

Министерство образования и науки

донецкой народной республики

Государственное образовательное учреждение

высшего профессионального образования

«донецкий национальный университет»

Физико-технический факультет

Кафедра компьютерных технологий

Направление подготовки 09.03.01 Информатика и вычислительная техника

**К защите допустить:**

Зав. кафедрой компьютерных технологий

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Ермоленко

(подпись)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**ДИПЛОМНАЯ работа**

на тему: **«Автоматизированная система учета и мониторинга методической деятельности физико-технического факультета»**

Студент: **Фурсов Дмитрий Владимирович** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Научный руководитель: ст. преподаватель **Котенко В.Н.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Работа представлена на кафедру «\_\_\_» \_\_\_\_ 2020 г. рег. № \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись принявшего)

Донецк 2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc41243441)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc41243442)

[1.1 Состояние вопроса 6](#_Toc41243443)

[1.2 Описание существующих бизнес-процессов объекта исследований 9](#_Toc41243444)

[1.3 Актуальность и цель работы 11](#_Toc41243445)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 13](#_Toc41243446)

[2.1 Описание области применения и исходных данных системы 13](#_Toc41243447)

[2.2 Требования к пользовательским интерфейсам 13](#_Toc41243448)

[2.3 Требования к аппаратным, программным и коммуникационным интерфейсам 17](#_Toc41243449)

[2.4 Требования к пользователям продукта 18](#_Toc41243450)

[2.5 Требования к адаптации на месте 18](#_Toc41243451)

[2.6 Функции продукта 19](#_Toc41243452)

[2.7 Сценарии работы системы 19](#_Toc41243453)

[3 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ И ПРОГРАММНЫХ СРЕДСТВ 27](#_Toc41243454)

[3.1 Язык программирования серверных скриптов PHP 27](#_Toc41243455)

[3.2 Реляционная система управления базами данных MYSQL 27](#_Toc41243456)

[3.3 WEB-сервер APACHE 27](#_Toc41243457)

[3.4 Laravel 5 28](#_Toc41243458)

[3.5 Bootstrap 4 28](#_Toc41243459)

[3.6 jQuery 29](#_Toc41243460)

[3.7 AJAX 29](#_Toc41243461)

[3.8 Архитектура «клиент-сервер» 30](#_Toc41243462)

[4. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ 31](#_Toc41243463)

[4.1 Входные и выходные данные 31](#_Toc41243464)

[4.2 Проектирование 31](#_Toc41243465)

[4.2.1 Проектирование структуры web-приложения 31](#_Toc41243466)

[4.2.2 Концептуальное проектирование БД 34](#_Toc41243467)

[4.2.3 Логическое проектирование БД 37](#_Toc41243468)

[4.2.4 Физическое проектирование БД 38](#_Toc41243469)

[4.3 Разработка и описание web-приложения и базы данных 41](#_Toc41243470)

[4.3.1 Аутентификация средствами Laravel 42](#_Toc41243471)

[4.3.2 Миграции для базы данных 43](#_Toc41243472)

[4.3.3 Классы-наполнители для базы данных 52](#_Toc41243473)

[4.3.4 Реализация базы данных в ORM Eloquent 58](#_Toc41243474)

[4.3.5 Шаблоны 59](#_Toc41243475)

[4.3.6 Контроллеры 60](#_Toc41243476)

[4.3.7 Маршруты 62](#_Toc41243477)

[4.3.8 Посредники 63](#_Toc41243478)

[4.4 Общая структура web-приложения 64](#_Toc41243479)

[5. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 66](#_Toc41243480)

[5.1 Аппаратные, системные и программные требования 66](#_Toc41243481)

[5.2 Тестирование веб-сайта 66](#_Toc41243482)

[5.3 Описание установки web-приложения. 68](#_Toc41243483)

[5.3.1 Установка, настройка и запуск Open Server 68](#_Toc41243484)

[5.3.2 Установка и настройка Composer и Laravel 69](#_Toc41243485)

[5.3.3 Установка, настройка и запуск сервера баз данных MySQL 69](#_Toc41243486)

[5.3.4 Настройка и запуск web-приложения 70](#_Toc41243487)

[5.4 Описание контрольных примеров 70](#_Toc41243488)

[6 ОХРАНА ТРУДА НА РАБОЧЕМ МЕСТЕ 72](#_Toc41243489)

[6.1 Общие сведения 72](#_Toc41243490)

[6.2 Опасные и вредные производственные факторы рабочего места программиста 73](#_Toc41243491)

[6.2.1 Данные микроклимата помещения 73](#_Toc41243492)

[6.2.2 Освещение рабочего места 74](#_Toc41243493)

[6.2.3 Воздействие электромагнитного излучения 76](#_Toc41243494)

[6.2.4 Шум и вибрация. Статическое электричество 77](#_Toc41243495)

[ЗАКЛЮЧЕНИЕ 79](#_Toc41243496)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 80](#_Toc41243497)

[ПРИЛОЖЕНИЕ А Экранные формы 82](#_Toc41243498)

[ПРИЛОЖЕНИЕ Б Фрагменты листинга 85](#_Toc41243499)

# ВВЕДЕНИЕ

В последнее десятилетие благодаря бурному развитию Интернета в программировании выделяют отдельное направление – Web-программирование. Поначалу оно не могло и сравниться по своей сложности с другими областями программистского ремесла, не "дотягиваясь" не только до системного, но даже и до прикладного уровня. В наши дни, однако, роль Web-программирования в структуре глобальной сети возрастает, соответственно увеличивается и средняя оценка сложности сценариев. Многие системы (например, поисковые) по объему кода приближаются к размеру исходных кодов серьезных прикладных программ. Доля же статических страниц в Web постоянно падает; на смену им приходят динамические страницы, сгенерированные автоматически тем или иным сценарием [1].

Методическая работа является важнейшей составной частью деятельности руководства и Ученого совета вуза, факультетов, кафедр, центров и лабораторий, всего профессорско-преподавательского, научного и учебно-вспомогательного состава вуза. Методическая работа в вузе осуществляется с целью качественной разработки образовательных программ, обеспечения эффективной организации учебно-воспитательного процесса, качественного проведения всех видов учебных занятий, повышения профессионального уровня руководящего, профессорско-преподавательского, научного и учебно-вспомогательного состава вуза [2].

В данной дипломной работе будет разработана автоматизированная система учета и мониторинга методической деятельности физико-технического факультета.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Состояние вопроса

Результаты учебно-методической деятельности университета являются важнейшей составляющей его успешности и конкурентоспособности на рынке образования. Над проблемой повышения результативности и эффективности деятельностей, приведенных выше, в последние годы работают не только различные государственные инстанции, но сотрудники различных вузов, в том числе и ДОННУ.

Уровень целостного образовательного процесса в вузе в большей мере зависит от его организации, профориентационной работы, разработанных учебных планов, контроля и оценки качества образования и т.п. Особое внимание уделяется квалификационным требованиям к сотрудникам, в которых выделены показатели учебно-методической работы. В основном эти виды деятельности относятся к преподавателям вуза и направлены на повышение уровня их научно-исследовательской деятельности, воспитания и квалификации будущих специалистов.

На сегодняшний день существует множество разных видов изданий. Список основных видов изданий приведен в таблице 1.1.

Таблица 1.1 ­– Основные виды изданий

|  |  |
| --- | --- |
| Вид издания | Описание |
| Учебное пособие | Учебное издание, дополняющее или частично (полностью) заменяющее учебник, официально утвержденное в качестве данного вида издания. |
| Учебное наглядное пособие | Учебное издание, содержащее материалы в помощь изучению, преподаванию или воспитанию. |
| Учебно-методическое пособие | Учебное издание, содержащее материалы по методике преподавания учебной дисциплины (ее раздела, части) или по методике воспитания. |
| Учебник | Учебное издание, содержащие систематическое изложение учебной дисциплины (ее раздела, части), соответствующее учебной программе и официально утвержденное в качестве данного вида издания. |
| Хрестоматия | Учебное пособие, содержащее литературно-художественные, исторические и иные произведения или отрывки из них, составляющие объект изучения учебной дисциплины. |
| Методические рекомендации (методические указания) | Учебное издание, содержащее разъяснения по определенной теме, разделу или вопросу учебной дисциплины, роду практической деятельности, определяющее методику выполнения отдельных заданий, определенного вида работ или мероприятий. |
| Курс лекций | Учебное издание, содержащее полное изложение тем учебной дисциплины, определенных программой. |
| Текст лекций | Учебной издание, содержащее изложение материала определенных разделов учебной дисциплины. |
| Конспект лекции | Учебное издание, содержащее краткое изложение курса лекций или отдельных разделов учебной дисциплины. |
| Учебная программа | Учебное издание, определяющее содержание, объем, а также порядок изучения и преподавания какой-либо учебной дисциплины (ее раздела, части). |
| Практикум | Учебное издание, содержащее практические задания и упражнения, способствующие усвоению пройденного. |
| Словарь | Справочное издание, содержащее упорядоченный перечень языковых единиц (слов, словосочетаний, фраз, терминов, имен, знаков), снабженных относящимися к ним справочными данными. |
| Энциклопедия | Справочное издание, содержащее в обобщенном виде основные сведения по одной или всем отраслям знаний и практической деятельности, изложенные в виде кратких статей, расположенных в алфавитном порядке. |
| Справочник | Справочное издание, носящее прикладной, практический характер, имеющее систематическую структуру или построенное по алфавиту заглавий статей. |

## 1.2 Описание существующих бизнес-процессов объекта исследований

Произведём оценку системы в терминах бизнес-процессов организации с помощью методологии IDEF0.

IDEF0 – это очень простой и одновременно наглядный язык описания бизнес-процессов. С помощью этого стандарта возможна передача информации между разработчиками, консультантами и пользователями. Функциональная модель IDEF0 представляет собой набор блоков, каждый из которых представляет собой «черный ящик» со входами и выходами, управлением и механизмами, которые детализируются (декомпозируются) до необходимого уровня. Соединяются блоки между собой при помощи стрелок и описаний функциональных блоков [3].

Стрелки и их вхождение обуславливают тип интерфейса:

1. если стрелка указывает на блок сверху – это управляющая информация;
2. если стрелка указывает на блок слева – это входная информация;
3. если стрелка указывает на блок справа – это результаты;
4. если стрелка указывает на блок снизу – это механизм.

Контекстная диаграмма, описывающая функционирование учета методической деятельности физико-технического факультета приведена на рисунке 1.1.



Рисунок 1.1 – Контекстная диаграмма

На рисунке 1.2 показана диаграмма декомпозиции.



Рисунок 1.2 – Диаграмма декомпозиции

## 1.3 Актуальность и цель работы

Модернизация системы образования рассматривается сегодня как потенциал инновационного развития общества, залог будущего благополучия граждан и безопасности страны. И это закономерно, так как именно изменения в образовании во многом обеспечивают интенсивное движение экономических, социальных и политических процессов в современном мире.

Преподаватель – ключевая фигура реформирования образования. В стремительно меняющемся открытом мире человеку предъявляются жесткие требования – это высокое качество образования, коммуникабельность, целеустремленность, креативность. Обучить человека современному мышлению, способного самореализоваться в жизни, могут только преподаватели, обладающие высоким профессионализмом.

Не секрет, что образовательные результаты студентов напрямую зависят от качества преподавания. Данный постулат объясняет актуальность эффективного механизма профессионального развития преподавателей, каким является методическая работа, и требует её развития и совершенствования.

Поскольку методическая работа может существенно влиять на качество обучения, на конечные результаты работы образовательного учреждения, можно рассматривать ее как важный фактор управления образовательным процессом [4].

Ситуация, описанная выше, обосновывает актуальность методической работы. Однако, имея большое количество видов методических изданий, преподавателей, кафедр и направлений, учёт методических изданий становится всё сложнее и запутаннее. В данном случае решением является автоматизированная система, обеспечивающая удобный способ учёта и мониторинга, а также формирования отчётов. На данный момент, в открытом доступе, практически нет систем, которые смогли бы выполнить все поставленные требования. Ближайшие аналоги, выполняющие похожие функции, это различные системы учёта библиотек, такие как 1С:Библиотека (рисунок 1.1).

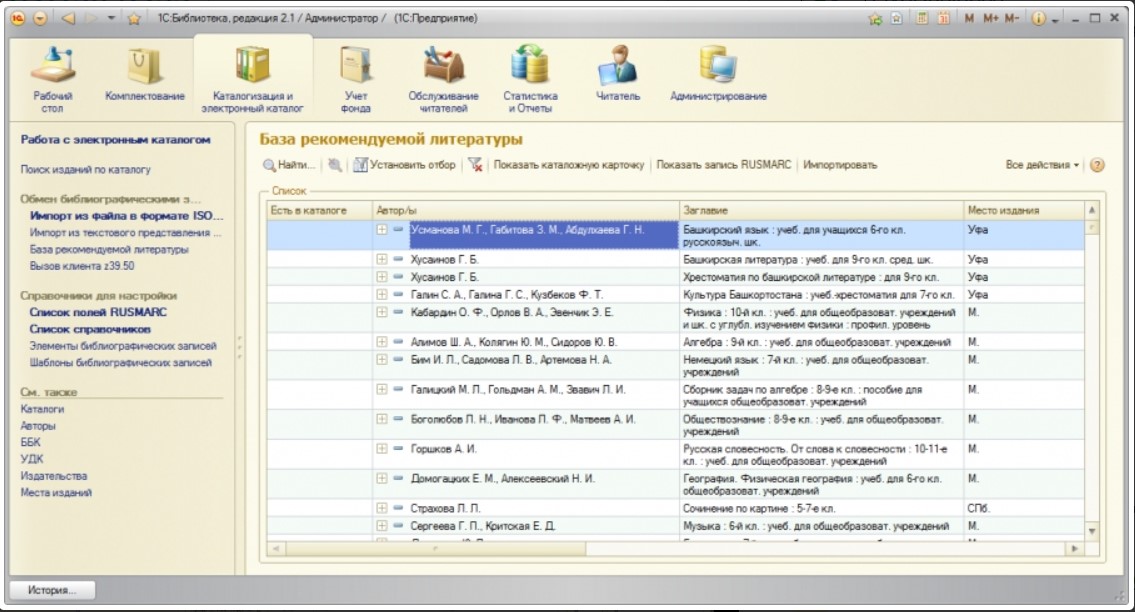


Рисунок 1.1 Скриншот из программы 1С:Библиотека

Из явных недостатков данной программы можно выделить наличие системы «1С:Предприятие» для использования, трудный для понимания пользовательский интерфейс, а также лишние функции, которые не нужны для выполнения поставленной задачи. Данную систему можно дописать, чтобы она смогла выполнить необходимые требования, однако для этого пользователю необходимо иметь нужные навыки и выделить немалое количество времени.

# 2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## 2.1 Описание области применения и исходных данных системы

Данное приложение предназначено для преподавателей и заведующим методической деятельности физико-технического факультета.

Исходными данными для сайта является информация о кафедрах, направлениях подготовки, видах методических изданий и преподавателях.

## 2.2 Требования к пользовательским интерфейсам

Пользовательский интерфейс должен предполагать черно-белые оттенки, анимацию для выделения элементов на сайте, навигационное меню с возможностью просмотра всех изданий, добавления новых изданий, формирование отчета, взаимодействия с данными о факультете, входом и выходом из аккаунта, подвал с информацией о факультете.

Пример навигационного меню приведен на рисунке 2.1



Рисунок 2.1 – Пример навигационного меню

Пример подвала сайта приведен на рисунке 2.2

Рисунок 2.2 – Пример подвала сайта

Перед входом на главную страницу отправить пользователя на страницу авторизации. Пример формы авторизации приведен на рисунке 2.3



Рисунок 2.3 – Пример формы авторизации

На главной странице вывести таблицу с информацией о методических изданиях физико-технического факультета, кнопки для редактирования и удаления записей (рисунок 2.4), фильтры для поиска нужных изданий по кафедрам, дисциплинам, авторам, видам изданий и году выпуска. Год издания сделать в виде текстового поля, остальные фильтры в виде выпадающего списка с возможностью поиска.



Рисунок 2.4 – Пример таблицы методических изданий

Все возможные данные, которые предоставляет база данных пользователю, отобразить пользователю в виде выпадающего списка с возможностью поиска и выбора нескольких вариантов (рисунок 2.5).



Рисунок 2.5 – Пример выпадающего списка

На рисунке 2.6 изображен общий вид веб-интерфейса системы. Такой интерфейс позволит сконцентрировать внимание пользователя на основной части сайта.



Рисунок 2.6 – Общий вид веб-интерфейса системы

Необходимо формировать отчёты о количествах опубликованных изданий в виде таблицы, где в качестве колонок указана кафедра или вид издания, и выбранные пользователем годы выпуска издания (рисунок 2.7).



Рисунок 2.7 – Пример отчёта

Также, для лучшего восприятия информации, помимо таблицы, необходимо формировать диаграмму, на основе этих данных (рисунок 2.8).



Рисунок 2.8 – Пример диаграммы

## 2.3 Требования к аппаратным, программным и коммуникационным интерфейсам

Для установки и работы программы необходимо иметь вычислительную систему следующей базовой конфигурации:

1. Windows (32-bit или 64-bit): Windows 8 / Windows 7 / Windows Server 2008 / Windows Vista / Windows XP SP3;
2. процессор: 2.0 ГГц;
3. оперативная память: 512 Mb;
4. HDD: 1 Гб.

Необходимо обеспечить программное взаимодействие системы с:

1. набором библиотек Microsoft Visual C++ 2005-2008-2010 Redistributable Package x86;
2. PHP версии 7.2 и выше;
3. Composer;
4. веб-фреймворк Laravel версии 5.8;
5. библиотекой  jQuery и набором инструментов Bootstrap 4;
6. web-сервером Apache;
7. сервером управления базами данных (БД) MySQL.

## 2.4 Требования к пользователям продукта

Системой должны иметь возможность пользоваться следующие категории пользователей: администратор, модератор, преподаватель.

Администратор:

1. базовыми навыками владения персональным компьютером;
2. базовые навыки работы с базой данных;
3. понимание принципов функционирования web-приложения;

Модератор:

1. базовыми навыками владения персональным компьютером;
2. базовые навыки работы с базой данных;

Преподаватель:

1. базовыми навыками владения персональным компьютером;

## 2.5 Требования к адаптации на месте

Необходимы программы-инсталляторы сервера html-страниц Apache, сервера баз данных MySQL, языка разработки серверных скриптов PHP, пакетного менеджера Composer, интернет-соединение для установки зависимостей Laravel.

## 2.6 Функции продукта

Основные функции:

1. cоздать миграции и наполнители для создания и заполнения данными о физико-техническом факультете;
2. авторизация и аутентификация пользователей;
3. запрет доступа к данным неавторизованным пользователям с помощью посредника;
4. подвал с отображением информации о назначении сайта и о физико-техническом факультете.
5. просмотр списка методических изданий;
6. добавление нового издания;
7. редактирование существующего издания;
8. удаление издания;
9. фильтрация списка методических изданий по авторам, дисциплинам и году издания;
10. добавление авторов, дисциплин и видов изданий;
11. редактирование существующих авторов, дисциплин и видов изданий;
12. удаление авторов, дисциплин и видов изданий, а также связанные с ними методические издания.
13. Формирование отчётов по кафедрам в виде таблицы и диаграммы;
14. Формирование отчётов по видам изданий в виде таблицы и диаграммы.

## 2.7 Сценарии работы системы

Сценарий «Одобренная авторизация пользователя»:

1. пользователь переходит на сайт по доменному имени;
2. вводит свой логин и пароль, нажимает кнопку «Войти»;
3. сайт проверяет введенные пользователем данные и находит пользователя с таким логином и паролем;
4. система загружает главную страницу с навигационным меню, находящемуся в секции “Header” (рисунок 2.6), выпадающими меню для фильтрации, таблицей методических изданий, кнопками для взаимодействия с изданиями и подвал сайта.

Сценарий «Ошибка авторизации пользователя»:

1. пользователь переходит на сайт по доменному имени;
2. вводит свой логин и пароль, нажимает кнопку «Войти»;
3. сайт проверяет введенные пользователем данные и не находит пользователя с таким логином и паролем;
4. система оставляет пользователя на странице авторизации и выводит сообщение об ошибке.

Сценарий «Переход на главную страницу от имени администратора»:

1. пользователь переходит на сайт по доменному имени и авторизуется;
2. на навигационном меню появляются кнопки «Просмотр издания», «Новое издание», «Взаимодействие с базой», «Сформировать отчёт» и имя авторизированного пользователя;
3. администратор имеет доступ ко всем методическим изданиям факультета;

Сценарий «Переход на главную страницу от имени модератора»:

1. пользователь переходит на сайт по доменному имени и авторизуется;
2. на навигационном меню появляются кнопки «Просмотр издания», «Новое издание», «Сформировать отчёт» и имя авторизированного пользователя;
3. модератор имеет доступ ко всем методическим изданиям факультета.

Сценарий «Переход на главную страницу от имени пользователя»:

1. пользователь переходит на сайт по доменному имени и авторизуется;
2. на навигационном меню появляются кнопки «Просмотр издания», «Новое издание» и имя авторизированного пользователя;
3. пользователь имеет доступ только к своим методическим изданиям.

Сценарий «Просмотр методических изданий от имени администратора или модератора»:

1. пользователь нажимает на логотип или кнопку «Просмотреть все издания» в навигационном меню;
2. система загружает главную страницу;
3. система проверяет права авторизованного пользователя, обнаруживает права администратора или модератора и дает доступ ко всем изданиям.
4. система проверяет наличие фильтров, не находит таковых и отправляет все данные;
5. cайт выводит в секции “Content” (рисунок 2.6) информацию в виде таблицы о методических изданиях кафедры компьютерных технологий.

Сценарий «Просмотр методических изданий от имени пользователя»:

1. пользователь нажимает на логотип или кнопку «Просмотреть все издания» в навигационном меню;
2. система загружает главную страницу;
3. система проверяет права авторизованного пользователя, у пользователя нет прав администратора или модератора, то выводит издания только этого пользователя.
4. система проверяет наличие фильтров, не находит таковых и отправляет все данные;
5. cайт выводит в секции “Content” (рисунок 2.6) информацию в виде таблицы о методических изданиях кафедры компьютерных технологий.

Сценарий «Просмотр методических изданий с фильтрами»:

1. пользователь нажимает на логотип или кнопку «Просмотреть все издания» в навигационном меню;
2. система загружает главную страницу;
3. система проверяет права авторизованного пользователя, если у пользователя нет прав администратора или модератора, то выводит издания только этого пользователя.
4. пользователь выбирает нужную кафедру, дисциплину, автора, тип издания, статус издания, вводит год издания и нажимает кнопку «Отфильтровать».
5. система проверяет наличие фильтров, находит нужные данные и отправляет пользователю;
6. сайт выводит в секции “Content” (рисунок 2.6) информацию в виде таблицы о методических изданиях физико-технического факультета с выбранными фильтрами.

Сценарий «Одобренное новое издание»:

1. пользователь нажимает на кнопку «Новое издание» в навигационном меню;
2. система загружает новую страницу с текстовыми полями для заполнения и выпадающими списками для данных о кафедре и кнопкой «Создать»;
3. по желанию, пользователь может добавить документ для онлайн просмотра.
4. пользователь вводит необходимые данные и нажимает на кнопку «Создать»;
5. система проверяет и подтверждает правильность ввода данных и отправляет пользователя на главную страницу с сообщением об удачной операции вместе со списком методических изданий.

Сценарий «Ошибка создания нового издания»:

1. пользователь нажимает на кнопку «Новое издание» в навигационном меню;
2. система загружает новую страницу с текстовыми полями для заполнения и выпадающими списками для данных о кафедре и кнопкой «Создать»;
3. система проверяет введенные пользователем данные и находит ошибку;
4. система оставляет пользователя на странице создания и выводит причину ошибки.

Сценарий «Одобренное редактирование существующего издания»:

1. пользователь нажимает на кнопку «Редактировать» напротив издания;
2. сайт переводит пользователя на страницу с заполненными, существующей информацией об издании, полями и выпадающими списками, и кнопкой «Обновить»;
3. система проверяет и подтверждает правильность ввода данных и отправляет пользователя на главную страницу с сообщением об удачной операции вместе со списком методических изданий.

Сценарий «Ошибка редактирования существующего издания»:

1. пользователь нажимает на кнопку «Редактировать» напротив издания;
2. сайт переводит пользователя на страницу с заполненными, существующей информацией об издании, полями и выпадающими списками, и кнопкой «Обновить»;
3. система проверяет и находит ошибку в введенных данных;
4. система отправляет пользователя на страницу редактирования с изначальными данными и сообщением о причине ошибки.

Сценарий «Удаление издания»:

1. пользователь нажимает на кнопку «Удалить» напротив издания;
2. система находит данное издание и удаляет из базы данных;
3. сайт обновляет страницу и оповещает пользователя об успешном удалении издания.

Сценарий «Онлайн-просмотр и скачивание методического издания»:

1. пользователь переходит на главную страницу с изданиями;
2. пользователь нажимает в колонке «Кнопки управления» на кнопку «Открыть в браузере». Если для этого издания нет документа, то кнопки не будет;
3. пользователь переходит на страницу с онлайн-просмотром документа;
4. по желанию пользователь может нажать на кнопку «Скачать» чтобы загрузить документ на устройство.

Сценарий «Добавление информации о факультете»:

1. пользователь нажимает на выпадающее меню «Взаимодействия с базой» и выбирает пункт «Добавить в базу»;
2. система переводит пользователя на страницу добавления, с выпадающим списком возможных таблиц для добавления информации.
3. пользователь выбирает таблицу в выпадающем списке;
4. система создает необходимые поля для заполнения;
5. пользователь заполняет нужные поля и нажимает кнопку «Добавить»;
6. система добавляет данные в нужную таблицу и отправляет пользователя обратно, с сообщением об успешном дополнении к таблице.

Сценарий «Редактировать информацию о факультете»:

1. пользователь нажимает на выпадающее меню «Взаимодействия с базой» и выбирает пункт «Обновить в базе»;
2. система переводит пользователя на страницу выбора таблицы для редактирования с выпадающим списком возможных таблиц и кнопкой «Далее»;
3. пользователь выбирает таблицу;
4. система создает еще один выпадающий список с данными из выбранной таблицы;
5. пользователь выбирает нужные данные и нажимает «Далее»;
6. система создает страницу с нужными полями для заполнения;
7. пользователь заполняет поля новыми данными и нажимает кнопку «Обновить»;
8. система редактирует данные в нужной таблице и отправляет пользователя на страницу выбора таблицы для редактирования, с сообщением об успешном редактировании данных.

Сценарий «Удалить информацию о факультете»:

1. пользователь нажимает на выпадающее меню “Взаимодействия с базой” и выбирает пункт “Удалить из базы”;
2. система переводит пользователя на страницу выбора таблицы для удаления с выпадающим списком возможных таблиц и кнопкой “Далее”;
3. пользователь выбирает таблицу,
4. система создает еще один выпадающий список с данными из этой таблицы;
5. пользователь выбирает данные из таблицы и нажимает кнопку “Удалить”;
6. система удаляет данные в нужной таблице, а также все связанные с ней методические издания, и отправляет пользователя на страницу выбора таблицы для удаления, с сообщением об успешном удалении данных.

Сценарий «Сформировать отчёт по кафедрам»:

1. пользователь нажимает на выпадающее меню “Сформировать отчёт” и выбирает пункт “По кафедрам”;
2. система переводит пользователя на формирования отчёта с выпадающим списком кафедр и текстовым полем для ввода года издания;
3. пользователь выбирает нужные кафедры из списка и вписывает нужные годы изданий через запятую.
4. система формирует пользователю страницу с таблицей и диаграммой (рисунок 2.8);

Сценарий «Сформировать отчёт по видам изданий»:

1. пользователь нажимает на выпадающее меню “Сформировать отчёт” и выбирает пункт “По кафедрам”;
2. система переводит пользователя на формирования отчёта с выпадающим списком видов изданий и текстовым полем для ввода года издания;
3. пользователь выбирает нужные кафедры из списка и вписывает нужные годы изданий через запятую.
4. система формирует пользователю страницу с таблицей и диаграммой (рисунок 2.8);

**2.8 Ограничения**

1. требуется соединение с web-сервером;
2. требуется MySQL версии 5.7 или выше;
3. требуется PHP версии 7.2 или выше;
4. не гарантируется корректная работа web-приложения, если пользователь будет изменять содержимое таблиц базы данных сторонними программами.

# 3 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ И ПРОГРАММНЫХ СРЕДСТВ

## 3.1 Язык программирования серверных скриптов PHP

PHP – это распространенный язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML.

Главная область применения PHP - написание скриптов, работающих на стороне сервера; таким образом, PHP способен обрабатывать данные форм, генерировать динамические страницы или отсылать и принимать cookies [5].

## 3.2 Реляционная система управления базами данных MYSQL

База данных сайта MySQL – это система, предназначенная для хранения и обработки информации. Комплекс таблиц, взаимосвязанных между собой, для доступа к которым применяется система управления базами данных (СУБД) MySQL. По сути, MySQL – это специальная программа с открытым кодом, которая используется на сервере SQL. Данная программа не способна обрабатывать большое количество информации, однако она идеальна для небольших и крупных веб-ресурсов [6].

## 3.3 WEB-сервер APACHE

Apache – это программное обеспечение с открытым исходным кодом, веб-сервер, [который обеспечивает работу около 46% сайтов](https://w3techs.com/technologies/details/ws-apache/all/all) по всему миру.

Преимущества:

1. Бесплатный даже для использования в коммерческих целях.
2. Надёжный, стабильное программное обеспечение.
3. Часто обновляемый, регулярные патчи безопасности.
4. Гибкий благодаря своей модульной структуре.
5. Легко настраиваемый, дружелюбный для начинающих.
6. Большое сообщество и легко доступная поддержка в случае любой проблемы [7].

## 3.4 Laravel 5

Laravel – PHP-фреймворк, быстро набирающий популярность среди веб-разработчиков и помогающий обеспечить максимально удобное [создание сайтов](http://it-devgroup.com/) различного уровня сложности с изящным и четко структурированным синтаксисом и архитектурой.

Laravel является достаточно гибким фреймворком и позволяет решать нестандартные задачи, структурировать веб-сайт в соответствии с существующей логикой и поставленными целями [8].

## 3.5 Bootstrap 4

Bootstrap – это фреймворк на основе HTML и CSS. Он содержит стили для основных элементов, которые применяются в верстке. Использование такого фреймворка значительно ускоряет процесс создания страниц. Стандартные стили легко менять, что обеспечивает гибкий и простой процесс создания макетов сайтов [9].

Преимущества, которые даёт фреймворк Bootstrap:

1. высокая скорость создания качественной адаптивной вёрстки даже начинающими веб-разработчиками;
2. кроссбраузерность и кроссплатформенность;
3. наличие большого количество готовых хорошо продуманных компонентов;
4. низкий порог вхождения;
5. наличие хорошо продуманного дизайна компонентов и согласованности.

## 3.6 jQuery

jQuery — специальная библиотека, написанная на JavaScript, в которой многие типовые действия прописаны как одна команда. Библиотека jQuery помогает легко получать доступ к любому элементу [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), обращаться к атрибутам и содержимому элементов [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), манипулировать ими. Также библиотека jQuery предоставляет удобный [API](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для работы с [AJAX](https://ru.wikipedia.org/wiki/AJAX) [10].

Преимущества jQuery:

1. кроссбраузерность;
2. компактность кода;
3. удобная работа с событиями и визуальными эффектами;
4. понятная и локализированная документация.

## 3.7 AJAX

AJAX – способ построения интерактивных пользовательских web-приложений посредством фонового обмена информацией браузера с сервером.

AJAX базируется на технологии обращения к серверу без перезагрузки страницы или использовании DHTML, позволяющего динамически изменять содержимое. Формат передачи данных – XML или JSON. AJAX можно реализовать в разных языках программирования: PHP, Ruby on Rails, ASP.NET и других. В коде web-страниц широко используется JavaScript для прозрачного обмена данными клиента с сервером. Пользователи взаимодействуют со стандартными HTML элементами, динамическое поведение которых описывается на JavaScript [11].

Преимущества AJAX:

1. экономия трафика пользователя, так как вместо обновления всей страницы, загружается ее небольшая изменившаяся часть;
2. снижение нагрузки на сервер. серверу не нужно повторно создавать страницу и передавать ее пользователю;
3. ускорение реагирования интерфейса на команды пользователя.

## 3.8 Архитектура «клиент-сервер»

Данная концепция нам говорит, что нужно разделять машины в сети на клиентские, которым что-то нужно и на серверные, которые отправляют то, что нужно.

Преимуществом модели взаимодействия клиент-сервер является то, что программный код клиентского приложения и серверного разделен. Если мы говорим про локальные компьютерные сети, то к преимуществам архитектуры клиент-сервер можно отнести пониженные требования к машинам клиентов, так как большая часть вычислительных операций будет производиться на сервере, а также архитектура клиент-сервер довольно гибкая и позволяет администратору сделать локальную сеть более защищенной [12]. Пример модели взаимодействия архитектуры «клиент-сервер» изображен на рисунке 3.1.



Рисунок 3.1 – Модель взаимодействия клиент-сервер

## 4. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

## 4.1 Входные и выходные данные

Входные данные динамического сайта:

1. список кафедр физико-технического факультета;
2. список дисциплин;
3. список преподавателей;
4. список видов методических изданий;
5. формат бумаги и тип обложки;
6. количество страниц;
7. тираж;
8. год и месяц выпуска;
9. контактный телефон преподавателя.

Результатом работы приложения является список методических изданий и отчёты о количестве выпущенных изданий физико-технического факультета.

## 4.2 Проектирование

### 4.2.1 Проектирование структуры web-приложения

Основные элементы, которые будут использоваться при построении web-приложения:

1. HTML;
2. CSS;
3. PHP;
4. Bootstrap 4;
5. фреймворк Laravel;
6. MySQL;
7. JavaScript;
8. jQuery
9. AJAX.

Структуру web-приложения сформируем с помощью Laravel   
(рисунок 4.1). Структура приложения Laravel спроектирована таким образом, чтобы стать удобной отправной точкой для динамического сайта.

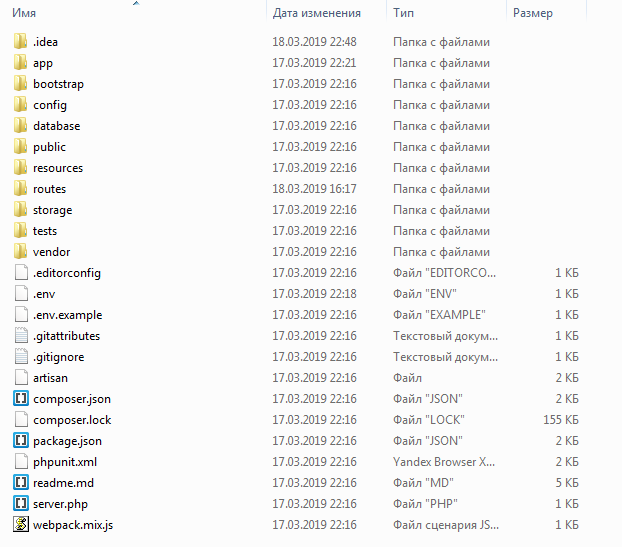


Рисунок 4.1 – Структура проекта

Подробнее о структуре каталога можно узнать из таблицы 4.1.

Таблица 4.1. Структура корневого каталога

|  |  |
| --- | --- |
| Название папки | Описание |
| app | Содержит классы, предоставляющие API к вашему приложению по протоколам CLI (командная строка) и HTTP (работа через браузер). В таблице 4.2 приведена подробная структура. |
| bootstrap | Содержит файлы, которые осуществляют первоначальную загрузку (bootstraping) фреймворка и настраивают автозагрузку классов. |
| Config | Здесь находятся конфигурационные файлы приложения. |
| database | Каталог для файлов миграций БД и "посева" данных. |
| public | Является DocumentRoot домена вашего приложения и содержит статические файлы - css, js, изображения и т.п. |
| resources | Здесь находятся шаблоны (Views), файлы локализации и, если таковые имеются, рабочие файлы LESS, SASS и js-приложения на фреймворках типа AngularJS или Ember, которые потом собираются внешним инструментом в папку public. |
| storage | Этот каталог должен иметь права для записи в него извне и в нём Laravel хранит скомпилированные Blade-шаблоны, файлы сессии, файловый кэш и другие сгенерированные файлы, нужные для работы |
| test | Каталог для юнит-тестов |
| vendor | В этот каталог Composer устанавливает пакеты, указанные в composer.json. |

Таблица 4.2. Структура каталога “app”

|  |  |
| --- | --- |
| Console | Находятся классы Artisan-команд |
| Http | Контроллеры, фильтры и реквесты (классы валидации пользовательского ввода) |
| Exceptions | Cодержит обработчик исключений приложения |
| Providers | Cодержит все сервис-провайдеры приложения |

### 4.2.2 Концептуальное проектирование БД

Концептуальное проектирование – построение семантической модели предметной области (информационной модели самого высокого уровня абстракции). Концептуальная модель базы данных включает в себя:

1. объекты, обозначающие вещи, обозначаются в виде прямоугольника;
2. атрибуты, характеризующие объект – в виде овала;
3. связи между объектами – ромбами.

В ходе изучения предметной области выделены следующие сущности, представляющие информационное значение для задачи:

1. методические издания – информация об изданиях физико-технического факультета;
2. кафедра – информация о кафедрах физико-технического факультета;
3. дисциплина – информация об имеющихся дисциплинах на физико-техническм факультете;
4. пользователь – информация о пользователях автоматизированной системы;
5. роль – информация о категориях пользователей автоматизированной системы;
6. автор – информация об авторах методических изданий;
7. вид издания – информация о видах методических изданий;
8. обложка – информация о типах обложек, которые будут использованы в методических изданиях;
9. формат бумаги – информация о формате бумаги, которую будут использовать для методических изданий;
10. месяц издания – информация о месяцах, в которых будут изданы методические издания.

Концептуальное представление базы данных приведено на рисунке 4.2.

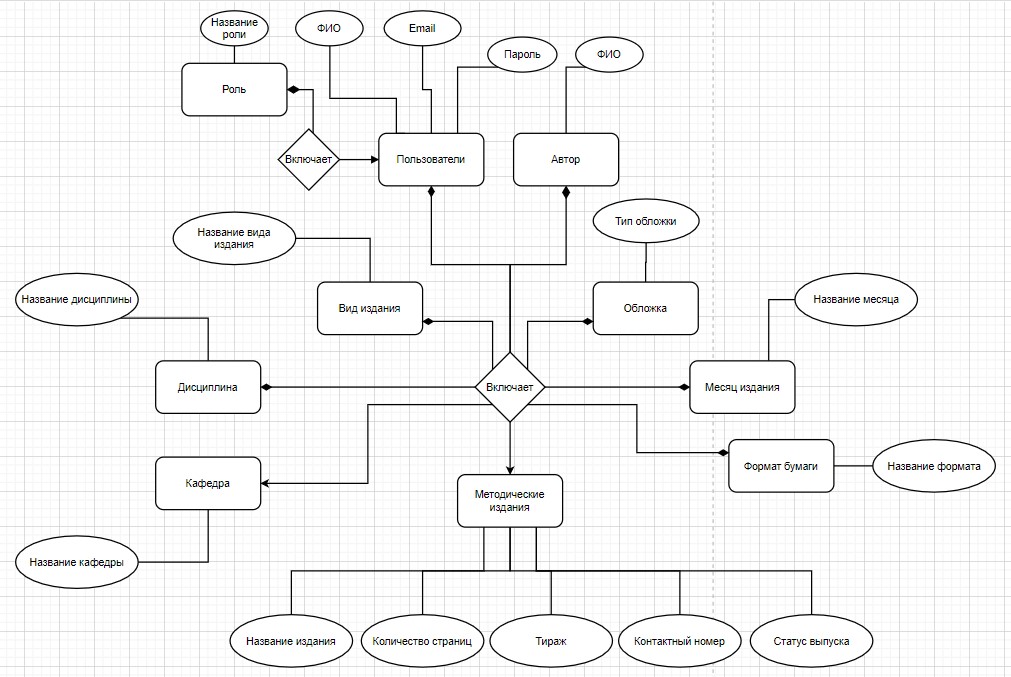


Рисунок 4.2 – Концептуальная модель базы данных

### 4.2.3 Логическое проектирование БД

Второй этап проектирования базы данных называется логическим проектированием базы данных. Его цель состоит в создании логической модели данных для исследуемой части предприятия. Концептуальная модель данных, созданная на предыдущем этапе, уточняется и преобразуется в логическую модель данных. Логическая модель данных учитывает особенности выбранной модели организации данных в целевой СУБД (например, реляционная модель).

Реляционная модель характеризуется своей простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для манипулирования данными.

В реляционных моделях данных объекты и взаимосвязи между ними представляются с помощью таблиц. Каждая таблица представляет один объект и состоит из строк и столбцов [13].

Так как для разрабатываемого программного продукта была выбрана реляционная модель данных MySQL.

Логическое представление базы данных приведено на рисунке 4.3.

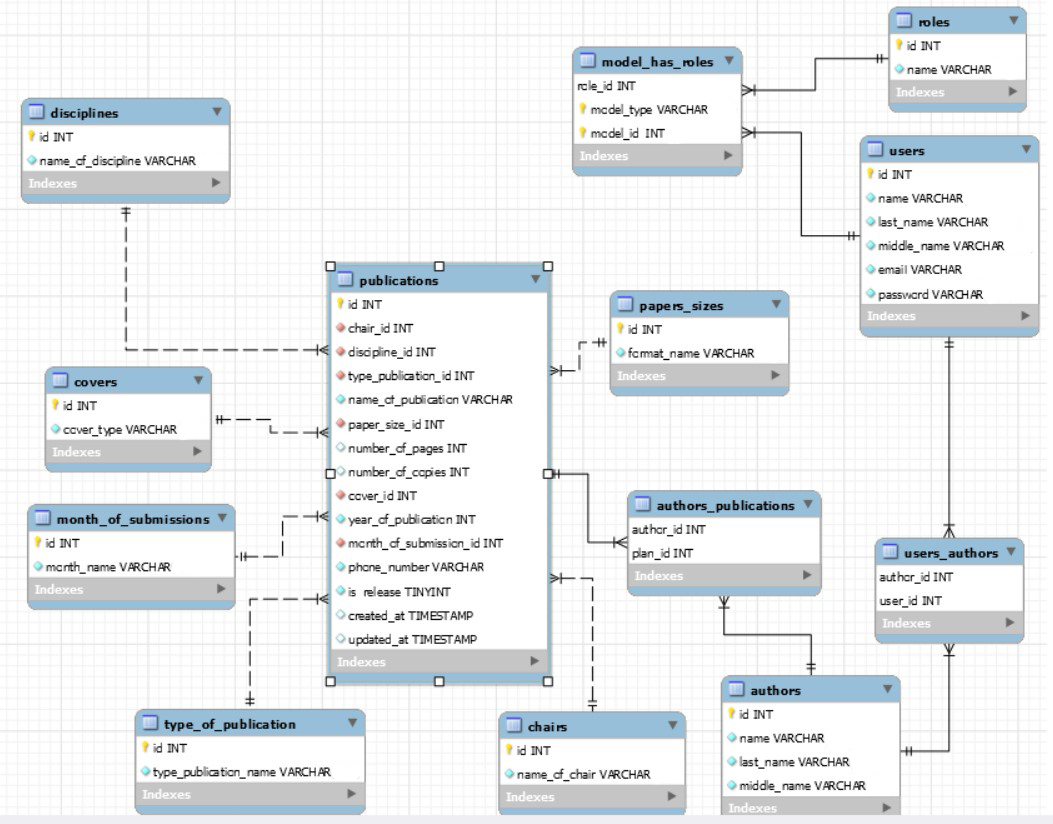


Рисунок 4.3 – Логическая модель базы данных

### 4.2.4 Физическое проектирование БД

Физическое проектирование – создание схемы базы данных для конкретной СУБД.

Физическая модель данных описывает то, как данные хранятся в компьютере, представляя информацию о структуре записей, их упорядоченности и существующих путях доступа.

В физической модели описываются типы, идентификаторы и разрядность полей. Физическая модель данных отражает физическое размещение данных на машинных носителях, а именно какой файл, какие объекты, с какими атрибутами содержит и каковы типы этих атрибутов.

Физическое представление таблиц базы данных приведено на  
 рисунках 4.4 – 4.11.



Рисунок 4.4 – Физическая модель таблицы «Publications»

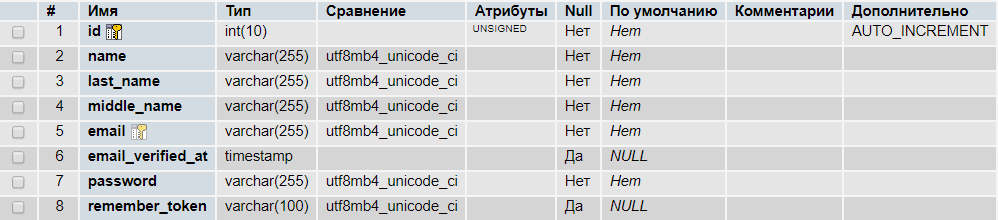


Рисунок 4.5 – Физическая модель таблицы «Users»



Рисунок 4.6 – Физическая модель таблицы «Authors»

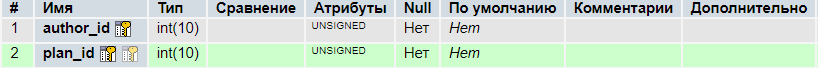


Рисунок 4.7 – Физическая модель таблицы «Authors\_Publications»



Рисунок 4.7 – Физическая модель таблицы «Users\_Authors»



Рисунок 4.8 – Физическая модель таблицы «Type\_of\_publication»



Рисунок 4.9 – Физическая модель таблицы «Papers\_sizes»



Рисунок 4.10 – Физическая модель таблицы «Month\_of\_publications»



Рисунок 4.11 – Физическая модель таблицы «Disciplines»



Рисунок 4.12 – Физическая модель таблицы «Chairs»



Рисунок 4.12 – Физическая модель таблицы «Covers»

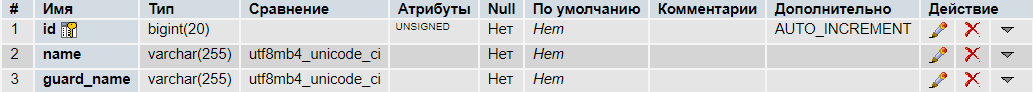


Рисунок 4.13 – Физическая модель таблицы «Roles»

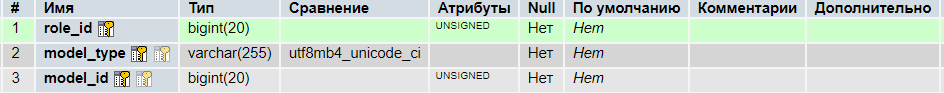


Рисунок 4.14 – Физическая модель таблицы «Models\_has\_Roles»

## 4.3 Разработка и описание web-приложения и базы данных

Для начала необходимо загрузить все необходимые компоненты командой “composer create-project --prefer-dist laravel/laravel MethodicalSystem”.

Далее необходимо настроить файл “.env” (рисунок 4.12). В поле “DB\_CONNECTION” нужно прописать используемую базу данных, в нашем случае это MySQL.

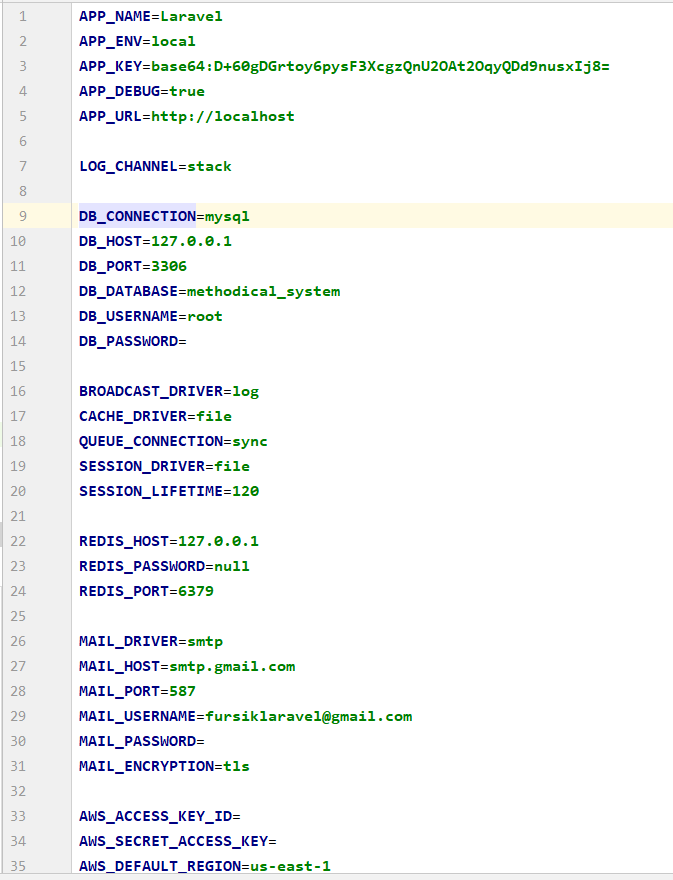


Рисунок 4.15 – Cтруктура файла “.env”

Посмотреть их полный список можно в файле “database.php” (рисунок 4.13)



Рисунок 4.16 – Содержимое файла “database.php ”

### 4.3.1 Аутентификация средствами Laravel

В Laravel сделать аутентификацию очень просто. Фактически, почти всё сконфигурировано уже из коробки. Конфигурационный файл аутентификации расположен в директории “config/auth.php”. Он содержит несколько хорошо описанных опций для тонкой настройки поведения служб аутентификации.

Чтобы сделать аутентификацию, достаточно вписать в консоли команды “php artisan make:auth” и “php artisan migrate”. Laravel самостоятельно создаст необходимые контроллеры, посредники, маршруты и модели для аутентификации.

### 4.3.2 Миграции для базы данных

Миграции – это набор средств, с помощью которого можно без особого труда создавать или обновлять таблицы в базе данных. Фасад Laravel “Schema” обеспечивает поддержку создания и изменения таблиц [14].

Чтобы создать миграцию, необходимо, находясь в директории с проектом, в консоли прописать команду “php artisan make:migration \*название миграции\*”. Для нашего проекта необходимо 8 миграций.

Миграции находятся в директории “../database/migrations”   
(рисунок 4.14)

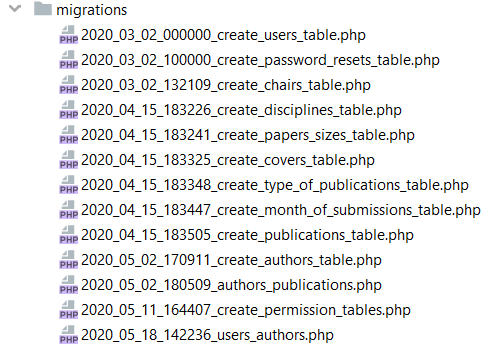


Рисунок 4.17 – Список миграций

Для запуска всех необходимых миграций приложения используем команду “php artisan migrate”. Подробнее о командах, которые понадобятся с для работы с миграциями в таблице 4.3.

Таблица 4.3 – Основные консольные команды для работы с миграциями

|  |  |
| --- | --- |
| Команда | Описание команды |
| php artisan migrate | Запуск всех миграций |
| php artisan migrate:reset | Отменяет изменения всех миграций вашего приложения |
| php artisan migrate:refresh | Отменяет изменения всех ваших миграций, а затем выполняет команду migrate |
| php artisan migrate:refresh --seed | Отменяет изменения всех ваших миграций, а затем выполняет команду migrate и запускает работу наполнителей. |

В миграциях существует два метода “up” и “down”. Метод “up” создает таблицы с нужными полями и свойствами, такими как внешний ключ или первичный ключ. Метод “down” отменяет действия метода “up”.

Laravel поддерживает создание ограничений для внешнего ключа, которые используются для обеспечения ссылочной целостности на уровне базы данных. Для этого, к полю добавляют свойство “unsigned()”, а затем указывают, откуда ссылается данное поле “foreign()->references()->on()”.

Порядок выполнения миграций очень важен. Чтобы не получить ошибку, необходимо заранее создавать таблицы, поля которых, в дальнейших миграциях будут использоваться внешними ключами.

Разработанные миграции:

1. CreateUsersTable – миграция пользователей (рисунок 4.15);



Рисунок 4.15 – Миграция пользователей

1. CreateDisciplinesTable – миграция дисциплин (рисунок 4.16);

На рисунке 4.16 можно заметить, что в миграциях используется подсистема InnoDB.

До MySQL 5.6 InnoDB  не поддерживал внешние ключи, и для этого нужно было использовать подсистему MyISAM. Однако главное преимущество InnoDB – в скорости работы. При выполнении запроса к базе InnoDB происходит блокировка строки, при выполнении же запроса к базе MyISAM блокируется таблица, это означает, что пока запрос выполнен не будет никакие другие обращения к таблице или строке будут невозможны.

В нашем проекте используется версия MySQL 5.7, поэтому мы можем использовать InnoDB с внешними ключами.



Рисунок 4.16 – Миграция дисциплин

1. CreatePapersSizes – миграция форматов бумаги (рисунок 4.17);



Рисунок 4.17 – Миграция форматов бумаги

1. CreateСoversTable – миграция обложек (рисунок 4.18);



Рисунок 4.18 – Миграция обложки издания

1. CreateTypeOfPublicationsTable – миграция видов изданий   
   (рисунок 4.19);



Рисунок 4.19 – Миграция видов изданий

1. CreateMonthOfSubmissionsTable – миграция месяцев изданий   
   (рисунок 4.20);

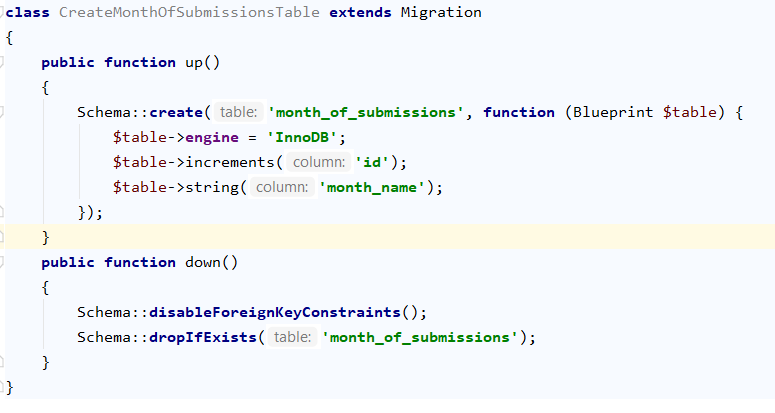


Рисунок 4.20 – Миграция месяцев издания

1. CreatePermissionTables – миграция, созданная пакетом “laravel-permissions’ (рисунок 4.21);

Для системы ролей и прав в системе воспользуемся пакетом “laravel-permission” третьей версии. Данный пакет включает миграции для таблицы “Roles’ и “Model\_has\_Role”. Первая таблица нужна для определения ролей, вторая – для соответствия роли и пользователя.

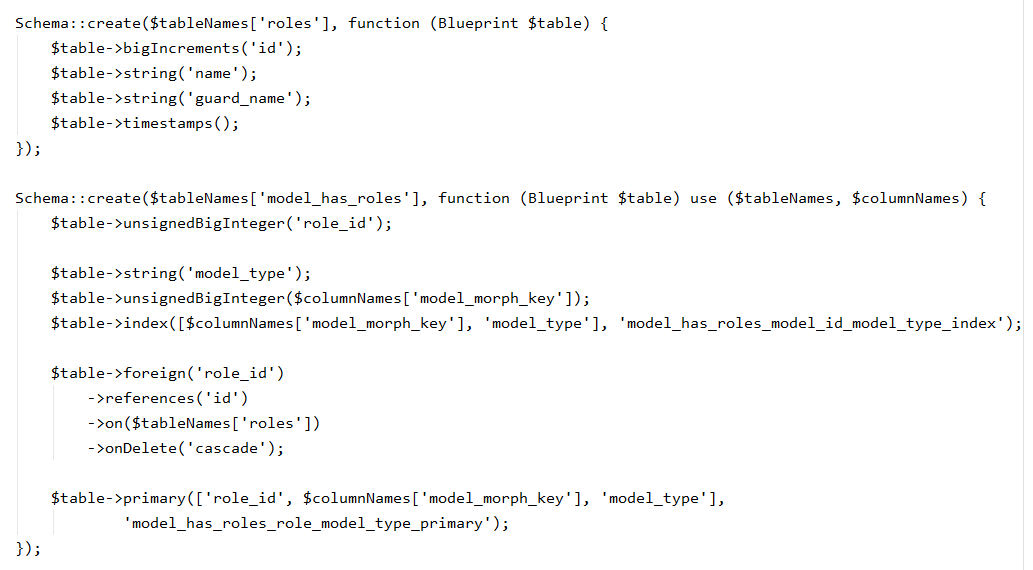


Рисунок 4.21 – Миграция ролей и прав

1. CreatePublicationTable – миграция методических изданий   
   (рисунок 4.22);

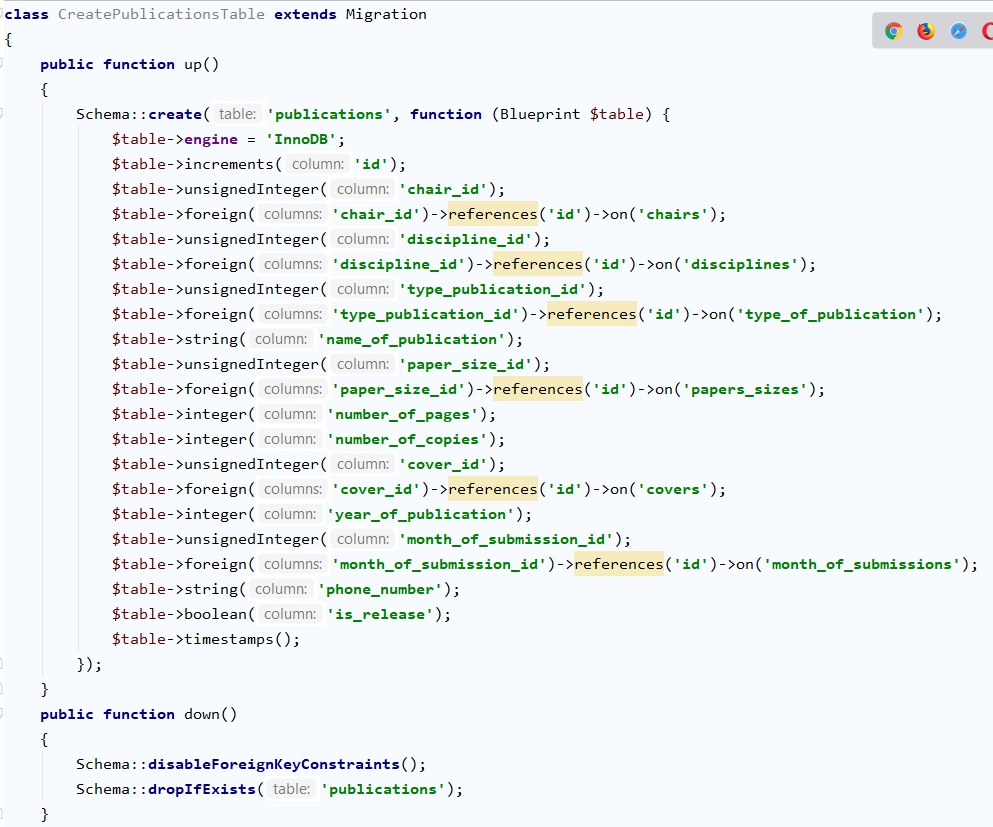


Рисунок 4.22 – Миграция методических изданий

Миграция методических изданий, по сути, является главной миграцией. В ней находится наибольшее скопление полей с внешними ключами других таблиц.

Более подробно о полях в таблице, созданной миграцией методических изданий в таблице 4.4.

Таблица 4.4. Поля таблицы, созданные миграцией методических изданий

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| id | increments | Первичный ключ для таблицы |
| chair\_id | int(10) | Id кафедры, которая издала методическое издание |
| discipline\_id | int(10) | Id дисциплины, для которой создается методическое издание |
| type\_publication\_id | int(10) | Id вида издания |
| name\_of\_publication | varchar(255) | Название методического издания |
| paper\_size\_id | int(10) | Id формата бумаги, в котором возможно выпустить издание |
| number\_of\_pages | int(10) | Количество страниц в методическом издании |
| number\_of\_copies | int(10) | Тираж методического издания |
| cover\_id | int(10) | Id обложки (переплёта), в котором возможно выпустить издание |
| year\_of\_publication | int(10) | Год издания |
| month\_of\_submission | int(10) | Id месяца, в котором возможно выпустить издания |
| phone\_number | varchar(255) | Дополнительная информация (контактный телефон) |
| is\_release | boolean | Указывает, выпущено ли методическое издание |

1. Authors\_Publications – миграция таблицы “Authors\_Publications” (рисунок 4.22)

Из-за связи “многие-ко-многим” мы также создаем миграцию “Authors\_Publications” с составным ключом из внешних ключей авторов и методических изданий. Это позволит приписать несколько изданий к одному автору или несколько авторов к одному изданию.



Рисунок 4.22 – Миграция для промежуточной таблицы “Authors\_Publications”

1. UsersAuthors – миграция соответствий между пользователями и авторами (рисунок 4.23);

Аналогичная ситуация из предыдущей таблицы с таблицей “Users\_Authors”. Данная таблица необходима, чтобы определить, с какими методическими изданиями сможет взаимодействовать пользователь, не обладающий правами администратора или модератора.



Рисунок 4.23 – Миграция для промежуточной таблицы “Users\_Authors’

### 4.3.3 Классы-наполнители для базы данных

Laravel имеет простой метод заполнения базы данных тестовыми данными, используя классы-наполнители.

Чтобы создать новый класс-наполнитель необходимо войти в консоль и в директории с проектом прописать команду “php artisan make:seeder \*Название наполнителя\*”.

Эти классы хранятся в директории “../database/seeds”. По умолчанию класс “DatabaseSeeder” уже создан в папке наполнителей. В этом классе вы можете использовать метод “call” для запуска других наполнителей, что позволяет вам контролировать порядок наполнения.

Чтобы запустить работы классов-наполнителей необходимо войти в консоль и в директории с проектом прописать команду   
“php artisan db:seed” [15].

Данные о факультете меняются нечасто. Поэтому зачастую, для начала работы нам нужны одни и те же данные для таблицы. Разработанные наполнители данных, которые наполнят таблицы необходимыми данными для работы с web-приложением:

1. CoverSeeder – класс-наполнитель для таблицы “Covers” (рисунок 4.24);

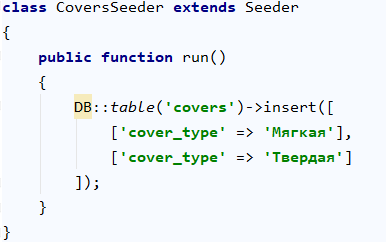


Рисунок 4.24 – Класс-наполнитель для таблицы “Covers”

1. PaperSizesSeeder – класс-наполнитель для таблицы “Papers\_sizes” (рисунок 4.25);

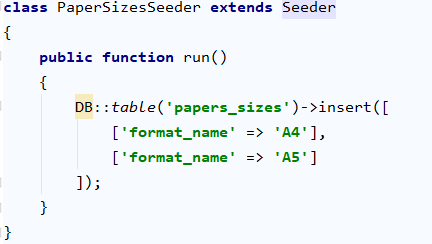


Рисунок 4.25 – Класс-наполнитель для таблицы “Papers\_sizes”

1. MonthSeeder – класс-наполнитель для таблицы “Month\_Of\_Submissions” (рисунок 4.26);



Рисунок 4.26 – Класс-наполнитель для таблицы “Month\_Of\_Submissions”

1. TypeOfPublicationSeeder – класс-наполнитель для таблицы “Type\_of\_publication” (рисунок 4.27).



Рисунок 4.27 – Класс-наполнитель для таблицы “Type\_of\_publication”

1. UsersSeeder – класс-наполнитель для таблицы “Users” (рисунок 4.28);



Рисунок 4.28 – Класс-наполнитель для таблицы “Users”

Для полей типа “timestamp” применяем функцию “now” из фасада “Carbon”. Это позволит получить текущее время.

1. DisciplinesSeeder – класс-наполнитель для таблицы “Disciplines” (рисунок 4.29);



Рисунок 4.29 – Класс-наполнитель для таблицы “Disciplines”

1. PublicationsSeeder – класс-наполнитель для таблицы “Publications” (рисунок 4.30);



Рисунок 4.30 – Класс-наполнитель для таблицы “Publications”

1. UsersPublicationsSeeder – класс-наполнитель для таблицы “Users\_Publications” (рисунок 4.31);

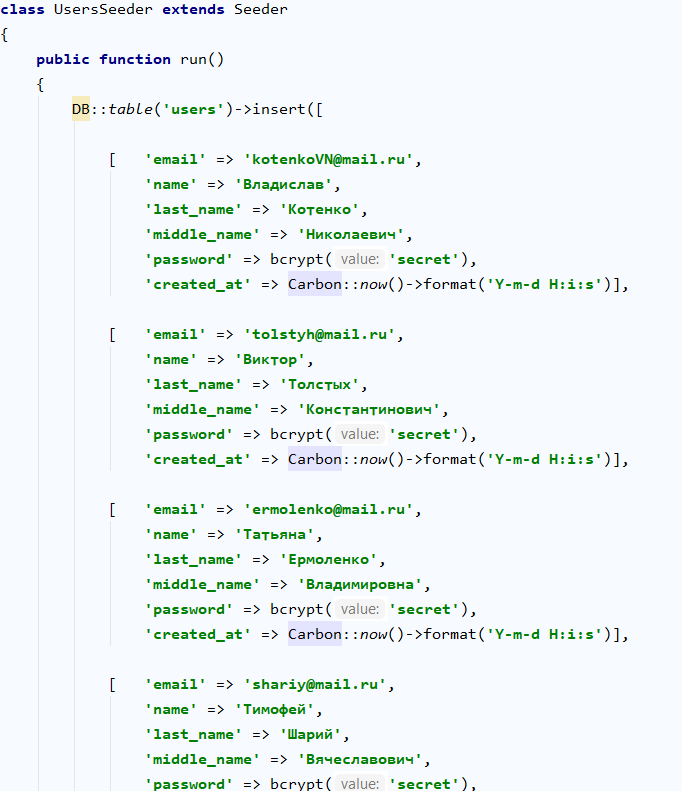


Рисунок 4.31 – Класс-наполнитель для таблицы “Users\_Publications”

1. DatabaseSeeder – главный класс-наполнитель. Именно этот класс вызывает все остальные классы-наполнители и задает порядок заполнения базы данных (рисунок 4.32);

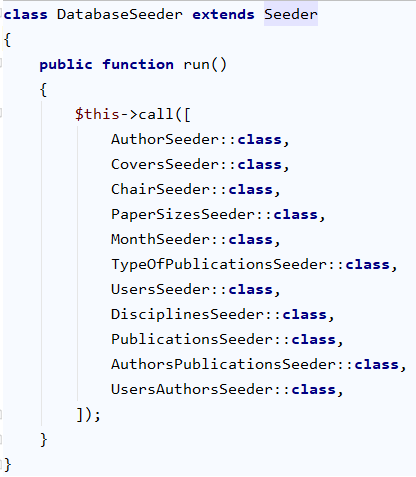


Рисунок 4.32 – Класс-наполнитель “DatabaseSeeder”

### 4.3.4 Реализация базы данных в ORM Eloquent

Система объектно-реляционного отображения (ORM) Eloquent – красивая и простая реализация шаблона ActiveRecord в Laravel для работы с базами данных. Каждая таблица имеет соответствующий класс-модель, который используется для работы с этой таблицей. Модели позволяют запрашивать данные из таблиц, а также вставлять в них новые записи [16].

Для того чтобы создать экземпляр модели нужно прописать Artisan-команду “php artisan make:model \*Название модели”. Имя будет использовано в нижнем регистре и во множественном числе чтобы привязать модель к таблице. Также можно определить таблицу определив свойство “$table”. Для web-приложения понадобится массовое заполнение, поэтому определим свойство “$fillable” и укажем нужные поля. Пример модели приведен на рисунке 4.32.



Рисунок 4.33 – Модель методических изданий

### 4.3.5 Шаблоны

Шаблоны обычно содержат HTML-код приложения и представляют собой удобный способ разделения бизнес-логики и логики отображения информации. Шаблоны хранятся в директории “../resources/views”.

В web-приложении используется шаблонизатор Blade – простой, но мощный шаблонизатор, поставляемый с Laravel. Для того, чтобы создать шаблон blade, нужно создать файл с расширением “.blade.php”. Структура шаблонов приведена на рисунке 4.32.

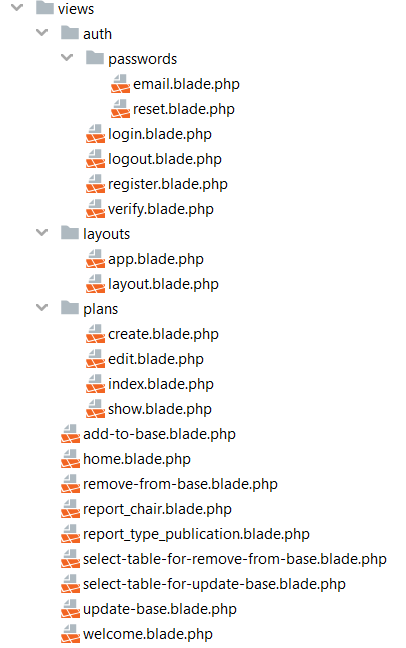


Рисунок 4.34 – Структура шаблонов web-приложения

### 4.3.6 Контроллеры

Вместо того, чтобы определять всю логику обработки запросов в виде замыканий в файлах маршрутов, организуем её с помощью классов контроллеров. Контроллеры могут группировать связанную с обработкой HTTP-запросов логику в отдельный класс. Контроллеры хранятся в директории “../app/Http/Controllers”. Чтобы создать контроллер нужно вписать команду “php artisan make:controller \*Название контроллера\*” в консоль. Создадим ресурсный контроллер “PublicationPlanController.php”.

Для данного проекта будет удобно использовать ресурсный контроллер. Маршрутизация ресурсов Laravel позволяет назначить обычные CRUD-маршруты на контроллеры одной строчкой кода – “php artisan make:controller \*Название контроллера\* –resource”. Подробное описание метода в таблице 4.5. Определим логику обработки запроса в функциях ресурсного контроллера (Листинг Б.1).

Таблица 4.5. Описание методов в ресурсном контроллере “PublicationPlanController.php”

|  |  |
| --- | --- |
| Метод | Описание |
| index | Здесь формируются данные для выпадающих списков (для фильтрации данных на главной странице), сопоставляются авторы и издания, формируются для таблицы изданий на основе выбранных фильтров (или без них) с помощью фасада “DB”. Возвращает шаблон главной страницы |
| create | Возвращает шаблон страницы для создания нового издания |
| store | Проверка корректности введенных пользователем данных. Добавление нового издания в базу данных |
| show | Просмотр отдельного элемента. Метод был сформирован ресурсным контроллером и пока не используется в данном проекте. |
| edit | Возвращает шаблон страницы для редактирования издания |
| update | Проверка корректности введенных пользователем данных. Обновление информации выбранного издания в базе данных |
| destroy | Удаляет выбранное издания из базы данных |

Однако “PublicationPlanController.php” занимается исключительно методическими изданиями. Для добавление, редактирования или удаления информации о факультете необходимо создать еще один контроллер “AdditionToBaseController.php”. Листинг кода “AdditionToBaseController.php” находится в листинге Б.2.

Таблица 4.6. Описание методов в контроллере “AdditionToBaseController.php”

|  |  |
| --- | --- |
| Метод | Описание |
| create | Возвращает шаблон страницы для выбора таблицы, для которой будут введены новые данные |
| store | Проверка корректности введенных пользователем данных. Добавление новой записи в базу данных |
| update\_table\_combobox | Обновление данных выпадающего списка в зависимости от выбора таблицы. |
| remove | Возвращает шаблон страницы для выбора данных, которые будут удалены. |
| destroy | Удаление данных и всех связанных с ними изданий |
| change | Возвращает шаблон страницы для выбора данных. |
| select\_table\_for\_update | Возвращает шаблон страницы для редактирования данных. |
| update | Сохранение изменений внесенных в таблицу |

Для формирования отчётов о методической работе факультета создадим контроллер “ReportController”. Листинг кода “ ReportController.php” находится в листинге Б.3.

Таблица 4.7. Описание методов в контроллере “ReportController.php”

|  |  |
| --- | --- |
| Метод | Описание |
| ReportForChair | Формирует отчёт по кафедрам |
| ReportForTypePublication | Формирует отчёт по видам изданий |

За авторизацию пользователя отвечает автоматически сгенерированный фреймворком Laravel контроллер LoginController.php.

### 

### 4.3.7 Маршруты

В Laravel простейшие маршруты принимают URI (путь) и функцию-замыкание, предоставляя очень простой и выразительный метод определения маршрутов. Все маршруты Laravel определены в файлах маршрутов, которые расположены в директории “../routes”. Эти файлы автоматически загружаются фреймворком. В файле “../routes/web.php” определены маршруты для web-интерфейса. Определение маршрутов приведено на рисунке 4.34.



Рисунок 4.35 – Содержимое файла web.php

### 4.3.8 Посредники

Посредники предоставляют удобный механизм для фильтрации HTTP-запросов приложения. К примеру, если пользователь не аутентифицирован, посредник перенаправит его на страницу входа в систему. Если же пользователь аутентифицирован, посредник позволит запросу пройти далее в приложение.

В Laravel есть несколько стандартных посредников, включая посредники для аутентификации и CSRF-защиты. Все они расположены в директории app/Http/Middleware.

В данном проекте будет использоваться стандартный посредник Authenticate.php. Таким образом, неавторизованные пользователи не смогут получить доступ к страницам сайта.  
Помимо необходимости в авторизации, необходимо обеспечить фильтрацию по уровню прав зарегистрированных пользователей. Для этого создадим еще один посредник “CheckRole”.

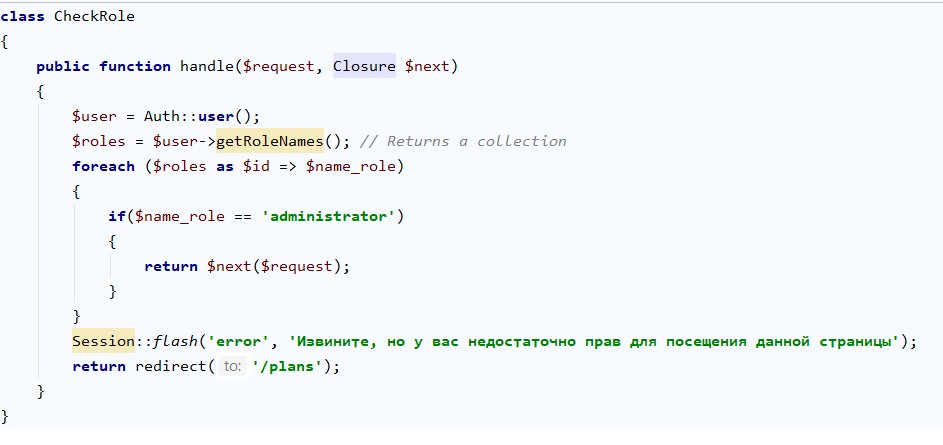


Рисунок 4.36 – Посредник “CheckRole”

Посредник проверяет наличие у пользователя роли «Администратор». Если пользователь обладает данными правами, то он получит доступ к странице. Если прав недостаточно, то его направят на главную страницу, вместе с сообщением об ошибке.

## 

## 4.4 Общая структура web-приложения

Общая структура динамического сайта приведена на рисунке 4.37.

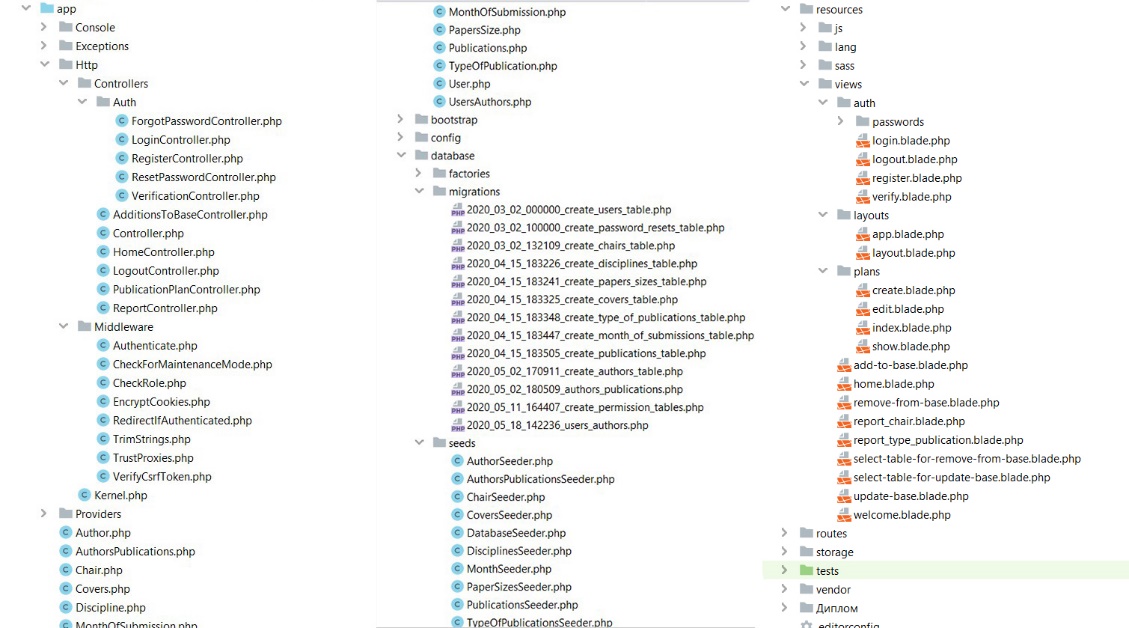


Рисунок 4.37 – Общая структура динамического сайта

Связь скриптов и шаблонов показана на рисунке 4.38.

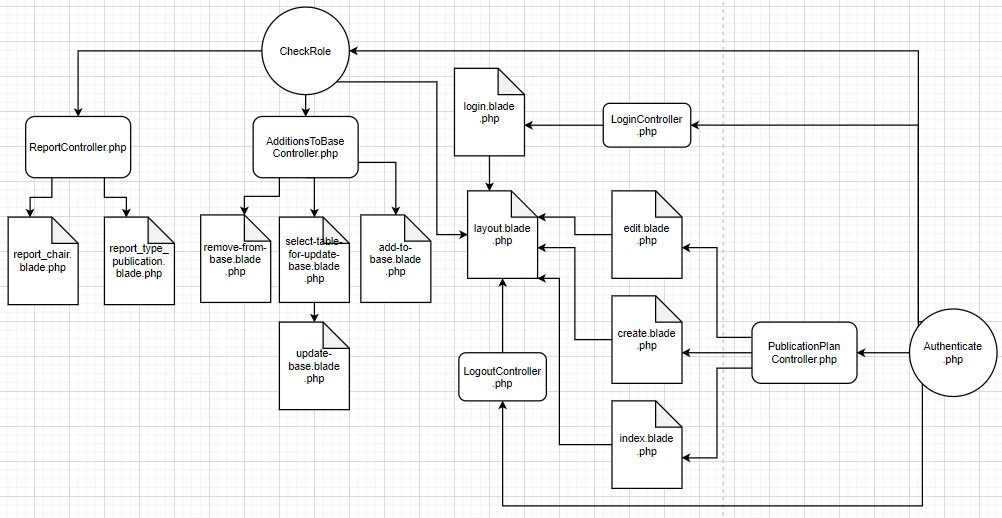


Рисунок 4.38 – Связь скриптов автоматизированной системы

# 5. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 5.1 Аппаратные, системные и программные требования

Для функционирования реализованного динамического сайта необходимы следующие аппаратные ресурсы:

1. Windows (32-bit или 64-bit): Windows 8 / Windows 7 / Windows Server 2008 / Windows Vista / Windows XP SP3;
2. процессор: 2.0 ГГц;
3. оперативная память: 512 Mb;
4. HDD: 1 Гб.

Для работы динамического сайта и взаимодействия его с созданной базой данных необходимо следующее программное обеспечение:

1. набором библиотек Microsoft Visual C++ 2005-2008-2010 Redistributable Package x86;
2. PHP и Composer;
3. веб-фреймворк Laravel;
4. библиотекой  jQuery и набором инструментов Bootstrap 4;
5. web-сервером Apache;
6. сервером управления базами данных (БД) MySQL.

## 5.2 Тестирование web-сайта

При помощи инструмента для функционального тестирования Selenium IDE, веб-сайт и пользовательский интерфейс были протестированы по следующим категориям:

1. протестированы корректность внутренних ссылок. Неработающие ссылки отсутствуют. Ссылки, ведущие к одной и той же странице, найдены в навигационном меню (рисунок 5.1). Кнопка «Просмотреть все издания» и логотип кафедры компьютерных технологий ведут на главную страницу.

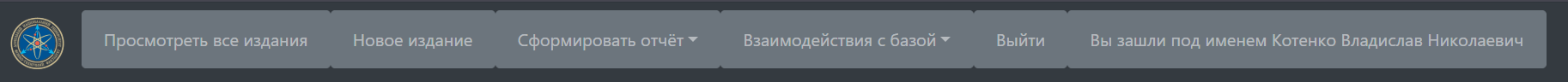


Рисунок 5.1 – Навигационное меню web-сайта

1. проверена работа системы по действительности входных данных, допустимые значения входных данных, для поля данных, недопустимые входные значения для полей данных. Проверена работа форм, в которых участвуют как текстовые поля, так и выпадающие списки.
2. проверено, что все страницы сайта простые в использовании и требуют только базовых умений от пользователя. Доступ к главному меню можно получить со всех страниц, если пользователь авторизован. Грамматические ошибки отсутствуют. Цветовая палитра выполнена в черно-белом стиле, оптимизированы удовлетворительно. Размеры шрифтов оптимизированы. Инструкции ясны и содержат правильную информацию. Из минусов можно выделить перезагрузку всей страницы при каждом действии;
3. интерфейс соответствует стандартам графических интерфейсов. Макет, цвета, шрифты, размеры шрифтов, текстовые поля, выпадающие списки, кнопки, значки, ссылки соответствуют спецификациям. Поведение сайта на различных расширениях экрана, а также поведение на планшетах и мобильных устройствах, приемлемо. Однако главная страница с таблицей методических указаний на мобильных устройствах работает не корректно;
4. протестирована правильность работы всех функций сайта.

Результат тестирования изображен на рисунке 5.2.

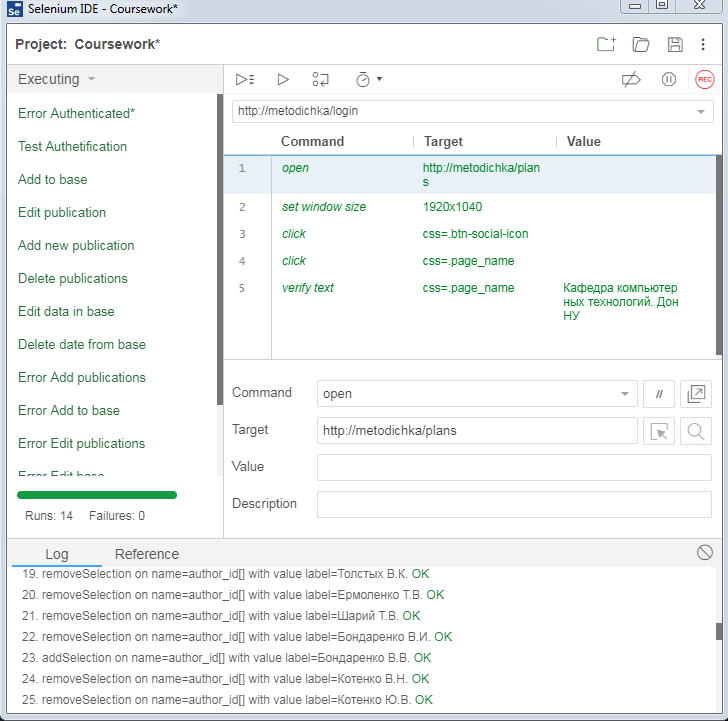


Рисунок 5.2 – Результаты тестирования

## 5.3 Описание установки web-приложения.

### 5.3.1 Установка, настройка и запуск Open Server

Для работы динамического сайта, прежде всего нужны Apache, MySQL и PHP. Всё это можно получить из локального веб-сервера Open Server.

Для установки Open Server необходимо скачать установщик с официального сайта “https://ospanel.io”. Следовать инструкции во время установки.

После установки необходимо настроить модули HTTP, PHP и MySQL. Для этого необходимо перейти в настройки Open Server, открыть вкладку «Модули» и выбрать:

1. HTTP ­­– Apache-PHP-7.2;
2. PHP – PHP-7.2;
3. MySQL/MariaDB – MySQL-5.7.

### 5.3.2 Установка и настройка Composer и Laravel

Для управления зависимостями Laravel, необходимо установить Composer, а затем Laravel.

Для начала, необходимо скачать и установить Composer. Во время установки можно взять PHP из Open Server из директории “../ospanel/modules/php/PHP-7.2”.

Для создания проекта Laravel необходимо открыть консоль и вписать команду “composer create-project --prefer-dist laravel/laravel <имя\_проекта>”.

### 5.3.3 Установка, настройка и запуск сервера баз данных MySQL

В PhpMyAdmin, установленном в Open Server, создать базу данных.

Далее, в папке с проектом создайте файл “.env” в корневой папке проекта, если он отсутствует, и заполните его соответствующей информацией, такой как информация о его базе данных. В качестве примера откройте файл “.env.example”.

После того как файл создан, нужно заполнить его исходными таблицами и данными о кафедре. Для этого в консоли нужно перейти в директорию с проектом и прописать команду “php artisan migrate:fresh –seed”

### 5.3.4 Настройка и запуск web-приложения

После установки проекта Laravel настроим Open Server так, чтобы проект запускался по его прямому доменному имени, без дополнительного указания в URL папки “/public”. В настройках Open Server переходим на вкладку «Домены». В списке "Управление доменами" выбираем «Ручное + Автопоиск». В поле «Имя домена» прописываем свой домен, в поле «Папка домена» выбираем каталог проекта и обязательно в конце указываем папку \public. Нажимаем кнопки «Добавить» и «Сохранить».

После проведения выше перечисленных действий, если все сделано верно, то при наборе в адресной строке только что добавленного доменного имени вы увидите страницу авторизации.

## 5.4 Описание контрольных примеров

После запуска в окне браузера отображается страница авторизации (рисунок A.1). Доступ к ссылкам из навигационного меню не будет разрешен, если пользователь не авторизован. В случае, если пользователь неправильно укажет логин или пароль, то сайт отправит пользователя на страницу авторизации с сообщением о неудачной аутентификации.

После авторизации выводится страница с методическими изданиями кафедры (рисунок A2). На этой странице возможно отредактировать существующие издания, удалить их, а также отфильтровать по категориям «Год выпуска», «Дисциплины», «Авторы», «Кафедры», «Статус наличия», «Вид издания» с помощью выпадающих списков. Фильтры можно комбинировать между собой (например, вывести издания за 2019 год по дисциплине «Архитектура ЭВМ и микроконтроллеров» автором которых является Котенко В.Н.). Чтобы упростить задачу поиска нужных данных в выпадающих списках, можно воспользоваться поиском по введенным символам.

При нажатии на пункт в меню «Новое издание», который находится в верхней части сайта, выводится форма для заполнения информации о новом методическом издании (рисунок A3). При вводе некорректных данных сайт вернет пользователя обратно на форму для создания нового издания с сообщением об ошибке.

После нажатия на выпадающее меню «Взаимодействие с базой», можно выбрать необходимое действие «Добавить в базу», «Обновить в базе», «Удалить из базы». Следуя инструкциям, вы сможете взаимодействовать с элементами в базе данных, которые в дальнейшем появятся в выпадающих списках (рисунок A4). Однако, если вы удалите какой-либо элемент, то вы удалите и все связанные с ним методические указания.

После нажатия на выпадающее меню «Сформировать отчёт», можно выбрать отчёт «По кафедре» или «По виду издания». Пользователь переходит на новую страницу, где отображается таблица с названием кафедры или вида издания, выпадающим списком и полем для ввода для фильтрации данных и диаграммой (рисунок А5). В выпадающем списке можно выбрать несколько значений, а годы перечислить через запятую. Диаграмма создается за каждый год, а значит количество диаграмм будет соответствовать количеству введенных лет пользователем.

В подвале сайта расположена ссылка на информацию о физико-техническом факультете на сайте ДОННУ и ссылка на группу Вконтакте.

Описать все

# 6 ОХРАНА ТРУДА НА РАБОЧЕМ МЕСТЕ

## 6.1 Общие сведения

Важным моментом в совершенствовании условий труда являются мероприятия по охране труда.

Факторы можно разделить на опасные и вредные. Опасным производственным фактором называется, воздействие на работающего человека в определенных условиях, приводящее к травме или другому внезапному резкому ухудшению самочувствия. Вредным производственным фактором называют воздействие, приводящее к заболеванию или снижению трудоспособности. В зависимости от уровня и продолжительности воздействия вредный производственный фактор может стать опасным.

К сожалению, на сегодняшний день, состояние условий труда программиста и его безопасности, все еще не удовлетворяют современным требованиям. Влияние на работников таких отрицательных факторов как повышенный уровень шума, повышенная температура внешней среды, отсутствие или недостаточная освещенность рабочей зоны, электрический ток, статическое электричество, приводит к снижению работоспособности, а при длительном нахождении в зоне комбинированного воздействия различных неблагоприятных факторов может привести к профессиональному заболеванию.

Охрана труда – система законодательных актов, социально-экономических, организационных, технических, гигиенических и лечебно-профилактических мероприятий и средств, обеспечивающих безопасность, сохранение здоровья и работоспособности человека в процессе труда.

Рабочее место программиста – это часть пространства, в котором он осуществляет трудовую деятельность, и проводит большую часть своего рабочего времени.

Научно-технический прогресс дал прирост производительности, снижении доли рутинного, монотонного труда, повышения скорости расчётов, скорости обмена информацией.

Вопросы безопасной жизнедеятельности программиста в значительной степени зависит от правильной оценки опасных и вредных производственных факторов. Существуют меры безопасности при работе с вычислительной техникой для того чтобы избежать вредного воздействия.

Рабочее место программиста должно выполнять следующие основные требования:

1. размещение оборудования должно быть удобным для эксплуатации;
2. необходимо иметь рабочее пространство для движения и перемещения;
3. необходимо иметь естественное и искусственное освещение для выполнения установленных задач;
4. уровень акустического шума должен не превышать допустимого значения [17].

## 6.2 Опасные и вредные производственные факторы рабочего места программиста

### 6.2.1 Данные микроклимата помещения

Производственное окружение, где работает разработчик программного обеспечения, характеризуется определенным сочетанием температуры и влажности воздуха, его подвижности, барометрическим давлением и тепловым излучением нагретых поверхностей. На рабочем месте разработчика должны обеспечиваться приемлемые параметры микроклимата. По определению ГОСТ 12.1.005-88 микроклимат производственных помещений - это климат внутренней среды этих помещений, который определяется действующими на организм человека сочетаниями температуры, влажности и скорости движения воздуха, а также температурой находящихся вокруг поверхностей. Микроклиматические условия в помещениях с вычислительной техникой, должны соответствовать требованиям, обозначенным в таблице 6.1

Таблица 6.1- Микроклимат производственных помещений

|  |  |  |  |
| --- | --- | --- | --- |
| Период  года | Температура  воздуха, oС | Относительная  влажность воздуха, % | Скорость движения  воздуха, м/с |
| Холодный | 22-24 | 40-60 | до 0,1 |
| Теплый | 23-25 | 40-60 | 0,1-0,2 |

Скорость перемещения воздуха должна быть 0,1 м/с, а условная влажность воздуха на рабочих местах обязана составлять 40-60%. Для влажности воздуха в помещениях следует использовать увлажнители воздуха. Воздух, поступающий в комнаты разработчиков, должен быть очищен от загрязнений, в том числе от пыли и микроорганизмов.

Комнаты с рабочими местами программиста должны оборудоваться системами отопления. Температуру в помещении следует регулировать с учетом тепловых потоков от вычислительных машин. Оборудование нужно ставить так, чтобы тепло, исходящее от него, не было направлено на сотрудника. В помещениях с температурой выше допустимой, нужно предусматривать регулирование подачи теплоносителя.

В помещениях, оснащенных ЭВМ, обязана проводиться систематическое проветривание и ежедневная влажная уборка [18].

В качестве материалов для отделки помещений должны соответствовать гигиеническим требованиям. Покрытие пола из гладких нескользящих материалов, обладающих антистатическими свойствами.

## 

### 6.2.2 Освещение рабочего места

На рабочем месте программистов, в случаях преимущественной работы с документами необходимо использовать систему комбинированного освещения. Система комбинированного освещения – это комбинация освещения, из основного освещения помещения и дополнительного, предназначенного для освещения зоны расположения документов. Освещенность на поверхности стола в зоне размещения рабочего документа должна быть в диапазоне от 300 до 500 люксов. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 люксов.

В помещениях, где работает разработчик программного обеспечения, искусственное освещение должно осуществляться системой общего равномерного освещения. В качестве источников света, при искусственном освещении, люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ) является оптимальным решением. Для местного освещения применяются лампы накаливания, в том числе галогенные.

Высокая нагрузка на глаза является следствием частой работы с монитором компьютера, поэтому следует ограничивать прямые блики от источников освещения, таких как окна или светильники, и отраженную блики на рабочих поверхностях, такие как экран или стол, находящихся в поле зрения. Избежать бликов можно за счет правильного выбора светильников и расположения рабочих мест к источникам естественного и искусственного освещения. Также следует обратить внимание на неравномерность распределения яркости в поле зрения программиста. Соотношение яркости между рабочими поверхностями не должно превышать 3:1-5:1, а между рабочими поверхностями и поверхностями стен и оборудования – 10:1. Для уменьшения бликов рекомендуется ставить защитный фильтр для экранов видеомониторов. При рядном размещении рабочих столов не разрешается расположение экранов дисплеев навстречу друг другу из-за их взаимного отражения. Данное расположение разрешается в случае, если между столами устанавливаются перегородки.

Необходимо чистить стекла, оконные рамы и светильники не реже двух раз в год для обеспечения нормируемых значений освещенности в помещениях с использованием вычислительной техники, а также проводить своевременную замену перегоревших ламп. Окна в помещениях, где работает программист, должны быть направлены на север и северо-восток. Жалюзи, занавесы, внешние козырьки и другие регулируемые устройства должны быть установлены на оконные проемы [19].

### 6.2.3 Воздействие электромагнитного излучения

ЭВМ являются источником нескольких излучений, среди них: мягкий рентгеновский; ультрафиолетовый 200–400 нм; видимый 400–700 нм, ближний инфракрасный 700–1050 нм; радиочастотный 3 кГц–30 МГц; излучение электростатических полей.

Рабочее место разработчика программного обеспечения, с использованием компьютера является источником переменных электрических и магнитных полей. Монитор, из составных частей ЭВМ, принято считать основным источником электромагнитного поля, определяющим электромагнитную обстановку. Но, как показали многочисленные исследования, кроме источников электромагнитного поля – блоки кадровой и строчной развертки, высоковольтные элементы, элементы питания, есть еще источник переменного электрического поля – экран дисплея на электронно-лучевых трубках. При изменении изображения на экране дисплеев уровни их электромагнитных полей могут меняться. В большинстве случаев, интенсивность электромагнитного поля создается элементами системы электроснабжения здания, трансформаторами, воздушными линиями электропередач и другими внешними источниками. Поэтому при установке персонального компьютера на рабочем месте он должен быть правильно подключен к электропитанию и заземлен.

Защитный фильтр, при использовании, нужно плотно устанавливать на экран дисплея и надежно заземлить. Следует, ежедневно его очищать от пыли, так же как и экран дисплея [20].

Ионизирующие излучения, каковым является рентгеновское, негативно отражаются на здоровье человека. Ионный состав воздуха на рабочем месте разработчика, должен содержать определенное количество положительных и отрицательных аэроионов. Контроль аэроионного состава воздуха помещений следует проводить на рабочих местах в зонах дыхания персонала. Если в результате выявляется его несоответствие нормированным показателям, осуществляется его нормализация.

## 

### 6.2.4 Шум и вибрация. Статическое электричество

Принтеры, плоттеры, множительная техника и оборудование для кондиционирования воздуха, вентиляторы систем охлаждения, трансформаторы являются источниками шумовых помех. Длительное воздействие шума и вибраций отрицательно сказываются на эмоциональном состоянии персонала.

В производственных помещениях при выполнении работ с использованием вычислительных машин уровни шума на рабочих местах должны быть пределах допустимых значений, установленных для данных видов деятельности в соответствии с действующими санитарно-эпидемиологическими нормативами. Мастерские, производственные цеха и другие производственные помещения, не должны быть расположены рядом с помещениями, где основной работой является взаимодействие с персональным компьютером.

Печатающее оснащение, являющееся источником шума, следует устанавливать на звукопоглощающей поверхности автономного рабочего места сотрудника. Если уровень шума от печатающего оснащения превышает нормируемый, оно должно быть расположено вне помещения с персональным компьютером.

Статическое электричество является причиной образования электрической искры. Искра может быть причиной воспламенения горючих или взрывоопасных газов, паров или пыли с воздухом. Помимо риска воспламенения, статическое электричество оказывает вредное воздействие на организм человека, причем не только из-за контакта с зарядом, но и за счет воздействия электрического поля, возникающего вокруг заряженных поверхностей. Для защиты от статического электричества принимают ряд мер, такие как увеличение поверхностей проводимости диэлектриков, заземление оборудования, увлажнение окружающего воздуха, ионизация воздуха, антистатическое покрытие на полу.

Размещено на Allbest.ru

# ЗАКЛЮЧЕНИЕ

В результате выполнения дипломной работы разработана автоматизированная система для учета методической деятельности физико-технического факультета.

Разработанный программный комплекс позволяет выполнять учет методических изданий физико-технического факультета, а именно аутентифицировать пользователя, добавлять, редактировать и удалять методические издания, фильтровать по категориям, добавлять, редактировать и удалять данные о факультете, формировать отчёт по кафедрам и видам изданий на основе данных о методических изданиях.

Система для учета методической деятельности физико-технического факультета может использоваться преподавателями и заведующим методической деятельностью факультета для мониторинга методической деятельности факультета.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Котеров Д. В. PHP 5. 2 изд. / Д.В. Котеров. – БХВ-Петербург, 2012.   
    – 1062 с.
2. Коробко А. И. Методическая работа в вузе: цели, задачи, пути и формы ее ведения // Коробко А. И. – 2012. – №. 16 (649).
3. Знакомство с нотацией IDEF0 и пример использования // Хабр сообщество IT-специалистов.

URL: https://habr.com/ru/company/trinion/blog/322832/

(дата обращения: 22.05.2020)

1. Актуальность методической работы в общеобразовательном учреждении // Научный журнал «Молодой ученый»

URL: https://moluch.ru/conf/ped/archive/309/14628/

(дата обращения: 10.04.2020)

1. Что такое PHP? // Официальный сайт PHP

URL: https://www.php.net/manual/ru/intro-whatis.php (дата обращения: 11.04.2020)

1. База данных сайта MySQL: что это такое, преимущества перед другими СУБД // Вебмастерье.

URL: https://webmasterie.ru/razrabotka/hosting/baza-dannih-mysql

(дата обращения: 09.04.2020)

1. Что Такое Apache? Подробный Обзор Веб-Сервера Apache // Руководства Hostinger.

URL: https://www.hostinger.com.ua/rukovodstva/web-server-apache/

(дата обращения: 09.04.2020)

1. Обзор Laravel 5. Основные преимущества и недостатки // IT DEV GROUP

URL:https://it-devgroup.com/blog/blog-razrabotka/obzor-laravel-5.-osnovnyie-preimushhestva-i-nedostatki/ (дата обращения: 12.04.2020)

1. Что такое Bootstrap и зачем он нужен? // ИТ Шеф – всё о веб-разработке и программировании. URL: https://itchief.ru/bootstrap/introduction
2. Зачем нам jQuery // Чудо-Саикт. Образовательное IT-пространство. URL: saikt-online.ru/zachem-nam-jquery/
3. AJAX // Ingate. Интернет-продвижение

URL: https://promo-ingate-ru.turbopages.org/s/promo.ingate.ru/seo-wikipedia/ajax/

1. О модели взаимодействия клиент-сервер // IT-блог о веб-технологиях, серверах, протоколах

URL:https://zametkinapolyah.ru/servera-i-protokoly/o-modeli-vzaimodejstviya-klient-server-prostymi-slovami-arxitektura-klient-server-s-primerami.html (дата обращения: 12.04.2020)

1. Карпова Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. ­– СПб: Питер, 2001. – 304 с.
2. Миграции // Laravel по-русски

URL: https://laravel.ru/posts/273 (дата обращения: 14.04.2020)

1. База данных: наполнение данными (seeding) // Laravel.su

URL: http://laravel.su/docs/5.2/seeding (дата обращения: 15.04.2020)

1. Eloquent ORM // Laravel по-русски

URL: https://laravel.ru/docs/v5/eloquent (дата обращения: 14.04.2020)

1. ГОСТ Р 50948-2001 "Средства отображения информации индивидуального пользования. Общие эргономические требования и требования безопасности" (утв. Постановлением Госстандарта РФ от 25.12.2001 №576-ст).
2. Санитарно-эпидемиологические правила и нормативы. СанПиН 2.2.2/2.4.1340-03. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы.
3. Петрова М.С. Охрана труда на производстве и в учебном процессе: учебное пособие // М.С.Петрова, С.В.Петров, С.Н.Вольхин. – М.: НЦЭНАС, 2006. – 232 с.
4. Плахов А.М. Безопасность жизнедеятельности: Учебное пособие // А.М.Плахов. – Томск: Изд-во ТПУ, 2006. – 180 с.

# ПРИЛОЖЕНИЕ А Экранные формы

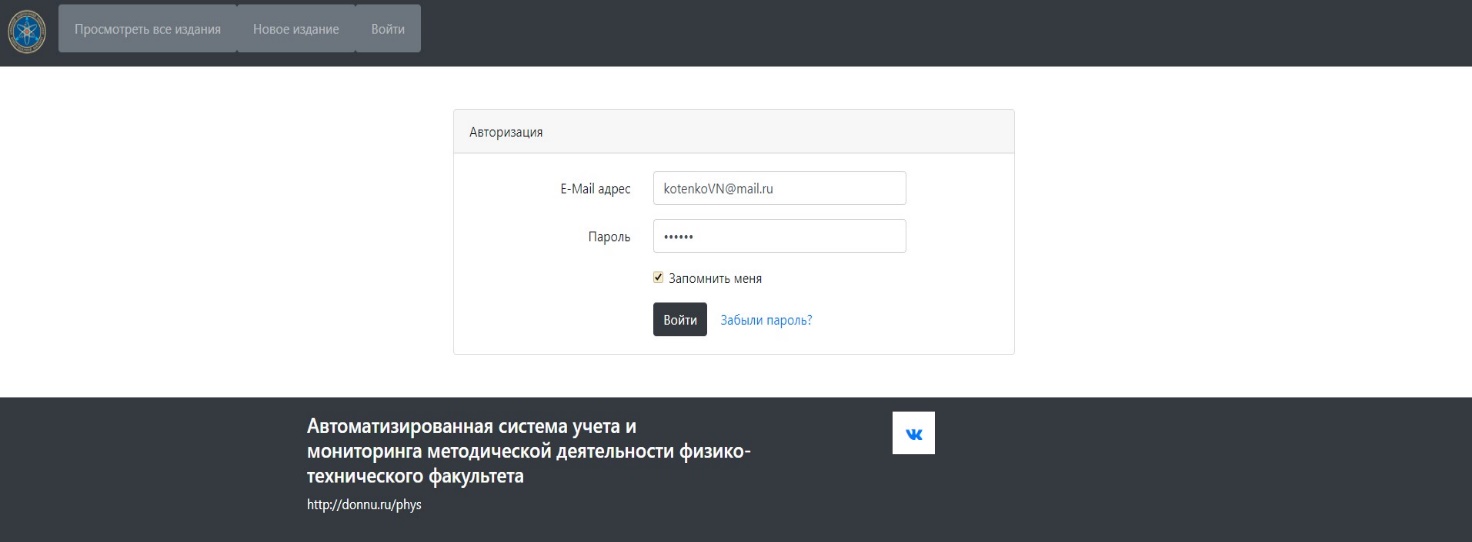
****

Рисунок А.1 – Страница с формой для входа

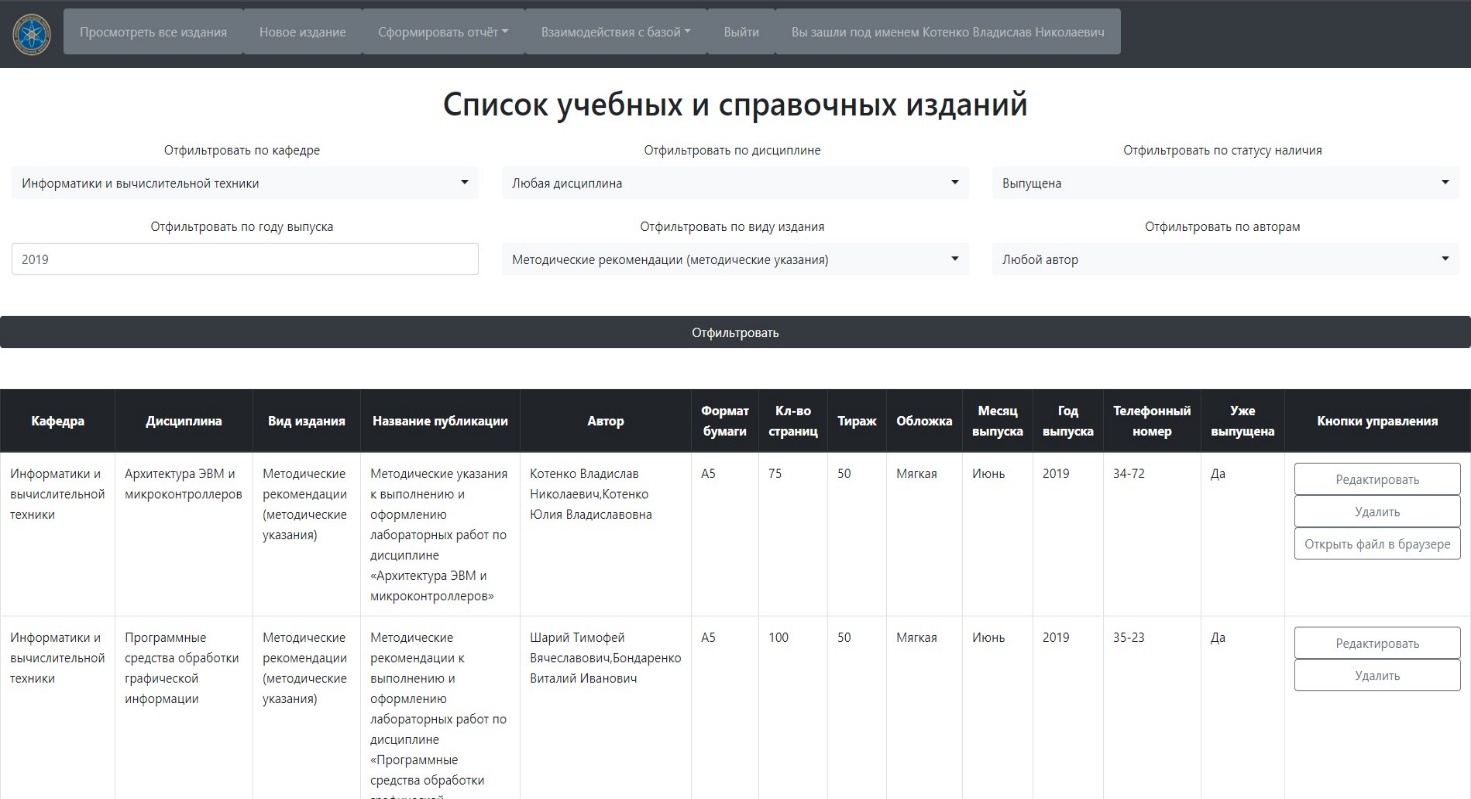
****

Рисунок А.2 – Страница с просмотром методических изданий

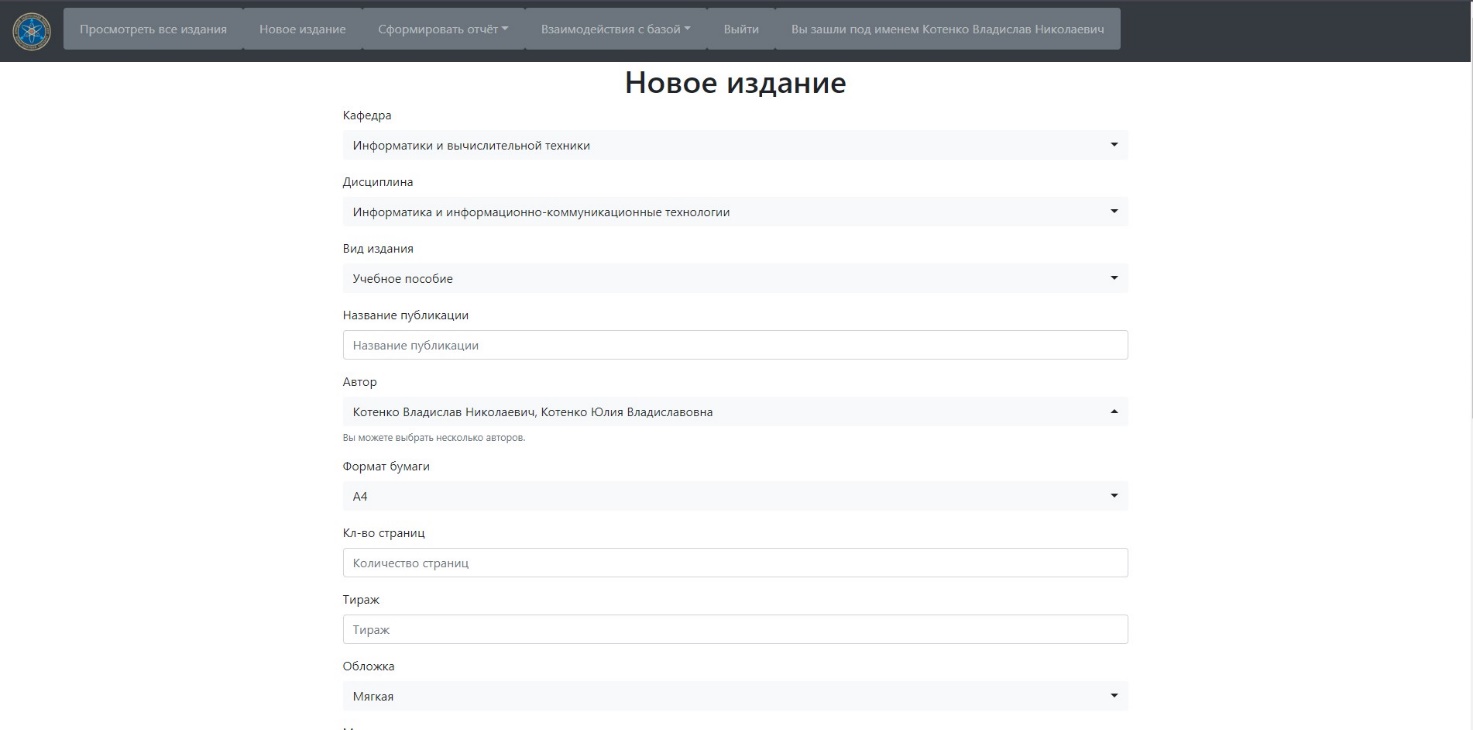
****

Рисунок А.3 – Страница с формой для добавления нового методического издания

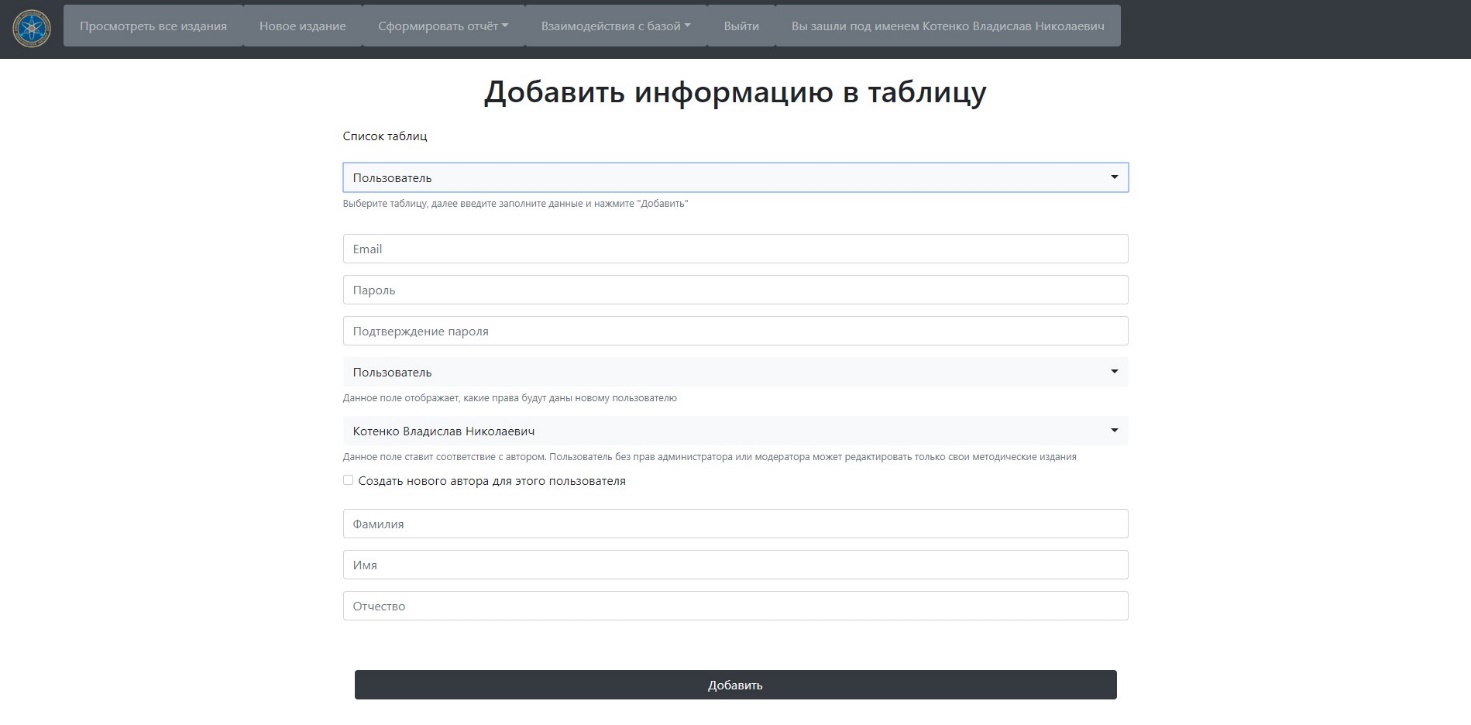
****

Рисунок А.4 – Страница с формой для добавления нового пользователя

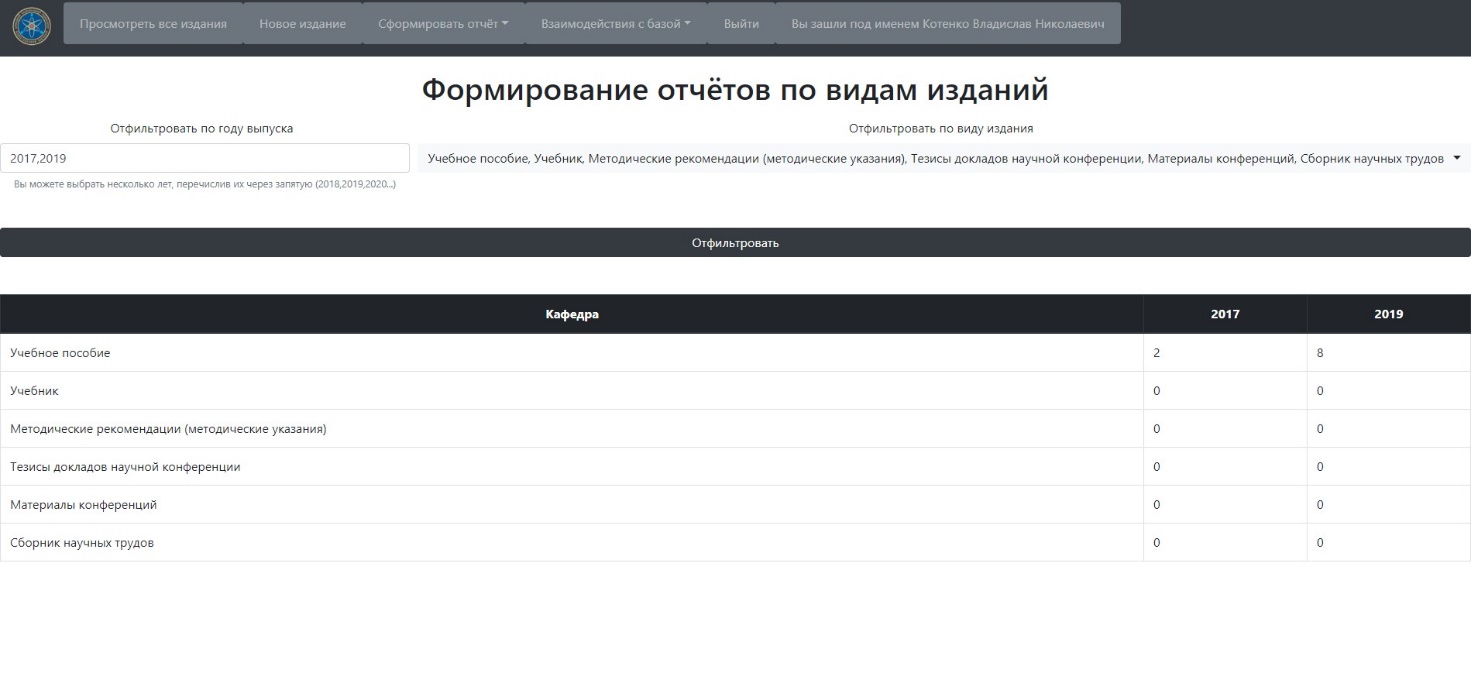
****

Рисунок А.5 – Страница с таблицей из отчёта

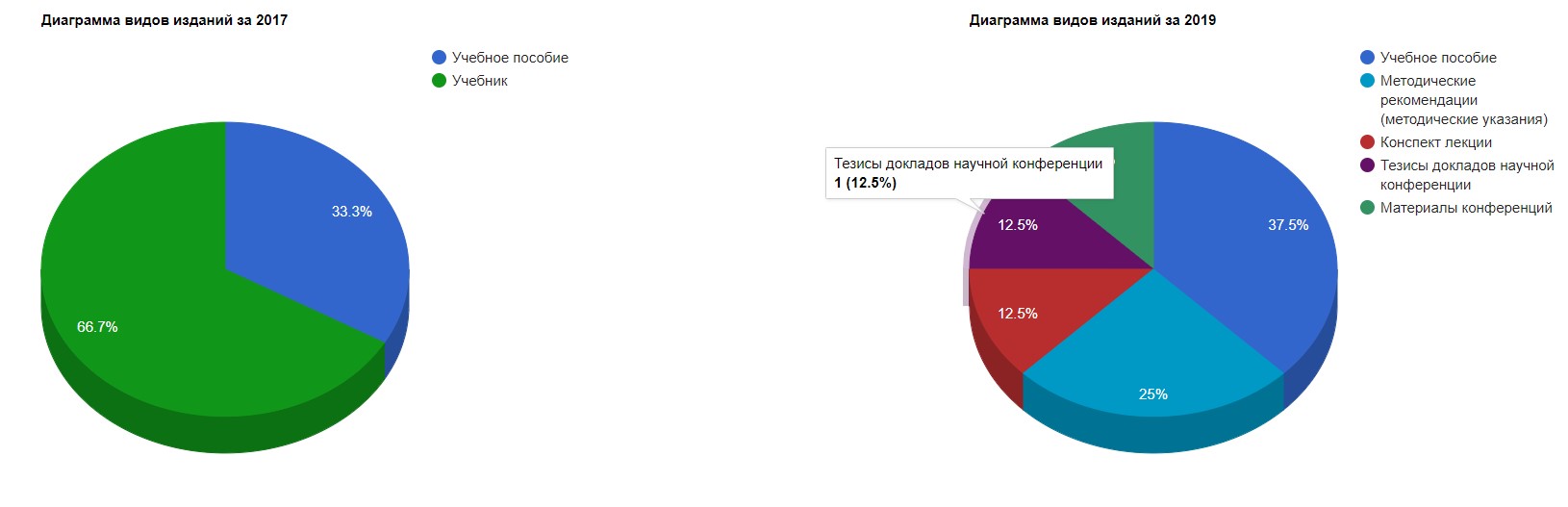
****

Рисунок А.6 – Страница с диаграммой из отчёта

# ПРИЛОЖЕНИЕ Б Фрагменты листинга

Листинг Б.1 **–** файл PublicationPlanController.php

class PublicationPlanController extends Controller  
{  
 public function \_\_construct()  
 {  
 $this->middleware('auth');  
 }  
 public function index(Request $request)  
 {  
 $query = 'SELECT publications.id, publications.chair\_id, chairs.name\_of\_chair, publications.discipline\_id, disciplines.name\_of\_discipline, publications.type\_publication\_id, type\_of\_publication.type\_publication\_name, publications.name\_of\_publication,  
 GROUP\_CONCAT(authors.last\_name,\' \', authors.name,\' \', authors.middle\_name) as authors,  
 papers\_sizes.format\_name , number\_of\_pages, number\_of\_copies, covers.cover\_type, publications.year\_of\_publication,  
 month\_of\_submissions.month\_name, publications.phone\_number, publications.is\_release  
 FROM  
 publications  
 LEFT JOIN authors\_publications  
 ON publications.id = authors\_publications.plan\_id  
 LEFT JOIN authors  
 ON authors.id = authors\_publications.author\_id  
 LEFT JOIN chairs  
 ON chairs.id = publications.chair\_id  
 LEFT JOIN disciplines  
 ON disciplines.id = publications.discipline\_id  
 LEFT JOIN type\_of\_publication  
 ON type\_of\_publication.id = publications.type\_publication\_id  
 LEFT JOIN papers\_sizes  
 ON papers\_sizes.id = publications.paper\_size\_id  
 LEFT JOIN covers  
 ON covers.id = publications.cover\_id  
 LEFT JOIN month\_of\_submissions  
 ON month\_of\_submissions.id = publications.month\_of\_submission\_id';  
 $first\_call = true; //был ли включен ранее фильтр, нужна для правильного формирования запроса  
 $select\_author = $request->input('select\_author');  
 $select\_year = $request->input('select\_year');  
 $select\_discipline = $request->input('select\_discipline');  
 $select\_chair = $request->input('select\_chair');  
 $select\_type = $request->input('select\_type');  
 $select\_status = $request->input('select\_status');  
 $disciplines\_table = Discipline::all();  
 $autors\_table = Author::all();  
 $chairs\_table = Chair::all();  
 $types\_table = TypeOfPublication::all();  
 if ($select\_author != null && $select\_author != -1) //Если выбрали фильтр по автору  
 {  
 $tmp\_query = 'SELECT authors\_publications.plan\_id FROM authors\_publications WHERE authors\_publications.author\_id = ' . $select\_author;

$selected\_plans = DB::select($tmp\_query, [1]);  
 foreach ($selected\_plans as $plan) {  
 $tmp\_query = '';  
 if ($first\_call == true) {  
 $first\_call = false;  
 $tmp\_query = $tmp\_query . ' WHERE authors\_publications.plan\_id = ' . $plan->plan\_id;  
 } else {  
 $tmp\_query = $tmp\_query . ' OR authors\_publications.plan\_id = ' . $plan->plan\_id;  
 }  
 $query = $query . $tmp\_query;  
 }  
 }  
 $query = $query . ' GROUP BY publications.id';  
 $plans = DB::select($query, [1]);  
  
 $user = Auth::user();  
 $roles = $user->getRoleNames(); // Returns a collection  
 $is\_admin = false;  
 foreach ($roles as $id => $name\_role) // проверяем права админа  
 {  
 if($name\_role == 'administrator' || $name\_role == 'moderator')  
 {  
 $is\_admin = true;  
 }  
 }  
 if ($is\_admin == false) //Если пользователь не админ  
 {  
 $id\_author = UsersAuthors::where('user\_id', '=', $user->id)->first();  
 if($id\_author == null)  
 {  
 $plans = null;  
 // load the view and pass the plans  
 return view('plans.index')->with([  
 'plans' => $plans,  
 'disciplines' => $disciplines\_table,  
 'autors' => $autors\_table,  
 'chairs' => $chairs\_table,  
 'types' => $types\_table,  
 'select\_chair' => $select\_chair,  
 'select\_type' => $select\_type,  
 'select\_year' => $select\_year,  
 'select\_author' => $select\_author,  
 'select\_discipline' => $select\_discipline,  
 'select\_status' => $select\_status,  
 ]);  
 }  
 $tmp\_query = 'SELECT authors\_publications.plan\_id FROM authors\_publications WHERE authors\_publications.author\_id = ' . $id\_author->author\_id; // одним запросом не выходит достать всех авторов, поэтому сначала ищем в каких публикациях они используются  
 $selected\_plans = DB::select($tmp\_query, [1]);  
 $tmp\_collection = collect();  
 foreach ($selected\_plans as $plan) {  
 foreach ($plans as $key => $value) {  
 if ($plan->plan\_id == $value->id) {  
 $tmp\_collection->push($value);  
 }  
 }  
 }  
 $plans = $tmp\_collection;  
 }  
 if ($select\_discipline != null && $select\_discipline != -1) //Если выбрали фильтр по дисциплине  
 {  
 $tmp\_collection = collect();  
 foreach ($plans as $key => $value) {  
 if ($value->discipline\_id == $select\_discipline) {  
 $tmp\_collection->push($value);  
 }  
 }  
 $plans = $tmp\_collection;  
 }  
 if ($select\_year != null && $select\_year != -1) //Если выбрали фильтр по году  
 {  
 $tmp\_collection = collect();  
 foreach ($plans as $key => $value) {  
 if ($value->year\_of\_publication == $select\_year) {  
 $tmp\_collection->push($value);  
 }  
 }  
 $plans = $tmp\_collection;  
 }  
 if ($select\_type != null && $select\_type != -1) //Если выбрали фильтр по типу издания  
 {  
 $tmp\_collection = collect();  
 foreach ($plans as $key => $value) {  
 if ($value->type\_publication\_id == $select\_type) {  
 $tmp\_collection->push($value);  
 }  
 }  
 $plans = $tmp\_collection;  
 }  
 if ($select\_chair != null && $select\_chair != -1) //Если выбрали фильтр по кафедре  
 {  
 $tmp\_collection = collect();  
 foreach ($plans as $key => $value) {  
 if ($value->chair\_id == $select\_chair) {  
 $tmp\_collection->push($value);  
 }  
 }  
 $plans = $tmp\_collection;  
 }  
 if($select\_status != null && $select\_status != -1)  
 {  
 $tmp\_collection = collect();  
 foreach ($plans as $key => $value) {  
 if ($value->is\_release == $select\_status) {  
 $tmp\_collection->push($value);  
 }  
 }  
 $plans = $tmp\_collection;  
 }  
 foreach ($plans as $key => $value) //проверяем, есть ли на сервере файл  
 {  
 $filePath = glob(public\_path() . '/docx/Document\_id\_' . $value->id . '.docx');  
 if ($filePath != null) {  
 $filePath[0] = str\_replace(public\_path() . '/', '', $filePath[0]);  
 $value->filePath = $filePath[0];  
 } else {  
 $value->filePath = 'none';  
 }  
 }  
 // load the view and pass the plans  
 return view('plans.index')->with([  
 'plans' => $plans,  
 'disciplines' => $disciplines\_table,  
 'autors' => $autors\_table,  
 'chairs' => $chairs\_table,  
 'types' => $types\_table,  
 'select\_chair' => $select\_chair,  
 'select\_type' => $select\_type,  
 'select\_year' => $select\_year,  
 'select\_author' => $select\_author,  
 'select\_discipline' => $select\_discipline,  
 'select\_status' => $select\_status,  
 ]);  
 }  
 public function create()  
 {  
 $papers\_size = PapersSize::all();  
 $chairs = Chair::all();  
 $months = MonthOfSubmission::all();  
 $cover = Covers::all();  
 $type\_publication = TypeOfPublication::all();  
 $disciplines = Discipline::all();  
 $authors = Author::all();  
 // load the create form (app/views/plans/create.blade.php)  
 return view('plans.create', [  
 'chairs' => $chairs,  
 'papers\_size' => $papers\_size,  
 'months' => $months,  
 'cover' => $cover,  
 'type\_publication' => $type\_publication,  
 'disciplines' => $disciplines,  
 'authors' => $authors  
 ]);  
 }  
 public function store(Request $request)  
 {  
 // validate  
 $request->validate([  
 'chair\_id' => 'required|numeric',  
 'discipline\_id' => 'required|numeric',  
 'type\_publication\_id' => 'required|numeric',  
 'name\_of\_publication' => 'required',  
 'author\_id' => "required|array|min:1",  
 'paper\_size\_id' => 'required|numeric',  
 'number\_of\_pages' => 'required|numeric',  
 'number\_of\_copies' => 'required|numeric',  
 'cover\_id' => 'required',  
 'month\_of\_submission\_id' => 'required|numeric',  
 'year\_of\_publication' => 'required|numeric',  
 'is\_release' => 'required|numeric',  
 ]);  
 // store  
 $plan = new Publications();  
 $plan->chair\_id = $request->input('chair\_id');  
 $plan->discipline\_id = $request->input('discipline\_id');  
 $plan->type\_publication\_id = $request->input('type\_publication\_id');  
 $plan->name\_of\_publication = $request->input('name\_of\_publication');  
 $plan->paper\_size\_id = $request->input('paper\_size\_id');  
 $plan->number\_of\_pages = $request->input('number\_of\_pages');  
 $plan->number\_of\_copies = $request->input('number\_of\_copies');  
 $plan->cover\_id = $request->input('cover\_id');  
 $plan->month\_of\_submission\_id = $request->input('month\_of\_submission\_id');  
 $plan->year\_of\_publication = $request->input('year\_of\_publication');  
 $plan->phone\_number = $request->input('phone\_number');  
 $plan->is\_release = $request->input('is\_release');  
 $plan->save();  
 $newid = Publications::latest()->first()->id;  
 if ($request->hasFile('doc\_file')) {  
 $file = $request->file('doc\_file');  
 $file->move(public\_path() . '/docx', 'Document\_id\_' . $newid . '.docx');  
 }  
 $array\_authors = $request->input('author\_id.\*');  
 if (count($array\_authors) > 0) {  
 for ($i = 0; $i < count($array\_authors); $i++) {  
 $plan\_author = new AuthorsPublications();  
 $plan\_author->plan\_id = $newid;  
 $plan\_author->author\_id = $array\_authors[$i];  
 $plan\_author->save();  
 }  
 }  
 // redirect  
 Session::flash('message', 'Новое издание создано!');  
 return redirect('/plans');  
 }  
 public function edit($id)  
 {  
 // get the plan  
 $plan = Publications::find($id); //selected plan  
 $chairs = Chair::all();  
 $papers\_size = PapersSize::all();  
 $months = MonthOfSubmission::all();  
 $cover = Covers::all();  
 $type\_publication = TypeOfPublication::all();  
 $disciplines = Discipline::all();  
 $authors = Author::all();  
 $plan\_authors = DB::table('authors\_publications')->where('plan\_id', '=', $id)->get();  
 $selected\_authors = collect();  
 $unselected\_authors = collect();  
 for ($i = 0; $i < count($authors); $i++) {  
 for ($j = 0; $j < count($plan\_authors); $j++) {  
 if ($authors[$i]->id == $plan\_authors[$j]->author\_id) {  
 $selected\_authors->push($authors->slice($i, 1));  
 }  
 }  
 }  
 for ($i = 0; $i < count($authors); $i++) {  
 for ($j = 0, $selected = false; $j < count($plan\_authors); $j++) {  
 if ($authors[$i]->id == $plan\_authors[$j]->author\_id) {  
 $selected = true;  
 }  
 }  
 if ($selected == false) {  
 $unselected\_authors->push($authors->slice($i, 1));  
 }  
 }  
 return view('plans.edit', [  
 'plan' => $plan,  
 'chairs' => $chairs,  
 'papers\_size' => $papers\_size,  
 'months' => $months,  
 'cover' => $cover,  
 'type\_publication' => $type\_publication,  
 'disciplines' => $disciplines,  
 'authors' => $authors,  
 'plan\_authors' => $plan\_authors,  
 'selected\_authors' => $selected\_authors,  
 'unselected\_authors' => $unselected\_authors,  
 ]);  
 }  
 public function update(Request $request, $id)  
 {  
 // validate  
 // read more on validation at http://laravel.com/docs/validation  
 $request->validate([  
 'chair\_id' => 'required|numeric',  
 'discipline\_id' => 'required|numeric',  
 'type\_publication\_id' => 'required|numeric',  
 'name\_of\_publication' => 'required',  
 'author\_id' => "required|array|min:1",  
 'paper\_size\_id' => 'required|numeric',  
 'number\_of\_pages' => 'required|numeric',  
 'number\_of\_copies' => 'required|numeric',  
 'cover\_id' => 'required',  
 'month\_of\_submission\_id' => 'required|numeric',  
 'year\_of\_publication' => 'required|numeric',  
 'is\_release' => 'required|numeric',  
 ]);  
 // store  
 DB::table('authors\_publications')->where('plan\_id', '=', $id)->delete();  
 $array\_authors = $request->input('author\_id.\*');  
 if (count($array\_authors) > 0) {  
 for ($i = 0; $i < count($array\_authors); $i++) {  
 $plan\_author = new AuthorsPublications();  
 $plan\_author->plan\_id = $id;  
 $plan\_author->author\_id = $array\_authors[$i];  
 $plan\_author->save();  
 }  
 }  
 $plan = Publications::find($id);  
 $plan->chair\_id = $request->input('chair\_id');  
 $plan->discipline\_id = $request->input('discipline\_id');  
 $plan->type\_publication\_id = $request->input('type\_publication\_id');  
 $plan->name\_of\_publication = $request->input('name\_of\_publication');  
 $plan->paper\_size\_id = $request->input('paper\_size\_id');  
 $plan->number\_of\_pages = $request->input('number\_of\_pages');  
 $plan->number\_of\_copies = $request->input('number\_of\_copies');  
 $plan->cover\_id = $request->input('cover\_id');  
 $plan->month\_of\_submission\_id = $request->input('month\_of\_submission\_id');  
 $plan->year\_of\_publication = $request->input('year\_of\_publication');  
 $plan->phone\_number = $request->input('phone\_number');  
 $plan->is\_release = $request->input('is\_release');  
 $plan->save();  
 if ($request->hasFile('doc\_file')) {  
 $filePath = glob(public\_path() . '/docx/Document\_id\_' . $id . '.docx');  
 if ($filePath != null) {  
 unlink($filePath[0]);  
 }  
 $file = $request->file('doc\_file');  
 $file->move(public\_path() . '/docx', 'Document\_id\_' . $id . '.docx');  
 }  
 // redirect  
 Session::flash('message', 'Издание было изменено!');  
 return redirect('/plans');  
 }  
 public function destroy($id)  
 {  
 // delete foreign key  
 $deletedRows = AuthorsPublications::where('plan\_id', '=', $id)->delete();  
 // delete  
 $plan = Publications::find($id);  
 $plan->delete();  
 $filePath = glob(public\_path() . '/docx/Document\_id\_' . $id . '.docx');  
 if ($filePath != null) {  
 unlink($filePath[0]);  
 }  
 // redirect  
 Session::flash('message', 'Издание было удалено!');  
 return redirect('/plans');  
 }  
}

Листинг Б.2 **–** файл AdditionsToBaseController.php

class AdditionsToBaseController extends Controller  
{  
 public function \_\_construct()  
 {  
 $this->middleware('auth');  
 }  
 public function create()  
 {  
 $authors = Author::all();  
 return view("add-to-base")->with([  
 'authors' => $authors  
 ]);  
 }  
 public function store(Request $request)  
 {  
 //what table we need  
 $select\_table = $request->input('table');  
 switch ($select\_table) {  
 case 'discipline':  
 // validate  
 $request->validate([  
 'field\_1' => 'required',  
 ]);  
 $discipline = new Discipline();  
 $discipline->name\_of\_discipline = $request->input('field\_1');  
 $discipline->save();  
 Session::flash('message', 'Дисциплина успешно создана!');  
 break;  
 case 'type\_publication':  
 // validate  
 $request->validate([  
 'field\_1' => 'required',  
 ]);  
 $type = new TypeOfPublication();  
 $type->type\_of\_publication = $request->input('field\_1');  
 $type->save();  
 Session::flash('message', 'Новый вид публикаций успешно создан!');  
 break;  
 case 'user':  
 // validate  
 $request->validate([  
 'field\_1' => 'required',  
 'field\_2' => 'required',  
 'field\_3' => 'required',  
 'type\_user' => 'required',  
 'field\_4' => 'required',  
 'field\_5' => 'required',  
 'field\_6' => 'required',  
 ]);  
 if($request->input('field\_2') != $request->input('field\_3')) //check confirm password  
 {  
 Session::flash('error', 'Пароль и подтверждение пароля не совпадают!');  
 return redirect('/add-to-base');  
 }  
  
 $user = new User();  
 $user->email = $request->input('field\_1');  
 $user->password = bcrypt($request->input('field\_2'));  
 $user->last\_name = $request->input('field\_4');  
 $user->name = $request->input('field\_5');  
 $user->middle\_name = $request->input('field\_6');  
 switch ($request->input('type\_user'))  
 {  
 case 'administrator':  
 $user->assignRole('administrator');  
 break;  
 case 'moderator':  
 $user->assignRole('moderator');  
 break;  
 case 'user':  
 $user->assignRole('user');  
 break;  
 }  
  
 $user->save();  
 if($request->add\_new\_authors == null)  
 {  
 $id\_author = $request->user\_author;  
 $is\_alredy\_taken = UsersAuthors::where('author\_id', $id\_author)->first();  
 if($is\_alredy\_taken != null)  
 {  
 $tmp\_user = User::find($is\_alredy\_taken->user\_id);  
 $user->delete();  
 Session::flash('error', 'У автора уже стоит соответствие с ' . $tmp\_user->last\_name .' ' . $tmp\_user->name . ' ' . $tmp\_user->middle\_name .'!');  
 return redirect('/add-to-base');  
 }  
 $author\_user = new UsersAuthors();  
 $author\_user->user\_id = $user->id;  
 $author\_user->author\_id = $id\_author;  
 $author\_user->save();  
 }  
 else  
 {  
 $new\_author = new Author();  
 $new\_author->last\_name = $request->input('field\_4');  
 $new\_author->name = $request->input('field\_5');  
 $new\_author->middle\_name = $request->input('field\_6');  
 $new\_author->save();  
 $author\_user = new UsersAuthors();  
 $author\_user->user\_id = $user->id;  
 $author\_user->author\_id = $new\_author->id;  
 $author\_user->save();  
 }  
 Session::flash('message', 'Пользователь успешно создан!');  
 break;  
 case 'author':  
 // validate  
 $request->validate([  
 'field\_1' => 'required',  
 'field\_2' => 'required',  
 'field\_3' => 'required',  
 ]);  
 $author = new Author();  
 $author->last\_name = $request->input('field\_1');  
 $author->name = $request->input('field\_2');  
 $author->middle\_name = $request->input('field\_3');  
 $author->save();  
 Session::flash('message', 'Автор успешно создан!');  
 break;  
 case 'chair':// validate  
 $request->validate([  
 'field\_1' => 'required',  
 ]);  
 $chair = new Chair();  
 $chair->name\_of\_chair = $request->input('field\_1');  
 $chair->save();  
 Session::flash('message', 'Кафедра успешно создана!');  
 break;  
 default:  
 Session::flash('message', 'Что-то пошло не так. Данные не сохранены!');  
 return redirect('/add-to-base');  
 break;  
 }  
 return redirect('/add-to-base');  
 }  
  
 public function update\_table\_combobox(Request $request)  
 {  
 $select\_table = $request->select\_table;  
 $values = null;  
 switch ($select\_table) {  
 case 'discipline':  
 $values = Discipline::all();  
 $all\_tmp\_values = collect();  
 $tmp\_values = collect();  
 foreach ($values as $key => $value)  
 {  
 $tmp\_values->put('key', $value->id);  
 $tmp\_values->put('value', $value->name\_of\_discipline);  
 $all\_tmp\_values->push($tmp\_values);  
 $tmp\_values = collect();  
 }  
 $values = $all\_tmp\_values;  
 break;  
 case 'type\_publication':  
 $values = TypeOfPublication::all();  
 $all\_tmp\_values = collect();  
 $tmp\_values = collect();  
  
 foreach ($values as $key => $value)  
 {  
 $tmp\_values->put('key', $value->id);  
 $tmp\_values->put('value', $value->type\_publication\_name);  
 $all\_tmp\_values->push($tmp\_values);  
 $tmp\_values = collect();  
 }  
 $values = $all\_tmp\_values;  
 break;  
 case 'user':  
 $values = User::all();  
 $all\_tmp\_values = collect();  
 $tmp\_values = collect();  
 foreach ($values as $key => $value)  
 {  
 $tmp\_values->put('key', $value->id);  
 $tmp\_values->put('value', $value->last\_name.' '.$value->name.' '.$value->middle\_name);  
 $all\_tmp\_values->push($tmp\_values);  
 $tmp\_values = collect();  
 }  
 $values = $all\_tmp\_values;  
 break;  
 case 'author':  
 $values = Author::all();  
 $all\_tmp\_values = collect();  
 $tmp\_values = collect();  
  
 foreach ($values as $key => $value)  
 {  
 $tmp\_values->put('key', $value->id);  
 $tmp\_values->put('value', $value->last\_name.' '.$value->name.' '.$value->middle\_name);  
 $all\_tmp\_values->push($tmp\_values);  
 $tmp\_values = collect();  
 }  
 $values = $all\_tmp\_values;  
 break;  
 case 'chair':  
 $values = Chair::all();  
 $all\_tmp\_values = collect();  
 $tmp\_values = collect();  
  
 foreach ($values as $key => $value)  
 {  
 $tmp\_values->put('key', $value->id);  
 $tmp\_values->put('value', $value->name\_of\_chair);  
 $all\_tmp\_values->push($tmp\_values);  
 $tmp\_values = collect();  
 }  
 $values = $all\_tmp\_values;  
 break;  
 }  
 return $values;  
 }  
 public function change()  
 {  
 return view("select-table-for-update-base");  
 }  
 public function select\_data\_for\_update(Request $request)  
 {  
 $select\_data = $request->select\_data;  
 $data = collect();  
 switch ($select\_data) {  
 case 'discipline':  
 $discipline = Discipline::find($request->data);  
 $data->push($discipline->name\_of\_discipline);  
 break;  
 case 'type\_publication':  
 $type\_publication = TypeOfPublication::find($request->data);  
 $data->push($type\_publication->type\_publication\_name);  
 break;  
 case 'user':  
 $authors = Author::all();  
 $user = User::find($request->data);  
 $data->push($user->email);  
 $data->push($user->last\_name);  
 $data->push($user->name);  
 $data->push($user->middle\_name);  
  
 //т.к. присутствует несколько ролей, нужно перебрать от высшего к низшему  
 $roles = $user->getRoleNames(); // Returns a collection  
 $role\_found = false;  
 foreach ($roles as $id => $name\_role)  
 {  
 if($name\_role == 'administrator')  
 {  
 $data->push($name\_role);  
 $role\_found = true;  
 break;  
 }  
 }  
 if($role\_found == false)  
 {  
 foreach ($roles as $id => $name\_role)  
 {  
 if($name\_role == 'moderator')  
 {  
 $data->push($name\_role);  
 $role\_found = true;  
 break;  
 }  
 }  
 }  
 if($role\_found == false) {  
 foreach ($roles as $id => $name\_role) {  
 if ($name\_role == 'user') {  
 $data->push($name\_role);  
 $role\_found = true;  
 break;  
 }  
 }  
 }  
 $selected\_author = UsersAuthors::where('user\_id', $user->id)->first();  
 return view("update-base")->with([  
 'select\_data' => $select\_data,  
 'data' => $data,  
 'id' => $request->data,  
 'authors' => $authors,  
 'select\_author' => $selected\_author  
 ]);  
 break;  
 case 'author':  
 $author = Author::find($request->data);  
 $data->push($author->last\_name);  
 $data->push($author->name);  
 $data->push($author->middle\_name);  
 break;  
 case 'chair':  
 $chair = Chair::find($request->data);  
 $data->push($chair->name\_of\_chair);  
 break;  
 }  
 return view("update-base")->with([  
 'select\_data' => $select\_data,  
 'data' => $data,  
 'id' => $request->data,  
 ]);  
 }  
 public function update(Request $request)  
 {  
 $id = $request->input('id');  
 switch ($request->input('select\_data')) {  
 case 'discipline':  
 $request->validate([  
 'discipline' => 'required',  
 ]);  
 $discipline = Discipline::find($id);  
 $old\_name = $discipline->name\_of\_discipline;  
 $discipline->name\_of\_discipline = $request->input('discipline');  
 $new\_name = $discipline->name\_of\_discipline;  
 $discipline->save();  
 break;  
 case 'type\_publication':  
 $request->validate([  
 'type\_publication' => 'required',  
 ]);  
 $type = TypeOfPublication::find($id);  
 $old\_name = $type->type\_publication\_name;  
 $type->type\_publication\_name = $request->input('type\_publication');  
 $new\_name = $type->type\_publication\_name;  
 $type->save();  
 break;  
 case 'user':  
 $request->validate([  
 'user\_email' => 'required',  
 'type\_user' => 'required',  
 'user\_author' => 'required',  
 'user\_last\_name' => 'required',  
 'user\_name' => 'required',  
 'user\_middle\_name' => 'required',  
 ]);  
 $password = $request->input('user\_password');  
 $password\_confirm = $request->input('user\_password\_confirm');  
 if($password != $password\_confirm)  
 {  
 Session::flash('error', 'Пароль и подтверждение пароля не совпадают!');  
 return redirect('/update-base');  
 }  
 $user = User::find($id);  
 $old\_name = $user->name;  
 $user->email = $request->input('user\_email');  
 if($password != '' || $password != null)  
 {  
 $user->password = bcrypt($request->input('user\_password'));  
 }  
 $selected\_user\_author = UsersAuthors::where('user\_id', $user->id)->first();  
 if($request->add\_new\_authors == null)  
 {  
 $id\_author = $request->user\_author;  
 if($selected\_user\_author != null && $selected\_user\_author->author\_id != $id\_author)  
 {  
 $tmp\_user = User::find($selected\_user\_author->user\_id);  
 Session::flash('error', 'У автора уже стоит соответствие с ' . $tmp\_user->last\_name .' ' . $tmp\_user->name . ' ' . $tmp\_user->middle\_name .'!');  
 return redirect('/select-table-for-update-base');  
 }  
 }  
 else  
 {  
 UsersAuthors::where('user\_id', $user->id)->delete();  
 $new\_author = new Author();  
 $new\_author->last\_name = $request->input('field\_4');  
 $new\_author->name = $request->input('field\_5');  
 $new\_author->middle\_name = $request->input('field\_6');  
 $new\_author->save();  
 $author\_user = new UsersAuthors();  
 $author\_user->user\_id = $user->id;  
 $author\_user->author\_id = $new\_author->id;  
 $author\_user->save();  
 }  
 $user->last\_name = $request->input('user\_last\_name');  
 $user->name = $request->input('user\_name');  
 $user->middle\_name = $request->input('user\_middle\_name');  
 $new\_name = $user->name;  
 DB::table('model\_has\_roles')->where('model\_id', '=', $id)->delete();  
 $user->assignRole($request->input('type\_user'));  
 $user->save();  
 break;  
 case 'author':  
 $request->validate([  
 'author\_last\_name' => 'required',  
 'author\_name' => 'required',  
 'author\_middle\_name' => 'required',  
 ]);  
 $autor = Author::find($id);  
 $old\_name = $autor->name;  
 $autor->last\_name = $request->input('author\_last\_name');  
 $autor->name = $request->input('author\_name');  
 $autor->middle\_name = $request->input('author\_middle\_name');  
 $new\_name = $autor->name;  
 $autor->save();  
 break;  
 case 'chair':  
 $request->validate([  
 'chair' => 'required',  
 ]);  
 $chair = Chair::find($id);  
 $old\_name = $chair->name\_of\_chair;  
 $chair->name\_of\_chair = $request->input('chair');  
 $new\_name = $chair->name\_of\_chair;  
 $chair->save();  
 break;  
 }  
 //Session::flash('message', $old\_name . ' изменён на ' . $new\_name);  
 Session::flash('message', 'Изменения сохранены!');  
 return redirect('/select-table-for-update-base');  
 }  
 public function remove()  
 {  
 return view("select-table-for-remove-from-base");  
 }

Листинг Б.3 **–** файл ReportController.php

class ReportController extends Controller  
{  
 public function \_\_construct()  
 {  
 $this->middleware('auth');  
 }  
  
 public function ReportForChair(Request $request)  
 {  
 //Формируем отчет какая кафедра, сколько изданий в год сделала  
 $first\_call = true;  
 $selected\_year = $request->input('select\_year');  
 $selected\_chair = $request->input('select\_chair.\*');  
 $query = "SELECT id, name\_of\_chair FROM Chairs";  
 $chairs = DB::select($query, [1]);;  
 $collection = collect();  
 if ($selected\_year != null) //Если пользователь ввел определенные год(а)  
 {  
 $first\_call = false;  
 $years = explode(',', $selected\_year);  
 foreach ($chairs as $chair) {  
 $counts = collect();  
 foreach ($years as $year) {  
 $count = DB::table('publications')  
 ->where('chair\_id', '=', $chair->id)  
 ->where('publications.year\_of\_publication', '=', $year)  
 ->where('publications.is\_release', '=', 1)  
 ->count();  
 $counts->push($count);  
 }  
 $collection->put($chair->name\_of\_chair, $counts);  
 }  
 }  
 if ($selected\_chair != null && $selected\_chair[0] != -1) //Если пользователь ввел определенные кафедры  
 {  
 if ($first\_call == true) {  
 $first\_call = false;  
 $years = Publications::groupBy('year\_of\_publication')  
 ->select("year\_of\_publication")  
 ->get();  
 foreach ($chairs as $chair) {  
 $counts = collect();  
 foreach ($years as $year) {  
 $count = DB::table('publications')  
 ->where('chair\_id', '=', $chair->id)  
 ->where('publications.year\_of\_publication', '=', $year->year\_of\_publication)  
 ->where('publications.is\_release', '=', 1)  
 ->count();  
 $counts->push($count);  
 }  
 $collection->put($chair->name\_of\_chair, $counts);  
 }  
 $tmp\_collection = collect();  
 foreach ($selected\_chair as $key => $value) {  
 foreach ($chairs as $k => $v) {  
 if ($value == $v->id) {  
 foreach ($collection as $c\_key => $c\_value) {  
 if ($v->name\_of\_chair == $c\_key) {  
 $tmp\_collection->put($c\_key, $c\_value);  
 break;  
 }  
 }  
 }  
 }  
 }  
 $collection = $tmp\_collection;  
 $tmp\_years = collect();  
 foreach ($years as $year) {  
 $tmp\_years->push((string)$year->year\_of\_publication);  
 }  
 $years = $tmp\_years;  
 }  
 else {  
 $first\_call = false;  
  
 foreach ($chairs as $chair) {  
 $counts = collect();  
 foreach ($years as $year) {  
 $count = DB::table('publications')  
 ->where('chair\_id', '=', $chair->id)  
 ->where('publications.year\_of\_publication', '=', $year)  
 ->where('publications.is\_release', '=', 1)  
 ->count();  
 $counts->push($count);  
 }  
 $collection->put($chair->name\_of\_chair, $counts);  
 }  
 $tmp\_collection = collect();  
 foreach ($selected\_chair as $key => $value) {  
 foreach ($chairs as $k => $v) {  
 if ($value == $v->id) {  
 foreach ($collection as $c\_key => $c\_value) {  
 if ($v->name\_of\_chair == $c\_key) {  
 $tmp\_collection->put($c\_key, $c\_value);  
 break;  
 }  
 }  
 }  
 }  
 }  
 $collection = $tmp\_collection;  
 }  
 }  
  
 if ($selected\_chair == null && $selected\_year == null) //Если пользователь ничего не вводил  
 {  
 $years = Publications::groupBy('year\_of\_publication')  
 ->select("year\_of\_publication")  
 ->get();  
 $tmp\_years = collect();  
 foreach ($years as $year) {  
 $tmp\_years->push((string)$year->year\_of\_publication);  
 }  
 $years = $tmp\_years;  
 foreach ($chairs as $chair) {  
 $counts = collect();  
 foreach ($years as $year) {  
 $count = DB::table('publications')  
 ->where('chair\_id', '=', $chair->id)  
 ->where('publications.year\_of\_publication', '=', $year)  
 ->where('publications.is\_release', '=', 1)  
 ->count();  
 $counts->push($count);  
 }  
 $collection->put($chair->name\_of\_chair, $counts);  
 }  
 }  
 return view('report\_chair')->with([  
 'collection' => $collection,  
 'select\_year' => $selected\_year,  
 'select\_chair' => $selected\_chair,  
 'chairs' => $chairs,  
 'years' => $years,  
 ]);  
 }  
 public function ReportForTypePublication(Request $request)  
 {  
 //Формируем отчет сколько и какие издания в год сделала  
 $first\_call = true;  
 $selected\_year = $request->input('select\_year');  
 $selected\_types = $request->input('select\_type.\*');  
 $types = TypeOfPublication::all();  
 $collection = collect();  
 if ($selected\_year != null) //Если выбрали определенный год(а)  
 {  
 $first\_call = false;  
 $years = explode(',', $selected\_year);  
 foreach ($types as $type)  
 {  
 $counts = collect();  
 foreach ($years as $year)  
 {  
 $count = DB::table('publications')  
 ->where('type\_publication\_id', '=', $type->id)  
 ->where('publications.year\_of\_publication', '=', $year)  
 ->where('publications.is\_release', '=', 1)  
 ->count();  
 $counts->push($count);  
 }  
 $collection->put($type->type\_publication\_name, $counts);  
 }  
 }  
 if ($selected\_types != null && $selected\_types[0] != -1) //Если пользователь ввел определенные кафедры  
 {  
 if ($first\_call == true) {  
 $first\_call = false;  
 $years = Publications::groupBy('year\_of\_publication')  
 ->select("year\_of\_publication")  
 ->get();  
 foreach ($types as $type) {  
 $counts = collect();  
 foreach ($years as $year) {  
 $count = DB::table('publications')  
 ->where('type\_publication\_id', '=', $type->id)  
 ->where('publications.year\_of\_publication', '=', $year->year\_of\_publication)  
 ->where('publications.is\_release', '=', 1)  
 ->count();  
 $counts->push($count);  
 }  
 $collection->put($type->type\_publication\_name, $counts);  
 }  
 //dd($collection);  
 $tmp\_collection = collect();  
 foreach ($selected\_types as $key => $value) {  
 foreach ($types as $k => $v) {  
 if ($value == $v->id) {  
 foreach ($collection as $c\_key => $c\_value) {  
 if ($v->type\_publication\_name == $c\_key) {  
 $tmp\_collection->put($c\_key, $c\_value);  
 break;  
 }  
 }  
 }  
 }  
 }  
 $collection = $tmp\_collection;  
 $tmp\_years = collect();  
 foreach ($years as $year) {  
 $tmp\_years->push((string)$year->year\_of\_publication);  
 }  
 $years = $tmp\_years;  
 }  
 else {  
  
 foreach ($types as $type) {  
 $counts = collect();  
 foreach ($years as $year) {  
 $count = DB::table('publications')  
 ->where('chair\_id', '=', $type->id)  
 ->where('publications.year\_of\_publication', '=', $year)  
 ->where('publications.is\_release', '=', 1)  
 ->count();  
 $counts->push($count);  
 }  
 $collection->put($type->type\_publication\_name, $counts);  
 }  
 $tmp\_collection = collect();  
 foreach ($selected\_types as $key => $value) {  
 foreach ($types as $k => $v) {  
 if ($value == $v->id) {  
 foreach ($collection as $c\_key => $c\_value) {  
 if ($v->type\_publication\_name == $c\_key) {  
 $tmp\_collection->put($c\_key, $c\_value);  
 break;  
 }  
 }  
 }  
 }  
 }  
 $collection = $tmp\_collection;  
 }  
 }  
 if($selected\_types == null && $selected\_year == null)//Если не выбрали год(а) то выводим за все года что есть  
 {  
 $years = Publications::groupBy('year\_of\_publication')  
 ->select("year\_of\_publication")  
 ->get();  
 foreach ($types as $type)  
 {  
 $counts = collect();  
 foreach ($years as $year)  
 {  
 $count = DB::table('publications')  
 ->where('type\_publication\_id', '=', $type->id)  
 ->where('publications.year\_of\_publication', '=', $year->year\_of\_publication)  
 ->where('publications.is\_release', '=', 1)  
 ->count();  
 $counts->push($count);  
 }  
 $collection->put($type->type\_publication\_name, $counts);  
 }  
 $tmp\_years = collect();  
 foreach ($years as $year) {  
 $tmp\_years->push((string)$year->year\_of\_publication);  
 }  
 $years = $tmp\_years;  
 }  
 return view('report\_type\_publication')->with([  
 'collection' => $collection,  
 'select\_year' => $selected\_year,  
 'types' => $types,  
 'select\_types' => $selected\_types,  
 'years' => $years,  
 ]);  
 }