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

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

РАЗРАБОТКА ИНФОРМАЦИОННОГО ПОРТАЛА ДЛЯ СТУДЕНЧЕСКОГО КОНСТРУКТОРСКОГО БЮРО

Выпускная квалификационная работа
обучающегося по направлению подготовки
09.03.02 Информационные системы и технологии
очной формы обучения,
группы 12001508
Сторожко Ольги Андреевны

Научный руководитель
ст. преподаватель
Гуль С.В.

БЕЛГОРОД 2019

РЕФЕРАТ

Разработка информационного портала для студенческого конструкторского бюро – Сторожко Ольга Андреевна, выпускная квалификационная работа бакалавра Белгород, Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 56, включая приложения 78, количество рисунков 59, количество таблиц 4, количество использованных источников 31.

КЛЮЧЕВЫЕ СЛОВА: студенческое конструкторское бюро, СКБ, информационный портал, web-ресурс.

ОБЪЕКТ ИССЛЕДОВАНИЯ: студенческое конструкторское бюро.

ПРЕДМЕТ ИССЛЕДОВАНИЯ: взаимодействие людей с существующими информационными ресурсами СКБ.

ЦЕЛЬ РАБОТЫ: улучшение информационного оборота между СКБ и его участниками за счет автоматизации деятельности студенческих конструкторских бюро с помощью информационного портала.

ЗАДАЧИ ИССЛЕДОВАНИЯ: изучить теоретические аспекты работы студенческого конструкторского бюро; исследовать существующие web-ресурсы, связанные с СКБ; разработать функциональную и информационную модели; реализовать информационный портал; протестировать и произвести отладку приложения.

МЕТОДЫ ИССЛЕДОВАНИЯ: экспериментальный метод, сравнение с существующими данными.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ: в результате работы был разработан информационный портал для студенческого конструкторского бюро.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире электронных технологий, практически невозможно представить компанию, магазин или организацию, которым не требуется обработка некоторого объёма информации. Информацию необходимо, где-то размещать, хранить, редактировать.

Любое научное сообщество, нацеленное на развитие и успех, первоочерёдно нуждается в представительстве, как в жизни, так и в глобальной сети интернет. Наличие собственного web-сайта позволяет привлечь гораздо больше посетителей, представить им не только актуальную и своевременную информацию, но и возможность участия в его деятельности [1]. Студенческое конструкторское бюро (СКБ) не является исключением, и также нуждается в представительстве.

Конструкторское бюро студентов представляет собой сообщество людей, объединённых общими интересами, целями и задачами. Данная структура играет большую роль в жизни начинающих научных деятелей, ведь своевременный толчок, поддержка, помощь и сопровождение в первых и последующих трудах, может помочь выбрать свое направление деятельности, развить таланты и обрести незаменимый опыт.

В связи, с чем возможность доступа к исчерпывающей информации о существующих структурах в городе, является необходимой в первую очередь для представителей молодого поколения, таких как школьники, студенты. С данной задачей может помочь представительство во всемирной паутине.

Одним из видов сайтов является информационный портал. Информационный портал – это крупный web-сайт, задачей которого является предоставление информации для большого количества пользователей по заданной тематике, помощь в её поиске, сортировке, размещении [2].

Актуальность создания информационного портала для студенческого конструкторского бюро обусловлена тем, что при поиске информации у

пользователя возникают трудности: данные отсутствуют, неактуальны, или их недостаточно. Соответственно, пользователю, желающему получить информацию об одном или нескольких конструкторских бюро, необходимо каждый искать отдельно, а для того чтобы стать его участником, требуется лично посетить заведение, что создает трудности.

Объект исследования – студенческое конструкторское бюро.

Предмет исследования – взаимодействие людей с существующими информационными ресурсами СКБ.

Целью выпускной квалификационной работы является улучшение информационного оборота между СКБ и его участниками за счет автоматизации деятельности студенческих конструкторских бюро с помощью информационного портала.

Для достижения поставленной цели, необходимо решить следующие задачи:

– изучить теоретические аспекты работы студенческого конструкторского бюро;

– исследовать существующие web-ресурсы, связанные с СКБ;

– разработать функциональную и информационную модели;

– реализовать информационный портал;

– протестировать и произвести отладку приложения.

Пояснительная записка выполнена на 56 страницах без учета приложения, содержит 59 рисунков, 4 таблицы и 3 приложения.

1 Аналитическая часть

1.1 Общая характеристика студенческого конструкторского бюро

Студенческое конструкторское бюро является неотъемлемой частью научной жизни не только университета, но и города. Оно вносит свой вклад в качественную подготовку молодых специалистов, обеспечивает доступность информации и актуальность новостей о деятельности организации. СКБ представляет собой организацию, в пределах которой происходит ускоренное взаимодействие будущих специалистов и компаний сектора высоких технологий за счёт наличия необходимой инфраструктуры и услуг.

Участниками СКБ могут стать не только студенты среднего и высшего профессионального образования, но также аспиранты, магистранты, инженеры, молодые ученые, и подрастающие поколение – школьники. Таким образом, бюро концентрируют заинтересованных в научной деятельности людей в определенном цифровом поле, что отвечает требованиям современных поколений и технологий [3].

Цели СКБ заключаются в привлечении в научную деятельность как можно больше молодежи, поддержке юных ученых, которые стремятся к самореализации через инновационную деятельность, путем организационной и финансовой поддержки проектов, увеличение числа вовлеченных в научную деятельность молодых специалистов, обеспечения взаимодействия между разными представителями научной деятельности в той или иной области [4].

Задачами студенческих конструкторских бюро являются:

– проведение научных исследований, достижение некоторых результатов на базе исследований;

– подготовка студентов во время учебного процесса, к практической работе по профилю;

– применение навыков, знаний и опыта, полученного во время процесса обучения в ходе научных исследований СКБ;

– подготовка и проведение экспериментов для достижения определённого результата, а также получение и анализ результатов от него в рамках учебного заведения;

– создание площадки для ускоренного взаимодействия будущих специалистов и компаний сектора высоких технологий.

В свою очередь студенческое конструкторское бюро предлагает следующие услуги:

– обеспечение инфраструктурой;

– предоставление помещений и оборудования коллективного пользования: конференц-зала, переговорной комнаты, офисного оборудования;

– содействие участникам СКБ в доступе к исследовательскому оборудованию лабораторий и мастерских и к библиотекам учебного заведения;

– информационные услуги;

– информация о возможности участия в международных конференциях, выставках, форумах;

– консультационные услуги;

– помощь в организации участия в выставках, ярмарках, конференциях;

– содействие в привлечении финансирования по федеральным целевым программам, средств государственных корпораций и частных инвесторов;

– организация встреч с потенциальными заказчиками и инвесторами;

– проведение семинаров и тренингов по управлению проектным циклом, инновационному менеджменту и защите интеллектуальной собственности;

– услуги по обучению и проведению исследований, консалтинговые услуги (юридическое сопровождение, экономическое консультирование, патентование и лицензирование, маркетинг, PR и реклама) [5].

Любое студенческое бюро является одним из элементов инновационной структуры университета, позволяющим молодым и начинающим специалистам получить практический опыт работы. СКБ обеспечивают подготовку студентов и аспирантов на этапе учебного процесса к практической работе в компаниях, производящих наукоёмкую продукцию.

1.2 Анализ существующих аналогов программных средств

Существующие СКБ, к примеру, НИУ «БелГУ» не имеют общей структуры, каждое является самостоятельной организацией, нет каких-то разработанных стандартов, общих для всех.

Как видно на рисунке 1 ниже, при поиске информации о студенческом конструкторском бюро НИУ «БелГУ» – отсутствует единый источник информации, а если зайти на одну из предлагаемых ссылок, то можно найти только упоминания, что в университете такие структуры есть.

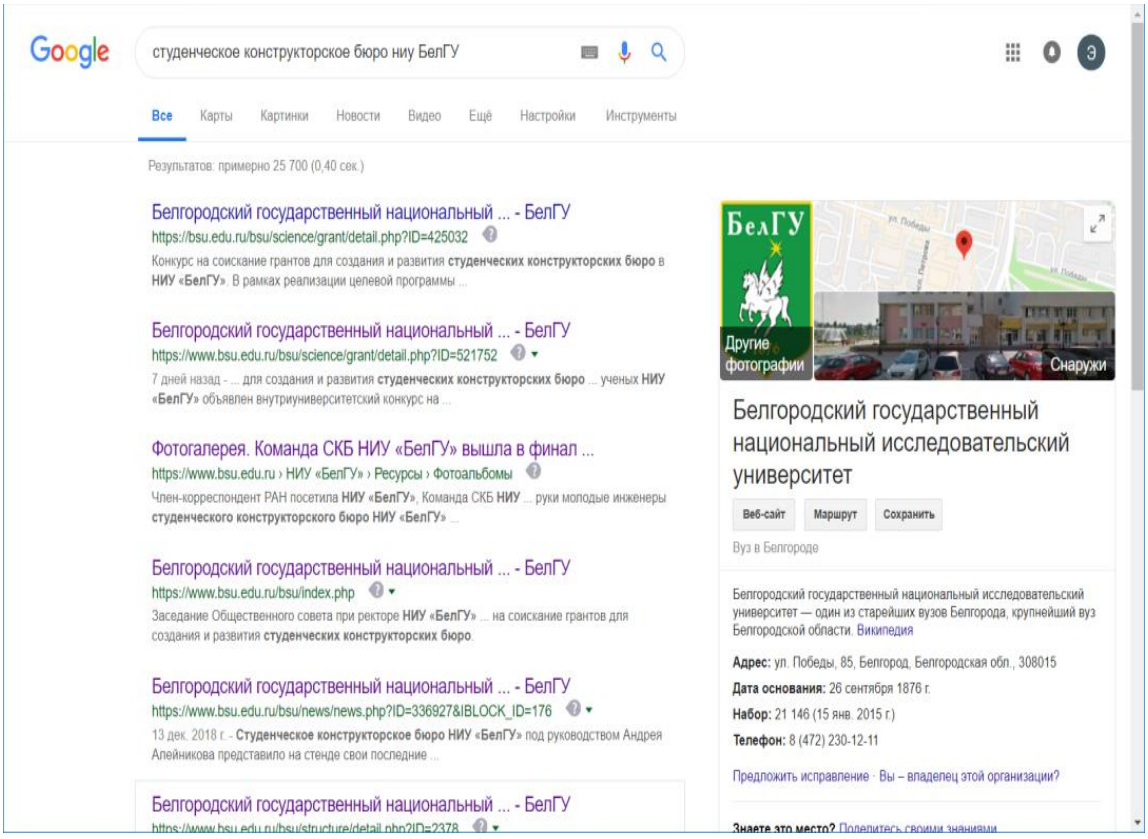


Рисунок 1 – Поиск информации о СКБ НИУ «БелГУ»

Кроме того, если поискать сайты других СКБ, вне Белгорода, то получим весьма скудный результат, информационных ресурсов, связанных со студенческими бюро мало, а те, которые есть, имеют существенные недостатки.

Таблица 1, представленная ниже, демонстрирует основные недостатки существующих ресурсов связанных с СКБ, среди которых – отсутствие online возможности вступления в студенческое конструкторское бюро (необходимо лично посетить организацию для принятия участия в жизни СКБ).

Таблица 1 – Анализ недостатков существующих web-ресурсов, связанных с СКБ

Название	Возможность вступления в СКБ online	Наличие новостей, информации о деятельности	Быстрый поиск	Возможность получения списка участников СКБ	Перечень оборудования СКБ
Технический Ун-т. МО.	Нет	Нет	Нет	Нет	Нет
СПБ Горный Ун-т.	Нет	Есть	Нет	Нет	Нет
ЕГУ им. И.А. Бунина	Нет	Есть	Нет	Нет	Нет
ЮГУ	Нет	Нет	Нет	Нет	Нет

В тоже время, на большинстве ресурсов информация не обновляется, нет разграничения по тематикам, возможности просмотра доступного оборудования, участников конструкторского бюро, а также поиска, в связи с чем, трудно проследить за успехами бюро, и найти актуальную, необходимую информацию.

1.3 Постановка задачи

Проанализировав web-ресурсы глобальной сети интернет, действующих СКБ, было обнаружено, что при поиске сведений о студенческом конструкторском бюро возникают трудности, информационные ресурсы не позволяют получить доступ к исчерпывающей информации и наладить информационный обмен между пользователем и бюро.

Учитывая вышеперечисленные в таблице 1 несовершенства существующих ресурсов, были поставлены следующие задачи, решение которых должен предоставлять информационный портал:

– регистрация/авторизация пользователя;

– разграничение полномочий пользователей;

– создание и обновление личного кабинета пользователя;

– создание новостей, редактирование их;

– добавление новых студенческих конструкторских бюро;

– добавление оборудования для СКБ;

– просмотр расписания оборудования и его бронирование для работы;

– просмотр истории эксплуатации оборудования;

– подача заявление на вступление в бюро, получение прав;

– формирование и выгрузка отчетов о количестве участников СКБ;

– обработка ошибок, в случае неверного ввода данных;

– обработка ошибок, связанных с недостаточными полномочиями пользователя.

Вывод по первому разделу:

В данном разделе выпускной квалификационной работы была рассмотрена предметная область, изучены недостатки существующих web-ресурсов СКБ, на их основе были поставлены задачи, решение которых должен осуществлять информационный портал студенческого конструкторского бюро.

2 Моделирование информационного портала

2.1 Разработка функциональной модели

Создание современных информационных систем представляет собой сложную задачу, решение которой возможно с помощью специальных методик и инструментов. Для реализации структурно-функциональной модели была использована программа AllFusion Process Modeler и методология IDEF0.

AllFusion Process Modeler – это CASE-средство для моделирования бизнес-процессов позволяющие собрать всю необходимую информацию о работе организации и отобразить ее в виде целостной модели [6]. Кроме того по построенной модели легко провести анализ работы предприятия (бюро), проверить его на соответствие стандартам, оптимизировать работу, спроектировать производство, уменьшить издержки, избежать ненужных операций, увеличить производственную эффективность [7].

Особенностями данного программного продукта является возможность создавать диаграммы в нотации IDEF0, IDEF3, DFD, а также во время модерирования свободно переключаться с нотации IDEF0 на любой ветви модели на нотацию IDEF3 или DFD и создать смешанную модель [8].

На рисунке 2 ниже представлена диаграмма, которая отображает предлагаемую технологию решения задачи, описание системы и ее взаимодействие с внешней средой, в приложении А расположены диаграммы потоков данных, демонстрирующих существующую технологию решения задач.

На вход контекстной диаграммы поступает информация о пользователе, его запрос (на вступление в одно из существующих СКБ, на получение различных прав, на использование оборудования, данные пользователя). Управляющими, регламентирующими и нормативными данными, которыми руководствуется работа портала, являются уровни доступа и законы РФ, в

частности предстандарт для сайтов. Механизмами являются администратор портала/пользователь и сам информационный портал. На выход поступает новое расписание работы оборудования, отчеты, информация о СКБ.

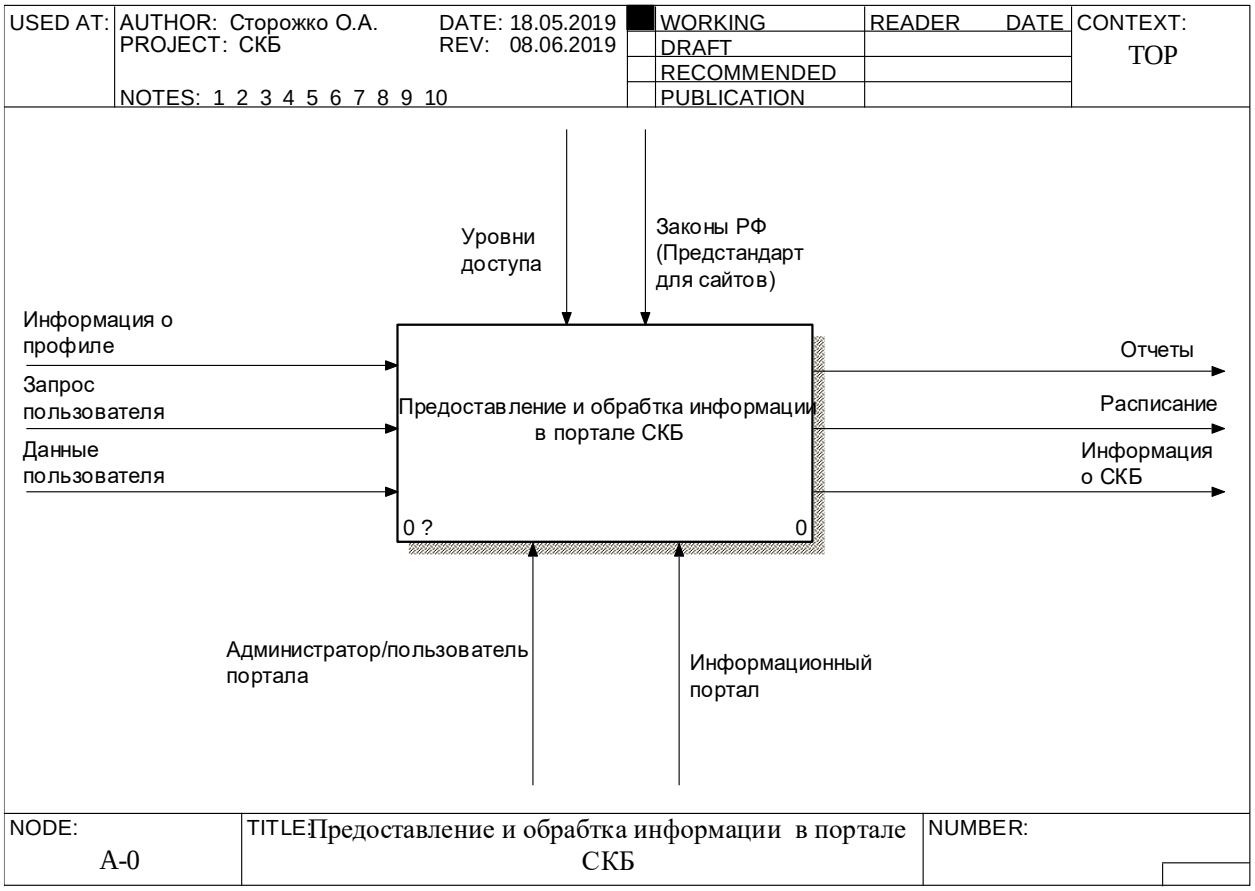


Рисунок 2 – Контекстная диаграмма

На рисунке 3 ниже представлена декомпозиция контекстной диаграммы. которая представляет собой разбиение на четыре основные процесса: авторизация пользователя, подача заявления на вступление/администрирование СКБ, создание нового СКБ, составление отчетов и расписания.

Входными данными для блока «Авторизация/регистрация пользователя» является информация о пользователе, выходными – информация о профиле пользователя, которая является входящими данными для блока «Подача и рассмотрение заявления».

Блок «Подача и рассмотрение заявления» отображает процесс оформления заявления пользователем, его обработку, на вход поступает

повторный запрос пользователя, выходными данными является обновленная информация о пользователе.

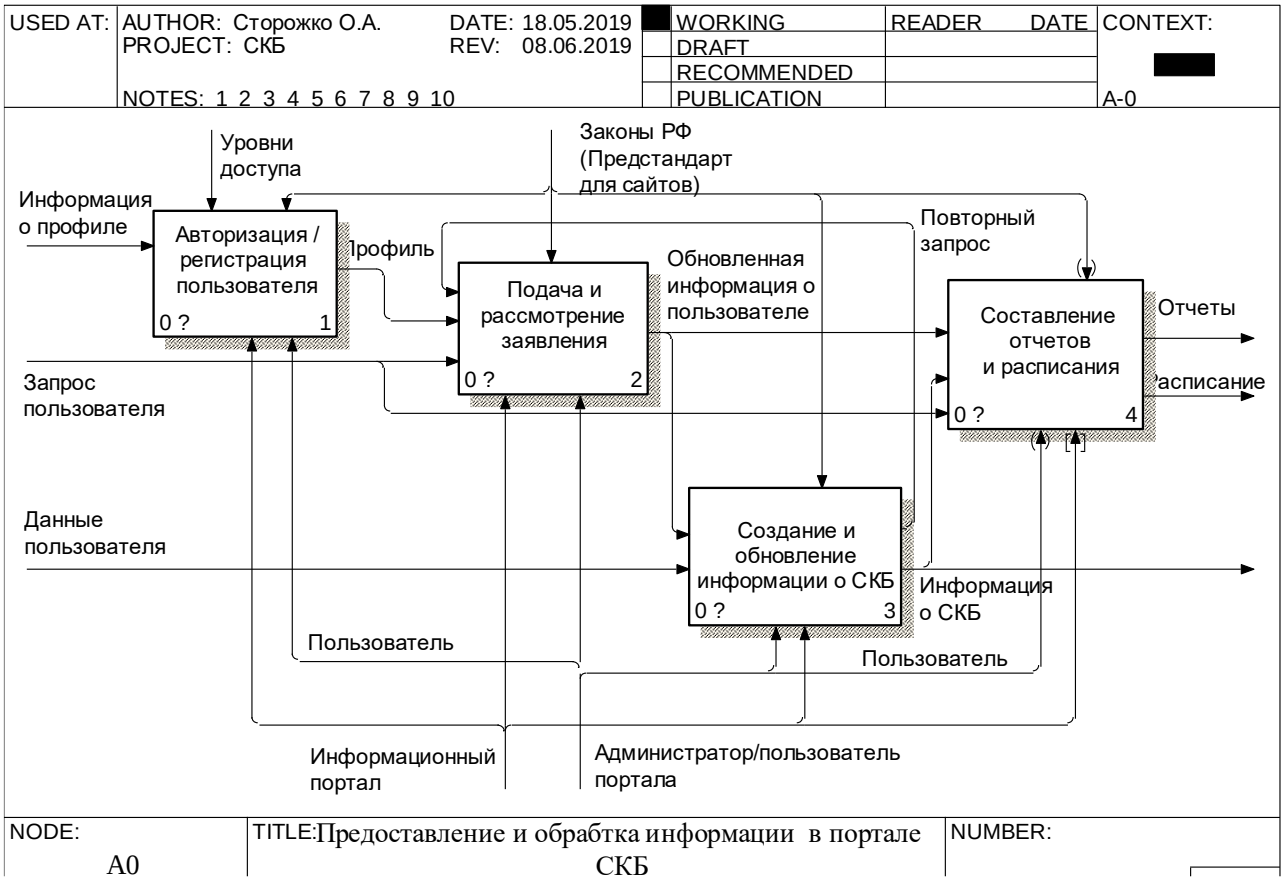


Рисунок 3 – Декомпозиция контекстной диаграммы

В блоке «Создание и обновление информации о СКБ» происходит создание в базе данных нового СКБ, либо обновление, изменение существующей информации. Входящими данными являются данные пользователя и обновленная информация о пользователе. Выходными данными является информация о СКБ.

Обновленная информация о пользователях, информация о СКБ являются входными данным для блока «Составление отчетов и расписания». В нем происходит формирование отчетов о количестве участников, изменение существующего расписания путем запроса пользователя на оборудования. Выходными данными являются отчеты и расписания, при декомпозиции этого блока была использована методология DFD, при описании которой управляющие данные и механизмы не используются, в связи, с чем на

декомпозиции контекстной диаграммы присутствуют туннели у соответствующих стрелок.

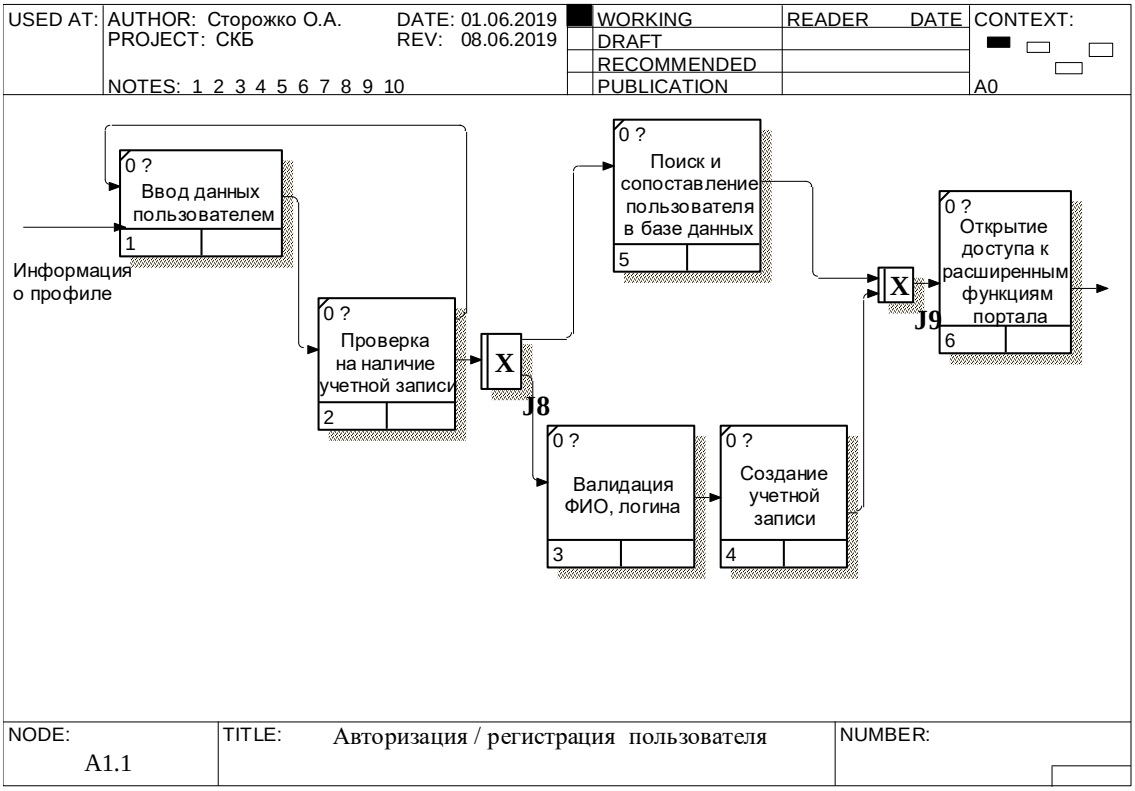


Рисунок 4 – Декомпозиция блока «Авторизация/регистрация пользователя»

На рисунке 4 выше представлена декомпозиция блока «Авторизация пользователя». В данном блоке происходит регистрация или авторизация пользователя, вначале он вводит данные, после происходит их проверка, в случае ввода правильных данных происходит поиск и сопоставление пользователя, если же данные введены не верно, осуществляется их ввод повторно или пользователь проходит регистрацию, после авторизированный пользователь получает доступ к расширенным функциям портала.

На рисунке 5 ниже представлена декомпозиция блока «Подача и рассмотрение заявления», в этом блоке происходит подача заявления пользователем, для получения прав стать участником СКБ, либо администратором. Вначале пользователь выбирает, какие права ему

необходимы, после оформляет заявление, затем происходит его проверка и выдача прав.

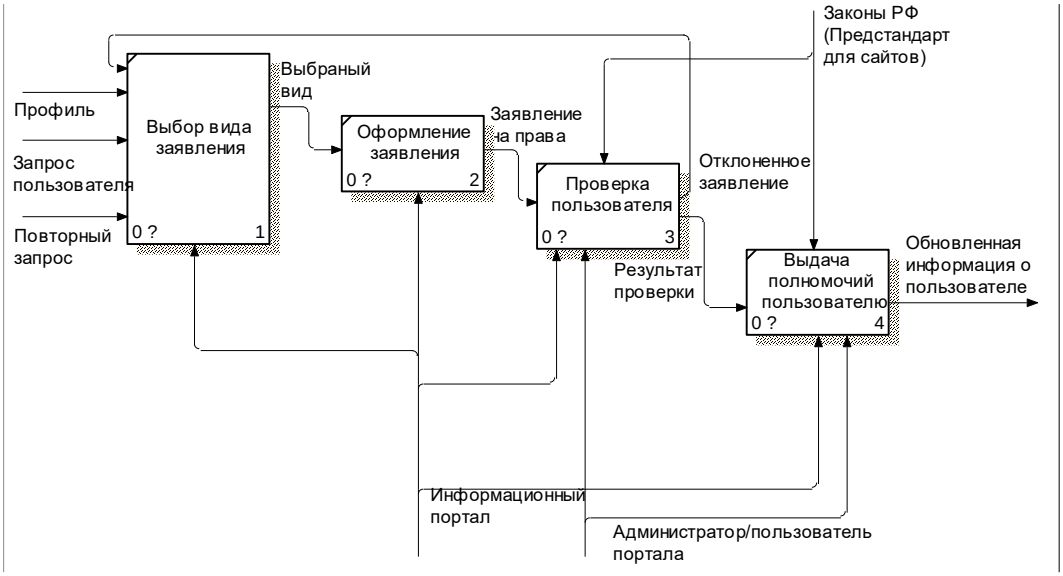


Рисунок 5 – Декомпозиция блока «Проверка запроса пользователя»

Рисунок 6 отображает декомпозицию блока «Создание и обновление информации СКБ», в нем происходит проверка прав пользователя, после пользователь либо создает новое СКБ, либо редактирует информацию ранее созданного СКБ.

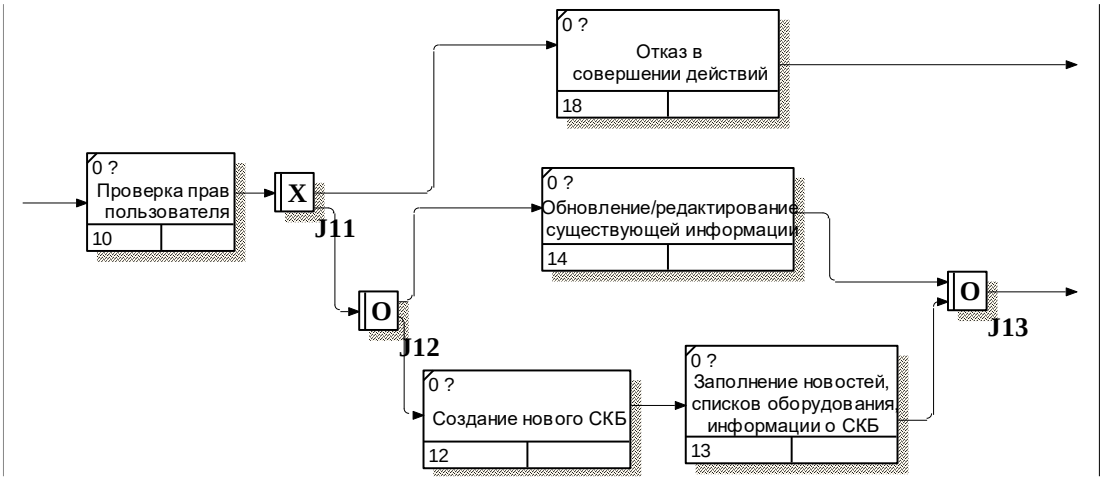


Рисунок 6 – Блок «Создание и обновление информации о СКБ»

На рисунке 7 ниже предоставлена декомпозиция блока «Составление отчетов и расписания».

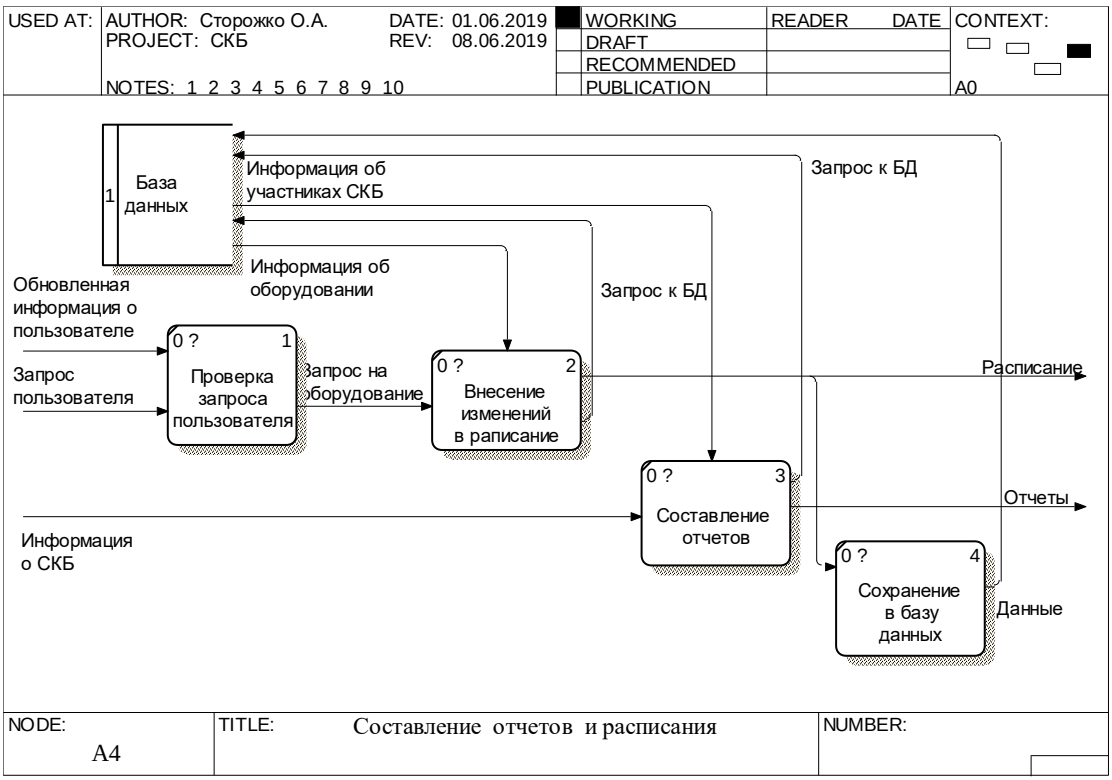


Рисунок 7 – Блок «Составление отчетов и расписания»

В данном блоке происходит проверка запроса пользователя, внесение изменений в расписание, составление отчетов о количестве участников СКБ и сохранение в базу данных.

2.2 Проектирование базы данных

Для построения информационной модели была выбрана программа Navicat Data Modeler, которая представляет собой мощный инструмент для проектирования БД поддерживающий различные СУБД в том числе PostgreSQL. Ключевой функцией программы является обратное проектирование, которое позволяет загружать структуру существующих баз данных и создавать новые ER диаграммы на их основе [9].

ER-диаграмма физического уровня в данной программе включает сущности, атрибуты, первичные ключи, связи между сущностями. Физический уровень демонстрирует целевую СУБД, имена объектов, индексы, типы данных.

На рисунке 8 ниже предоставлена физическая модель информационного портала СКБ.

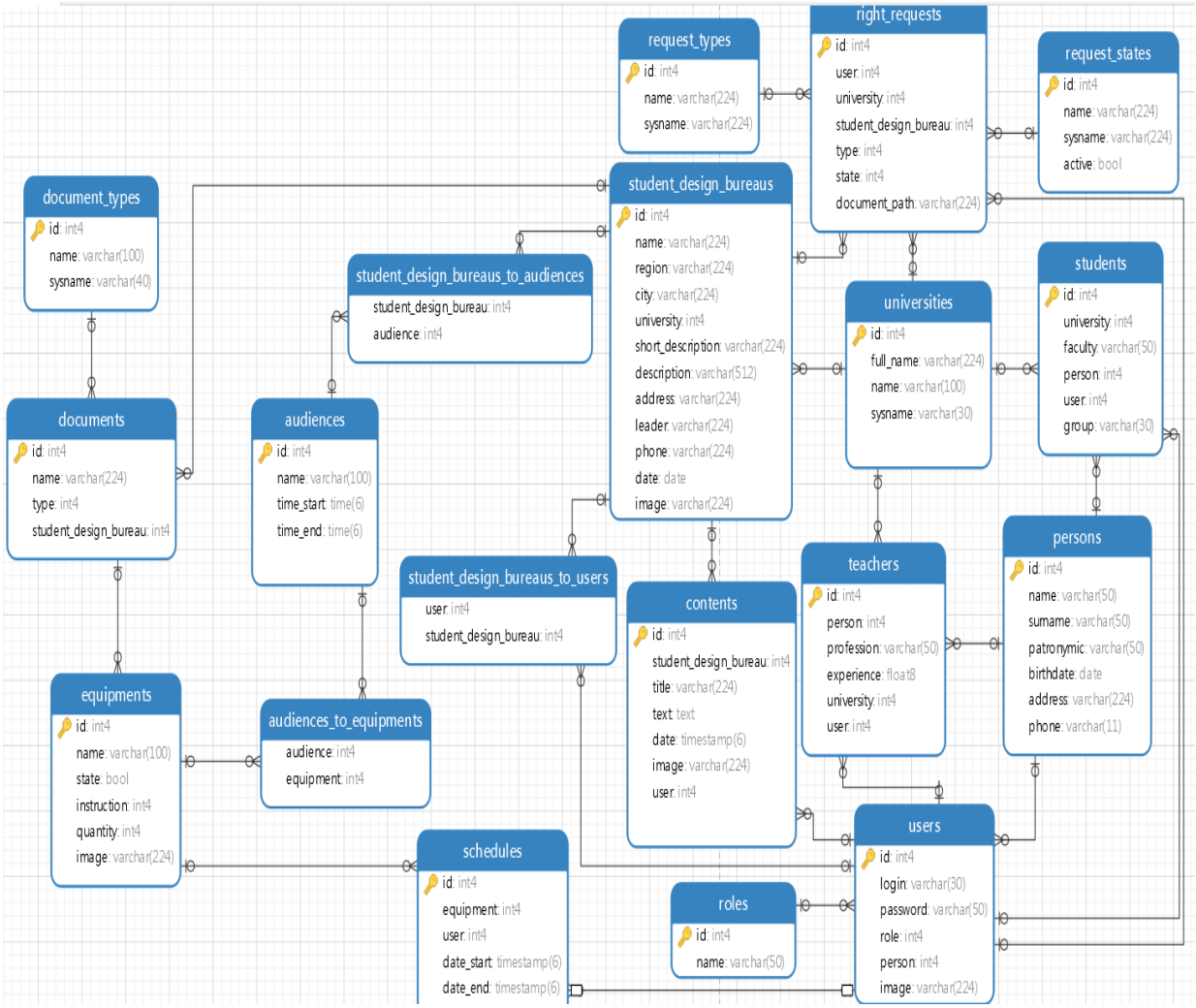


Рисунок 8 – Физическая модель

Главной таблицей в модели является student_design_bureaus, Данная таблица содержит сведенья о наименовании студенческого конструкторского бюро, точном его расположении (область, город, адрес), главе СКБ, дате основания, и фото.

Таблица right_requests необходима для подачи заявления пользователем и содержит данные о статусе, кем совершён запрос, тип запроса, для какого СКБ.

Shedules таблица отображает занятость оборудования, какое есть в наличии у СКБ, кем оно занято и на какой период.

Таблица student_design_bureaus представлена на рисунке 9 ниже.

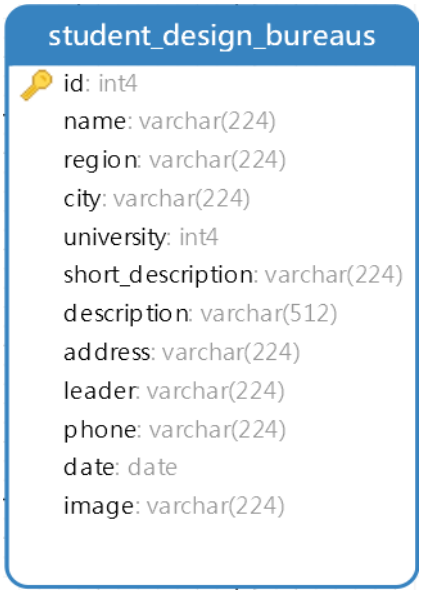


Рисунок 9 – Таблица СКБ

Каждое СКБ имеет свои новости, аудитории, оборудование, расписание, участников и привязку к учебному заведению.

2.3 Выбор программного обеспечения

Для разработки серверной части портала был выбран язык программирования Java, среда разработки IntelliJ IDEA, СУБД PostgreSQL, а также, для удобства работы с базой данных был использован PgAdmin.

Java – это высокоуровневый, объектно-ориентированный язык, особенностью которого является трансляция кода в специальный байт-код, который не зависит от платформы, а после выполняется на Java Virtual Machine в отличии от других языков как PHP, Perl. Данная архитектура обеспечивает кроссплатформенность и аппаратную переносимость [10].

Одной из лучших сред для разработки, поддерживающей Java является IntelliJ IDEA. Она является высокотехнологичным комплексом, который включает интеллектуальный редактор исходных текстов, современные инструменты рефакторинга кода, встроенную поддержку технологий J2EE, механизмы интеграции со средой тестирования Ant/JUnit и системами управления версиями, уникальный инструмент оптимизации и проверки кода

Code Inspection, а также инновационный визуальный конструктор графических интерфейсов [11].

Данные технологии помогают избавить разработчика от рутины, быстро устранить ошибки, улучшить качество кода, повысить продуктивность и производительность программиста, а также подходят для создания различных приложений, включая веб-приложения[12].

Для создания базы данных была выбрана объектно-реляционной СУБД PostgreSQL, которая поддерживает пользовательские объекты, их поведение, функции, операции, индексы, домены и типы данных, за счет чего становится очень гибким и надёжным средством [13]. Данная программа использует язык PL/pgSQL, позволяющий группировать блок вычислений и последовательность запросов внутри сервера БД, в результате чего пропадает необходимость в дополнительном взаимодействии между сервером и клиентом, передаче промежуточных ненужных результатов [14].

Процедурный язык используется для:

– создания новых функций и триггеров;

– добавления управляющих структур к языку SQL;

– выполнения сложных вычислений;

– наследования всех пользовательских типов, функций и операторов[15].

Кроме того PL/pgSQL предоставляет возможность избежать многочисленных разборов одного запроса. Что в итоге приводит к значительному увеличению производительности по сравнению с приложением, которое не использует хранимых функций.

Также для работы с базой данных был использован pgAdmin – это кроссплатформенное приложение, предоставляющее графический интерфейс для работы с БД, через который можно создавать, удалять, редактировать базы данных, а также управлять ими [16].

Создание интуитивно понятного и удобного интерфейса для портала является очень важным, для его создания были выбраны HTML, CSS, JavaScript.

HTML – специальный язык разметки документов в сети интернет, большая часть web-страниц содержит описание разметки на данном языке, сам язык интерпретируется, а полученный текст отображается на мониторе пользователя [17].

CSS – в свою очередь является языком, формально описывающим внешний вид документа, написанного с HTML. Применяется для назначения шрифтов, цветовых гамм, расположения и т.д., с его помощью можно представить один документ в разных стилях или методах вывода [18].

JavaScript (JS) – язык программирования, который зачастую используется как встроенный язык для доступа к объектам приложения, в браузере они подключаются сразу к HTML, в связи с чем выполнение скриптов происходит сразу при загрузке web-страницы [19]. С помощью JS можно создавать, удалять, скрывать, изменять стили элементов, создавать (осуществлять) реакцию на действия посетителя, обрабатывать перемещения курсора мыши, нажатие клавиш, выводить сообщения, отправлять запросы на сервера и загружать данные без перезагрузки страницы и т.д. [20].

Вывод по второму разделу

В данном разделе выпускной квалификационной работы были разработаны функциональная и информационная модели для информационного портала студенческого конструкторского бюро, а также выбраны программные средства и языки программирования для реализации портала.

3 Реализация информационного портала

3.1 Разработка базы данных

Для хранения информации информационного портала в СУБД PostgreSQL была создана база данных, которая состоит из 19 таблиц, таблица 2 ниже отображает созданные таблицы в базе данных.

Таблица 2 – Список таблиц базы данных

Название таблицы	Описание таблицы
audiences	Содержит данные об аудиториях.
contents	Содержит новости связанные с СКБ.
document_types	Содержит сведения о типах документов.
documents	Содержит документы, связанные с СКБ.
equipments	Перечень оборудования СКБ.
persons	Содержит общие сведения о человеке походящие как студенту, так и преподавателю.
request_states	Содержит информацию о статусах запроса.
request_types	Содержит информацию о типах запроса.
right_ requests	Содержит информацию о запросах.
roles	Содержит информацию о ролях, используемых в портале.
schedules	Перечень доступного оборудования.

Продолжение таблицы 2

Название таблицы	Описание таблицы
student_design_bureaus	Содержит информацию о студенческом конструкторском бюро.
students	Содержит информацию о студенте
teachers	Содержит информацию о преподавателе.
universities	Содержит информацию о заведении.
users	Содержит информацию о пользователе.
student_design_bureaus_to_audiences	Связующая таблица для СКБ и аудиторий.
student_design_bureaus_to_users	Связующая таблица для СКБ и пользователей.
audiences_to_equipments	Связующая таблица для аудиторий и оборудования.

Структура таблицы student_design_bureaus (студенческое конструкторское бюро) представлена в таблице 3.

Таблица 3 – Структура таблицы student_design_bureaus

Имя поля	Ключ	Тип, дина	Обязательность значения
id		serial	not null
name		varchar(224)	
region		varchar(224)	
city		varchar(224)	
university		integer	
short_description		varchar(224)	

Продолжение таблицы 3

Имя поля	Ключ	Тип, дина	Обязательность значения
description		varchar(512)	
address		varchar(224)	
leader		varchar(224)	
phone		varchar(224)	
date		date	
image		varchar(224)	

Создание таблицы student_design_bureaus было осуществлено с помощью кода представленного на рисунке 10 ниже.

```
CREATE TABLE student_design_bureaus(id SERIAL PRIMARY KEY,
name VARCHAR(224),region VARCHAR(224), city VARCHAR(224),
university INTEGER,short_description VARCHAR(224),
  description VARCHAR(512), address VARCHAR(224),
  leader VARCHAR(224), phone VARCHAR(224), "date" DATE,
  image VARCHAR(224));

ALTER TABLE student_design_bureaus ADD CONSTRAINT   student_design_bureau_to_university_fkey
FOREIGN KEY (university)  REFERENCES universities (id);

insert into  student_design_bureaus (name, region ,city, university, short_description,
 description, address, leader, phone, date) values ('СКБ ЮИ', 'Белгородская область',
 'Белгород', 1, 'Юный инженер, проектируем вместе!!', 'Студенческое конструкторское
 бюро "Юный инженер" представляет собой сообщество людей, объединённых едиными целями,
стремящихся к развитию своих навыков проектирования.',
 'г. Белгород, ул. Студенческая 14', 'Иванов И.И.', '88287336201',  '01/02/2016');
```

Рисунок 10 – Создание таблицы СКБ

При создании таблицы был задан первичный ключ (PRIMARY KEY), который используется для уникальной идентификации записи таблицы, кроме того, ключу был задан тип данных serial выполняющий функцию автоинкремента.

После необходимо было задать внешний ключ (FOREIGN KEY), что бы связать между собой несколько таблиц, кроме того после созданную таблицу необходимо было заполнить, заполнение таблицы возможно осуществить используя pgAdmin.

Для всех остальных таблиц создание структуры, первичных, вторичных ключей, ввод начальных данных было выполнено по аналогии с таблицей student_design_bureaus, листинг кода их создания расположен в приложении Б.

3.2 Разработка информационного портала

Первым шагом при разработке информационного портала было создание нового Gradle проекта в IntelliJ IDEA 2018. Gradle является системой автоматической сборки, разработанной для расширяемых многопроектных сборок, умеет определять какие составляющие дерева сборки не были подвержены изменению и какие зависимые задачи не требуют перезапуска. Большинство плагинов предназначено для работы с Java, Groovy[21].

Структура Gradle проекта включает в себя место для хранения всех исходных кодов java, файла build.gradle, используемого для построения проекта и объявления библиотек, а также, файла с настройками settings.gradle.

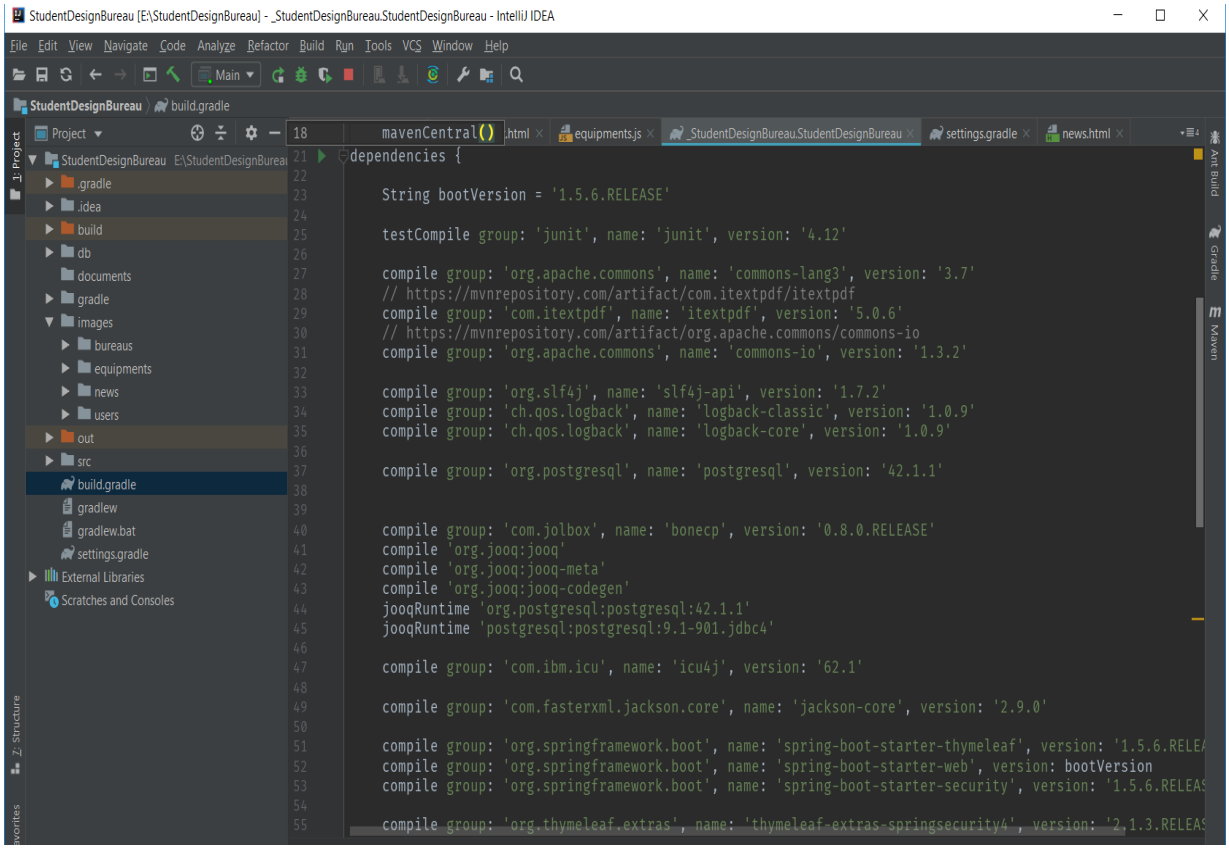


Рисунок 11 – Структура файла build.gradle

На рисунке 11 выше представлено содержание файла build.gradle, в котором dependencies описывают сторонние библиотеки необходимые во время работы проекта и автоматически закачивает их с Java репозитория.

Одной из описанных зависимостей является Spring boot, которая представляет собой инструмент для быстрой настройки разрабатываемого приложения, с помощью данной утилиты встраивается web-сервер в приложение, становится доступна функция для запуска проекта как jar-файла, а разработчик может указывать Spring, какие из модулей необходимо использовать [22].

Также была описана зависимость «spring-boot-starter-security», позволяющая сделать ограничения в доступе на некоторые функции портала пользователю и «spring-boot-starter-web», которая обеспечивает поддержку web-разработки с полным стеком, включая Tomcat (используется как отдельный web-сервер и позволяет запускать web-приложения) [23].

С помощью Model-View-Controller (MVC) архитектуры было описано строение приложения, на рисунке 12 ниже продемонстрировано схематическое отображение данной архитектуры.

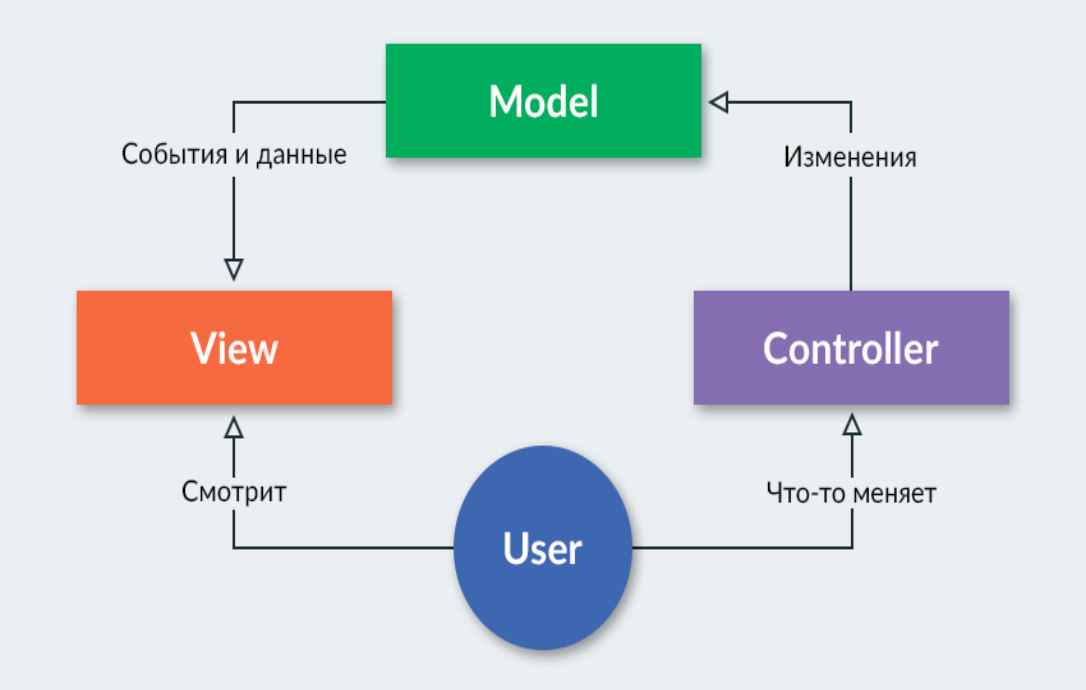


Рисунок 12 – Структура архитектуры MVC

Принцип построения заключается в разбиении приложения на три части - модели(model), виды(view), контроллеры(controllers); первая отображают бизнес-логику приложения, вторая отвечает за отображение пользовательских данных, третья обрабатывает действия пользователя (реагирует на действие пользователя и меняет модель) [24]. Сначала были созданы модели, которым соответствуют таблицы из БД, пример создания модели аудитории показан на рисунке 13.

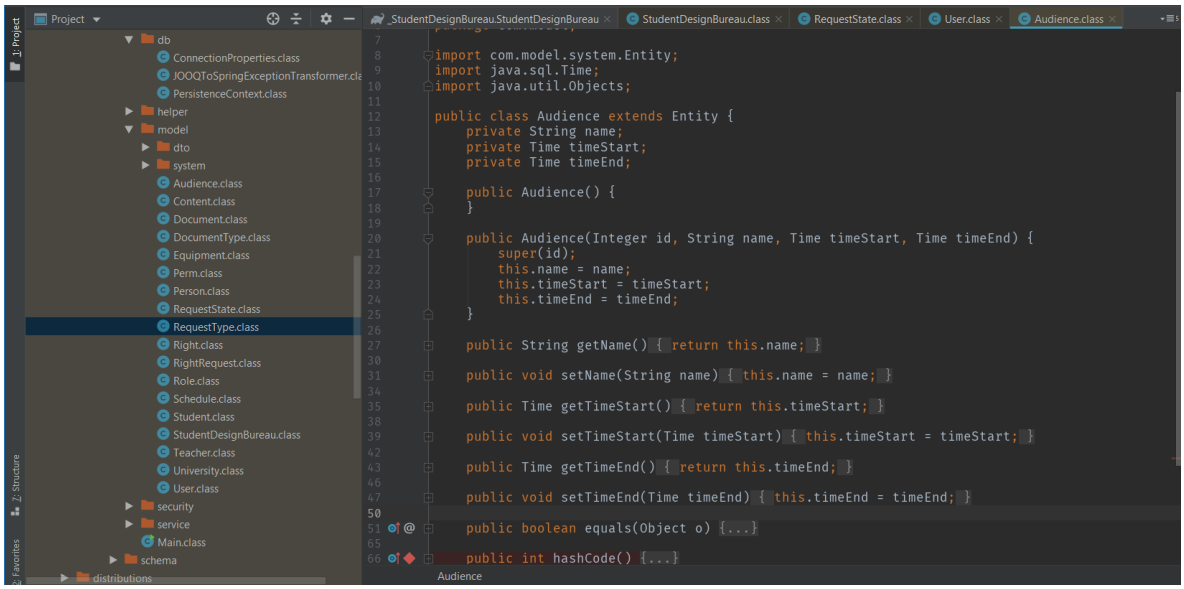


Рисунок 13 – Пример создания модели

Аналогично были созданы модели для всех существующих таблиц. Для организации доступа к базе данных необходимо было создать соединение между приложением и базой данных в PostgresSQL, для этого в файле build.gradle была прописана новая зависимость, рисунок 14 ниже.

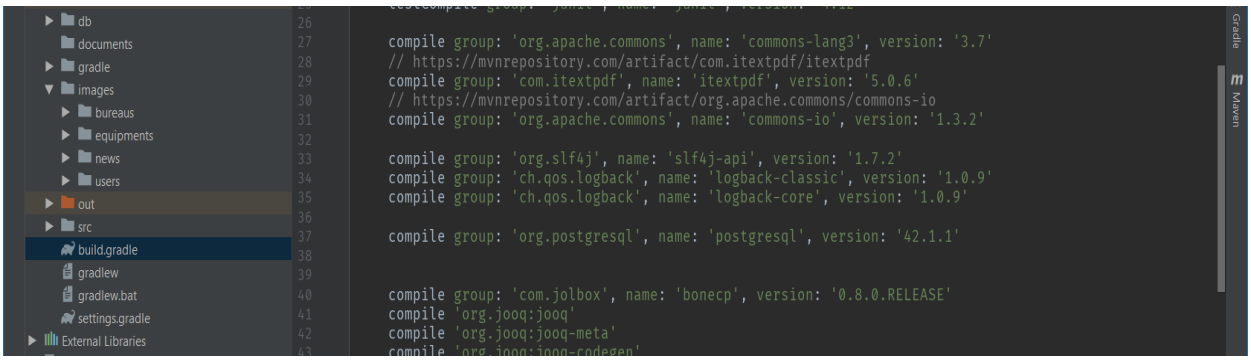


Рисунок 14 – Создание зависимости для подключения к БД

Кроме того, для построения SQL запросов в Java коде была задана зависимость для типизированного построителя Jooq показанная на рисунке 14 ниже, позволяющая парсить созданную БД и создавать необходимые Java-классы.

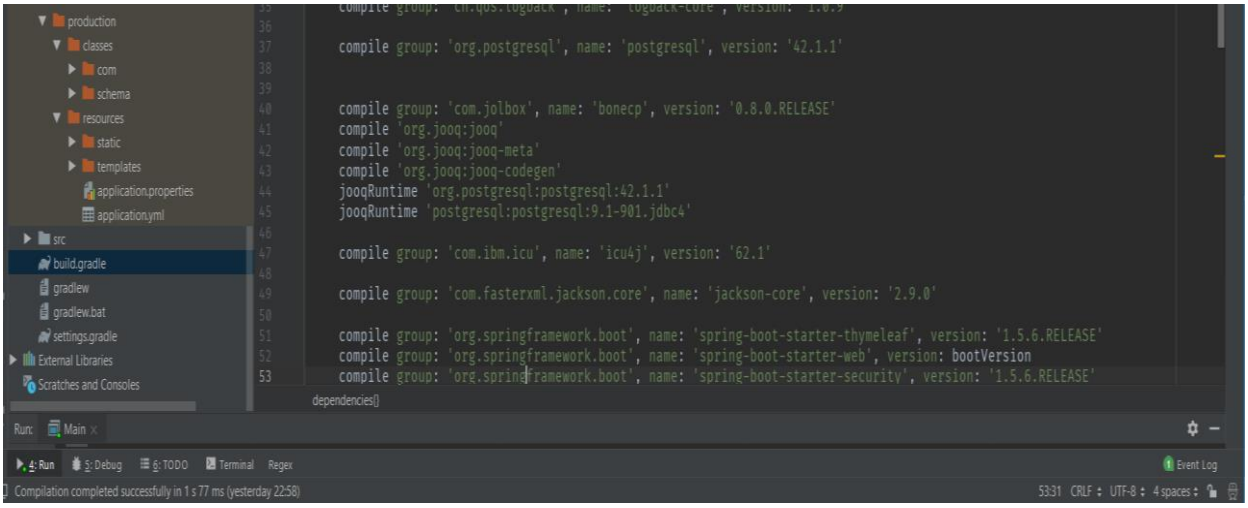


Рисунок 15 – Создание зависимости для подключения к БД

Для получения доступа к имеющимся таблицам в базе данных требовалось создать шаблон Data Access Object (DAO) – используется для соединения реляционной модели и объектной. На рисунке 16 ниже показано создание DAO для аудитории, аналогично соединение было создано и для других моделей.

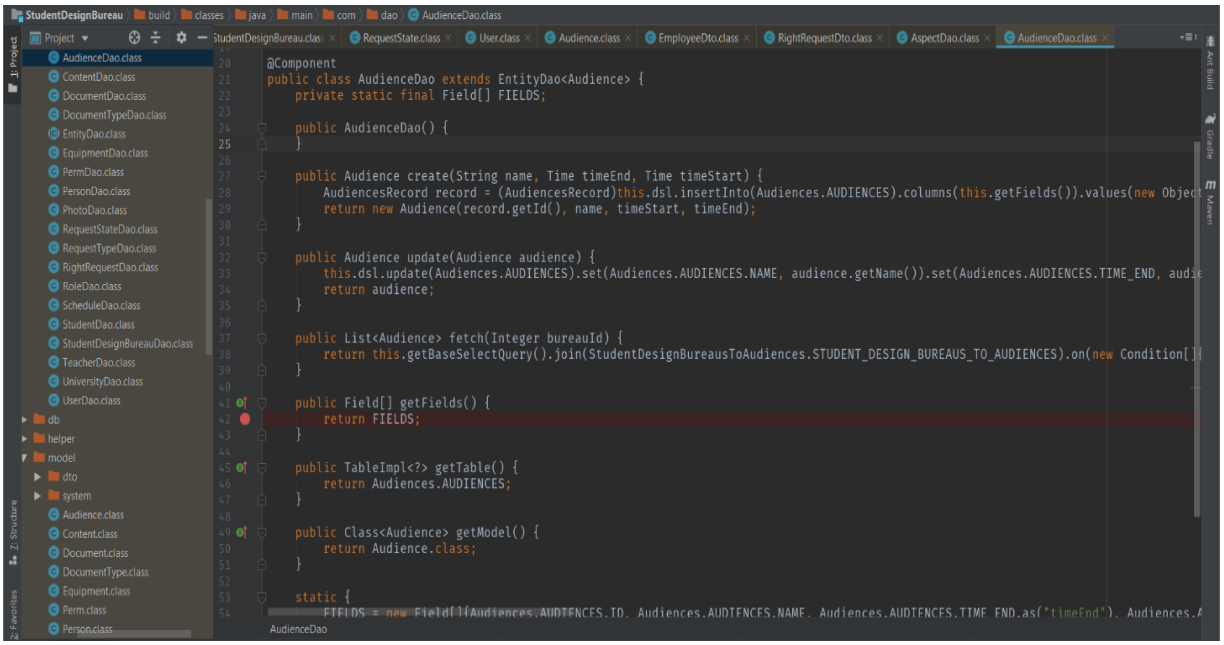


Рисунок 16 – Создание DAO для аудитории

Для автоматического соединения с базой данных, необходимо было в файле application. properties прописать логин, пароль и url базы данных, структура файла показана на рисунке 17 ниже.

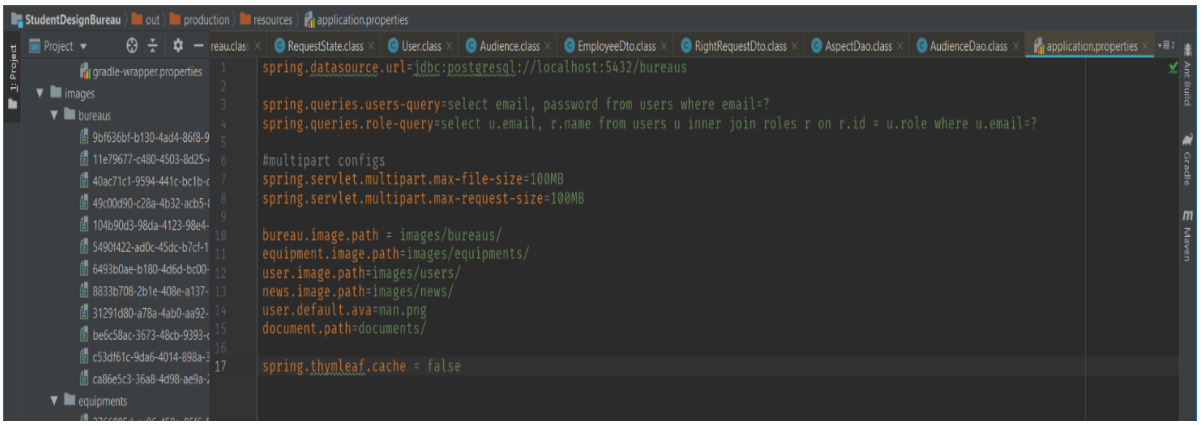


Рисунок 17 – Файл application. Properties с заданными значениями

В дальнейшем фреймворк будет использовать эти данные для подключения к базе. Большинство базовых операций, таких как сохранение, удаление данных – реализовано в Spring, однако недостающие методы можно дописать самостоятельно [25].

Следующим шагом после установки связи с базой данных является создание сервисов. Для всех классов был разработан собственный интерфейс, описывающий поведение класса, пример создания сервиса для аудитории показан на рисунке 18 ниже.

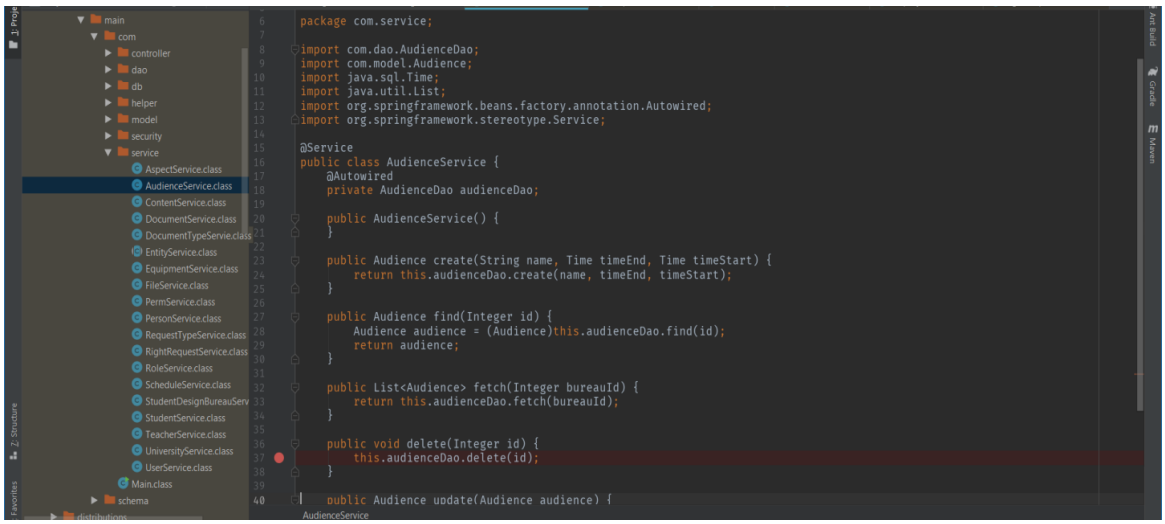


Рисунок 18 – Создание сервиса для аудитории

Заключающим этапом реализации серверной части являлось создание классов контроллеров, они позволяют обрабатывать события, ввод/вывод и любые действия пользователей, создание класса Controller для аудитории показано на рисунке 19 ниже.

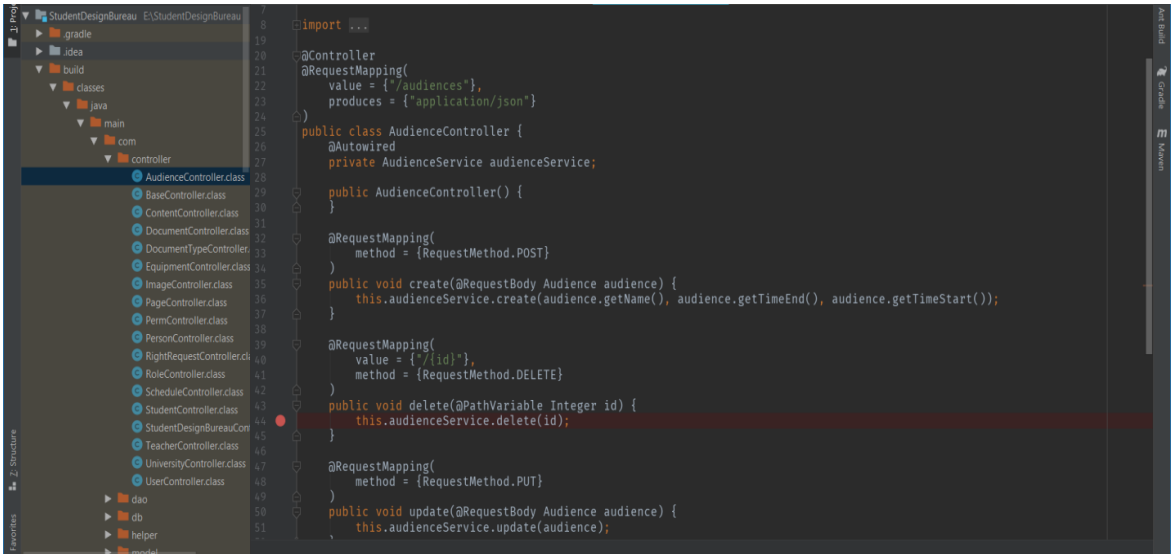


Рисунок 19 – Создание контроллера для аудитории

В качестве построителя HTML кода был использован Thymeleaf – серверный механизм Java-шаблонов, возможности которого позволяют обрабатывать HTML, XML, JavaScript, CSS и даже простой текст [26]. Также был прописан в файле build.gradle в зависимостях. Использование данного механизма показано на рисунке 20 ниже.

Рисунок 20 – Создание HTML при помощи Thymeleaf

Листинг кода, отображающий создание всех вышеописанных объектов находиться в приложении В.

3.3 Разработка интерфейса информационного портала

Одной из важнейших составляющих любого web-ресурса является его интерфейс, в частности от того насколько привлекателен, удобен, прост в использовании и соответствует поставленным целям сайт зависит не только посещаемость ресурса, но и его эффективность [27].

На первых этапах разработки была спроектирована главная страница сайта, показанная на рисунке 21 ниже. На ней отображены ранее созданные в информационном портале студенческие конструкторские бюро и краткая информация о них, так же в правом верхнем углу закреплена иконка для авторизации пользователя, а левее от нее кнопка создания нового СКБ в портале.

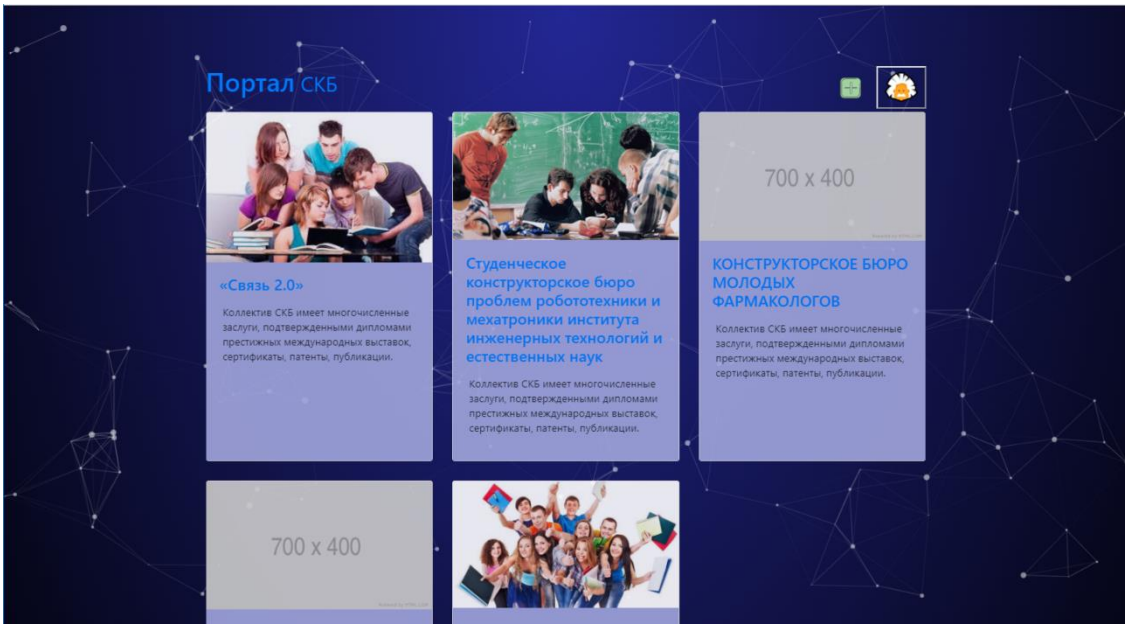


Рисунок 21 – Главная страница информационного портала

Кроме того, при наведении курсора мышки на пустую область происходит взаимодействие движущихся частиц с ним, данное явление продемонстрирована в левом верхнем углу на рисунке 22.

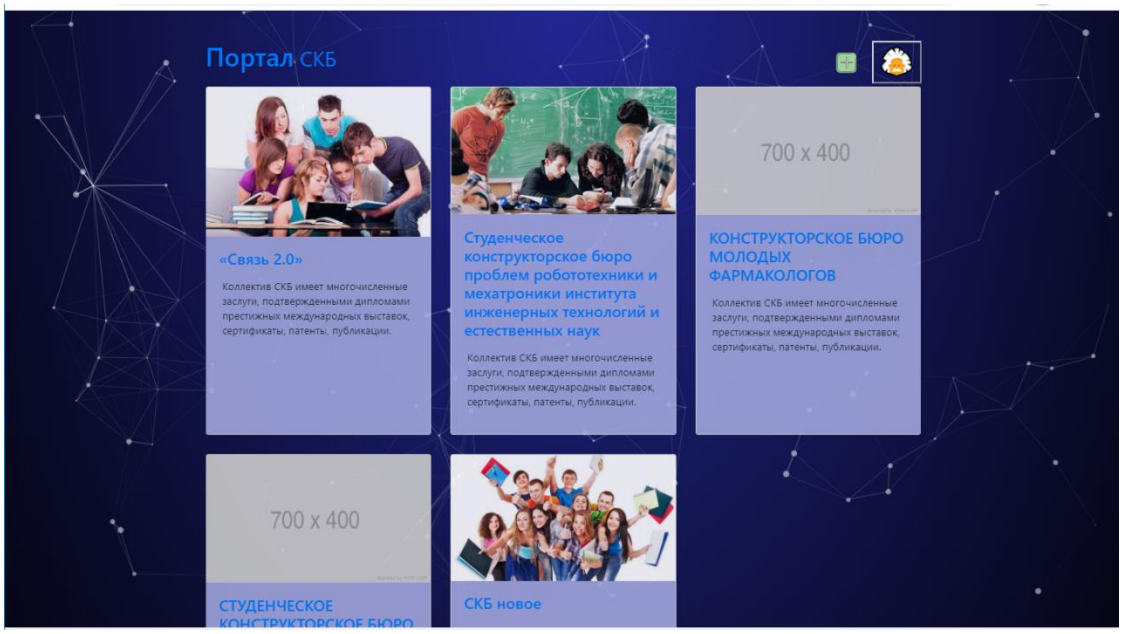


Рисунок 22 – Реакция движущихся частиц на действие пользователя

Следующим шагом было создание формы для регистрации и авторизации пользователя, форма авторизации включает в себя два поля логин, пароль и представлена на рисунке 23.

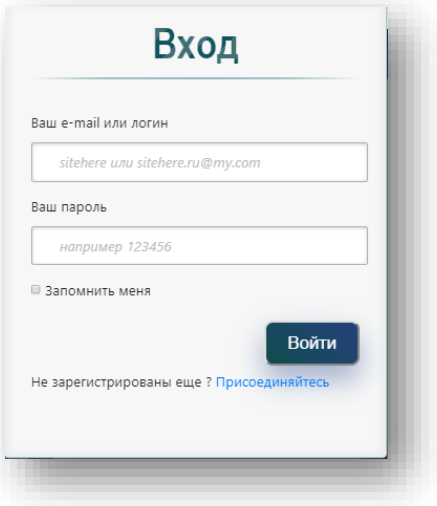


Рисунок 23 – Форма для авторизации пользователя

Форма для регистрации пользователя включает в себя поля «e-mail» «Фамилия», «Имя», «Пароль», «Подтверждение пароля» и представлена на рисунке 24 ниже.

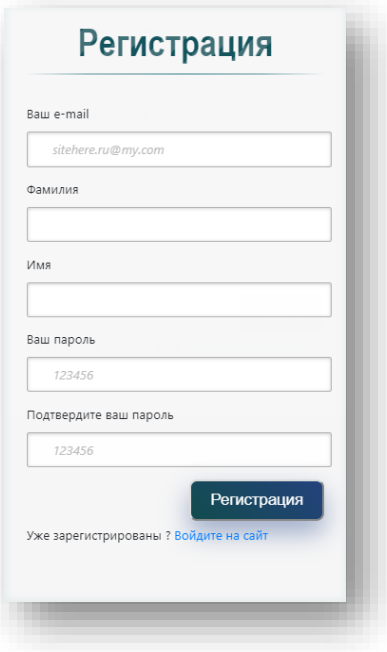


Рисунок 24 – Форма для регистрации пользователя

После того как были созданы формы для регистрации и авторизации, была начата разработка личного кабинета пользователя, в зависимости от полномочий внешний вид кабинета меняется рисунок 25.

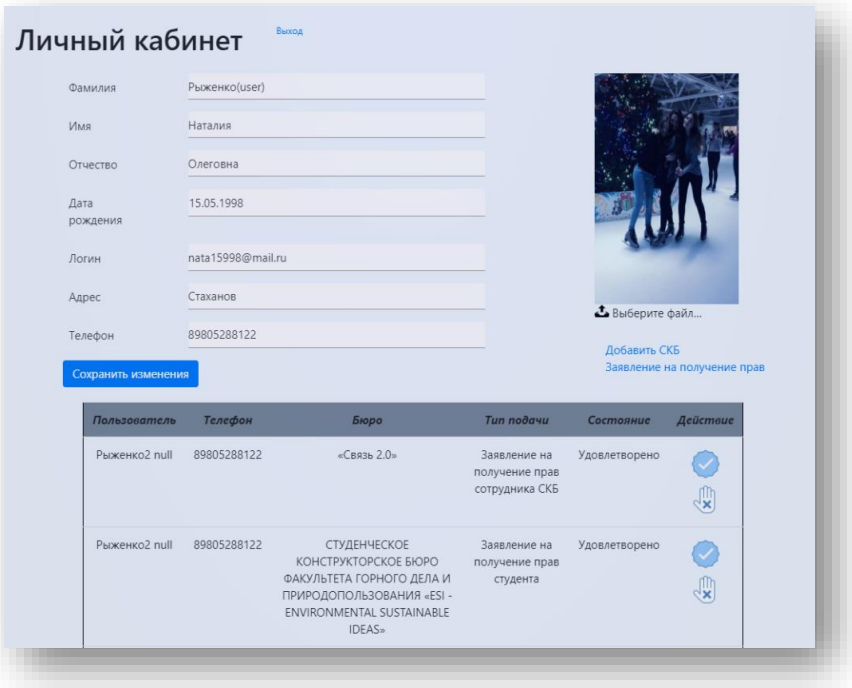


Рисунок 25 – Личный кабинет пользователя

Если пользователь является администратором, то в личном кабинете под данными профиля появляется панель администратора, представленная на рисунке 26 ниже.

Рисунок 26 – Панель администратора

Справа под фотографией пользователя было реализовано меню для подачи заявления и создания нового СКБ, интерфейс окна для подачи заявление показан на рисунке 27 ниже.

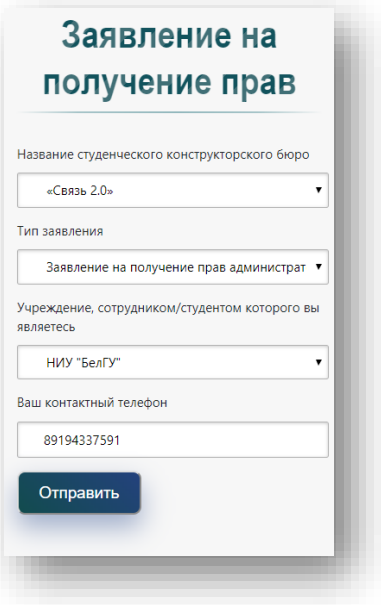


Рисунок 27– Интерфейс для подачи заявления на права

В случае выбора пункта «Добавить СКБ» перед пользователем возникнет форма, для заполнения показанная на рисунке 28 ниже.

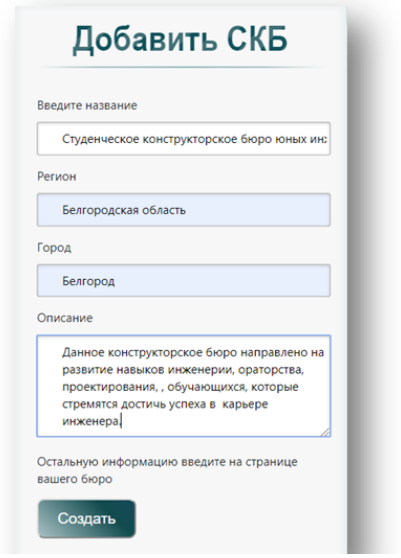


Рисунок 28 – Интерфейс для добавления нового СКБ

Для демонстрации более подробной информации о студенческом конструкторском бюро, был создан профиль для каждого существующего СКБ, представленный на рисунке 29.

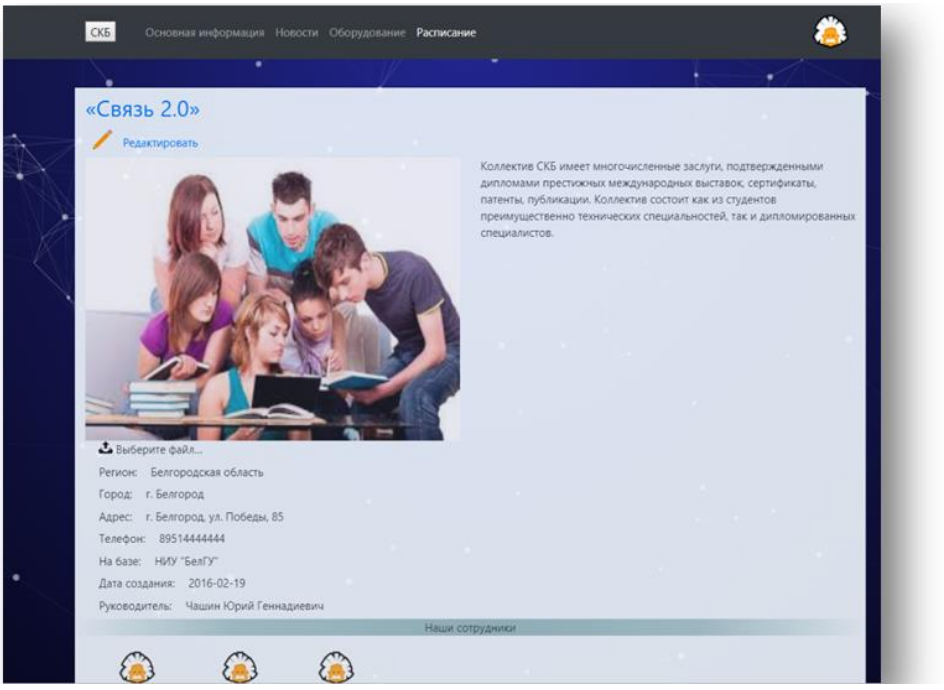


Рисунок 29 – Интерфейс профиля СКБ

Как видно из рисунка 29, в верхней части портала расположено меню навигации, для просмотра связанной с бюро информации (новости, оборудование, расписание). На рисунке 30 изображена форма для создания новости.

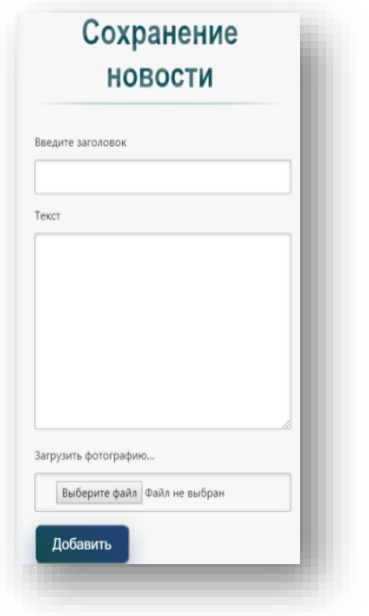


Рисунок 30 – Интерфейс для добавления новости

На рисунке 31 отображено как выглядит информационный портал в разделе отображения новостей СКБ.

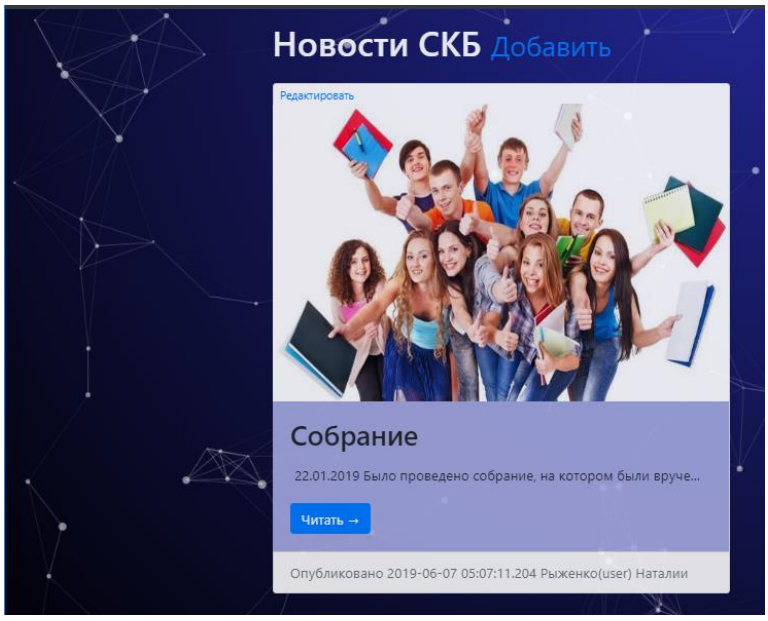
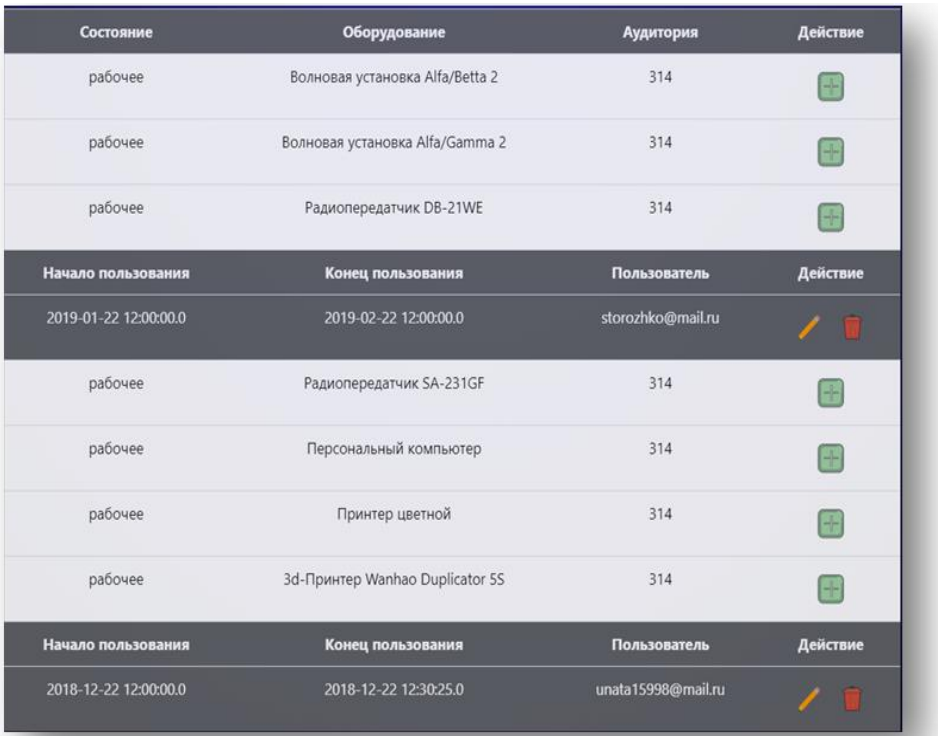


Рисунок 31 – Интерфейс портала в разделе «Новости»

После создания формы для добавления новостей был разработан интерфейс представленный на рисунке 32, для просмотра оборудования и его бронирования.



Состояние	Оборудование	Аудитория	Действие
рабочее	Волновая установка Alfa/Betta 2	314	
рабочее	Волновая установка Alfa/Gamma 2	314	
рабочее	Радиопередатчик DB-21WE	314	

Начало пользования	Конец пользования	Пользователь	Действие
2019-01-22 12:00:00.0	2019-02-22 12:00:00.0	storozhko@mail.ru	
рабочее	Радиопередатчик SA-231GF	314	
рабочее	Персональный компьютер	314	
рабочее	Принтер цветной	314	
рабочее	3d-Принтер Wanhao Duplicator 5S	314	

Начало пользования	Конец пользования	Пользователь	Действие
2018-12-22 12:00:00.0	2018-12-22 12:30:25.0	unata15998@mail.ru	

Рисунок 32 – Форма для расписания оборудования

Данная форма позволяет отобразить доступное оборудование, его состояние и историю пользования.

3.4 Тестирование и отладка портала для студенческого бюро

Процесс разработки любого программного обеспечения невозможен без контроля качества, тестирование позволяет обнаружить и устранить дефекты, снизить уровень риска и повысить качество продукта [28].

Проверка включает места пользовательского интерфейса, где пользователь может допустить ошибку и устойчивость системы к злонамеренным действиям [29].

Вначале была протестирована форма регистрации пользователя при различных действиях пользователя (вводе не верных, ошибочных данных).

При вводе логина и пароля, которые уже существуют, в базе данных появляется соответствующее окно, представленное на рисунке 33.

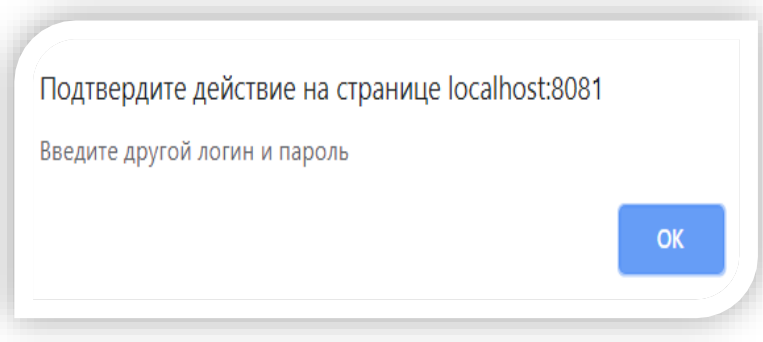


Рисунок 33 – Диалоговое окно при вводе существующих данных

В случае ввода уникального логина, но отсутствия заполнения полей «Имя» и «Фамилия», форма для ввода данных приобретёт вид, показанный на рисунке 34.

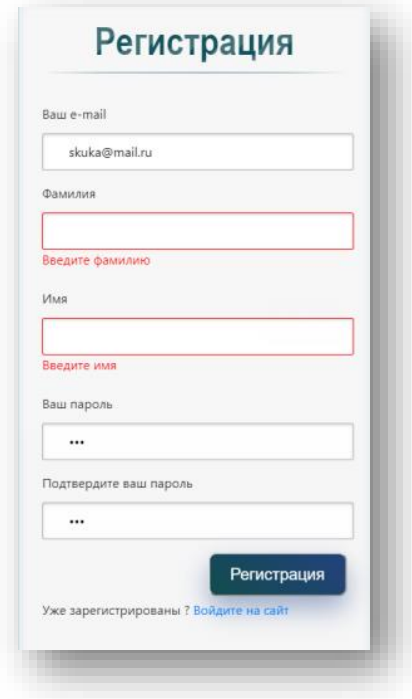


Рисунок 34 – Изменение формы при не заполнении обязательных полей

После того как пользователь заполнит все поля формы для регистрации, происходит проверка, если пароли не совпадают после проверки, то, выводиться соответствующие предупреждение представленное на рисунке 35.

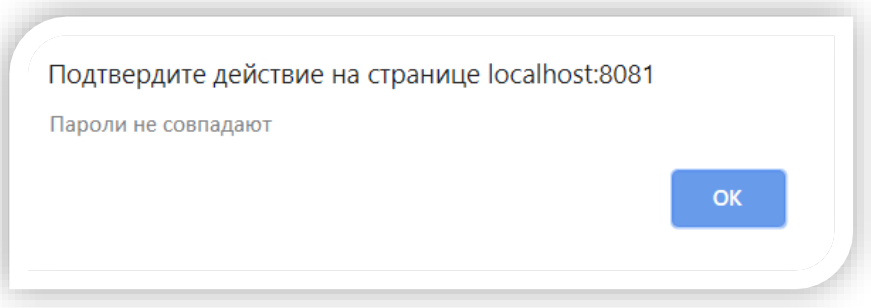


Рисунок 35 – Предупреждение о несоответствии паролей

Аналогичная проверка ошибок была сделана и для авторизации пользователя, в случае неверного ввода пароля или логина пользователю предлагалось пройти регистрацию.

В случае если пользователь не осуществлял авторизации на сайте, то функционал портала был для него ограничен, к примеру, он не мог создать новое студенческое конструкторское бюро, редактировать новости, вносить изменения в существующую на сайте информацию и бронировать оборудование, сообщение о недостаточных полномочиях представлено на рисунке 36.

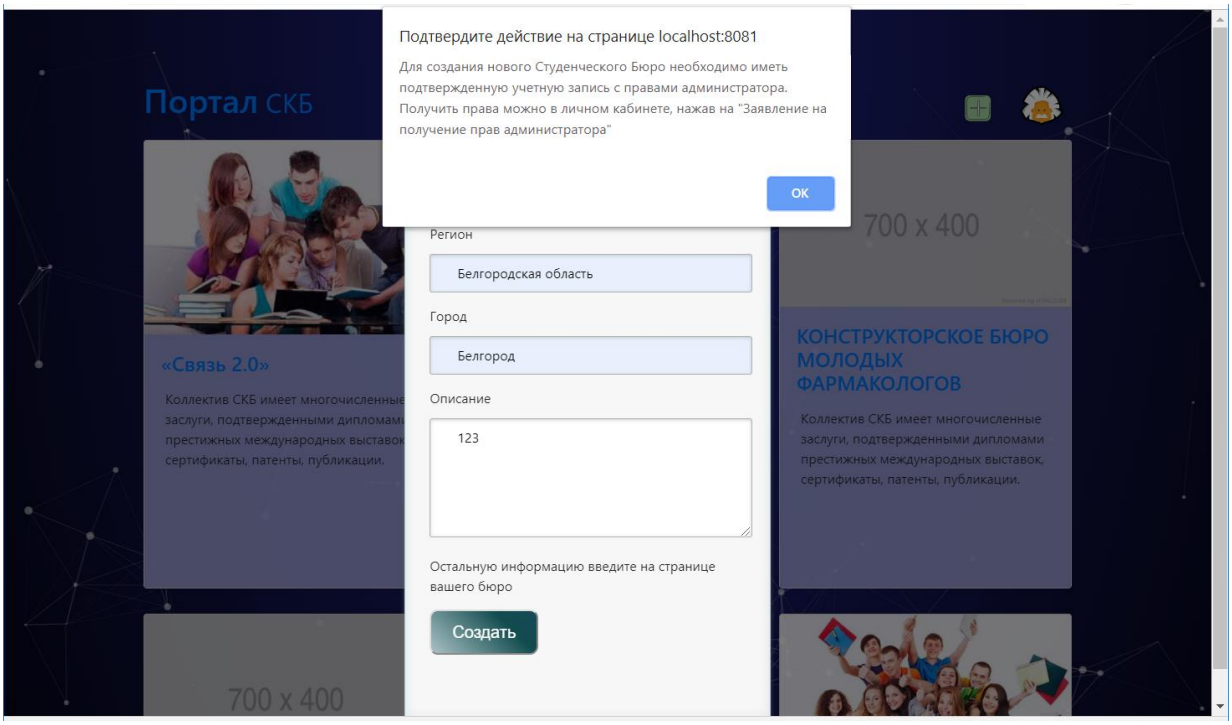


Рисунок 36 – Предупреждение об отсутствии прав

После того как пользователь прошел авторизацию/регистрацию и перешел в личный кабинет, он мог редактировать информацию о своем профиле, после ее изменения появлялось диалоговое окно продемонстрированное на рисунок 37 ниже.

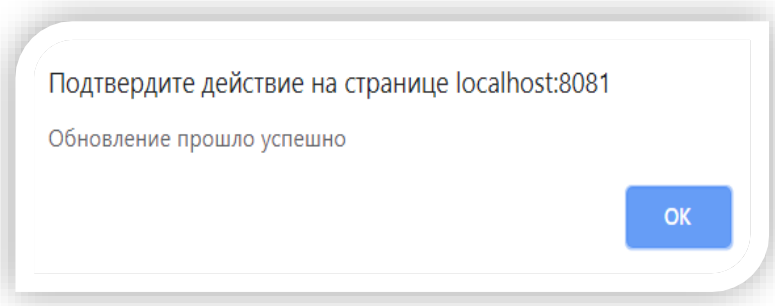


Рисунок 37 – Диалоговое окно об успешном обновлении данных

В случае ввода некорректных данных появляется также предупреждение, кроме того в личном кабинете была осуществлена возможность подачи заявления для получения прав, показанная на рисунке 38 ниже.

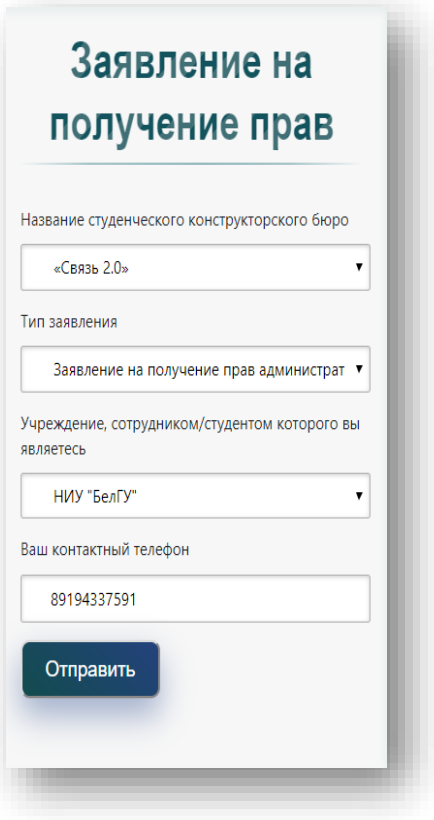


Рисунок 38 – Заявление на получение прав

После того как пользователь отправил любой вид заявления, оно появлялось на панели у администратора показанной на рисунке 39.

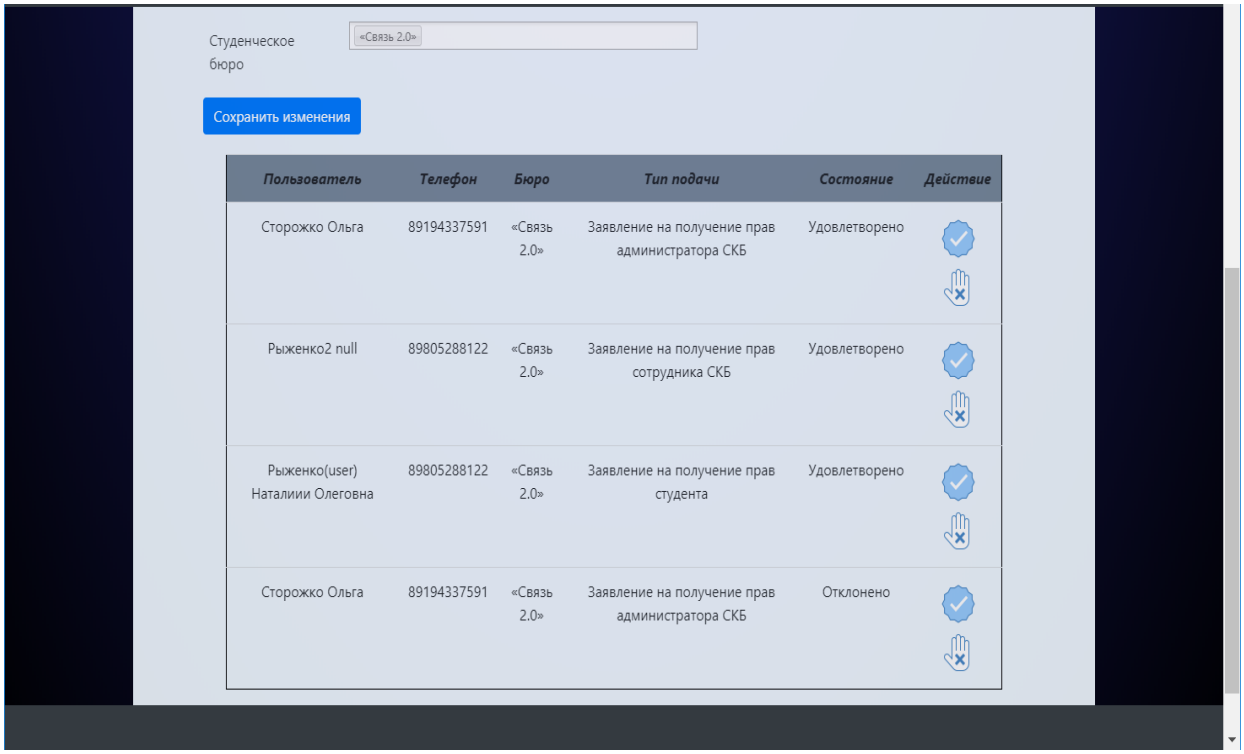


Рисунок 39 – Панель администратора портала

В случае отклонения/одобрения заявления на права, появляется соответствующие уведомление, представленное на рисунке 40.

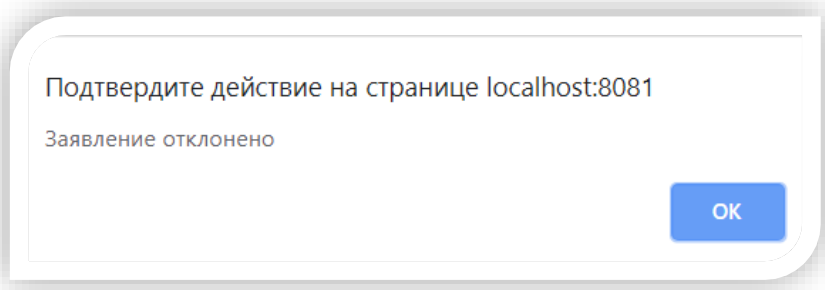


Рисунок 40 – Сообщение об отклонении заявления

Если администратор ошибся, с назначением прав, то он всегда может снова рассмотреть заявление и одобрить/отклонить его. В случае повторного одобрения либо отклонения заявления пользователя, изменение прав и запись

в базу данных не произойдет, но появиться соответствующие сообщение, представленное на рисунке 41.

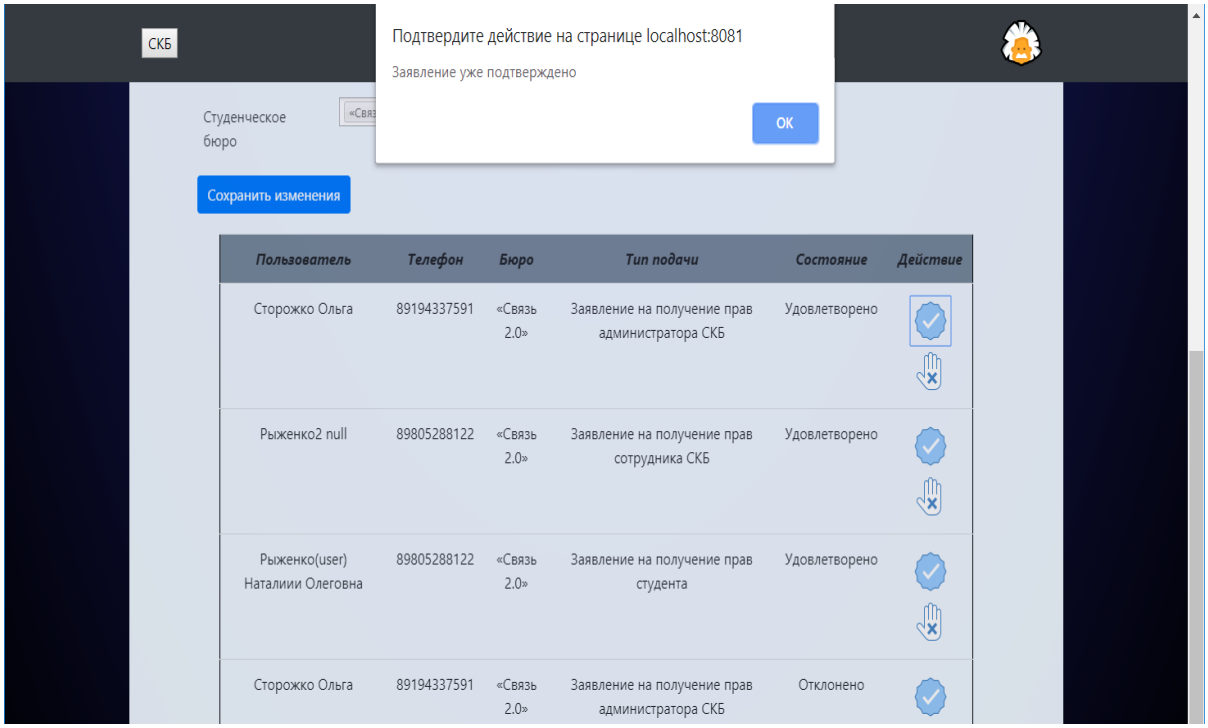


Рисунок 41 – Сообщение о повторном действии

В случае, если авторизированный пользователь обладающий правами администратора решит изменить информацию о СКБ, за которое он несет ответственность появится уведомление, показанное на рисунке 42.

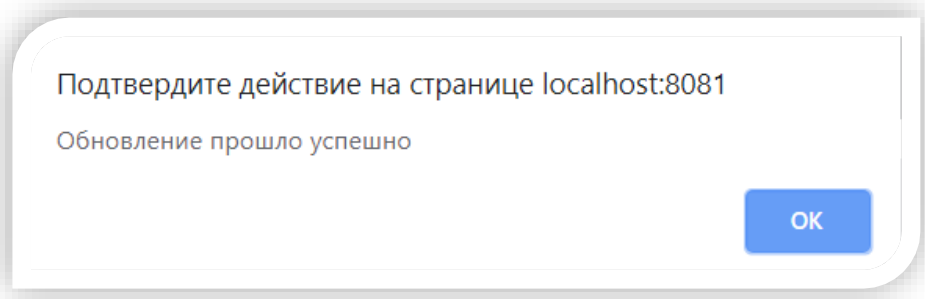


Рисунок 42 – Сообщение об успешном изменении данных

В личном кабинете администратор также может сформировать отчет об участниках бюро, результат форматирования отчёта показан на рисунке 43 ниже.

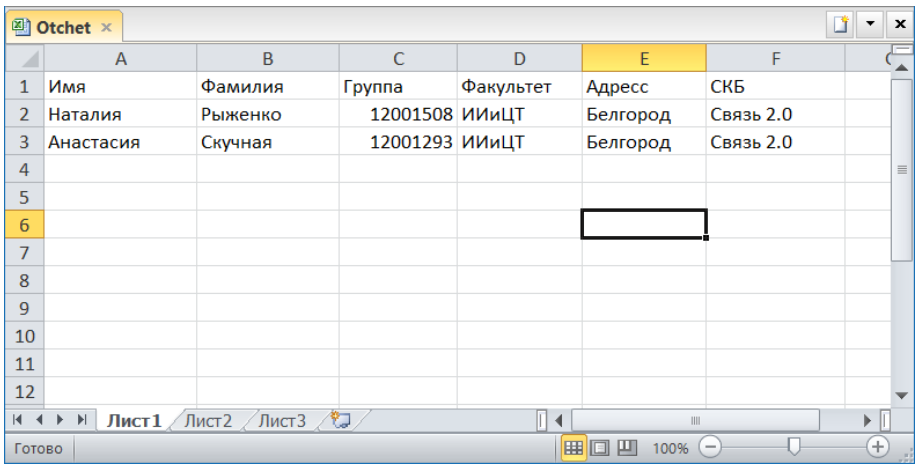


Рисунок 43 – Отчет об участниках СКБ

Кроме того администратор СКБ может не только обновлять информацию, но также добавлять новости, оборудование, формировать отчет о количестве участников СКБ. На рисунке 44 предоставлен пример создания новой новости.

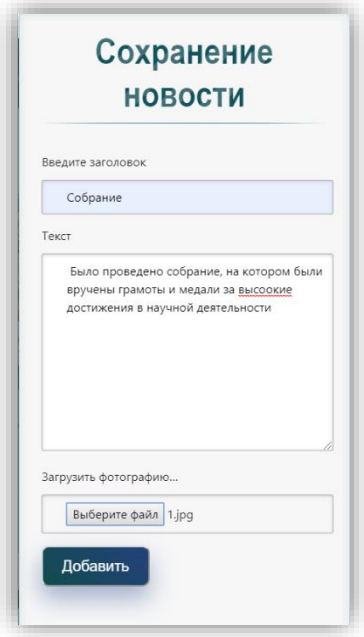


Рисунок 44 – Создание новой новости

При создании новости пользователю обладающими достаточными правами необходимо указать заголовок, основной текст новости и выбрать изображения.

На рисунке 45 продемонстрирована как будет выглядеть созданная пользователем новость после выбора ее из списка новостей.

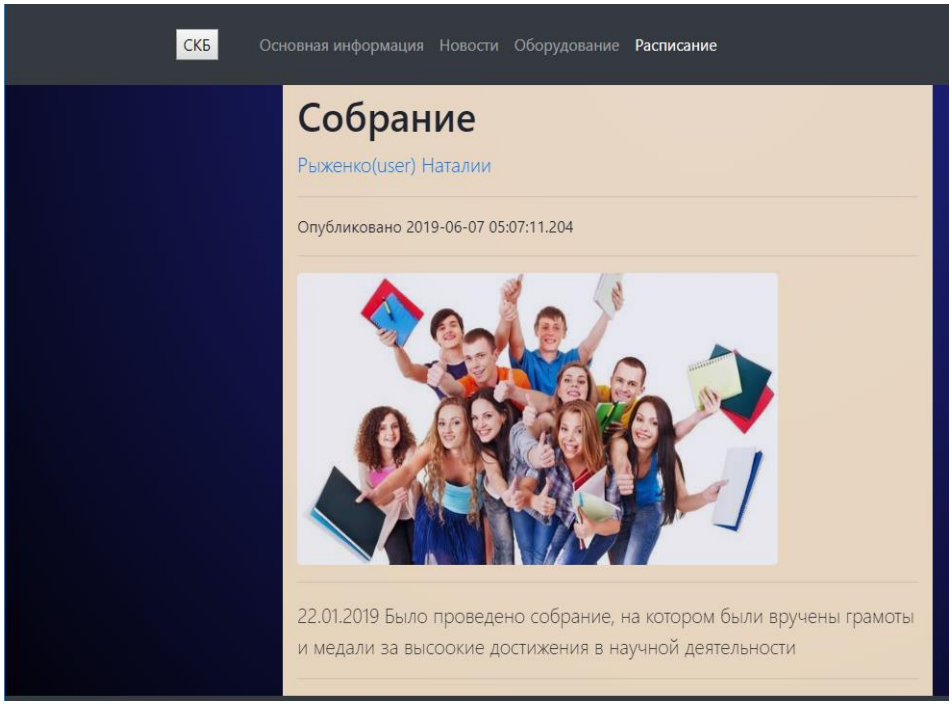


Рисунок 45 – Созданная новость

В случае если другой пользователь попытается изменить эту новость, то увидит сообщение, показанное на рисунке 46.

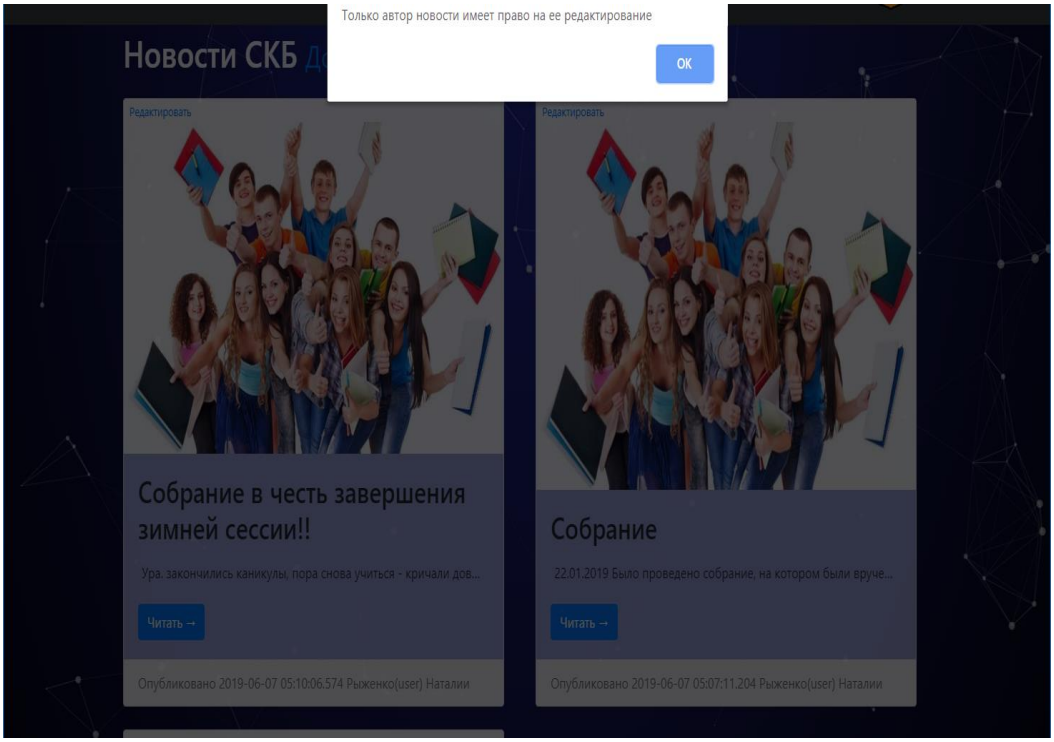


Рисунок 46 – Уведомление о невозможности редактировать новости

У каждого СКБ есть перечень оборудования, редактировать и добавлять который может только администратор, тестирование функции создание нового оборудования показно на рисунке 47 ниже.

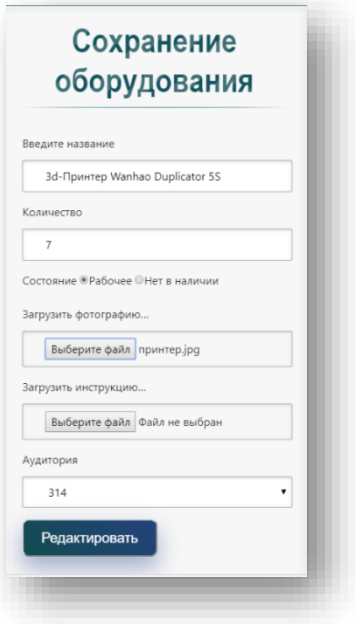


Рисунок 47 – Добавление нового оборудования

В результате внесения новой информации в перечень уже существующего у СКБ оборудования, список обновится как на рисунке 48 ниже.

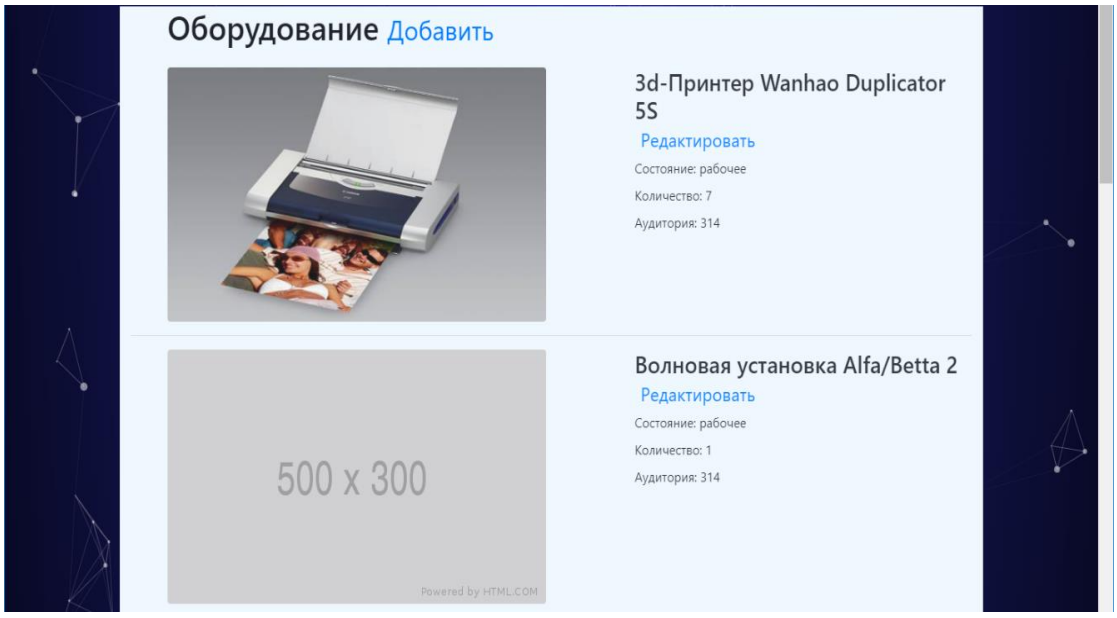


Рисунок 48 – Результат добавление оборудования

Редактировать и вносить информацию о дате бронирования оборудования может любой зарегистрированный пользователь, принадлежащий к соответствующему СКБ, пример бронирования представлен на рисунке 49 .

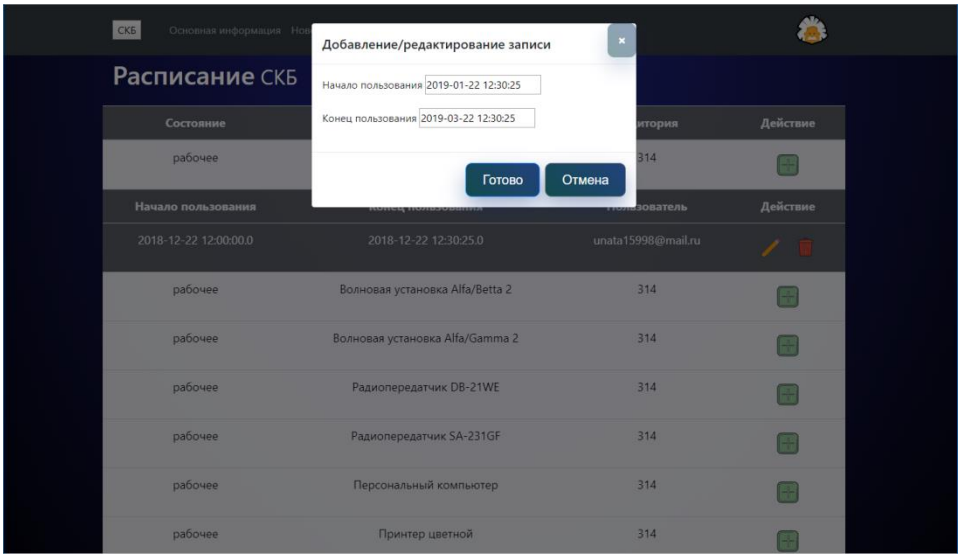


Рисунок 49 – Внесение информации в расписание оборудования

В результате внесения информации пользователя, расписание изменилось, кроме того если даты пользования уже прошли, пользователь не может удалить/изменить информацию в истории, будет показано соответствующие сообщение представленное на рисунке 50.

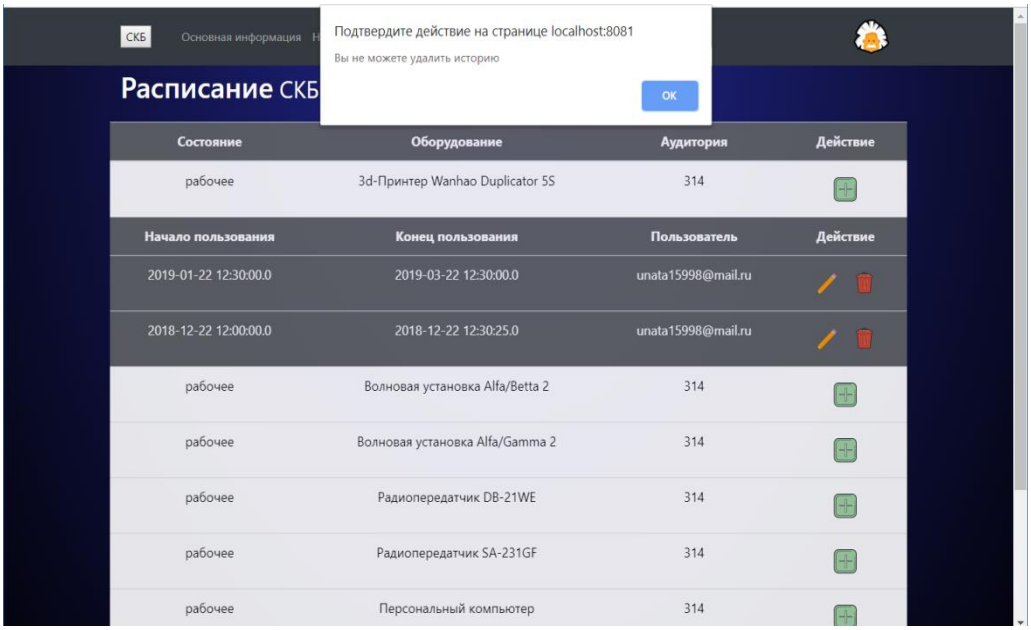


Рисунок 50 – Внесение информации в расписание оборудования

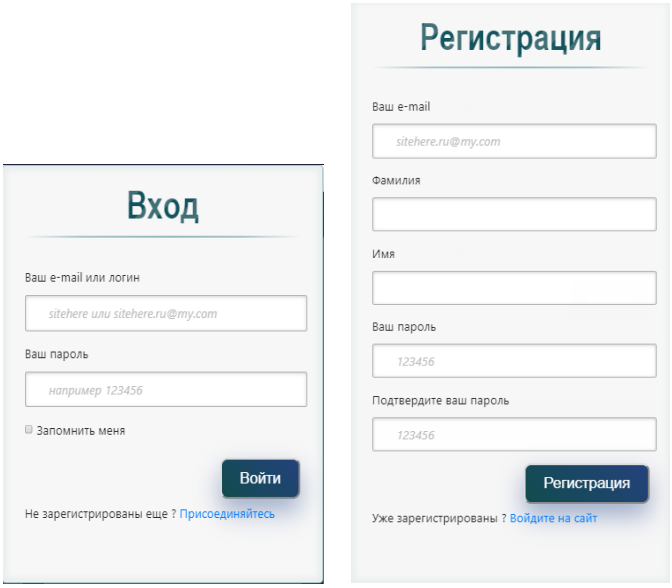
3.5 Руководство пользователя и администратора

3.5.1 Инструкция для пользователя

Для совершения большинства действий на информационном портале пользователю необходимо авторизоваться/зарегистрироваться, для этого следует на главной странице нажать на иконку справа, представленной на рисунке 51 ниже.

Рисунок 51 – Иконка для перехода к регистрации/авторизации

После нажатия на икону перед пользователем появится форма для авторизации, представленная на рисунке 52 а, если пользователь ранее не проходил регистрацию, то ему следует нажать на ссылку «Присоединитесь» и перед ним появится окно для заполнения данных для регистрации, представленное на рисунке 52 б.



а) б)

Рисунок 52 – Окно авторизации (а) и окно регистрации (б)

После того как будет осуществлен вход на портал под учетной записью, пользователь можете заново нажать на иконку авторизации и перейти в личный кабинет представленный на рисунке 53.

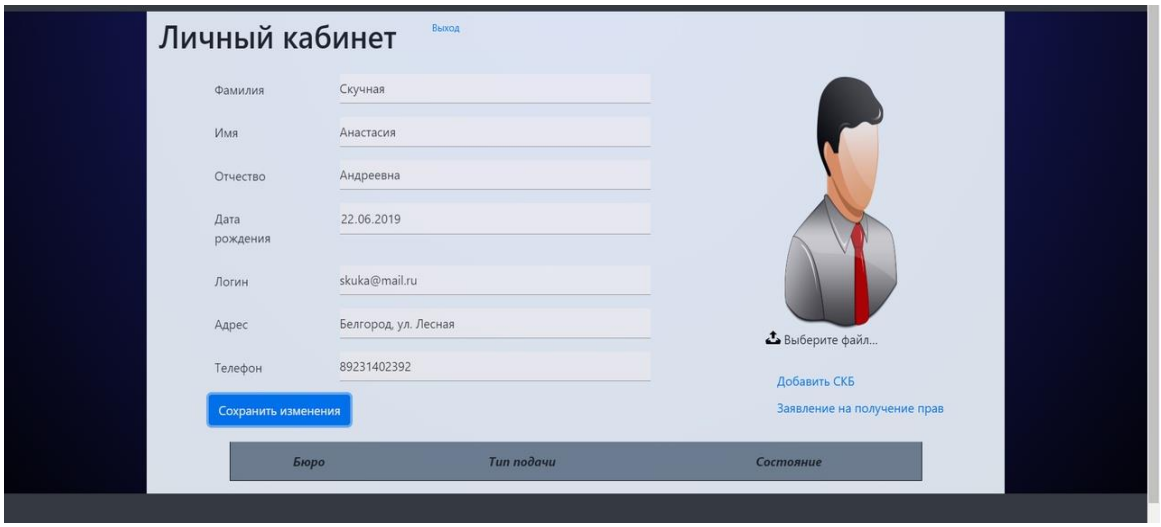


Рисунок 53 – Личный кабинет пользователя

В кабинете обычный пользователь может отредактировать информацию своего профиля (добавить фотографию, личные данные), кроме того нажав на пункт «Заявление на получение прав» появиться форма для оформления заявления представленная на рисунке 54.

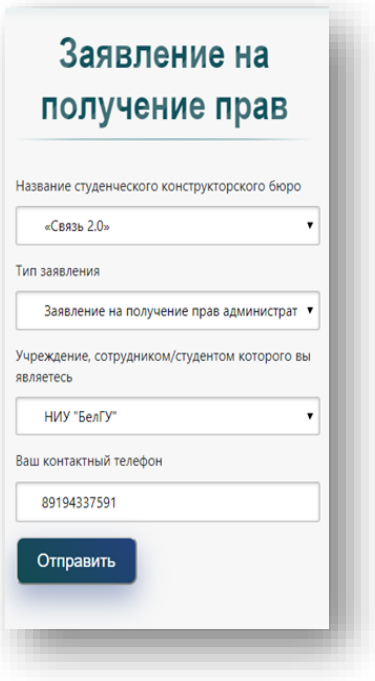


Рисунок 54 – Окно для подачи заявления

Подача заявления представляет собой диалоговое окно, в котором пользователю необходимо выбрать СКБ, тип заявления и указать свои данные.

После того как пользователю будут назначены права, он сможет взаимодействовать со структурами портала, к примеру просматривать информации о студенческом бюро, его новостях, оборудовании и расписании данные функции будут доступны любому пользователю, однако брать оборудование в пользование сможет только участник данного СКБ.

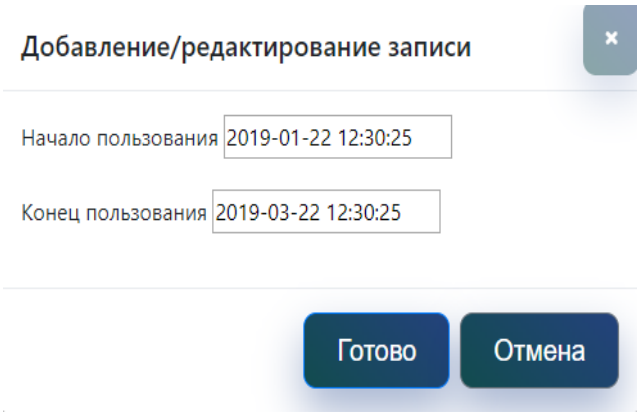


Рисунок 55 – Окно для внесения периода использования

Для бронирования оборудования в расписании необходимо нажать на зеленую иконку с изображением плюса, после чего появится форма для заполнения периода использования, представленная на рисунке 55 выше.

3.5.2 Инструкция для администратора

К полномочиям администратора СКБ кроме вышеперечисленных возможностей пользователя добавляются несколько новых:

– формирование отчета об участниках СКБ;

– добавление нового бюро;

– возможность рассмотрения заявления пользователя;

– назначение новых прав пользователю.

На рисунке 56 продемонстрирован личный кабинет администратора.

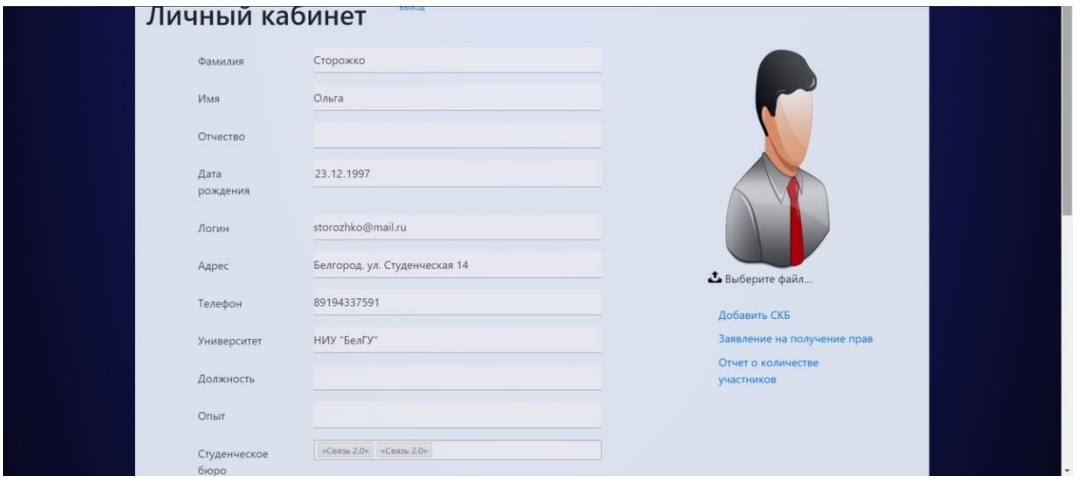


Рисунок 56 – Личный кабинет администратора

На панели администратор представленной на рисунке 56 показаны сведенья о пользователе и его запрос на права, нажав на кнопку одобрения или отклонения, администратор может назначить, отклонить или снять назначенные права с пользователя соответственно.

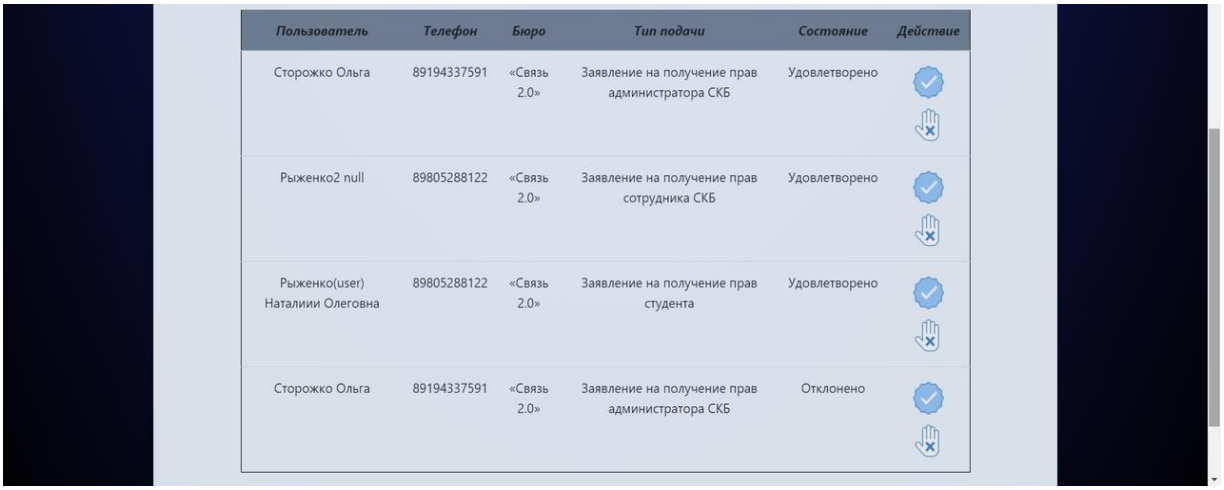
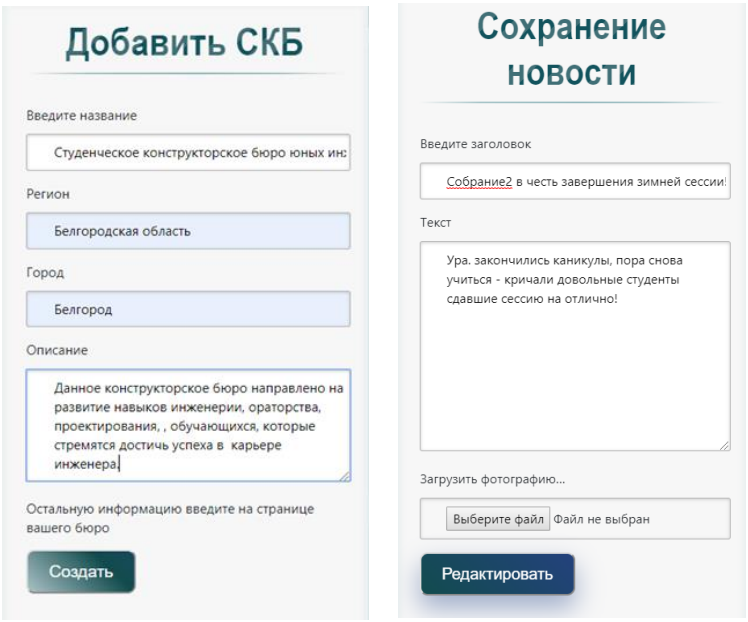


Рисунок 57 – Панель администратора

Выбрав соответствующий пункт в личном кабинете, управляющий, может сформировать отчет или добавить СКБ, также он обладает правом обновления информации о СКБ, добавления оборудования, новостей, аналогично заполнению информации в профиле пользователя. На рисунке 58 а ниже представлен пример создания СКБ и создание новости для СКБ на рисунок 58 б.



а) б)

Рисунок 58 – Окно создания СКБ (а) и окно добавления новости (б)

Ответственный за СКБ может изменять, удалять и редактировать историю использования оборудования, нажимая на соответствующие иконки (карандаш для редактирования, мусорное ведро для удаления).

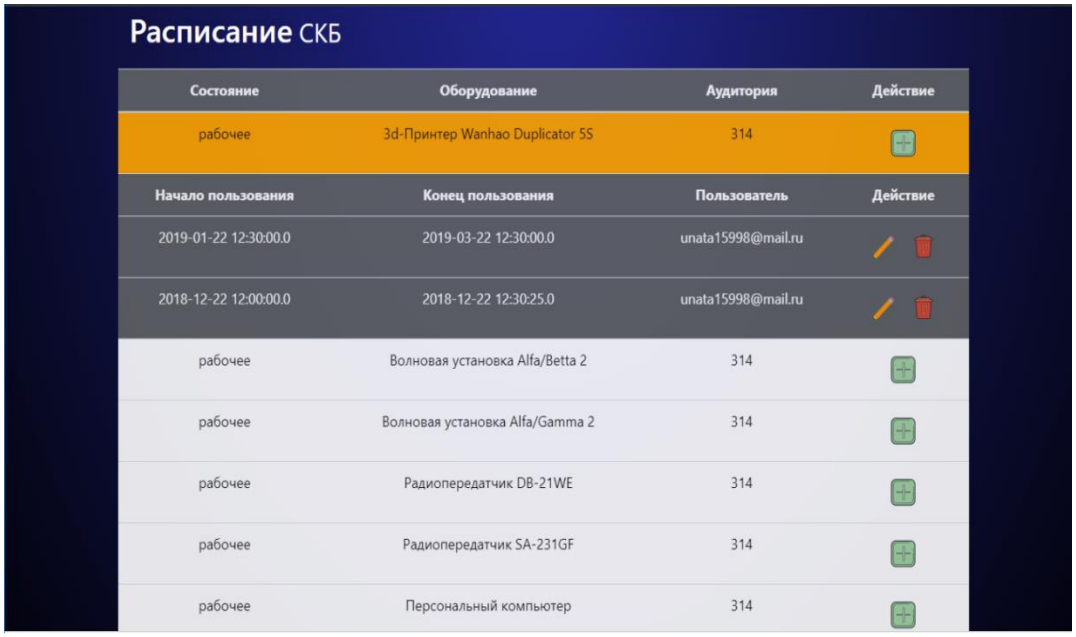


Рисунок 59 – Просмотри истории пользования оборудования

Кроме того просматривать информацию об истории использования оборудования, кликнув два раза на интересующий объект как на рисунке 59.

3.6 SWOT- анализ сильных и слабых сторон

SWOT анализ – является одним из самых эффективных инструментов для анализа внутренних и внешних факторов организации, оценке рисков и конкурентоспособности в отрасли. Объектом изучения может быть продукт, компания, магазин, завод, страна, образовательное учреждение, человек [31].

Таблица 4 – SWOT матрица

Сильные стороны	Возможности		Угрозы		Итого
	Улучшение разработки	Охват новых целевых групп	Выход на рынок нового конкурента	Быстрое устаревание	
Доступ к уникальным ресурсам/информации	++	++	+	+	6
Низкая вероятность ошибки	+	+	+	+	4
Высокая скорость работы	+	+	+	+	4
Многофункциональность	+	++	+	+	5
Подача заявления online	0	+	+	+	3
Итого	5	7	5	5	22
Слабые стороны					
Необходимость поддержки и сопровождения портала	–	–	–	– –	-5
Обучение персонала работы с порталом	–	–	0	–	-3
Итого	-2	-2	-1	-3	-8
Общий итог					14

Разработанный информационный портал имеет следующие сильные стороны:

– доступ к уникальным ресурсам/информации;

– низкая вероятность ошибки;

– высокая скорость работы;

– многофункциональность;

– подача заявления online.

Слабые стороны информационного портала включает в себя:

–необходимость поддержки и сопровождения портала;

–обучение персонала работы с порталом.

Возможности системы позволяют:

– улучшить разработку;

– охватить новые целевые группы.

Угрозами для информационного портала студенческого конструкторского бюро являются:

– выход на рынок нового конкурента;

– быстрое устаревание.

Проанализировав таблицу 4, был сделан вывод, что наиболее важным достоинством информационного портала для студенческого конструкторского бюро является его многофункциональность и доступ к уникальным ресурсам. Все выше перечисленные слабые стороны проекта несут угрозу, однако наиболее опасна необходимость поддержки портала в рабочем состоянии. На данный момент сложившиеся условия угрозы не являются существенными, однако, для стабильного функционирования, необходимо его поддерживать, обновлять структуру и информацию.

Вывод по третьему разделу

В данном разделе выпускной квалификационной работы была разработана база данных, информационный портал для СКБ, интерфейс для портала, протестировано созданное приложение, написано руководство для пользователя и администратора и проведен SWOT - анализ.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был разработан информационный портал для студенческого конструкторского бюро.

В процессе разработки и проектирования были решены следующие задачи:

– произведен анализ предметной области и аналогов существующих программных средств;

– разработана функциональная и информационная модель;

– выбраны программные средства для реализации;

– разработано программное обеспечение;

– произведено тестирование и отладка программного продукта.

В ходе первых тестирований программы были выявлены незначительные ошибки в проектировании и реализации, которые были устранены. Программа успешно работает в стандартном режиме эксплуатации, а также при различных несанкционированных действиях пользователя.

В результате была достигнута основная цель работы: разработан информационный портал для студенческого конструкторского бюро.

В будущем, возможны улучшения разработанного портала в следующих направлениях:

– добавление чата для общения участников СКБ;

– добавление фотогалереи для каждого СКБ;

– добавление отчета о научных достижениях каждого пользователя и СКБ в целом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дронов, В. А. HTML 5, CSS 3 и Web 2.0: Разработка современных Web-сайтов / В.А. Дронов. - М.: Горячая линия-Телеком, 2011. - 414 с.

2. Разновидности сайтов [Электронный ресурс]. – Режим доступа: http://sait-sozdat.ru/vidy-sait/info-portal/chto-takoe-informatsionniy-portal.php.

3. ЮниТех[Электронный ресурс]. – Режим доступа: https://unitech-mo.ru/science/research-activities-/youth-science/kruek/.

4. Киберлинк [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/studencheskoe-konstruktorskoe-byuro-kak-uslovie-podgotovki-konkurentosposobnyh-spetsialistov.

5. Отчет по годовому этапу НИР [Электронный ресурс]. – Режим доступа:http://zabgu.ru/files/html_document/pdf_files/fixed/Rezul'taty'_proektny'x_NIR/645634331.pdf#3

6. Калянов, Г.Н. CASE-технологии: Консалтинг при автоматизации бизнес-процессов /Г.Н. Калянов. - М.: Горячая линия-Телеком, 2014. - 318 с.

7. Лаврищева, Е.М. Программная инженерия: Парадигмы, технологии и CASE-средства. Учебник для вузов / Е.М. Лаврищева. - М.: Горячая линия-Телеком, 2017. - 165 с.

8. Остроух, А.В. Проектирование информационных систем / А.В. Остроух, Н.Е. Суркова. - СПб.: Лань, 2019. -164 с.

9. Волкова, В. Н. Информационные модели и автоматизированные процедуры для управления инновациями / В. Н. Волкова, А. В. Логинова. - М.: ДМК Пресс, 2019. - 7 с.

10. Бэзинс, Б. Java для начинающих: Объектно-ориентированный подход / Э. Бэкил, З. Брукее. - СПб.: Лань, 2018. - 688 с.

11. Давыдов, С. IntelliJ IDEA. Профессиональное програмирование на Java / С. Давыдов, А. Ефимов. - М: Вильямс, 2011. -800 с.

12. Волков, А. Изучаем PostgreSQL 10 / А. Волков, Д. Салахалдин. –М.: ДМК Пресс, 2019. - 400 с.

13. Новиков, Б. А. Основы технологий баз данных: учеб. пособие / Б. А. Новиков, Е. А. Горшкова; под ред. Е. В. Рогова. - М.: ДМК Пресс, 2019. -240 с.

14. Моргунов, Е. П. PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов, П. В. Лузанова. - СПб.: БХВ-Петербург, 2018. - 336 с.

15. Салахалдин, Д. Изучаем PostgreSQL 10 / Д. Салахалдин, А. Волков. - М.: ДМК Пресс, 2019. -400 с.

16. Фримен, Э. Изучаем HTML, XHTML и CSS 2-е изд. / Э. Фримен. - СПб.: Питер, 2013. - 720 с.

17. Макфарланд, Д. Новая большая книга CSS / Д. Макфарланд. - СПб.: Питер,2018.- 720 с.

18. Браун, Э. Изучаем JavaScript. Руководство по созданию современных веб-сайтов / Э. Браун.- М.: Альфа-книга, 2017. - 368 с.

19. Сьерра, К. Изучаем Java / К. Сьерра, Б. Бейтс. - М.: Эксмо, 2018. - 720 с.

20. Лафоре, Р. Структуры данных и алгоритмы в Java / Р. Лафоре. - СПб.: Питер, 2018.- 704 с.

21. Эккель, Б. Философия Java / Б. Эккель.- СПб.: Питер, 2019. - 1168 с.

22. Уоллс, К. Spring в действии / К. Уолс. -М.: ДМК Пресс, 2015. -754 с.

23. Чугреев, В. Л. Особенности реализации MVC-архитектуры в веб-приложениях / В.Л. Чугреев.- М.: Горячая линия-Телеком, 2015. - 200 с.

24. Флэнаган, Д. JavaScript. Подробное руководство / Д. Флэнаган. -М.: Символ-Плюс, 2012. – 1080 с.

25. Крокфорд, Д. JavaScript: сильные стороны / Д. Крокфорд. - М.: Эксмо, 2012. - 176с.

26. Ноубл, Д. HTML, XHTML и CSS для чайников/ Д. Ноубл. - М.: «Диалектика», 2011. – 313 с.

27. Лабберс, П. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений/ П.Лабберс. - М.: «Вильямс», 2011. – 453 с.

28. Гради, Б. Объектно-ориентированный анализ и проектирование с примерами приложений / Б. Гради, А. Роберт, У. Энгл и др .- М.:Вильямс, 2010.- 720 с.

29. Седжвик, Р. Алгоритмы на Java / Р. Седжвик, К. Уэйн. - М.: «Вильямс», 2016. – 848 с.

30. Арутюнова, Д.В. Стратегический менеджмент. Учебное пособие/ Д.В. Артюнова. Таганрог: ТТИ ЮФУ, 2010.- 122 с.

31. ГОСТ 7.32 - 2001 Система стандартов по информации, библиотечному и издательскому делу. Отчѐт о научно-исследовательской работе. Структура и правила оформления.

ПРИЛОЖЕНИЕ А

Диаграммы потоков данных, демонстрирующие существующую технологию решения задач «Как есть» на аналогичных ресурсах

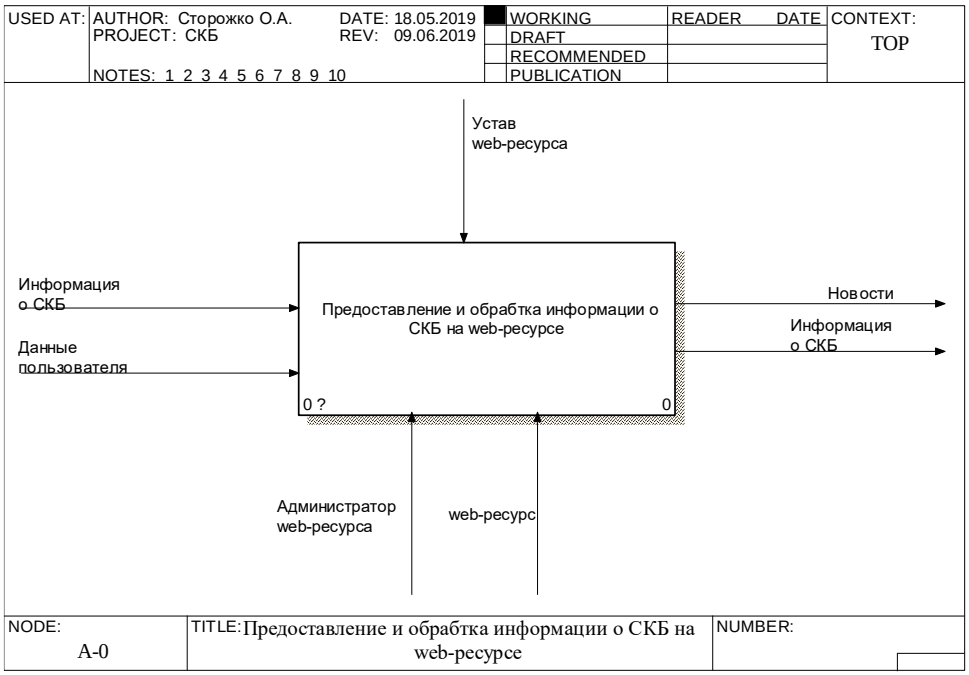


Рисунок А.1– Контекстная диаграмма «Как есть»

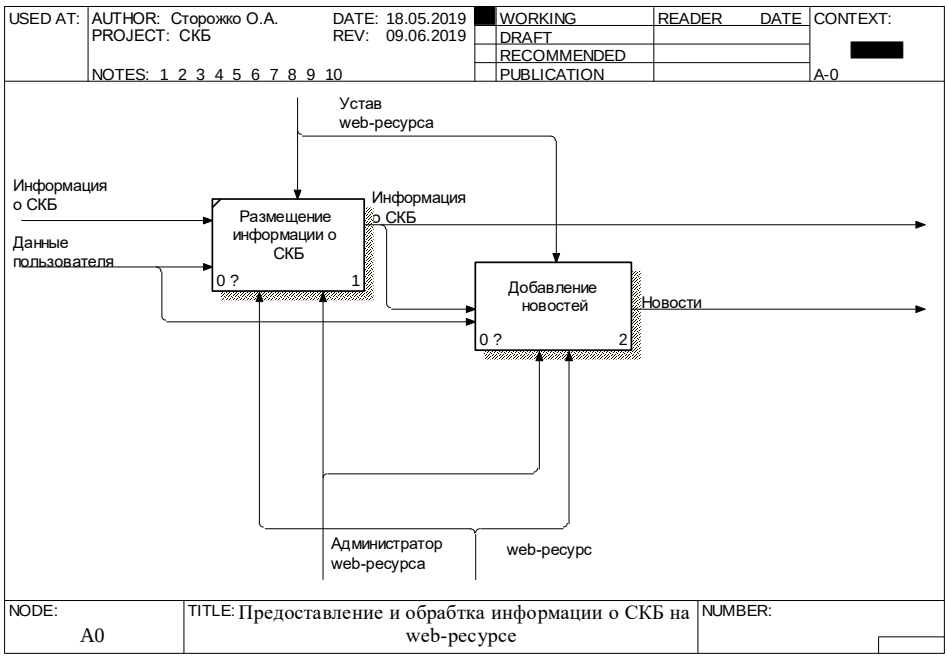


Рисунок А.2 – Декомпозиция контекстная диаграмма «Как есть»

ПРИЛОЖЕНИЕ Б

Листинг кода для создания базы данных

```
CREATE DATABASE bureaus;
CREATE TABLE roles(id SERIAL PRIMARY KEY , name VARCHAR(50), aspect INTEGER);

CREATE TABLE users(id SERIAL PRIMARY KEY , login VARCHAR(30), password VARCHAR(50), role INTEGER, person INTEGER, image VARCHAR(224), aspect INTEGER);
ALTER TABLE users ADD CONSTRAINT user_to_person_fkey FOREIGN KEY(person) REFERENCES persons(id);
ALTER TABLE users ADD CONSTRAINT user_to_role_fkey FOREIGN KEY(role) REFERENCES roles(id);

CREATE TABLE universities(id SERIAL PRIMARY KEY ,full_name VARCHAR(224), name VARCHAR(100), sysname VARCHAR(30), aspect INTEGER);

CREATE TABLE students(id SERIAL PRIMARY KEY , university INTEGER, faculty VARCHAR(50),person integer, "user" INTEGER, "group" VARCHAR(30));
ALTER TABLE students ADD CONSTRAINT student_to_user_fkey FOREIGN KEY("user") REFERENCES users(id);
ALTER TABLE students ADD CONSTRAINT student_to_uni_fkey FOREIGN KEY(university) REFERENCES universities(id);
ALTER TABLE students ADD CONSTRAINT  student_to_person_fkey FOREIGN KEY(person) REFERENCES persons(id);

CREATE TABLE teachers(id SERIAL PRIMARY KEY  , person integer,profession VARCHAR(50), experience DOUBLE PRECISION, university INTEGER, "user" INTEGER);
ALTER TABLE teachers ADD CONSTRAINT teacher_to_user_fkey FOREIGN KEY("user") REFERENCES users(id);
ALTER TABLE teachers ADD CONSTRAINT teacher_to_uni_fkey FOREIGN KEY(university) REFERENCES universities(id);
ALTER TABLE teachers ADD CONSTRAINT  teacher_to_person_fkey FOREIGN KEY(person) REFERENCES persons(id);
```

CREATE TABLE audiences(id SERIAL PRIMARY KEY ,name VARCHAR(100), time_start TIME, time_end TIME, aspect INTEGER);
ALTER TABLE audiences ADD CONSTRAINT audience_to_aspect_fkey FOREIGN KEY(aspect) REFERENCES aspects(id);

CREATE TABLE student_design_bureaus(id SERIAL PRIMARY KEY,name VARCHAR(224),region VARCHAR(224), city VARCHAR(224),university INTEGER, short_description VARCHAR(224), description VARCHAR(512), address VARCHAR(224), leader VARCHAR(224), phone VARCHAR(224), "date" DATE, aspect INTEGER, image VARCHAR(224));
ALTER TABLE student_design_bureaus ADD CONSTRAINT student_design_bureau_to_university_fkey FOREIGN KEY (university) REFERENCES universities(id);

CREATE TABLE student_design_bureaus_to_audiences(student_design_bureau INTEGER, audience INTEGER);
ALTER TABLE student_design_bureaus_to_audiences ADD CONSTRAINT student_design_bureau_fkey FOREIGN KEY(student_design_bureau) REFERENCES student_design_bureaus(id);
ALTER TABLE student_design_bureaus_to_audiences ADD CONSTRAINT audience_fkey FOREIGN KEY(audience) REFERENCES audiences(id);

CREATE TABLE document_types(id SERIAL PRIMARY KEY, name VARCHAR(100), sysname VARCHAR(40));

CREATE TABLE documents(id SERIAL PRIMARY KEY, name VARCHAR(224), type INTEGER, student_design_bureau INTEGER);
ALTER TABLE documents ADD CONSTRAINT document_to_student_design_bureau_fkey FOREIGN KEY(student_design_bureau) REFERENCES student_design_bureaus(id);
ALTER TABLE documents ADD CONSTRAINT document_to_type_fkey FOREIGN KEY(type) REFERENCES document_types(id);

CREATE TABLE equipments(id SERIAL PRIMARY KEY, name VARCHAR(100), state BOOLEAN, instruction INTEGER, quantity INTEGER, image VARCHAR(224));

ALTER TABLE equipments ADD CONSTRAINT equipment_to_instruction_fkey FOREIGN KEY(instruction) REFERENCES documents(id);

CREATE TABLE audiences_to_equipments(audience INTEGER, equipment INTEGER);
ALTER TABLE audiences_to_equipments ADD CONSTRAINT audience_fkey FOREIGN KEY(audience) REFERENCES audiences(id);
ALTER TABLE audiences_to_equipments ADD CONSTRAINT equipment_fkey FOREIGN KEY(equipment) REFERENCES equipments(id);

CREATE TABLE schedules(id serial PRIMARY KEY, equipment INTEGER,"user" INTEGER,date_start TIMESTAMP, date_end TIMESTAMP);
ALTER TABLE schedules ADD CONSTRAINT schedules_to_equipment_fkey FOREIGN KEY(equipment) REFERENCES equipments(id);
ALTER TABLE schedules ADD CONSTRAINT schedules_to_user_fkey FOREIGN KEY("user") REFERENCES users(id);

CREATE TABLE contents(id SERIAL PRIMARY KEY, student_design_bureau INTEGER, title VARCHAR(224), text TEXT, date TIMESTAMP, image VARCHAR(224), "user" INTEGER, aspect INTEGER);
ALTER TABLE contents ADD CONSTRAINT content_to_student_design_bureau_fkey FOREIGN KEY(student_design_bureau) REFERENCES student_design_bureaus(id);
ALTER TABLE contents ADD CONSTRAINT content_to_user_fkey FOREIGN KEY("user") REFERENCES users(id);
ALTER TABLE contents ADD CONSTRAINT content_to_aspect_fkey FOREIGN KEY(aspect) REFERENCES aspects(id);

CREATE TABLE student_design_bureaus_to_users("user" INTEGER, student_design_bureau INTEGER);
ALTER TABLE student_design_bureaus_to_users ADD CONSTRAINT student_design_bureau_fkey FOREIGN KEY(student_design_bureau) REFERENCES student_design_bureaus(id);
ALTER TABLE student_design_bureaus_to_users ADD CONSTRAINT user_fkey FOREIGN KEY("user") REFERENCES users(id);

```
CREATE TABLE request_types(id SERIAL PRIMARY KEY, name VARCHAR(224), sysname VARCHAR(224));
CREATE TABLE request_states(id SERIAL PRIMARY KEY, name VARCHAR(224), sysname VARCHAR(224), active BOOLEAN);

CREATE TABLE right_requests(id SERIAL PRIMARY KEY, "user" INTEGER, university INTEGER, student_design_bureau INTEGER,
    type INTEGER, state INTEGER, document_path VARCHAR(224));
ALTER TABLE right_requests ADD CONSTRAINT request_to_univesity_fkey FOREIGN KEY(university) REFERENCES universities(id);
ALTER TABLE right_requests ADD CONSTRAINT request_to_bureau_fkey FOREIGN KEY(student_design_bureau) REFERENCES student_design_bureaus(id);
ALTER TABLE right_requests ADD CONSTRAINT request_user_fkey FOREIGN KEY("user") REFERENCES users(id);
ALTER TABLE right_requests ADD CONSTRAINT request_to_type_fkey FOREIGN KEY(type) REFERENCES request_types(id);
ALTER TABLE right_requests ADD CONSTRAINT request_to_state_fkey FOREIGN KEY(state) REFERENCES request_states(id);
```

ПРИЛОЖЕНИЕ В

Листинг кода для создания иформационного портала

Листинг 1 – создание класса main:

```
package com;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ContentNegotiationConfigurer;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@SpringBootApplication
public class Main {

    public static void main(String[] args) {

        System.out.println(System.getProperty("java.class.path"));
        SpringApplication.run(Main.class, args);
    }

    @Configuration
    public static class PathMatchingConfigurationAdapter extends WebMvcConfigurerAdapter {

        @Override
        public void configureContentNegotiation(ContentNegotiationConfigurer configurer) {
            configurer.favorPathExtension(false);
        }
    }
}
```

Листинг 2 – создание зависимостей:

```
plugins {
    id 'java'
    id 'nu.studer.jooq' version '2.0.9'
}

group ' StudentDesignBureau'
version '1.0-SNAPSHOT'

apply plugin: 'idea'

apply plugin: 'application'

mainClassName = 'com.Main'

sourceCompatibility = 1.8

repositories {
```

```
    mavenCentral()
}

dependencies {

    String bootVersion = '1.5.6.RELEASE'

    testCompile group: 'junit', name: 'junit', version: '4.12'

    compile group: 'org.apache.commons', name: 'commons-lang3', version: '3.7'
    // https://mvnrepository.com/artifact/com.itextpdf/itextpdf
    compile group: 'com.itextpdf', name: 'itextpdf', version: '5.0.6'
    // https://mvnrepository.com/artifact/org.apache.commons/commons-io
    compile group: 'org.apache.commons', name: 'commons-io', version: '1.3.2'

    compile group: 'org.slf4j', name: 'slf4j-api', version: '1.7.2'
    compile group: 'ch.qos.logback', name: 'logback-classic', version: '1.0.9'
    compile group: 'ch.qos.logback', name: 'logback-core', version: '1.0.9'

    compile group: 'org.postgresql', name: 'postgresql', version: '42.1.1'


    compile group: 'com.jolbox', name: 'bonecp', version: '0.8.0.RELEASE'
    compile 'org.jooq:jooq'
    compile 'org.jooq:jooq-meta'
    compile 'org.jooq:jooq-codegen'
    jooqRuntime 'org.postgresql:postgresql:42.1.1'
    jooqRuntime 'postgresql:postgresql:9.1-901.jdbc4'

    compile group: 'com.ibm.icu', name: 'icu4j', version: '62.1'

    compile group: 'com.fasterxml.jackson.core', name: 'jackson-core', version: '2.9.0'

    compile group: 'org.springframework.boot', name: 'spring-boot-starter-thymeleaf', version:
'1.5.6.RELEASE'
    compile group: 'org.springframework.boot', name: 'spring-boot-starter-web', version:
bootVersion
    compile group: 'org.springframework.boot', name: 'spring-boot-starter-security', version:
'1.5.6.RELEASE'

    compile group: 'org.thymeleaf.extras', name: 'thymeleaf-extras-springsecurity4', version:
'2.1.3.RELEASE'

    compile group: 'org.springframework', name: 'spring-tx', version: '4.3.4.RELEASE'

    compile group: 'org.springframework', name: 'spring-jdbc', version: '3.0.4.RELEASE'

    testCompile group: 'org.springframework.boot', name: 'spring-boot-starter-test', version:
bootVersion

}
```

```
def dbAddress = 'jdbc:postgresql://localhost:5432/StudentDesignBureau'

jooq {
    version = '3.10.1'
    edition = 'OSS'
    sample(sourceSets.main) {
        jdbc {
            driver = 'org.postgresql.Driver'
            url = dbAddress
            password = 'root'
            user = 'postgres'
            schema = 'public'
        }
        generator {
            name = 'org.jooq.util.DefaultGenerator'
            strategy {
                name = 'org.jooq.util.DefaultGeneratorStrategy'
            }
            database {
                name = 'org.jooq.util.postgres.PostgresDatabase'
                inputSchema = 'public'
            }
            generate {
                daos = true
            }
            target {
                packageName = 'schema'
                directory = 'src/main/java/com/model/jooq'
            }
        }
    }
}
```

Листинг 3 – создание моделей:

```
package com.model;

import com.model.system.Entity;

import java.sql.Time;
import java.util.Objects;

public class Audience extends Entity {

    private String name;
    private Time timeStart;
    private Time timeEnd;

    public Audience() {
    }

    public Audience(Integer id, String name, Time timeStart, Time timeEnd) {
```

```
    super(id);
    this.name = name;
    this.timeStart = timeStart;
    this.timeEnd = timeEnd;
}

public String getName() {
    return name;
}

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

public Time getTimeStart() {
    return timeStart;
}

public void setTimeStart(Time timeStart) {
    this.timeStart = timeStart;
}

public Time getTimeEnd() {
    return timeEnd;
}

public void setTimeEnd(Time timeEnd) {
    this.timeEnd = timeEnd;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    if (!super.equals(o)) return false;
    Audience audience = (Audience) o;
    return Objects.equals(name, audience.name) &&
        Objects.equals(timeStart, audience.timeStart) &&
        Objects.equals(timeEnd, audience.timeEnd);
}

@Override
public int hashCode() {
    return Objects.hash(super.hashCode(), name, timeStart, timeEnd);
}

@Override
public String toString() {
    return "Audience{" +
        "name='" + name + '\'' +
        ", timeStart=" + timeStart +
        ", timeEnd=" + timeEnd +
```

```
            ", id=" + id +
            '}';
    }
}
```

Листинг 4 – создание DAO:

```
package com.dao;

import com.model.Audience;
import org.jooq.Field;
import org.jooq.impl.TableImpl;
import org.springframework.stereotype.Component;
import schema.tables.records.AudiencesRecord;

import java.sql.Time;
import java.util.List;

import static schema.Sequences.AUDIENCES_ID_SEQ;
import static schema.tables.Audiences.AUDIENCES;
import static
schema.tables.StudentDesignBureausToAudiences.STUDENT_DESIGN_BUREAUS_TO_AUDI
ENCES;

@Component
public class AudienceDao extends EntityDao<Audience> {

    private static final Field[] FIELDS = new Field[]{
            AUDIENCES.ID, AUDIENCES.NAME, AUDIENCES.TIME_END.as("timeEnd"),
AUDIENCES.TIME_START.as("timeStart")
    };

    public Audience create(String name, Time timeEnd, Time timeStart) {

        AudiencesRecord record = dsl.insertInto(AUDIENCES)
                .columns(getFields())
                .values(AUDIENCES_ID_SEQ.nextval(), name, timeEnd, timeStart)
                .returning(AUDIENCES.ID)
                .fetchOne();

        return new Audience(record.getId(), name, timeStart, timeEnd);
    }

    public Audience update(Audience audience) {

        dsl.update(AUDIENCES)
            .set(AUDIENCES.NAME, audience.getName())
            .set(AUDIENCES.TIME_END, audience.getTimeEnd())
            .set(AUDIENCES.TIME_START, audience.getTimeStart())
            .where(AUDIENCES.ID.eq(audience.getId()))
            .execute();

        return audience;
```

```
    }

    public List<Audience> fetch(Integer bureauId) {

        return getBaseSelectQuery()

.join(STUDENT_DESIGN_BUREAUS_TO_AUDIENCES).on(STUDENT_DESIGN_BUREAU
S_TO_AUDIENCES.AUDIENCE.eq(AUDIENCES.ID))

.where(STUDENT_DESIGN_BUREAUS_TO_AUDIENCES.STUDENT_DESIGN_BUREAU.eq
(bureauId))
            .fetchInto(getModel());
    }

    @Override
    public Field[] getFields() {
        return FIELDS;
    }

    @Override
    public TableImpl<?> getTable() {
        return AUDIENCES;
    }

    @Override
    public Class<Audience> getModel() {
        return Audience.class;
    }
}
```

Листинг 5 – Создание сервисов:

```
package com.service;

import com.dao.AudienceDao;
import com.model.Audience;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.sql.Time;

import java.util.List;

@Service
public class AudienceService {
    @Autowired
    private AudienceDao audienceDao;

    public Audience create(String name, Time timeEnd, Time timeStart){
        return audienceDao.create(name,timeEnd,timeStart);
    }

    public Audience find(Integer id) {
```

```
        Audience audience = audienceDao.find(id);
        return audience;
    }

    public List<Audience> fetch(Integer bureauId) {
        return audienceDao.fetch(bureauId);
    }

    public void delete(Integer id) {
        audienceDao.delete(id);
    }

    public Audience update(Audience audience) {
        return audienceDao.update(audience);
    }
}
```

Листинг 6 – Создание контроллеров:

```
package com.controller;

import com.model.Audience;
import com.service.AudienceService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import java.util.List;

import static org.springframework.http.MediaType.APPLICATION_JSON_VALUE;


@Controller
@RequestMapping(value = "/audiences", produces = APPLICATION_JSON_VALUE)
public class AudienceController {
    @Autowired
    private AudienceService audienceService;

    @RequestMapping(method = RequestMethod.POST)
    public void create(@RequestBody final Audience audience) {

        audienceService.create(
            audience.getName(), audience.getTimeEnd(), audience.getTimeStart()
        );
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.DELETE)
    public void delete(@PathVariable final Integer id) {

        audienceService.delete(id);
    }

    @RequestMapping(method = RequestMethod.PUT)
```

```
    public void update(@RequestBody final Audience audience) {

        audienceService.update(audience);
    }

    @ResponseBody
    @RequestMapping(method = RequestMethod.GET)
    public List<Audience> fetch(@RequestParam Integer bureauId) {

        return audienceService.fetch(bureauId);
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.GET)
    public Audience find(@PathVariable final Integer id) throws Throwable {

        return audienceService.find(id);
    }
}
```

Листинг 7 – Создание security:

```
package com.security;

import com.model.User;
import com.service.UserService;
import javassist.NotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.stereotype.Component;

@Component
public class AccountAuthenticationProvider implements AuthenticationProvider {
    private UserService userService;
    @Autowired
    public AccountAuthenticationProvider(UserService userService) {
        this.userService = userService;
    }

    @Override
    public Authentication authenticate(Authentication authentication) throws
AuthenticationException {

        System.out.println("Start auth");
        String email = authentication.getName();
        String password = authentication.getCredentials().toString();
        User user = null;
        try {
            user = userService.authenticateUser(email, password);
        } catch (NotFoundException e) {
            e.printStackTrace();
```

```
        }
        if (user != null) {
            return createSuccessAuthentication(user, authentication);
        }
        return null;
    }

    @Override
    public boolean supports(Class<?> authentication) {
        return UsernamePasswordAuthenticationToken.class.equals(authentication);
    }

    private Authentication createSuccessAuthentication(User principal, Authentication
authentication) {
        UsernamePasswordAuthenticationToken result = new
UsernamePasswordAuthenticationToken(
            principal, authentication.getCredentials(), principal.getAuthorities()
        );
        result.setDetails(authentication.getDetails());
        return result;
    }
}
package com.security;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBu
ilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecuri
ty;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapte
r;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

import javax.sql.DataSource;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    private DataSource dataSource;
```

```
//    private PasswordEncoder encoder;

    private AccountAuthenticationProvider authenticationProvider;

    @Autowired
    public SecurityConfiguration(DataSource dataSource,
                            AccountAuthenticationProvider authenticationProvider) {
        this.dataSource = dataSource;
//        this.encoder = encoder;
        this.authenticationProvider = authenticationProvider;
    }

    @Value("${development.usersquery}")
    private String usersQuery;

    @Value("${development.rolequery}")
    private String rolesQuery;

    @Override
    protected void configure(AuthenticationManagerBuilder auth)
            throws Exception {

        auth
                .authenticationProvider(authenticationProvider)
                .jdbcAuthentication()
//                .passwordEncoder(encoder)
                .usersByUsernameQuery(usersQuery)
                .authoritiesByUsernameQuery(rolesQuery)
                .dataSource(dataSource);
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http.authorizeRequests()
                .antMatchers(HttpMethod.GET, "/").permitAll()

                .antMatchers("/login").permitAll()
                .antMatchers("/register").permitAll()
                .antMatchers(HttpMethod.POST,"/").hasAnyAuthority("admin", "admin_skb")
                .antMatchers(HttpMethod.POST,"/bureaus/*/employees").hasAnyAuthority("admin",
"admin_skb")
//                .antMatchers(HttpMethod.GET, "/api/comment/**").permitAll()
//                .antMatchers(HttpMethod.POST, "/api/comment/**").hasAuthority("USER")
//                .antMatchers("/api/image/**").permitAll()
//                .antMatchers("/api/shop/**").permitAll()
//                .antMatchers("/cart").hasAuthority("USER")
//                .antMatchers("/api/personal").hasAuthority("USER")
//                .antMatchers("/admin/**").hasAuthority("ADMIN")
//                .anyRequest().authenticated()
//
                .and().csrf().disable()
```

```
                .formLogin()
                .loginPage("/login")
//                .defaultSuccessUrl("/")
            .failureUrl("/login?error=true")
                .usernameParameter("login")
                .passwordParameter("password")
                .and().logout()
                .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
                .logoutSuccessUrl("/");
//            .and().exceptionHandling()
//            .accessDeniedPage("/access-denied");

    }

    @Override
    public void configure(WebSecurity web) {
        web
                .ignoring()
                .antMatchers("/resources/**", "/static/**", "/css/**", "/js/**",
                        "/img/**", "/html/**", "/**/favicon.ico", "/vendor/**");
    }
}
```

Листинг 8 – Создание dto:

```
package com.model.dto;

import com.model.Person;
import com.model.University;

import java.util.Objects;

public class EmployeeDto {

    private String profession;
    private String name;
    private String surname;
    private String patronymic;
    private Integer bureauId;
    private Integer userId;

    public EmployeeDto() {
    }

    public EmployeeDto(String profession, String name, String surname, String patronymic, Integer
bureauId, Integer userId) {
        this.profession = profession;
        this.name = name;
        this.surname = surname;
        this.patronymic = patronymic;
        this.bureauId = bureauId;
        this.userId = userId;
    }
```

```
public String getProfession() {
    return profession;
}

public void setProfession(String profession) {
    this.profession = profession;
}

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 Integer getBureauId() {
    return bureauId;
}

public void setBureauId(Integer bureauId) {
    this.bureauId = bureauId;
}

public Integer getUserId() {
    return userId;
}

public void setUserId(Integer userId) {
    this.userId = userId;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
```

```
        if (o == null || getClass() != o.getClass()) return false;
        EmployeeDto that = (EmployeeDto) o;
        return Objects.equals(profession, that.profession) &&
                Objects.equals(name, that.name) &&
                Objects.equals(surname, that.surname) &&
                Objects.equals(patronymic, that.patronymic) &&
                Objects.equals(bureauId, that.bureauId) &&
                Objects.equals(userId, that.userId);
    }

    @Override
    public int hashCode() {
        return Objects.hash(profession, name, surname, patronymic, bureauId, userId);
    }

    @Override
    public String toString() {
        return "EmployeeDto{" +
                "profession='" + profession + '\'' +
                ", name='" + name + '\'' +
                ", surname='" + surname + '\'' +
                ", patronymic='" + patronymic + '\'' +
                ", bureauId=" + bureauId +
                ", userId=" + userId +
                '}';
    }
}
```

Листинг 9 – Создание ConnectionProperties:

```
package com.db;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class ConnectionProperties {

    private String driver;
    private String url;
    private String username;
    private String password;
    private String sqlDialect;

    @Autowired
    public ConnectionProperties(@Value("${development.url}") String url,
                                @Value("${development.driver}") String driverClass,
                                @Value("${development.username}") String username,
                                @Value("${development.password}") String password,
                                @Value("${development.dialect}") String sqlDialect) {

        this.url = url;
```

```
        this.driver = driverClass;
        this.username = username;
        this.password = password;
        this.sqlDialect = sqlDialect;
    }


    public String getDriver() {
        return driver;
    }

    public String getUrl() {
        return url;
    }

    public String getUsername() {
        return username;
    }

    public String getSqlDialect() {
        return sqlDialect;
    }

    public String getPassword() {
        return password;
    }
}
```

Листинг 10 – Создание OOQToSpringExceptionTransformer:

```
package com.db;

import org.jooq.ExecuteContext;
import org.jooq.SQLDialect;
import org.jooq.impl.DefaultExecuteListener;
import org.springframework.jdbc.support.SQLErrorCodeSQLExceptionTranslator;
import org.springframework.jdbc.support.SQLExceptionTranslator;
import org.springframework.jdbc.support.SQLStateSQLExceptionTranslator;


public class JOOQToSpringExceptionTransformer extends DefaultExecuteListener {

    @Override
    public void exception(ExecuteContext ctx) {
        SQLDialect dialect = ctx.configuration().dialect();
        SQLExceptionTranslator translator = (dialect != null)
            ? new SQLErrorCodeSQLExceptionTranslator(dialect.name())
            : new SQLStateSQLExceptionTranslator();

        ctx.exception(translator.translate("jOOQ", ctx.sql(), ctx.sqlException()));
    }
}
```

Листинг 11 – Создание PersistenceContext:

```
package com.db;

import com.jolbox.bonecp.BoneCPDataSource;
import org.jooq.SQLDialect;
import org.jooq.impl.DataSourceConnectionProvider;
import org.jooq.impl.DefaultConfiguration;
import org.jooq.impl.DefaultDSLContext;
import org.jooq.impl.DefaultExecuteListenerProvider;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.context.annotation.Profile;
import org.springframework.core.io.ClassPathResource;
import org.springframework.jdbc.datasource.TransactionAwareDataSourceProxy;
import org.springframework.jdbc.datasource.init.DataSourceInitializer;
import org.springframework.jdbc.datasource.init.ResourceDatabasePopulator;

import javax.sql.DataSource;

@Configuration
public class PersistenceContext {

    @Autowired
    private ConnectionProperties connectionProperties;

    @Autowired
    private DataSource dataSource;

    @Bean
    @Primary
    public DataSource dataSource() {

        final BoneCPDataSource dataSource = new BoneCPDataSource();

        dataSource.setDriverClass(connectionProperties.getDriver());
        dataSource.setJdbcUrl(connectionProperties.getUrl());
        dataSource.setUsername(connectionProperties.getUsername());
        dataSource.setPassword(connectionProperties.getPassword());

        return dataSource;
    }

    @Bean
    public TransactionAwareDataSourceProxy transactionAwareDataSource() {
        return new TransactionAwareDataSourceProxy(dataSource);
    }

    @Bean
    public DataSourceConnectionProvider connectionProvider() {
        return new DataSourceConnectionProvider(transactionAwareDataSource());
```

```
    }

    @Bean
    public JOOQToSpringExceptionTransformer jooqToSpringExceptionTransformer() {
        return new JOOQToSpringExceptionTransformer();
    }


    public DefaultConfiguration configuration() {

        final DefaultConfiguration jooqConfiguration = new DefaultConfiguration();

        jooqConfiguration.set(connectionProvider());
        jooqConfiguration.set(new DefaultExecuteListenerProvider(
                jooqToSpringExceptionTransformer()
        ));

        final String sqlDialectName = connectionProperties.getSqlDialect();
        final SQLDialect dialect = SQLDialect.valueOf(sqlDialectName);
        jooqConfiguration.set(dialect);

        return jooqConfiguration;
    }

    @Bean
    public DefaultDSLContext dsl() {
        return new DefaultDSLContext(configuration());
    }

    @Bean
    public DataSourceInitializer dataSourceInitializer() {

        final DataSourceInitializer initializer = new DataSourceInitializer();
        initializer.setDataSource(dataSource());

        final ResourceDatabasePopulator populator = new ResourceDatabasePopulator();
        populator.addScript(
                new ClassPathResource("schema.sql")
        );

        initializer.setDatabasePopulator(populator);
        return initializer;
    }
}
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____