользу MySQL 5.7. Данная версия не имеет никаких проблем с соединением с другим программным обеспечением и имеет все необходимые функции. Также, MySQL предоставляет возможность установить MySQL Workbench – инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и

эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL. Это упростит разработку и тестирование проекта.

Для создания ядра web-ресурса существует множество языков. К ним относятся: C, Java, Python, PHP, Ruby, Perl и многие другие. В данном проекте использован Java. Java – это строго типизированный язык высокого уровня, основным преимуществом которого является кроссплатформенность [13]. Данный язык является объектно-ориентированным. Это преимущество позволит уменьшить нагрузку на базу данных, создав сущности и загрузив информацию в память программы. Важно и то, что разрабатывать на Java программы, которые не содержат ошибок, значительно легче, чем на C++.

В качестве среды программирования идеально подходит IntelliJ IDEA Ultimate Edition – интегрированная среда разработки. С ее помощью можно работать как с языком Java, так и с SQL, HTML, CSS, JS файлами. Это позволит, используя лишь данную среду разработать web-портал.

Для того, чтобы создать портал и проводить тестирование необходим локальный сервер, который позволит запустить web-ресурс. Для этого подойдет один из самых популярных контейнеров сервлетов Apache Tomcat. Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server [14].

3.2 Реализация web-портала

Для того, чтобы осуществить верстку страницы, необходимо для начала определиться с дизайном интерфейса. Для разработки дизайна, в рамках данной работы, был использован редактор Adobe Photoshop CC. Первым этапом на пути к реализации web-ресурса является создание макета портала. Одна из страниц создания макета представлена на рисунке 3.1. Вверху макета располагается логотип портала, его название, кнопки входа и регистрации, а также меню.

Меню состоит из четырех пунктов:

– Научные конференции

– Новости

– Статьи

– О нас

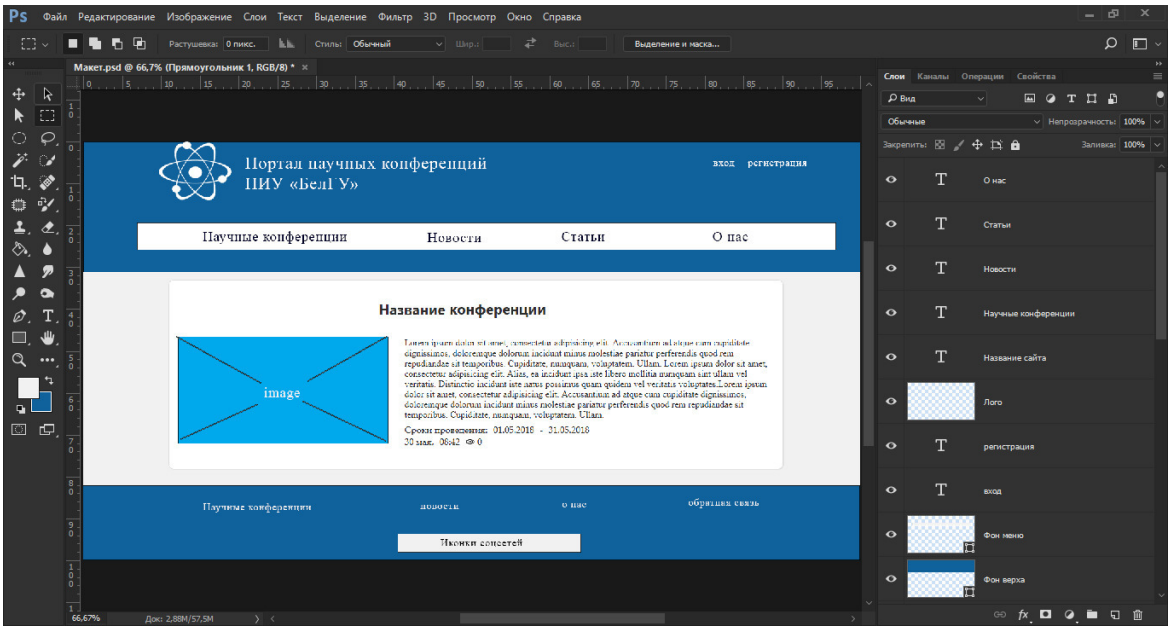


Рисунок 3.1 – Макет страницы

Верхняя часть страницы, как и нижняя, выполнены в светло-синем цвете с шрифтом белого цвета. Меню – белого цвета с темно-синим шрифтом. Тело страницы выполнено светло-серым цветом и содержит в себе научные конференции.

Каждая научная конференция имеет следующую информацию:

– название конференции;

– картинку конференции;

– текст;

– сроки проведения конференции;

– дату публикации;

– количество просмотров конференции.

В нижней части страницы дублируется меню, а также располагается поле с ссылками на социальные сети.

За этапом составление макетов следует этап создания страниц с помощью CSS и HTML кода. Верстка страницы практически полностью соответствует макету. Дальнейшее представление рисунков макетов излишне, так как внешний вид страниц будет виден уже на заключающем этапе и не несет за собой достаточной информативности.

Страница, составленная по макету с помощью языка разметки и языка стилей, представлена на рисунке 3.2.

Рисунок 3.2 – Список актуальных мероприятий
(страница составлена по макету)

Следующим этапом будет создание базы данных. База данных создается полностью, как было составлено на этапе проектирования базы данных и описано во второй главе.

На рисунке 3.3 демонстрируется созданная структура в MySQL Workbench. Схема имеет название «events». Согласно проекту, схема содержит следующие страницы:

– content;

– event;

– event_organizer;

– flags;

– organizer;

– user;

– user_event.

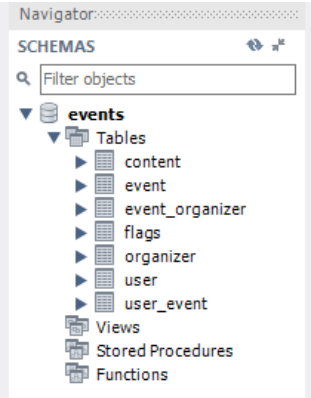


Рисунок 3.3 – Структура базы данных «events»

На рисунке 3.4 демонстрируется одна из таблиц – таблица event. Согласно проекту, она содержит в себе предусмотренные поля. Практически все поля не могут содержать пустые значения. На данном этапе важно задать кодировку, которая используется во всем проекте (UTF-8) для верного отображения всех символов и корректного заполнения базы данных.

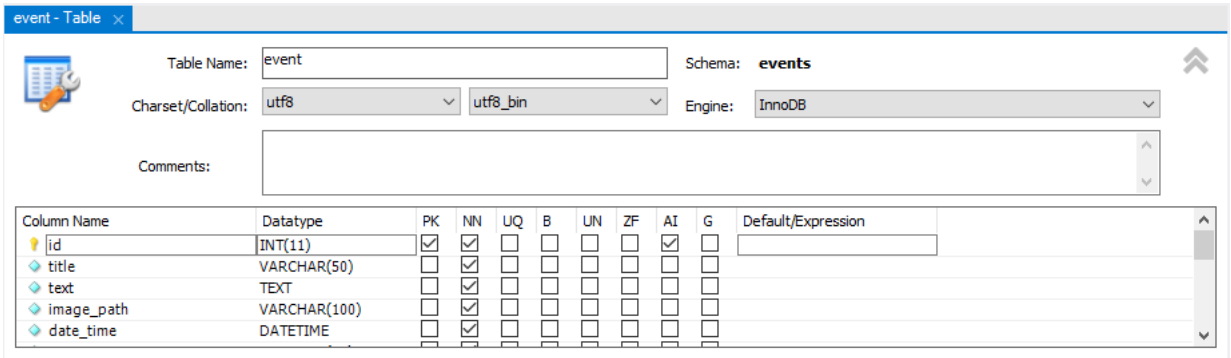
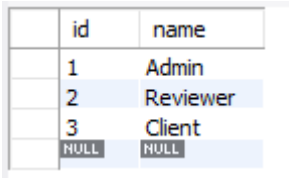


Рисунок 3.4 – Таблица «event»

Стоит отметить особенность таблицы «flags». Данная таблица содержит в себе роли пользователей. На странице администратора не будет к ней

доступа. Следовательно, уже на данном этапе целесообразно ее заполнить. Заполненная таблица представлена на рисунке 3.5.



	id	name
	1	Admin
	2	Reviewer
	3	Client
	NULL	NULL

Рисунок 3.5 – Заполненная таблица «flags»

Данная таблица заполнена следующими записями:

– admin – администратор;

– reviewer – рецензент;

– client – обычный пользователь.

Следующим этапом является перевод html страницы в страницу JSP или сервлета. Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ. При осуществлении данного этапа страницы приобретут динамичность и перестанут быть обычными страницами со статическими элементами.

Но у обычных сервлетов есть недостаток. При достаточном объеме информации, его код становится слишком громоздким и трудночитаемым. Данные недостатки приводят к падению производительности программиста и как следствие падению экономической выгоды. Исходя из этого, в Java были добавлены JSP-страницы. JSP – технология, позволяющая веб-разработчикам создавать содержимое, которое имеет как статические, так и динамические компоненты. Эта технология лишена названных недостатков [15].

Все страницы, созданные с помощью HTML, следует преобразовать в JSP. Статические части HTML остаются неизменными. Динамические же части помещаются в скриптовые элементы JSP. Существуют следующие скриптовые элементы:

– объявления;

– выражения;

– скриплеты.

Объявления, как и в любом другом языке используются для резервирования в памяти какого-либо элемента и, возможно, присвоения ему значения. Выражения используются для вывода статических переменных на страницу.

Скриплеты дают возможность вставить любой код в метод сервлета, который будет создан при обработке страницы, позволяя использовать большинство конструкций Java [16]. Схема, отображающая порядок запросов к JSP представлен на рисунке 3.6.

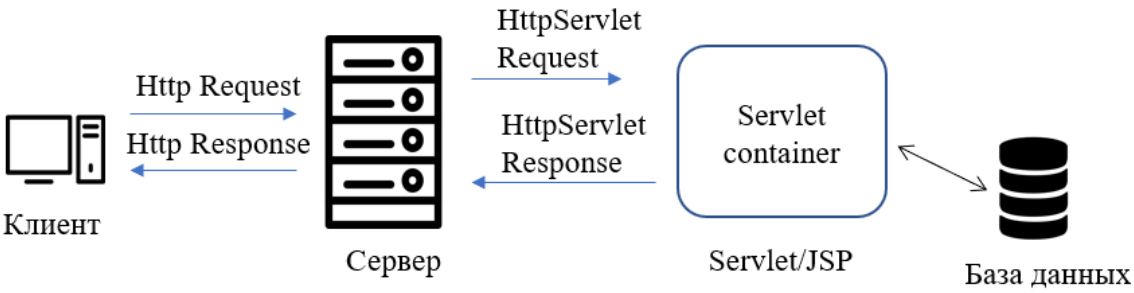


Рисунок 3.6 – Схема работы сервлета

Из схемы видно, что запрос от клиента по сети отправляется на сервер, который преобразует его в запрос, понятный сервлетам и передает в контейнер сервлетов. Контейнер взаимодействует с JSP или Servlet файлами, где и обрабатывается запрос. Если необходимо, из контейнера происходит обращение к базе данных. Далее контейнер посылает назад ответ.

В данном случае, динамическими объектами являются поля, отображающие содержимое базы данных. Далее следует заполнить таблицы базы данных тестовыми значениями. Одной из таких таблиц является таблица «event», представленная на рисунке 3.7.

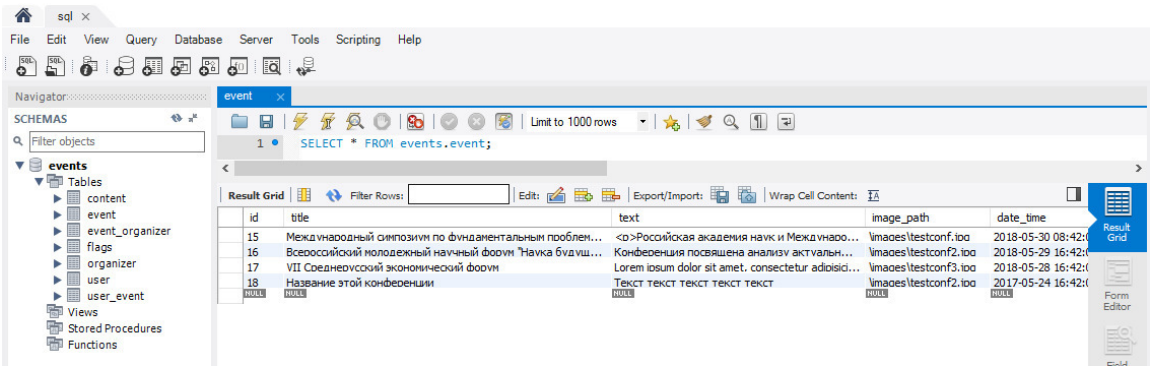


Рисунок 3.7 – Заполнение тестовыми данными таблицы

После заполнения тестовыми данными можно приступить к разработке кода. На странице с конференциями выходной информацией являются:

– название конференции;

– картинка конференции;

– preview текст;

– научное направление;

– сроки проведения конференции;

– дата и время публикации;

– количество просмотров.

Каждый из этих элементов соответствует полям таблицы, а именно:

– title;

– preview_text;

– image_path;

– tag;

– date_start;

– date_finisg;

– date_time

– view.

Для того, чтобы поместить необходимые данные на страницу, необходимо сначала получить их из базы данных. В языке Java для связи с базами данных используются JDBC (Java DataBase Connectivity) – стандарт

взаимодействия Java-приложений с различными СУБД. Для получения соединения требуется загрузить драйвер. Для MySQL таким драйвером является MySQL Connector/J.

Для того, чтобы соединить в себе драйвер базы данных и технологию сервлетов был создан проект, использующий Maven.

Apache Maven – фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, являющемся подмножеством XML. С его помощью не составляет труда подключить все необходимые для работы элементы [17].

Далее происходит создание класса для получения соединения. Таким классом является DBConnector. Данный класс имеет приватные поля, такие как: DB_CONNECTION_URL, строкового типа, содержащий URL для доступа к базе данных, DB_USER_NAME – имя пользователя базы данных и DB_PASSWORD – пароль для доступа к базе. Класс DBConnector имеет два метода:

– getConnection – возвращает соединение с базой данных;

– closeConnection(Statement, Connection) – полностью закрывает соединений.

Для того, чтобы получить информацию из базы данных, чаще всего используют шаблон проектирования DAO (Data Access Object). Данный шаблон представляет собой абстрактный интерфейс к какому-либо типу базы данных или механизму хранения [18].

Реализация данного шаблона происходит следующим образом: создаются классы, соответствующие таблицам, затем создается интерфейс, который содержит в себе объявления методов, которые будут реализованы в классах, расширяющих интерфейс. Методы, объявляемые в нем, могут быть следующие:

– Entity getById(int id) – возвращает сущность по id;

– Entity getByLogin(String login) – возвращает сущность по логину;

– void update(Entity entity) – обновляет запись в БД;

– void delete(int id) – удаляет запись из БД;

– List<Entity> getAll() – возвращает лист со всеми записями и др.

Сущности создаются для того, чтобы не строить слишком сложные запросы в базу данных и выгружать необходимую информацию в память программы. Данный прием позволяет ускорить работу программного приложения, упростить работу с данными, а также избежать множества ошибок.

Далее создаются сервис-классы, которые наследуются от DBConnector и расширяют интерфейс DAO. Данные классы получают соединение из родительского метода getConnection() и реализуют все методы интерфейса, а затем закрывает соединение с помощью родительского метода closeConnection. Как результат – возможность получать, редактировать и удалять данные из базы данных. Структура программного кода представлена в приложении А.

Таким образом, обратившись из JSP-страницы к сервису с помощью скриплета, и получив значения из базы данных с помощью метода getAll(), представленных в виде листа, не составляет труда вставить их через выражения.

Стоит также отметить, что в JSP существует такая технология, как Expression Language, которая позволяет еще проще получать информацию из сущностей. Для этого следует лишь установить необходимые атрибуты, а после воспользоваться ими.

Результат реализации приведенного выше алгоритма представлен на рисунке 3.8. Запись, представленная на рисунке, соответствует по дизайну предварительно созданной HTML странице, а также отображает информацию, заполненную в базу данных. Кроме того, не только одна запись выгружается из базы данных. На JSP странице создан цикл, который отображает все записи, занесенные туда. Это позволяет сделать страницу динамической и автоматизировать процесс добавления и удаления контента на портале.

Рисунок 3.8 – Страница научных конференций с информацией из БД

Важно также отметить, что таблица event имеет поле visible булевого типа. Если поле в записи имеет значение false, то запись не отобразится на странице.

При переходе по ссылке на индивидуальную страницу мероприятия, перед пользователем открывается окно (рисунок 3.9).

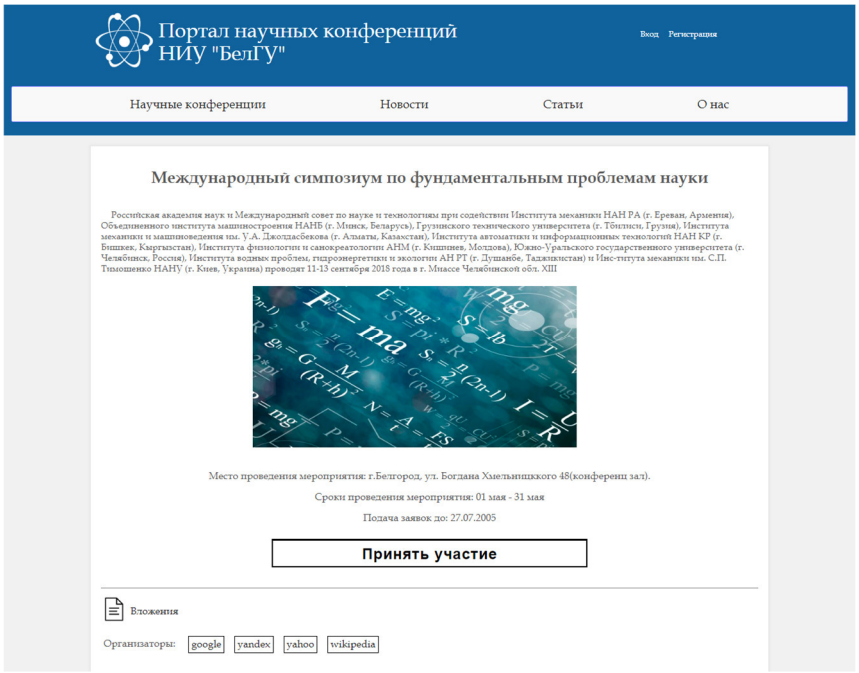


Рисунок 3.9 – Страница выбранного научного мероприятия

Данная страница состоит практически из тех же полей, что и предыдущая. Отличие состоит в измененном дизайне, тексте, который

выгружается из другого поля базы данных. Здесь имеется информация о дате окончания принятия заявок, об организаторах, которые выгружаются из таблицы event_organizer, реализующую связь многое-ко-многому между таблицами event и organizer, файл с вложением, который можно скачать, а также имеется кнопка регистрации на мероприятие.

Стоит отметить, что если пользователь не вошел в систему под своим логином, то он перейдет на страницу авторизации, передав на нее id мероприятия в параметрах. После авторизации этот id позволит вернуться обратно на страницу регистрации на мероприятие.

Если пользователь зарегистрирован, то он перейдет на страницу регистрации на мероприятие (рисунок 3.10). В передаваемых на страницу параметрах – id мероприятия.

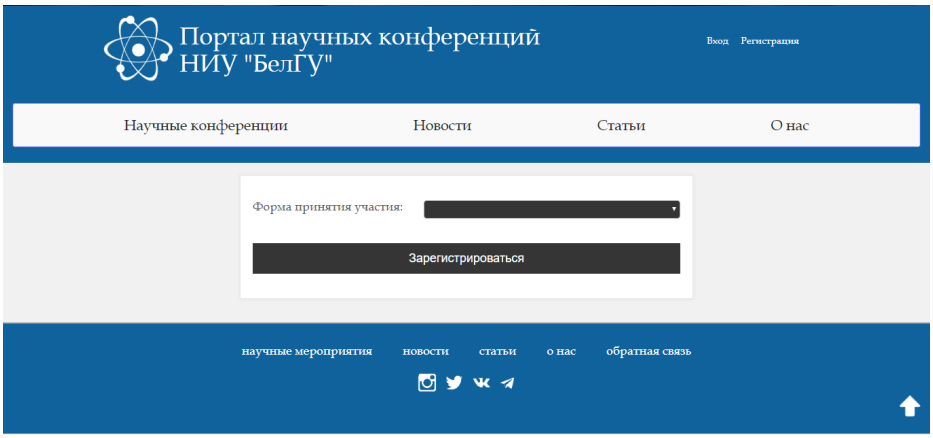


Рисунок 3.10 – Форма регистрации на мероприятие

Бланк раскрывается в дальнейшем в зависимости от введенной формы принятия участия. Далее пользователь выбирает тип принятия участия. Если выбран докладчик, то форма раскрывается полностью, позволяя ввести тему доклада, а также прикрепить его к своей заявке.

Раскрытие формы организовано с помощью языка JavaScript, а также с помощью CSS. JS меняет классы элементов, скрывая или наоборот отображая элементы в зависимости от выбранного значения поля. Полностью раскрытая форма демонстрируется на рисунке 3.11.

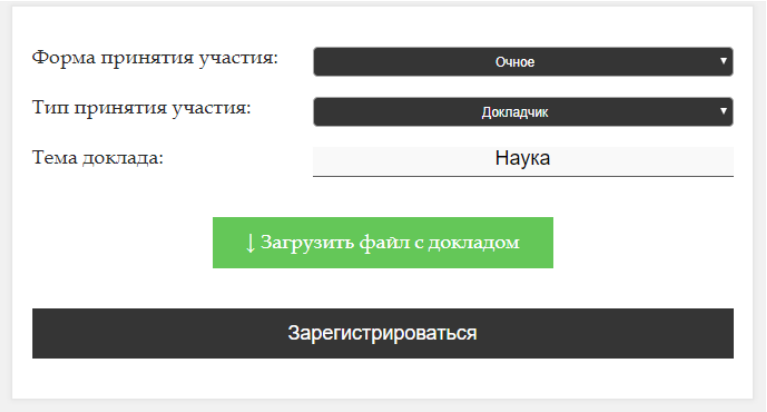


Рисунок 3.11 – Раскрытая форма регистрации

На рисунке 3.12 демонстрируется форма регистрации на портале.

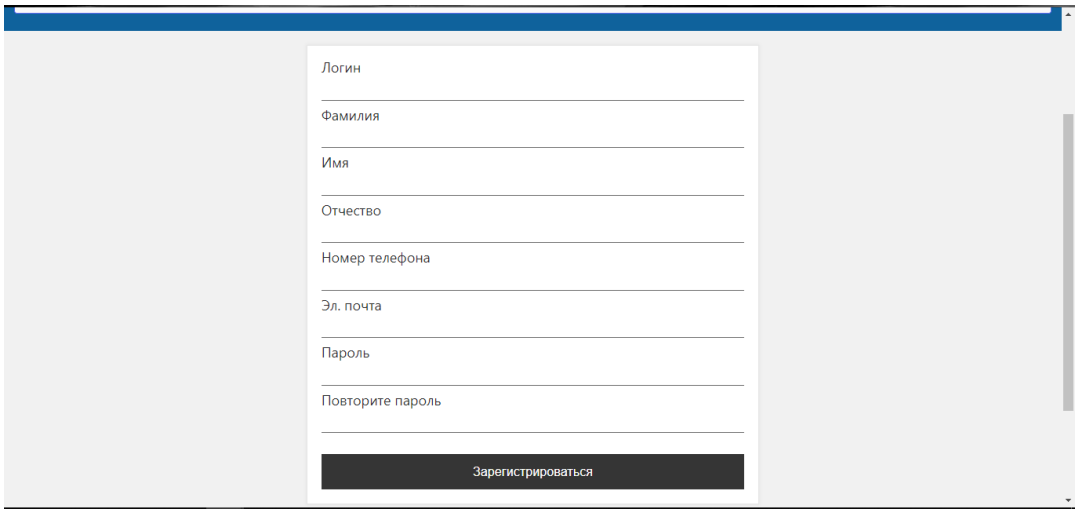


Рисунок 3.12 – Форма регистрации на портале

Поля для регистрации следующие:

– логин;

– фамилия;

– имя;

– отчество;

– номер телефона;

– адрес электронной почты;

– пароль;

– повтор пароля.

Данные поля соответствуют полям таблицы user в базе данных, за исключением поля flag. Данное поле по умолчанию заполняется значением 3 (client). Получение иных прав доступа контролируется администратором. На самой странице ведется контроль заполнения полей. При отправке формы, все поля должны быть заполнены, поле «пароль» и «повтор пароля» должны совпадать. Данные проверки ведутся с помощью языка JavaScript. При некорректном заполнении, пользователю выводится соответствующее сообщение.

Также, поле «логин» не должно повторяться с уже имеющимися в таблице. Данную проверка осуществляется на JSP-странице.

При корректном заполнении и нажатии на кнопку «зарегистрироваться», данные пользователя записываются в базу данных, автоматически генерируя идентификатор записи.

Стоит отметить, что пароль опасно хранить в открытом виде в базе данных. При получении доступа к базе, злоумышленник может воспользоваться паролем пользователя для авторизации под его логином. Кроме того, зачастую, люди используют пароли для множества сайтов одинаковые. Таким образом, злоумышленник получает возможность авторизоваться и на других web-ресурсах, используя личность другого человека.

Для того, чтобы этого избежать, пароли хранятся не в открытом виде, а в виде хеш-кода. Безопасный хеш-код можно получить с помощью множества алгоритмов шифрования. В рамках данной работы был выбран алгоритм SHA-11(Secure Hash Algorithm 1).

SHA-1 – алгоритм криптографического хеширования. Описан в RFC 3174. Для входного сообщения произвольной длины алгоритм генерирует 160-битное хеш-значение, называемое также дайджестом сообщения. Используется во многих криптографических приложениях и протоколах.

SHA-1 реализует хеш-функцию, построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения. Выход представляет собой значение всех хеш-блоков до этого момента. Хеш-значением всего сообщения является выход последнего блока. Одна из итераций алгоритма SHA-1 приведена на рисунке 3.13.

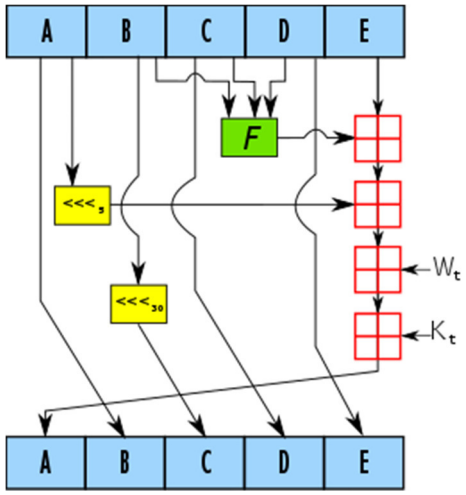


Рисунок 3.13 – Одна итерация алгоритма SHA-1

При прохождении алгоритма, полученный хеш записывается в базу данных. При авторизации, пароль, введенный пользователем, хешируется повторно. Сервер сравнивает полученный хеш с хеш-кодом пользователя. При положительном исходе пользователь авторизуется, при отрицательном – пользователю выводится соответствующее сообщение.

Данный алгоритм предусмотрен для шифрования в одну сторону. Алгоритмического подхода к дешифрованию не существует. Однако, при простых паролях возможен взлом, так как существуют базы данных с хеш-кодами простых паролей и соответствующими им значениями.

На рисунке 3.14 изображена страница авторизации на портале.

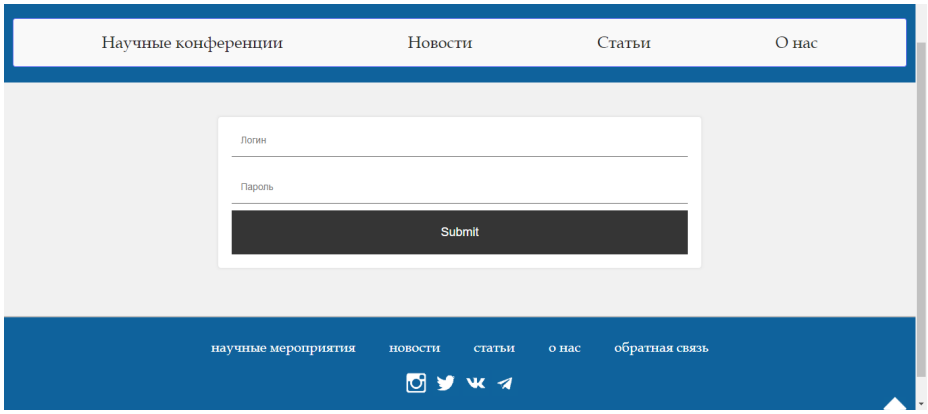


Рисунок 3.14 – Страница авторизации

Форма авторизации обладает типичными полями: логин и пароль. При отправке формы осуществляется поиск по логину. Как было сказано ранее, значение поля «пароль» преобразуется в хеш-код, который сравнивается со значеним соответствующего кода в базе данных. При корректно введенных данных пользователь проходит авторизацию, при некорректных выводится выводится сообщение.

На рисунке 3.15 представлена страница с новостями.

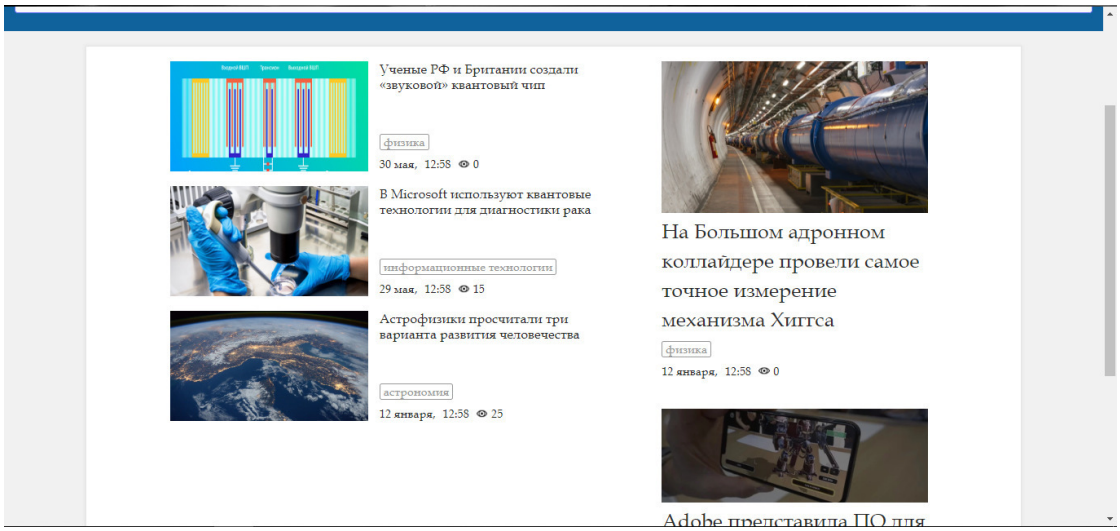


Рисунок 3.15 – Страница новостей

Страница разделена на две части: левая – новости, правая – главные новости. Все отличия между ними заключаются в размере и типе расположения данных. Каждая новость имеет в своем составе заголовок, изображение, тэг, дату и время поста, а также количество просмотров. Все данные выгружаются из базы данных аналогично научным конференциям, из таблицы content. Данная таблица имеет поле type. В зависимости от этого поля, записи выгружаются на разные страницы ресурса: статьи или новости. К тому же, новости подразделяются на малые и большие, что определяется также полем type.

При переходе по ссылке новости, открывается индивидуальная страница новости (рисунок 3.16), на которой располагается заголовок новости, текст и дата создания. В поле с текстом представляется возможным загрузить и изображение, так же, как и на индивидуальной странице научной конференции.

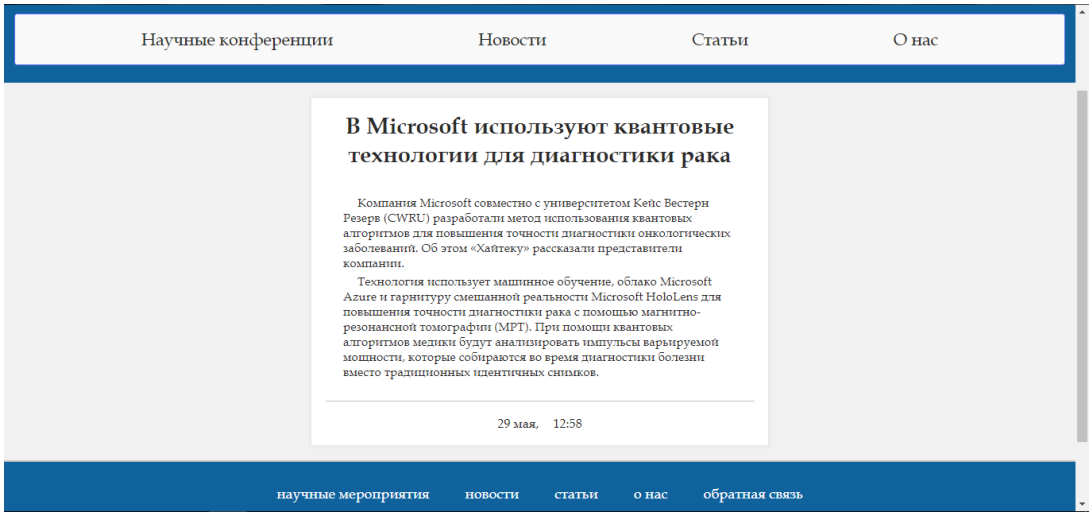


Рисунок 3.16 – Индивидуальная страница новостного сообщения

В данном случае, различия между индивидуальными страницами главной новости и обычной новости отсутствует. Новость также можно скрывать с помощью поля visible.

На рисунке 3.17 располагается изображение страницы со статьями. Данная страница содержит в себе заголовок, текст, изображение, автора, дату и время публикации, а также количество просмотров. Выгрузка осуществляется из базы данных аналогичным образом.

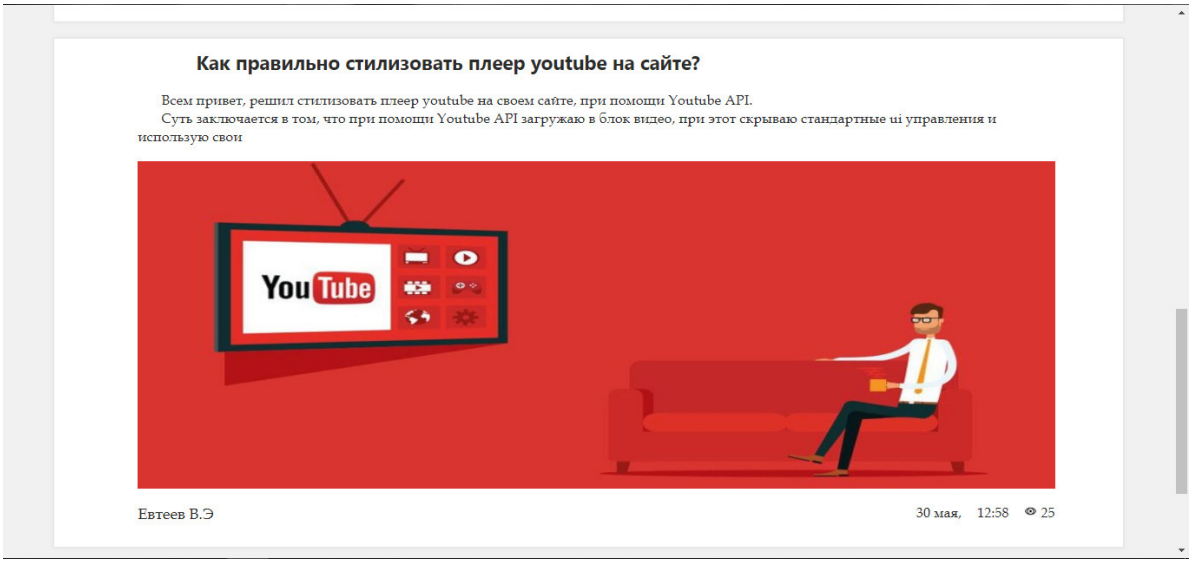


Рисунок 3.17 – Страница просмотра статей

При переходе на индивидуальную страницу статей, оформление остается прежним. Из содержания меняется только текст с preview_text на text.

Следующим этапом проводилась разработка CMS (система управления содержимым). CMS – это программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления содержимым. Данная система требуется для управления содержимым на сайте для пользователя, не обладающего знаниями по языкам разметки и программирования.

На рынке существует множество различных CMS, но в рамках данной работы создана собственная CMS (рисунок 3.18). Ее преимущество состоят в том, что она настроена конкретно под данный проект и не содержит больших объемов дополнительных функций.

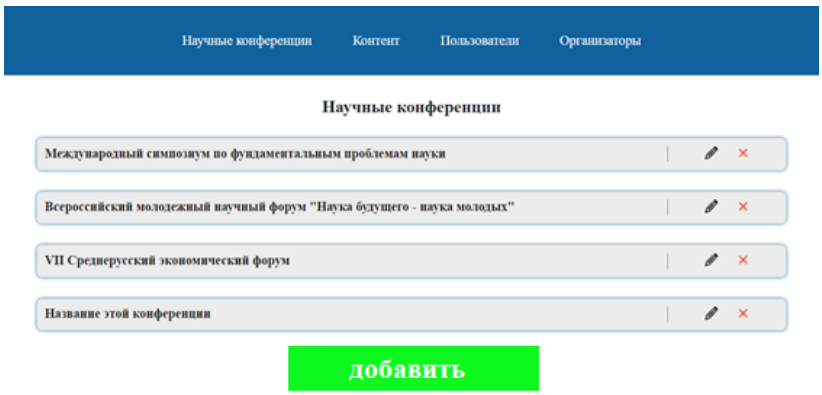


Рисунок 3.18 – CSM администратора

Для того, чтобы воспользоваться CMS, следует пройти по адресу /admin домашнего каталога и быть авторизованным с полномочиями администратора. Меню представляет собой таблицы базы данных, которые предусмотрены для редактирования. При переходе по ссылке, пользователь остается на той же странице, но в запрос передаются параметры, которые определяют, какая из ссылок была нажата. С помощью функции «forvard» содержимое соответствующих запросу страниц передается на страницу администратора.

CMS имеет интуитивный интерфейс: чтобы удалить запись, следует нажать на «крестик», чтобы редактировать – на «карандаш», чтобы добавить – на кнопку «добавить».

На рисунке 3.19 представлена страница редактирования мероприятия.

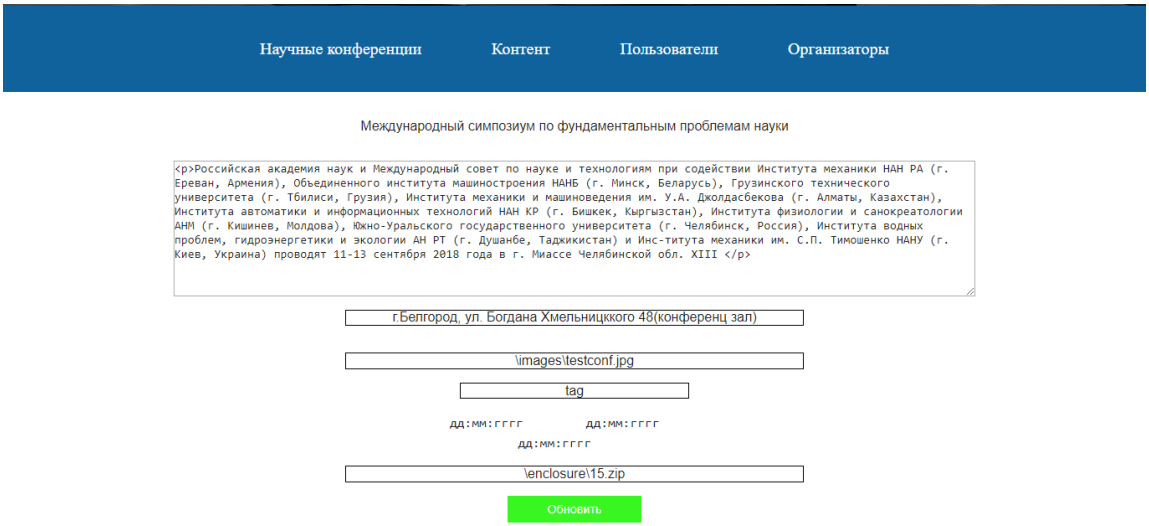


Рисунок 3.19 – Страница редактирования научного мероприятия

При редактировании или добавлении появляется одинаковый интерфейс. Если запись редактируется, то поля заполнены значениями из базы данных, если создается – они пустые.

Также, существует и страница рецензента. При подаче заявки, если пользователь желает выступать, он отправляет документ или статью. Задача рецензента – проверить этот документ и определить: соответствует участник требованиям или нет. Поэтому соответствующая страница необходима рецензенту (рисунок 3.20).

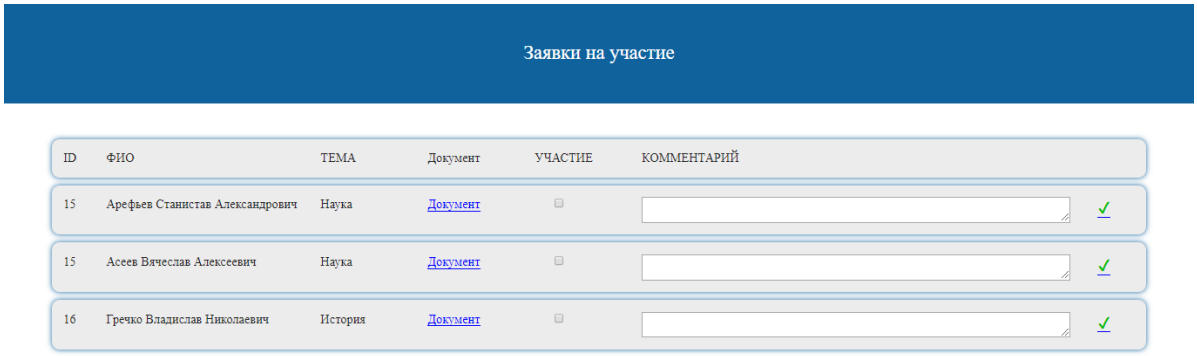


Рисунок 3.20 – Страница управления для рецензента

На данной странице, рецензент может увидеть id мероприятия, ФИО участника, тему его доклада, скачать присланный документ, проверить его и принять решение об участии пользователя. Автоматически в базу данных запишется ФИО рецензента, который поставил отметку о допуске. При нажатии на «галочку», пользователю автоматически отправляется письмо на электронную почту с результатом. Код программного продукта представлен в приложении Б.

3.3 Технические требования

Для сервера минимальными системными требованиями являются:

– процессор: Intel Core 2 Quad 2.20Ghz;

– платформа: 64-разрядная;

– оперативная память: не менее 4Gb;

– жесткий диск: не менее 300Mb свободного объема.

Кроме того, сервер должен иметь следующее программное обеспечение:

– Java EE 7 и выше;

– Java SE 8 и выше;

– Apache Tomcat версии 7 и выше;

– MySQL Server 5.7.22 и выше.

Пользователь должен иметь браузер, поддерживающий:

– HTML5;

– CSS3;

– JavaScript 2.0.

3.4 Обоснование экономической эффективности разработки

Данный программный продукт позволит снизить трудоемкость работ по организации научных конференций. Как следствие, это позволит уменьшить затраты на заработную плату сотрудникам.

В результате внедрения, данная разработка повысит оперативность проведения конференций, улучшит качество организационных моментов, а также снизит до минимума количество ошибок, возникающих при организации мероприятия.

Портал позволяет уменьшить до минимума работу с бумагами, что увеличит продуктивность сотрудников. Кроме того, портал повысит конкурентоспособность. С его помощью увеличится охват аудитории.

Также портал позволит сократить количество задействованных сотрудников за счет автоматизации процесса организации.

Заключительным этапом данной выпускной квалификационной работы является проведение SWOT-анализа (Strengths Weaknesses Opportunities

Threats). SWOT-анализ представляет собой метод стратегического планирования, заключающийся в выявлении факторов внутренней и внешней среды организации и разделении их на четыре категории: возможности, угрозы, сильные и слабые стороны. Сущность SWOT заключается в анализе факторов, оценке рисков и конкурентоспособности продукта в отрасли.

Анализ SWOT проводится в таблице, в которую вносятся сильные стороны продукта, слабые стороны, возможности и угрозы компании. На основании данной информации составлена таблица 3.1, в которой отражен SWOT-анализ программного продукта.

Таблица 3.1 – SWOT-анализ разработанного интернет-портала

Сильные стороны	Возможности			Угрозы	Итого
	Увеличение охвата аудитории	Улучшение доступности ресурсов	Увеличение роста спроса	Уменьшение количества поступающих в ВУЗ студентов	
Портал разработан для известного университета (20 место в топе ВУЗов России)	++	++	+	+	6
Возможность онлайн регистрации на мероприятие	++	++	++	0	6
Автоматическое оповещение о результатах рецензирования	0	+	+	0	2
Автономность	0	++	+	0	3
Лучшее соотношение цены-качества	+	+	+	0	3
Отсутствие сторонней рекламы	+	+	+	0	3
Итого	7	7	8	2	23

Продолжение таблицы 3.1

Слабые стороны					
Новая разработка, нуждающаяся в дополнительной отладке	-	0	0	-	-2
Отсутствие поиска в данной версии	-	0	-	0	-2
Итого	2	0	1	1	-4
Общий итог	5	7	7	1	39

Как видно по данным таблицы, результат общего итога достаточно велик, что показывает эффективность разработанного портала и доказывает целесообразность его создания.

Выводы по третьему разделу

В заключительном разделе представлена реализация этапов разработки интернет-портала для обеспечения проведения научных мероприятий НИУ «БелГУ». Сформулированы системные требования для сервера и для клиента. Проведен анализ экономической эффективности разработки и SWOT-анализ.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы разработан интернет-портал для обеспечения проведения научных мероприятий в НИУ «БелГУ».

На первом этапе был проведен анализ предметной области и рассмотрены существующие решения для обеспечения реализации проекта. В результате этого были определены: примерный перечень элементов, которые должен содержать ресурс; структура интернет-портала; информационное обеспечение порталов научной тематики. Далее были разработаны диаграммы потоков данных и база данных. Сформулированы системные требования для сервера и для клиента. Это позволило определить логику клиент-серверного приложения и его приблизительную структуру.

Следует отметить, что в процессе разработки клиент-серверного приложения использовались следующие инструменты и программные приложения: HTML, CSS, SQL, JavaScript, Java, Intellij idea, MySQL Workbench. Также были применены инструменты для адаптивной верстки интернет-портала.

Разработанный портал выполнен в стилистике НИУ «БелГУ» и системы электронного обучения «Пегас». Как следствие, внедрение не только расширит границы университета во всемирной сети, но и повысит качество проведения научных мероприятий. Портал был создан с возможностью оптимизации. Дальнейшее направление развития: расширение базы данных, улучшение ее структуры, расширение контента портала, оптимизация работы портала.

Поставленные задачи решены, цель выпускной квалификационной работы достигнута.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Студопедия [Электронный ресурс]. / Этапы проведения конференции. Режим доступа: https://studopedia.su/10_127862_etapi-provedeniya-konferentsii.html

2. УчитьсяНа5 [Электронный ресурс]. / Лекция 13. Конференции: виды, формы и правила участия. Режим доступа: http://u4isna5.ru/konspektlekcii/38-putvnauku/156-putvnauku213

3. Cyberpedia Информационный ресурс [Электронный ресурс]. / Виды и специфика научных форумов. Режим доступа: https://cyberpedia.su/12x6c11.html.

4. Конференции.ru [Электронный ресурс]. / Открытый каталог научных конференций, выставок и семинаров. Режим доступа: http://konferencii.ru/

5. Свободная энциклопедия [Электронный ресурс]. / Научная конференция. Режим доступа: https://ru.wikipedia.org/wiki/Научная_конференция

6. Социальная научная сеть [Электронный ресурс]. / Научные конференции в 2018 году. Режим доступа: https://www.science-community.org/ru/conferences

7. Маторин, С.И. Теория систем и системный анализ: Учебное пособие / С.И. Маторин., О.А. Зимовец. – Белгород: Изд-во НИУ «БелГУ», 2012. – 288 с.

8. Когаловский, М. Р. Энциклопедия технологий баз данных/М.Р. Когаловский. – М.: Финансы и статистика, 2002. — 800 с.

9. Научная библиотека избранных естественно-научных изданий. / Проектирование баз данных

10. Гаевский, А. Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript/А.Ю. Гаевский, В.А. Романовский. — М.: Триумф, 2018. — 464 с.

11. Фримен, Э. Изучаем HTML, XHTML и CSS/Э. Фримен. — М.: Питер, 2013. — 973 с.

12. Флэнаган, Д. JavaScript. Подробное руководство/Д. Флэнаган. — М.: Символ-плюс, 2014. —773 с.

13. Программирование на языке Java. — М.: Мультимедиа Технологии и Дистанционное Обучение, 2016. — 416 с.

14. Свободная энциклопедия [Электронный ресурс]. / Apache HTTP Server. Режим доступа: https://ru.wikipedia.org/wiki/Apache_HTTP_Server

15. Oracle [Электронный ресурс]. / Java™ EE at a Glance. Режим доступа: http://www.oracle.com/technetwork/java/javaee/overview/index.html

16. Гупта, А. Java EE 7. Основы/А. Гупта. – М.: Вильямс, 2014. – 336 с.

17. Гонсалвес, Э. Изучаем Java EE 7/Э. Гонсалвес. – М.: Питер, 2016. – 640 с.

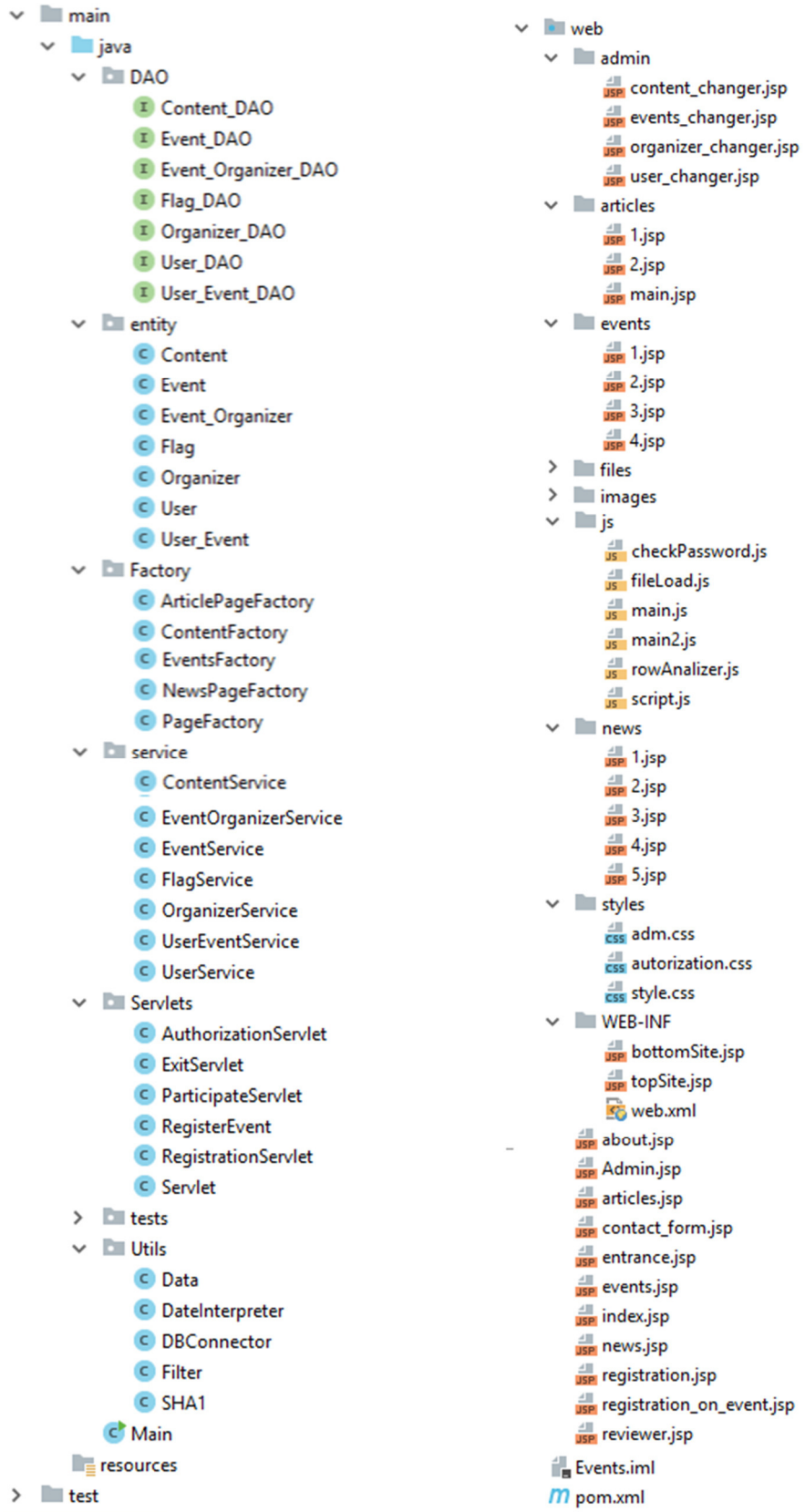
18. Мурат, Й. Java EE. Паттерны проектирования для профессионалов /Й. Мурат. – М.: Питер, 2016. – 976 с.

19. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке С./Б. Шнайер. – М.: Триумф, 2002. – 816 с.

20. Фергюсон, Н. Практическая криптография/Н. Фергюсон, Б.Шнайер. – М.: Диалектика, 2004. – 432 с.

ПРИЛОЖЕНИЕ А

Дерево проекта:



ПРИЛОЖЕНИЕ Б

Листинг программы:

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>Evteev</groupId>
    <artifactId>Events</artifactId>
    <version>1.0</version>

    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>LATEST</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.6</version>
        </dependency>
    </dependencies>
</project>
```

DAO

```
package entity;

public class User {
    private int id;
    private String login;
    private String name;
    private String surname;
    private String patronymic;
    private String password;
    private String phone;
    private String mail;
    private int flag;

    public User(String login, String surname, String name, String patronymic, String
password, String phone, String mail, int flag) {
        this.login = login;
        this.name = name;
        this.surname = surname;
        this.patronymic = patronymic;
        this.password = password;
        this.phone = phone;
        this.mail = mail;
        this.flag = flag;
    }
    public User(int id, String login,String surname, String name,  String patronymic,
String password, String phone, String mail, int flag) {
        this.id = id;
        this.login = login;
```

```
        this.name = name;
        this.surname = surname;
        this.patronymic = patronymic;
        this.password = password;
        this.phone = phone;
        this.mail = mail;
        this.flag = flag;
    }
    public User() { }

    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 getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getPatronymic() {
        return patronymic;
    }

    public void setPatronymic(String patronymic) {
        this.patronymic = patronymic;
    }

    public String getPassword() {
        return password;
    }

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

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getMail() {
        return mail;
    }
```

```
    public void setMail(String mail) {
        this.mail = mail;
    }

    public int getFlag() {
        return flag;
    }

    public void setFlag(int flag) {
        this.flag = flag;
    }

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

        User user = (User) o;

        if (id != user.id) return false;
        if (flag != user.flag) return false;
        if (login != null ? !login.equals(user.login) : user.login != null) return
false;
        if (name != null ? !name.equals(user.name) : user.name != null) return false;
        if (surname != null ? !surname.equals(user.surname) : user.surname != null)
return false;
        if (patronymic != null ? !patronymic.equals(user.patronymic) : user.patronymic
!= null) return false;
        if (password != null ? !password.equals(user.password) : user.password !=
null) return false;
        if (phone != null ? !phone.equals(user.phone) : user.phone != null) return
false;
        return mail != null ? mail.equals(user.mail) : user.mail == null;
    }

    @Override
    public int hashCode() {
        int result = id;
        result = 31 * result + (login != null ? login.hashCode() : 0);
        result = 31 * result + (name != null ? name.hashCode() : 0);
        result = 31 * result + (surname != null ? surname.hashCode() : 0);
        result = 31 * result + (patronymic != null ? patronymic.hashCode() : 0);
        result = 31 * result + (password != null ? password.hashCode() : 0);
        result = 31 * result + (phone != null ? phone.hashCode() : 0);
        result = 31 * result + (mail != null ? mail.hashCode() : 0);
        result = 31 * result + flag;
        return result;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", login='" + login + '\'' +
                ", name='" + name + '\'' +
                ", surname='" + surname + '\'' +
                ", patronymic='" + patronymic + '\'' +
                ", password='" + password + '\'' +
                ", phone='" + phone + '\'' +
                ", mail='" + mail + '\'' +
                ", flag=" + flag +
                '}';
    }
}

public interface User_DAO {
    void add(User user);
    User getById(int id);
    User getByLogin(String login);
    void update(User user);
```

```
    void delete(User user);
    List<User> getAll();
}

public class UserService extends DBConnector implements User_DAO {

    public void add(User user) {
        Connection connection = getConnection();
        PreparedStatement preparedStatement = null;
        String SQL = "INSERT INTO `events`.`user` (`login`, `surname`, `name`,
`patronymic`, `password`, `phone`, `mail`, `flag`) VALUES (?, ?, ?, ?, ?, ?, ?, ?);";
        try {
            preparedStatement = connection.prepareStatement(SQL);
            preparedStatement.setString(1, user.getLogin());
            preparedStatement.setString(2, user.getSurname());
            preparedStatement.setString(3, user.getName());
            preparedStatement.setString(4, user.getPatronymic());
            preparedStatement.setString(5, user.getPassword());
            preparedStatement.setString(6, user.getPhone());
            preparedStatement.setString(7, user.getMail());
            preparedStatement.setInt(8, user.getFlag());
            preparedStatement.executeUpdate();
            System.out.println("Запись добавлена");
        } catch (SQLException e) {
            System.out.println("Ошибка запроса на добавление класса User");
            e.printStackTrace();
        } finally {
            closeConnection(preparedStatement, connection);
        }
    }

    public User getById(int id) {
        Connection connection = getConnection();
        User user = new User();
        String SQL = "SELECT * FROM events.user WHERE id = ?;";
        PreparedStatement preparedStatement = null;
        try {
            preparedStatement = connection.prepareStatement(SQL);
            preparedStatement.setInt(1, id);
            ResultSet resultSet = preparedStatement.executeQuery();
            resultSet.last();
            user.setId(resultSet.getInt("id"));
            user.setLogin(resultSet.getString("login"));
            user.setSurname(resultSet.getString("surname"));
            user.setName(resultSet.getString("name"));
            user.setPatronymic(resultSet.getString("patronymic"));
            user.setPassword(resultSet.getString("password"));
            user.setPhone(resultSet.getString("phone"));
            user.setMail(resultSet.getString("mail"));
            user.setFlag(resultSet.getInt("flag"));
        } catch (SQLException e) {
            System.out.println("Запись не найдена");
        } finally {
            closeConnection(preparedStatement, connection);
        }
        return user;
    }
    public User getByLogin(String login) {
        Connection connection = getConnection();
        User user = new User();
        String SQL = "SELECT * FROM events.user WHERE login = ?;";
        PreparedStatement preparedStatement = null;
        try {
            preparedStatement = connection.prepareStatement(SQL);
            preparedStatement.setString(1, login);
            ResultSet resultSet = preparedStatement.executeQuery();
            resultSet.last();
            user.setId(resultSet.getInt("id"));
            user.setLogin(resultSet.getString("login"));
            user.setSurname(resultSet.getString("surname"));
```

```
            user.setName(resultSet.getString("name"));
            user.setPatronymic(resultSet.getString("patronymic"));
            user.setPassword(resultSet.getString("password"));
            user.setPhone(resultSet.getString("phone"));
            user.setMail(resultSet.getString("mail"));
            user.setFlag(resultSet.getInt("flag"));
        } catch (SQLException e) {
            System.out.println("Запись не найдена");
        } finally {
            closeConnection(preparedStatement, connection);
        }
        return user;
    }

    public void update(User user) {
        Connection connection = getConnection();
        PreparedStatement preparedStatement = null;
        String SQL = "UPDATE `events`.`user` SET `login` = ?, `surname` = ?, `name` =
?, `patronymic` = ?, `password` = ?, `phone` = ?, `mail` = ?, `flag` = ? WHERE (`id` =
?);";
        try {
            preparedStatement = connection.prepareStatement(SQL);
            preparedStatement.setString(1,user.getLogin());
            preparedStatement.setString(2, user.getSurname());
            preparedStatement.setString(3, user.getName());
            preparedStatement.setString(4, user.getPatronymic());
            preparedStatement.setString(5, user.getPassword());
            preparedStatement.setString(6, user.getPhone());
            preparedStatement.setString(7, user.getMail());
            preparedStatement.setInt(8, user.getFlag());
            preparedStatement.setInt(9, user.getId());
            preparedStatement.executeUpdate();
            System.out.println("Запись обновлена");
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            closeConnection(preparedStatement, connection);
        }
    }

    public void delete(User user) {
        Connection connection = getConnection();
        PreparedStatement preparedStatement = null;
        String SQL = "DELETE FROM `events`.`user` WHERE (`id` = ?);";
        try {
            preparedStatement = connection.prepareStatement(SQL);
            preparedStatement.setInt(1, user.getId());
            preparedStatement.executeUpdate();
            System.out.println("Запись удалена");
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            closeConnection(preparedStatement, connection);
        }
    }

    public List<User> getAll() {
        Connection connection = getConnection();
        Statement statement = null;
        String SQL = "SELECT * FROM events.user;";
        User user;
        List<User> users = new ArrayList<User>();
        try {
            statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(SQL);
            while (resultSet.next()) {
                user = new User();
                user.setId(resultSet.getInt("id"));
                user.setLogin(resultSet.getString("login"));
                user.setSurname(resultSet.getString("surname"));
```

```
                user.setName(resultSet.getString("name"));
                user.setPatronymic(resultSet.getString("patronymic"));
                user.setPassword(resultSet.getString("password"));
                user.setPhone(resultSet.getString("phone"));
                user.setMail(resultSet.getString("mail"));
                user.setFlag(resultSet.getInt("flag"));
                users.add(user);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            closeConnection(statement, connection);
        }
        return users;
    }
}
```

Подключение к базе данных

```
public class DBConnector {
    private String DB_CONNECTION_URL =
"jdbc:mysql://localhost:3306/events?useUnicode=true&characterEncoding=utf8";
    private String DB_USER_NAME = "root";
    private String DB_PASSWORD = "root";
    Connection connection = null;

    public Connection getConnection() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection(DB_CONNECTION_URL, DB_USER_NAME,
DB_PASSWORD);
            Statement statement = connection.createStatement();
            statement.execute("set character set utf8");
            statement.execute("set names utf8");
            statement.close();
        } catch (SQLException e) {
            System.out.println("Ошибка подключения базы данных");
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return connection;
    }
    protected void closeConnection(Statement statement, Connection connection) {
        try {
            if (statement != null)
                statement.close();
            if (connection != null)
                connection.close();
        } catch (Exception e) {
            System.out.println("Не удалось закрыть соединение с Базой Данных");
        }
    }
}
```

Создание страниц

```
public class NewsPageFactory {
    private Content content;

    public NewsPageFactory(Content content) {
        this.content = content;
    }

    public void createNewsPage() {
        String path = Data.getCorePath() + content.getHref();
        String finalHTML = createJSP();
        System.out.println(path);
```

```
        File file = new File(path);
        try {
            FileWriter fileWriter = new FileWriter(file);
            fileWriter.write(finalHTML);
            System.out.println("Страница создана");
            fileWriter.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        //System.out.println(finalHTML);
    }

    private String createJSP() {
        String jsp = "<%@ page contentType=\"text/html;charset=UTF-8\"
language=\"java\" %>\n" +
                "<%@include file=\"/WEB-INF/topSite.jsp\"%>\n" +
                "\n" +
                "<main>\n" +
                "    <%\n" +
                "        ContentService contentService = new ContentService();\n" +
                "        Content content =
contentService.getById("+content.getId()+");\n" +
                "        String date =
DateInterpreter.translate(content.getDate_time());\n" +
                "        SimpleDateFormat simpleDateFormat = new
SimpleDateFormat(\"HH:mm\");\n" +
                "        String time =
simpleDateFormat.format(content.getDate_time().getTime());\n" +
                "        request.setAttribute(\"content\", content);\n" +
                "    %>\n" +
                "    <div class=\"news_page container\">\n" +
                "        <h1>${content.title}</h1>\n" +
                "        ${content.text}\n" +
                "        <div class=\"main_news_footer\">\n" +
                "            <ul>\n" +
                "                <li><%=date%>,</li>\n" +
                "                <li><%=time%></li>\n" +
                "            </ul>\n" +
                "        </div>\n" +
                "    </div>\n" +
                "</main>\n" +
                "<%@include file=\"/WEB-INF/bottomSite.jsp\"%>";
        return jsp;
    }

    public Content getContent() {
        return content;
    }

    public void setContent(Content content) {
        this.content = content;
    }
}
```

Jsp-файл регистрации:

```
<%@ page import="entity.*" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@include file="WEB-INF/topSite.jsp" %>

<%!String status;%>
<%
    status = request.getParameter("status");
    if (status != null) {
        if (status.equals("repeat")) {%>
    <script>

    </script>
<%
```

```
        }
    }
%>
<main>
    <div class="container">
        <form class="registration_form" name="form" id="form" method="post"
action="/registrationServlet"
              accept-charset="UTF-8" enctype="application/x-www-form-urlencoded">
            <label for="login">Логин</label>
            <input type="textarea" id="login" name="login" class="input" required>
            <label for="surname">Фамилия</label>
            <input type="textarea" id="Surname" name="surname" class="input" required>
            <label for="Name">Имя</label>
            <input type="textarea" id="Name" name="name" class="input" required>
            <label for="patronymic">Отчество</label>
            <input type="textarea" id="patronymic" name="patronymic" class="input"
required>
            <label for="phone">Номер телефона</label>
            <input type="textarea" id="phone" name="phone" class="input" required>
            <label for="mail">Эл. почта</label>
            <input type="email" id="mail" name="mail" class="input" required>
            <label for="password">Пароль</label>
            <input type="password" id="password" name="password" class="input">
            <label for="repeatPass">Повторите пароль</label>
            <input type="password" id="repeatPass" name="repeatPass" class="input"
required>
            <script src="../js/checkPassword.js"></script>
            <div class="button">
                <button type="submit" onclick="return
checkPass(document.getElementById('form'))">Зарегистрироваться
                </button>
            </div>
        </form>
    </div>
</main>
<%@include file="WEB-INF/bottomSite.jsp" %>
```

Сервлет авторизации

```
@WebServlet("/authorization")
public class AuthorizationServlet extends HttpServlet {
    static String repeatStatus = "repeat";

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        String login = req.getParameter("name");
        String password = req.getParameter("password");
        Pair<String, String> pair = new Pair<String, String>(login, password);
//        outPair(name, password);
        boolean isRightUser = Filter.checkPair(pair);
        if (isRightUser) {
            UserService userService = new UserService();
            User userToSession = userService.getByLogin(pair.getKey());
            HttpSession session = req.getSession();
            session.setAttribute("user", userToSession);
            resp.sendRedirect("/events.jsp");
        } else {
            resp.sendRedirect("/entrance.jsp?status=" + repeatStatus);
        }
    }
}
```