

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**   
**«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»**

Факультет Физико-технический

Кафедра Компьютерных технологий (КТ)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | КТ |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Т. В. Ермоленко |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе бакалавра 3 курса

на тему:

|  |
| --- |
| РАЗРАБОТКА ДИНАМИЧЕСКОГО САЙТА ДЛЯ УЧЕТА МЕТОДИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ КАФЕДРЫ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| Автор работы |  |  | Д.И. Фурсов |

подпись

|  |  |  |
| --- | --- | --- |
| Направление | 09.03.01 | Информатика и вычислительная техника |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель В.Н. Котенко

подпись

Консультанты по разделам:

Сетевые технологии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель А.Е. Гукай

подпись

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. лаборант В.Г. Медведева

подпись

Курсовая работа защищена \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата итоговая оценка комиссия

Подписи членов комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Донецк  
2019

ГОУ ВПО «Донецкий национальный университет»

кафедра Компьютерных технологий

Утверждаю

Зав. кафедрой КТ

\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

**ЗАДАНИЕ**

на курсовую работу студента 3 курса Фурсова Д.В.

*Тема курсовой работы*: Разработка динамического сайта для учета методической деятельности кафедры компьютерных технологий

*Краткая постановка задачи*: 1. Изучить и проанализировать предметную область методической деятельности кафедры 2. Описать бизнес-процессы системы. 3. Ознакомиться с программными продуктами для создания динамических сайтов, выполнить установку Web-сервера Apache, реляционной системы управления базами данных MySQL, языка разработки серверных скриптов PHP, веб-фреймворка Laravel. 4. Разработать техническое задание на создание динамического сайта. 5. Разработать проект программного обеспечения системы: спроектировать концептуальную, логическую и физическую модели базы данных, создать базу данных для хранения информации о методических изданиях кафедры компьютерных технологий, преподавателях, дисциплин, видах изданий. 6. Разработать средствами языка программирования серверных приложений PHP, веб-фреймворка Laravel и набора инструментов Bootstrap 4 динамический сайт, который позволяет выполнять учет методических изданий кафедры компьютерных технологий. 8. Протестировать программное обеспечение; 9. Оформить отчёт.

*Исходные данные*: 1. Рекомендации для учета методической деятельности кафедры. 2. Документация по веб-фреймворку Laravel. 3. Документация по набору инструментов Bootstrap 4. 4. Документация по системе управления базами данных MySQL.

*Ожидаемые результаты*: Разработка динамического сайта для учета методической деятельности кафедры компьютерных технологий.

*Календарный план работы:*

|  |  |  |
| --- | --- | --- |
| Даты консультаций | Этапы выполнения работы | Отметки о выполнении |
| 01.02.2019 | Постановка задачи и обсуждение литературы | Выполнено |
| 08.02.2019 | Предварительное утверждение содержания отчёта | выполнено |
| 15.02.2019 | Утверждение проекта, алгоритмов, методов, технологий | выполнено |
| 13.03.2019 | Ход реализации проекта | выполнено |
| 21.03.2019 | Обсуждения организации тестирования программы | выполнено |
| 01.02.2019 | Демонстрация программного продукта руководителю | выполнено |
| 01.02.2019 | Оформление отчёта | выполнено |
| 01.02.2019 | Предоставление отчёта руководителю | выполнено |

Дата выдачи задания 01.02.2019 года

Студент Д.В. Фурсов

Руководитель В.Н. Котенко

**АННОТАЦИЯ**

Отчёт о курсовой работе: 81 с., 50 рис., 7 табл., 2 приложения,   
13 источников.

Объект исследования – методической деятельность кафедры компьютерных технологий.

Предмет исследования – автоматизация учета методической деятельности кафедры компьютерных технологий.

Цель работы – разработать динамический сайт для автоматизации учета методической деятельности кафедры компьютерных технологий.

Метод исследования – анализ возможностей языка PHP и фреймворка Laravel для создания и обработки баз данных и разработки серверных приложений.

В курсовой работе был разработан динамический сайт для автоматизации учета методической деятельности кафедры компьютерных технологий.

Динамический сайт может использоваться преподавателем, который ведет учёт методической деятельности кафедры компьютерных технологий

Дальнейшее развитие системы связано с добавлением веб-фреймворка Vue.JS, расширением базы для возможности учета методических изданий физико-технического факультета, расширением круга пользователей, включая студентов, возможностью онлайн просмотра и скачивания методических изданий.

УЧЕТ МЕТОДИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ КАФЕДРЫ, ДИНАМИЧЕСКИЙ САЙТ, APACHE, MYSQL, PHP, LARAVEL.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc6866314)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc6866315)

[1.1 Состояние вопроса 8](#_Toc6866316)

[1.2 Моделирование существующих бизнес-процессов 11](#_Toc6866317)

[1.3 Актуальность и цель работы 11](#_Toc6866318)

[2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 13](#_Toc6866319)

[2.1 Описание области применения и исходных данных приложения 13](#_Toc6866320)

[2.2 Требования к пользовательским интерфейсам 13](#_Toc6866321)

[2.3 Требования к аппаратным и программным интерфейсам 16](#_Toc6866322)

[2.4 Требования к пользователям продукта 16](#_Toc6866323)

[2.5 Функции продукта 16](#_Toc6866324)

[2.6 Сценарии работы приложения 17](#_Toc6866325)

[2.7 Ограничения 22](#_Toc6866326)

[3. ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕАЛИЗАЦИИ ПОСТАВЛЕННОЙ ЗАДАЧИ 23](#_Toc6866327)

[3.1 Язык программирования серверных скриптов PHP 23](#_Toc6866328)

[3.2 Реляционная система управления базами данных MYSQL 23](#_Toc6866329)

[3.3 WEB-сервер APACHE 23](#_Toc6866330)

[3.4 Laravel 5 24](#_Toc6866331)

[3.5 Bootstrap 4 24](#_Toc6866332)

[3.6 Архитектура «клиент-сервер» 25](#_Toc6866333)

[4. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ДИНАМИЧЕСКОГО САЙТА И БАЗЫ ДАННЫХ 26](#_Toc6866334)

[4.1 Входные и выходные данные 26](#_Toc6866335)

[4.2 Проектирование 26](#_Toc6866336)

[*4.2.1 Проектирование структуры сайта* 26](#_Toc6866337)

[*4.2.2 Концептуальное проектирование БД* 29](#_Toc6866338)

[*4.2.3 Логическое проектирование БД* 31](#_Toc6866339)

[*4.2.4 Физическое проектирование БД* 33](#_Toc6866340)

[4.3 Разработка и описание динамического сайта и базы данных 35](#_Toc6866341)

[*4.3.1 Аутентификация для сайта средствами Laravel* 36](#_Toc6866342)

[*4.3.2 Миграции для базы данных* 36](#_Toc6866343)

[*4.3.3 Классы-наполнители для базы данных* 44](#_Toc6866344)

[*4.3.4 Реализация базы данных в ORM Eloquent* 52](#_Toc6866345)

[*4.3.5. Шаблоны* 52](#_Toc6866346)

[*4.3.6 Контроллеры* 53](#_Toc6866347)

[*4.3.4 Маршруты* 57](#_Toc6866348)

[4.4 Общая структура динамического сайта 59](#_Toc6866349)

[5. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 60](#_Toc6866350)

[5.1 Аппаратные, системные и программные требования 60](#_Toc6866351)

[5.2 Тестирование веб-сайта 60](#_Toc6866352)

[5.3 Описание установки динамического сайта. 62](#_Toc6866353)

[*5.3.1 Установка, настройка и запуск Open Server* 62](#_Toc6866354)

[*5.3.2 Установка и настройка Composer и Laravel* 63](#_Toc6866355)

[*5.3.3 Установка, настройка и запуск сервера баз данных MySQL* 63](#_Toc6866356)

[*5.3.4 Настройка и запуск динамического сайта* 64](#_Toc6866357)

[5.4 Описание контрольных примеров 64](#_Toc6866358)

[ЗАКЛЮЧЕНИЕ 66](#_Toc6866359)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 67](#_Toc6866360)

[ПРИЛОЖЕНИЕ А 69](#_Toc6866361)

[ПРИЛОЖЕНИЕ Б 71](#_Toc6866362)

# 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 курсовой работе будет разработан динамический сайт для автоматизации учета методической деятельности кафедры компьютерных технологий.

# 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 Моделирование существующих бизнес-процессов

Контекстная диаграмма, описывающая функционирование учета методической деятельности кафедры компьютерных технологий приведена на рисунке 1.1.

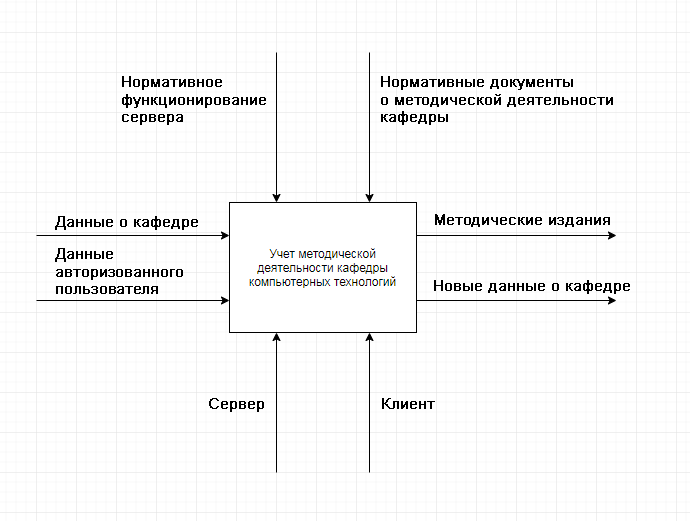


Рисунок 1.1 – Контекстная диаграмма

## 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. [3]

Ситуация, описанная выше, обосновывает актуальность методической работы. Однако, имея большое количество видов методических изданий, преподавателей и дисциплин на кафедре, учёт методических изданий становится всё сложнее и запутаннее. Данная курсовая работа будет направлена на устранение этой проблемы.

# 2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## 2.1 Описание области применения и исходных данных приложения

Данное приложение предназначено для преподавателя, ведущего учет методической деятельности кафедры компьютерных технологий.

## 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, взаимодействия с данными о кафедре, входом и выходом из аккаунта, подвал с информацией о кафедре.

Пример навигационного меню приведен на рисунке 2.1



Рисунок 2.1 – Пример навигационного меню

Пример подвала сайта приведен на рисунке 2.2

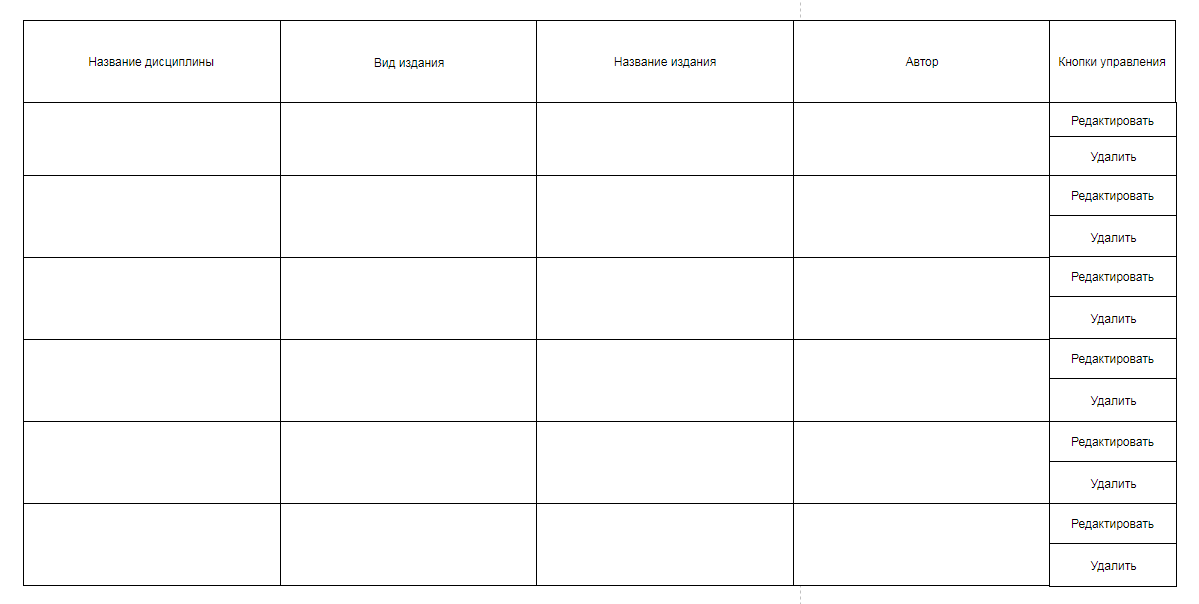
Рисунок 2.2 – Пример подвала сайта

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 кафедры компьютерных технологии, кнопки для редактирования и удаления записей (рисунок 2.4), фильтры для поиска нужных изданий по авторам, дисциплинам и году издания. Фильтр для авторов и дисциплин сделать в виде выпадающих списков, а год издания – текстового поля.

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, а также позволит легко адаптировать сайт под мобильные устройства с минимальными изменениями в интерфейсе.



Рисунок 2.6 – Общий вид веб-интерфейса системы

## 2.3 Требования к аппаратным и программным 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 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.

## 2.4 Требования к пользователям продукта

Пользователь должен обладать базовыми навыками владения персональным компьютером, а также владеть навыками работы с Open Server.

## 2.5 Функции продукта

Основные функции:

1. работа с базой данных по средствам ORM, чтобы сократить возможность ошибки при обмене данными;
2. cоздать миграции и наполнители для создания и заполнения данными о кафедре компьютерных технологий;
3. зарегистрировать пользователей с помощью наполнителей;
4. авторизация и аутентификация пользователей;
5. запрет доступа к данным неавторизованным пользователям с помощью посредника;
6. подвал с отображением информации о назначении сайта и о кафедре компьютерных технологий.
7. просмотр списка методических изданий;
8. добавление нового издания;
9. редактирование существующего издания;
10. удаление издания;
11. фильтрация списка методических изданий по авторам, дисциплинам и году издания;
12. добавление авторов, дисциплин и видов изданий;
13. редактирование существующих авторов, дисциплин и видов изданий;
14. удаление авторов, дисциплин и видов изданий, а также связанные с ними методические издания.

## 2.6 Сценарии работы приложения

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 «Просмотр методических изданий»:

1. пользователь нажимает на логотип или кнопку «Просмотреть все издания» в навигационном меню;
2. система загружает главную страницу;
3. система проверяет наличие фильтров, не находит таковых и отправляет все данные;
4. динамический сайт 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. пользователь выбирает нужную дисциплину, автора, вводит год издания и нажимает кнопку «Отфильтровать».
4. система проверяет наличие фильтров, находит нужные данные и отправляет пользователю;
5. динамический сайт выводит в секции “Content” (рисунок 2.6) информацию в виде таблицы о методических изданиях кафедры компьютерных технологий 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. пользователь вводит необходимые данные и нажимает на кнопку «Создать»;
4. 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 «Добавление информации о кафедре»:

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, текстовым полем для данных и кнопкой «Добавить»;
3. пользователь выбирает таблицу в выпадающем списке, вводит новые данные и нажимает кнопку «Создать»;
4. система добавляет данные в нужную таблицу и отправляет пользователя на страницу добавления, с сообщением об успешном дополнении к таблице.

Сценарий «Редактировать информацию о кафедре»:

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, нажимает кнопку «Далее»;
4. система переводит пользователя на страницу редактирования таблицы с выпадающим списком данных из таблицы, текстовым полем и кнопкой «Обновить»;
5. пользователь выбирает нужные данные из таблицы, заполняет текстовое поле и нажимает кнопку «Обновить»;
6. 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 кафедре»:

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, нажимает кнопку “Далее”;
4. система переводит пользователя на страницу удаления данных из таблицы с выпадающим списком данных из таблицы и кнопкой “Удалить”;
5. пользователь выбирает нужные 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.

## 2.7 Ограничения

1. динамический сайт не будет работать без заранее установленного проекта Laravel;
2. после каждого действия на сайте, страница будет полностью перезагружаться;
3. требуется соединение с web-сервером;
4. требуется MySQL версии 5.7 или выше.

# 3. ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕАЛИЗАЦИИ ПОСТАВЛЕННОЙ ЗАДАЧИ

## 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 [4].

## 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 [5].

## 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 [6].

## 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 [7].

## 3.5 Bootstrap 4

Bootstrap – это WEB-фреймворк, содержащий готовые CSS, HTML и JavaScript компоненты.

Преимущества фреймворка Bootstrap:

1. Кроссбраузерность и адаптивность сайта. Все элементы фреймворка адаптивны под все устройства и корректно отображаются во всех современных браузерах.
2. Легкость в использовании. Даже человек, имеющий базовые знания о HTML и CSS, может свободно создавать web-страницы с использованием фреймворка.
3. Простота в обучении. У Bootstrap очень хорошая документация с большим количеством примеров готового кода [10].

## 3.6 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 [8]. Пример модели взаимодействия архитектуры «клиент-сервер» изображен на рисунке 3.1.



Рисунок 3.1 – Модель взаимодействия клиент-сервер

# 4. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ДИНАМИЧЕСКОГО САЙТА И БАЗЫ ДАННЫХ

## 4.1 Входные и выходные данные

Входные данные динамического сайта:

1. Список дисциплин кафедры компьютерных технологий;
2. Список преподавателей кафедры компьютерных технологий;
3. Список видов методических изданий;
4. Формат бумаги и тип обложки;
5. Количество страниц;
6. Тираж;
7. Год и месяц выпуска;
8. Контактный телефон преподавателя.

Результатом работы приложения является список методических изданий кафедры компьютерных технологий.

## 4.2 Проектирование

### *4.2.1 Проектирование структуры сайта*

Динамический сайт – сайт, состоящий из динамичных страниц – шаблонов, контента, скриптов и прочего. Отличие динамического сайта от статического в том, что в статических сайтах готовые веб-страниц находятся на сервере и ждут своей очереди для отправки в браузер пользователя. При этом, если страницы буду иметь любое различие, это все равно будут отдельные страницы.

В динамических сайтах формирования страниц происходит на сервере при каждом запросе пользователя в зависимости от запрашиваемой информации.

Основные элементы, которые будут использоваться при построении динамического сайта:

1. HTML;
2. CSS;
3. PHP;
4. Bootstrap 4;
5. Фреймворк Laravel;
6. MySQL.

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

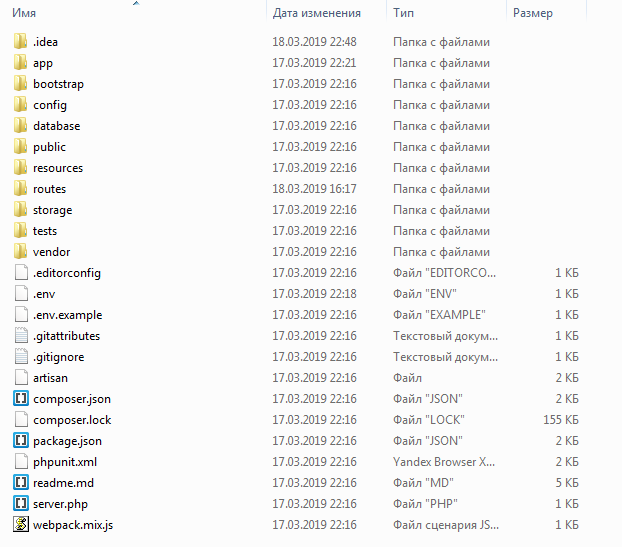


Рисунок 4.1 – Структура проекта

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 для вашего 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 кафедры компьютерных технологий;
2. Дисциплина – информация об имеющихся дисциплинах на кафедре компьютерных технологий;
3. Преподаватели – информация о преподавателях кафедры компьютерных технологий;
4. Вид издания – информация о видах методических изданий;
5. Обложка – информация о типах обложек, которые будут использованы в методических изданиях;
6. Формат бумаги – информация о формате бумаги, которую будут использовать для методических изданий;
7. Месяц издания – информация о месяцах, в которых будут изданы методические издания.

Концептуальное представление базы данных приведено на рисунке 4.2.

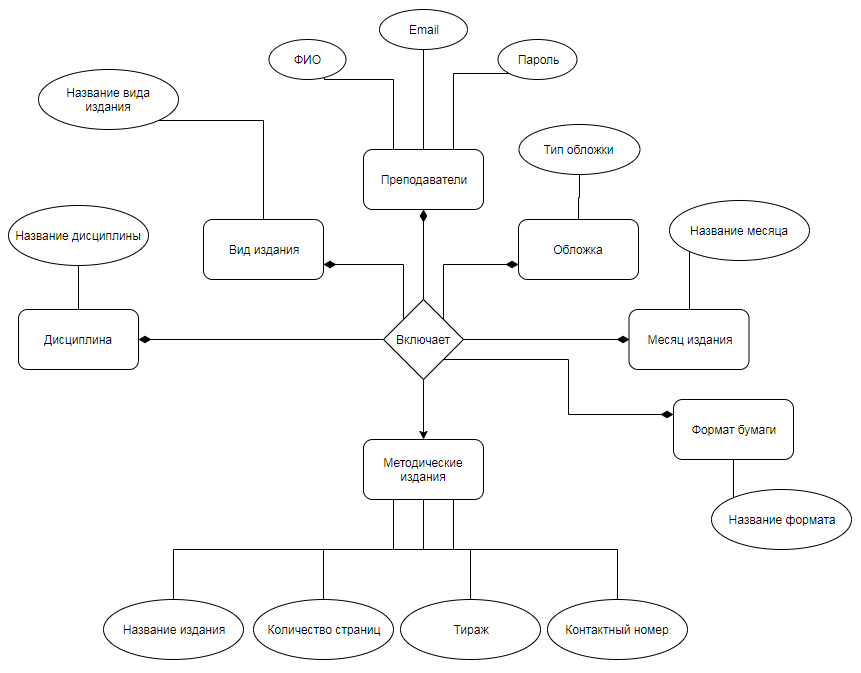


Рисунок 4.2 – Концептуальная модель базы данных

### *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. [9]

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

Логическое представление базы данных приведено на рисунке 4.3.

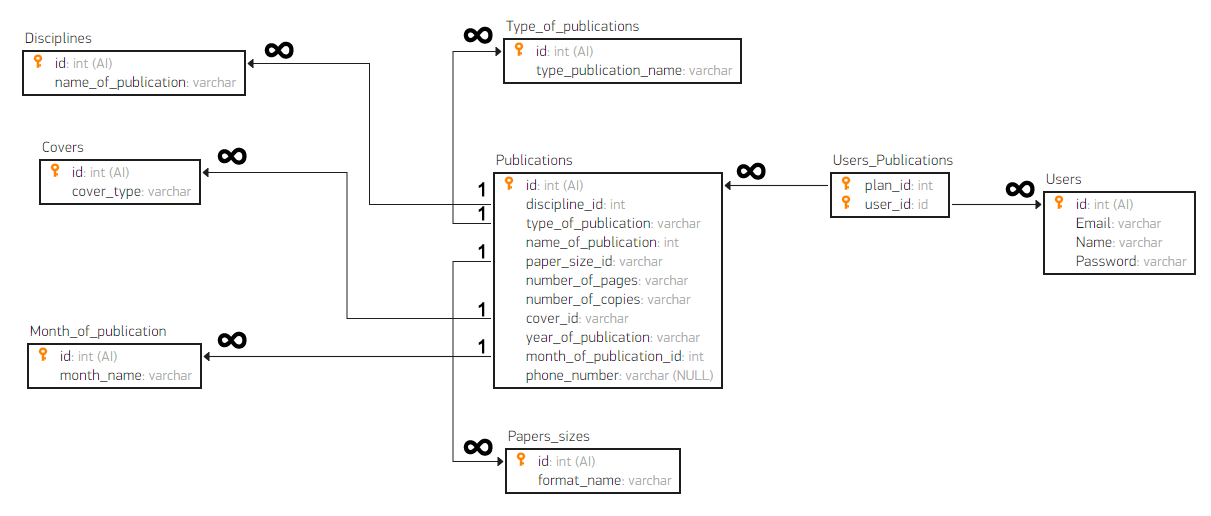


Рисунок 4.3 – Логическая модель базы данных

### *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»

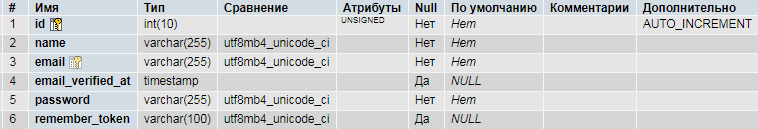


Рисунок 4.5 – Физическая модель таблицы «Users»



Рисунок 4.6 – Физическая модель таблицы «Users\_Publications»



Рисунок 4.7 – Физическая модель таблицы «Type\_of\_publication»



Рисунок 4.8 – Физическая модель таблицы «Papers\_sizes»



Рисунок 4.9 – Физическая модель таблицы «Month\_of\_publications»



Рисунок 4.10 – Физическая модель таблицы «Disciplines»



Рисунок 4.11 – Физическая модель таблицы «Covers»

## 4.3 Разработка и описание динамического сайта и базы данных

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

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

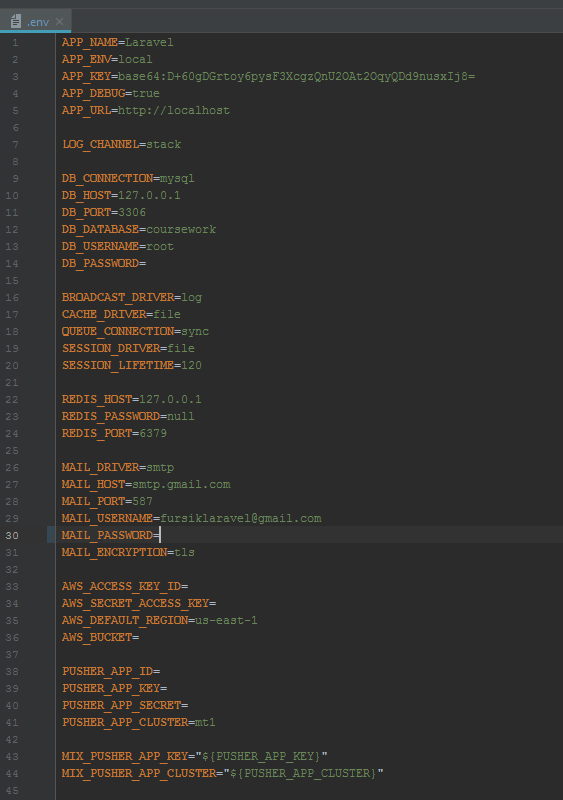


Рисунок 4.12 – Cтруктура файла “.env”

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



Рисунок 4.13 – Содержимое файла “database.php ”

### *4.3.1 Аутентификация для сайта 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. [12].

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)

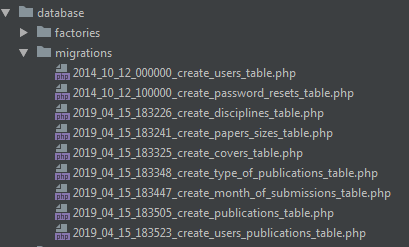


Рисунок 4.14 – Список миграций

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 создавать таблицы, которые в дальнейших миграциях будут использоваться внешними ключами, раньше таблиц, в которых они будут использоваться.

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.

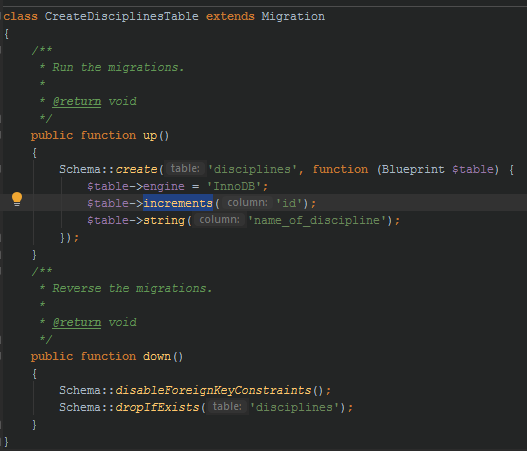
’

Рисунок 4.16 – Миграция дисциплин

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

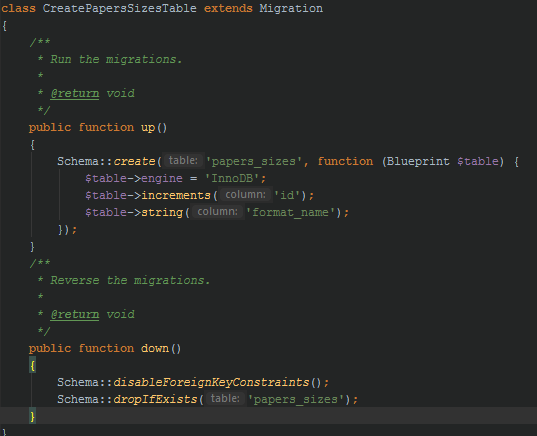


Рисунок 4.17 – Миграция форматов бумаги

1. CreateСoversTable – миграция обложек (рисунок 4.18);



Рисунок 4.18 – Миграция обложки издания

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

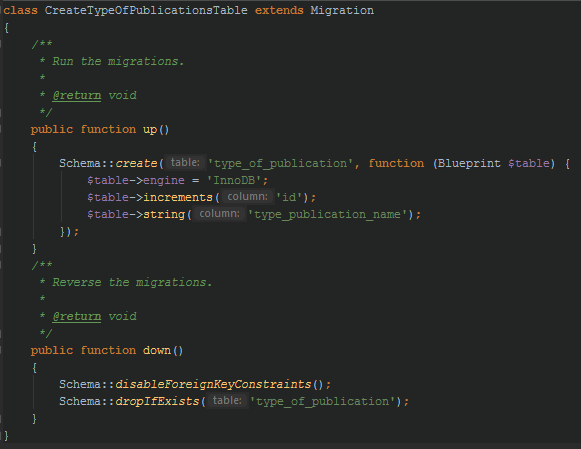


Рисунок 4.19 – Миграция видов изданий

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

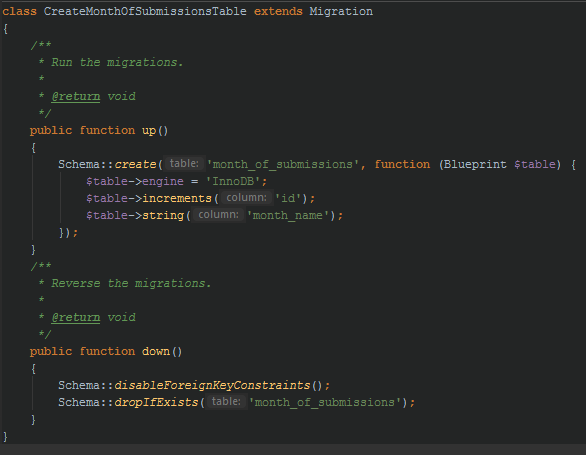


Рисунок 4.20 – Миграция месяцев издания

1. CreatePublicationTable – миграция методических изданий (рисунок 4.21);

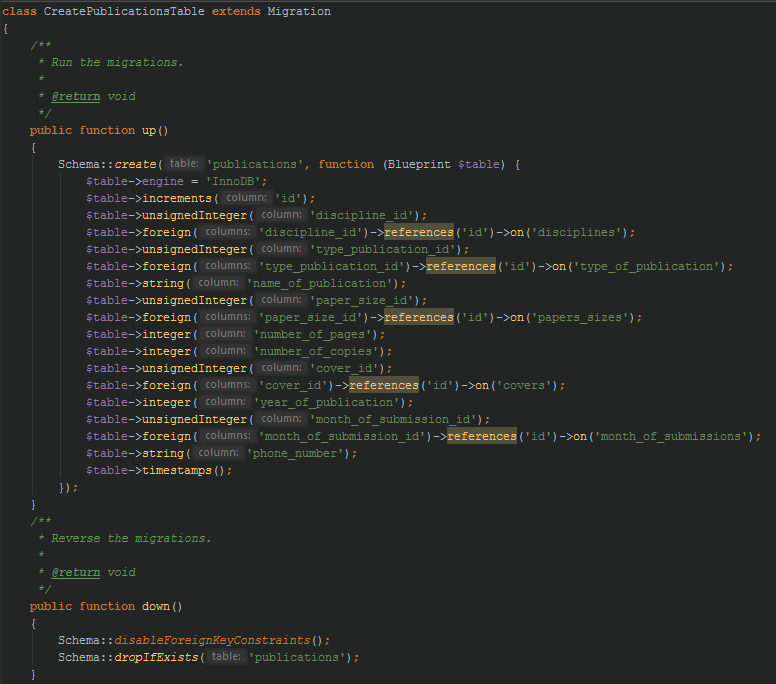


Рисунок 4.21 – Миграция методических изданий

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 |
| 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(11) | Количество страниц в методическом издании |
| number\_of\_copies | int(11) | Тираж методического издания |
| cover\_id | int(10) | Id обложки (переплёта), в котором возможно выпустить издание |
| year\_of\_publication | int(11) | Год издания |
| month\_of\_submission | int(10) | Id месяца, в котором возможно выпустить издания |
| phone\_number | varchar(255) | Дополнительная информация (контактный телефон) |

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

Из-за связи “многие-ко-многим” мы также создаем миграцию “Users\_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.

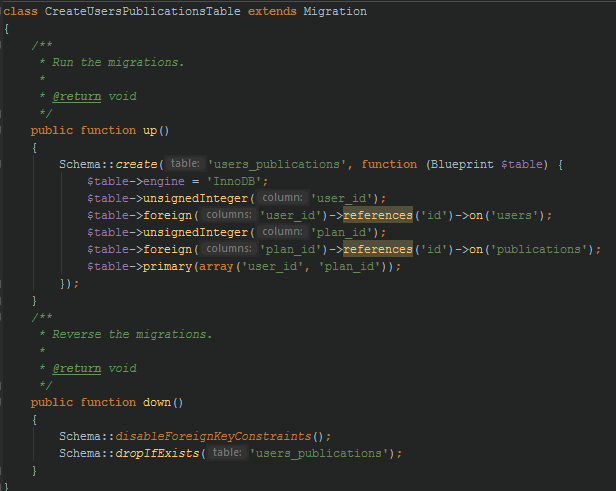


Рисунок 4.22 – Миграция таблицы “Users\_Publications”

### *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” [13].

Данные о кафедре 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 динамическим сайтом:

1. CoverSeeder – класс-наполнитель для таблицы “Covers” (рисунок 4.23);

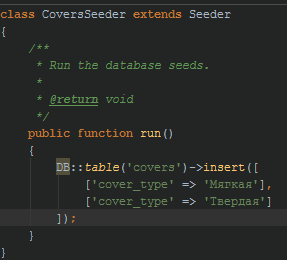


Рисунок 4.23 – Класс-наполнитель для таблицы “Covers”

1. PaperSizesSeeder – класс-наполнитель для таблицы “Papers\_sizes” (рисунок 4.24);

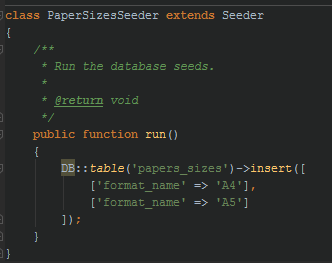


Рисунок 4.24 – Класс-наполнитель для таблицы “Papers\_sizes”

1. MonthSeeder – класс-наполнитель для таблицы “Month\_Of\_Submissions” (рисунок 4.25);

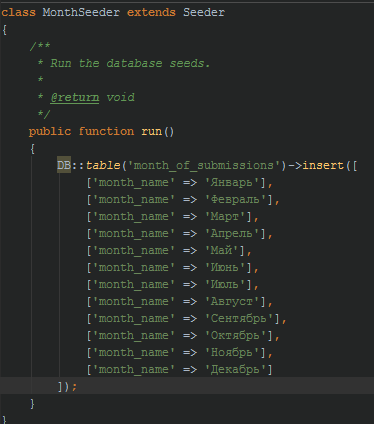


Рисунок 4.25 – Класс-наполнитель для таблицы “Month\_Of\_Submissions”

1. TypeOfPublicationSeeder – класс-наполнитель для таблицы “Type\_of\_publication” (рисунок 4.26);

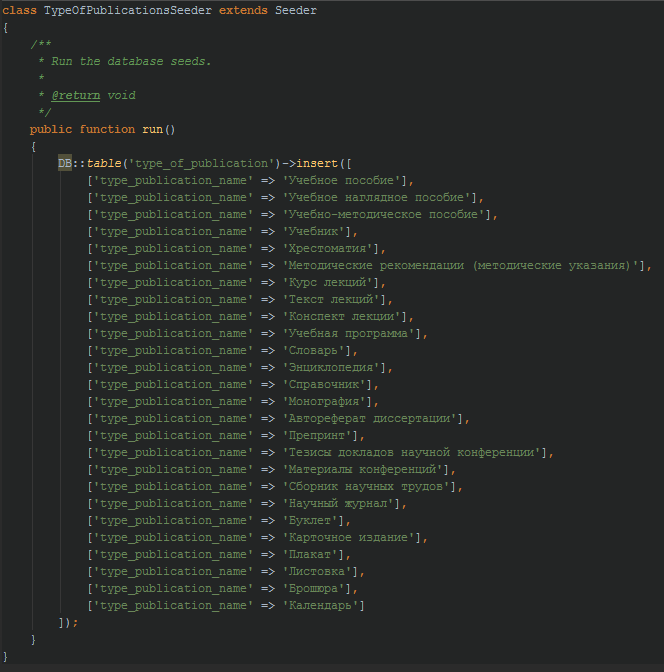


Рисунок 4.26 – Класс-наполнитель для таблицы “Type\_of\_publication”

1. UsersSeeder – класс-наполнитель для таблицы “Users” (рисунок 4.27);

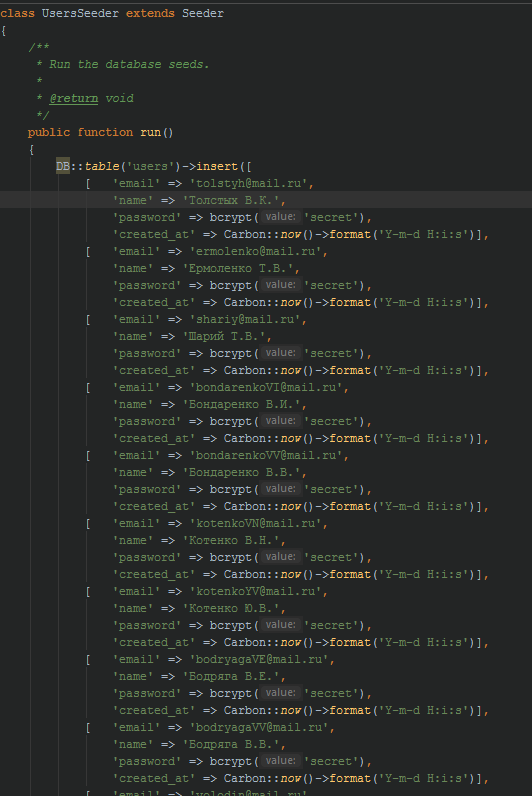


Рисунок 4.27 – Класс-наполнитель для таблицы “Users”

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

1. DisciplinesSeeder – класс-наполнитель для таблицы “Disciplines” (рисунок 4.28);



Рисунок 4.28 – Класс-наполнитель для таблицы “Disciplines”

1. PublicationsSeeder – класс-наполнитель для таблицы “Publications” (рисунок 4.29);

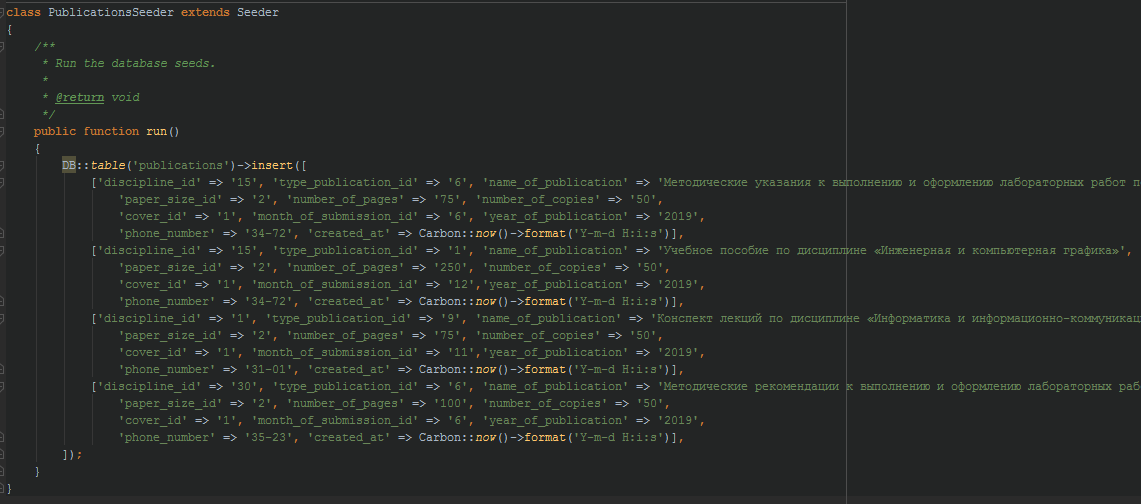


Рисунок 4.29 – Класс-наполнитель для таблицы “Publications”

1. UsersPublicationsSeeder – класс-наполнитель для таблицы “Users\_Publications” (рисунок 4.30);

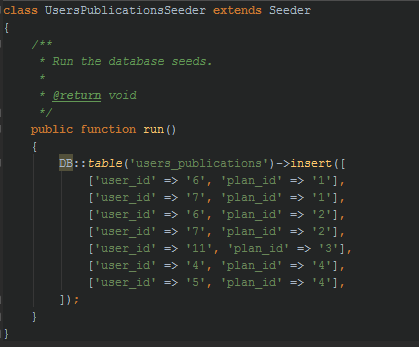


Рисунок 4.30 – Класс-наполнитель для таблицы “Users\_Publications”

1. DatabaseSeeder – класс-наполнитель, который вызывает все остальные классы-наполнители (рисунок 4.30);

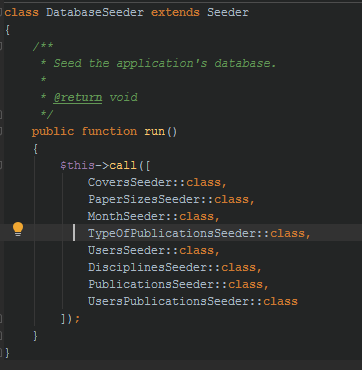


Рисунок 4.31 – Класс-наполнитель “DatabaseSeeder”

В классе-наполнителе “DatabaseSeeder” в методе “call” вызываем классы наполнители в определенном порядке, чтобы не нарушить добавление внешних ключей.

### *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. [11]

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 “$table”. Также для нашего проекта понадобится массовое заполнение, поэтому определим свойство “$fillable” и укажем нужные поля. Пример модели приведен на рисунке 4.31.

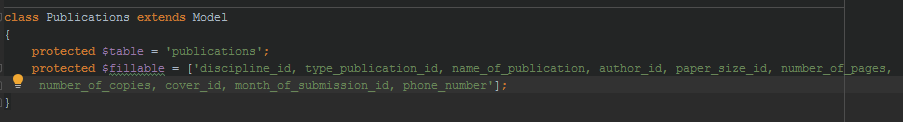


Рисунок 4.31 – Модель методических изданий

### *4.3.5. Шаблоны*

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

В проекте будем использовать шаблонизатор Blade – простой, но мощный шаблонизатор, поставляемый с Laravel. Для того, чтобы создать шаблон blade, создадим файл с расширением “.blade.php”. Структура шаблонов приведена на рисунке 4.32.

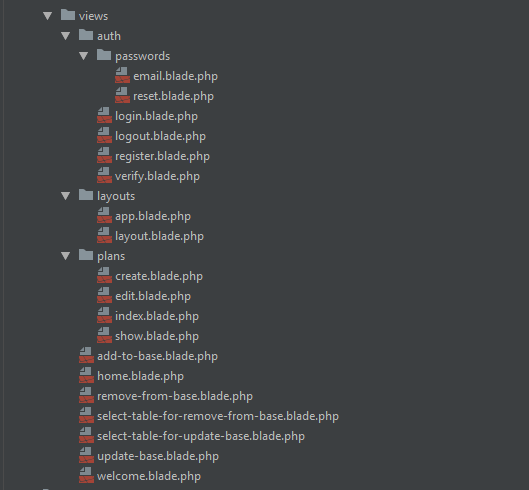


Рисунок 4.32 – Структура шаблонов динамического сайта

### *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”.

Однако, 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 кафедре необходимо создать еще один контроллер “AdditionToBaseController.php”. Листинг кода “AdditionToBaseController.php” находится в листинге Б.2.

Таблица 4.6. Описание методов в ресурсном 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 |
| remove | Возвращает шаблон страницы для выбора таблицы, из которой будут удалены данные |
| select\_table\_for\_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 таблицы, в которой будут редактироваться данные |
| select\_table\_for\_update | Возвращает шаблон страницы для выбора данных и ввода вместо них новых данных |
| update | Сохранение изменений внесенных в таблицу |

За авторизацию пользователя отвечает автоматически сгенерированный фреймворком Laravel контроллер LoginController.php и посредник Authenticate.php. Также, стоит упомянуть, что через посредник Authenticate.php проходят все запросы. Таким образом, неавторизованные пользователи не смогут получить доступ к страницам сайта.

Связь контроллеров и шаблонов показана на рисунке 4.33.

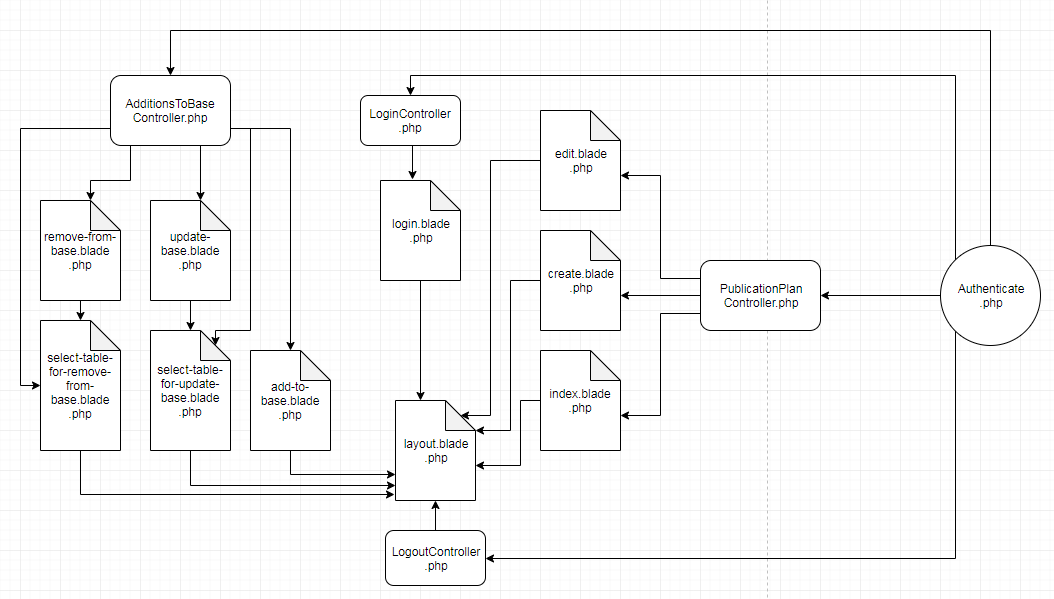


Рисунок 4.33 – Связь скриптов динамического сайта

### *4.3.4 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 вашего web-интерфейса. Определение маршрутов приведено на рисунке 4.34.

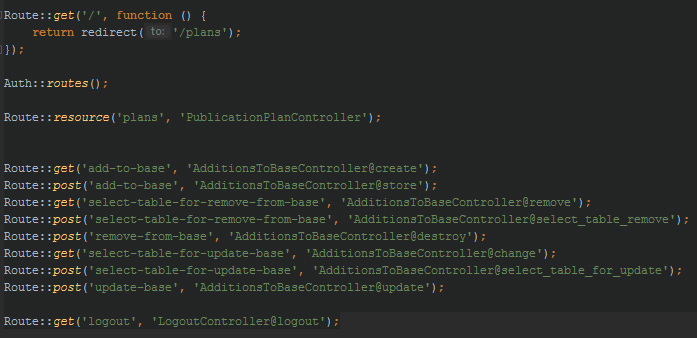


Рисунок 4.34 – Содержимое файла web.php

## 4.4 Общая структура динамического сайта

Общая структура динамического сайта приведена на рисунке 4.35.

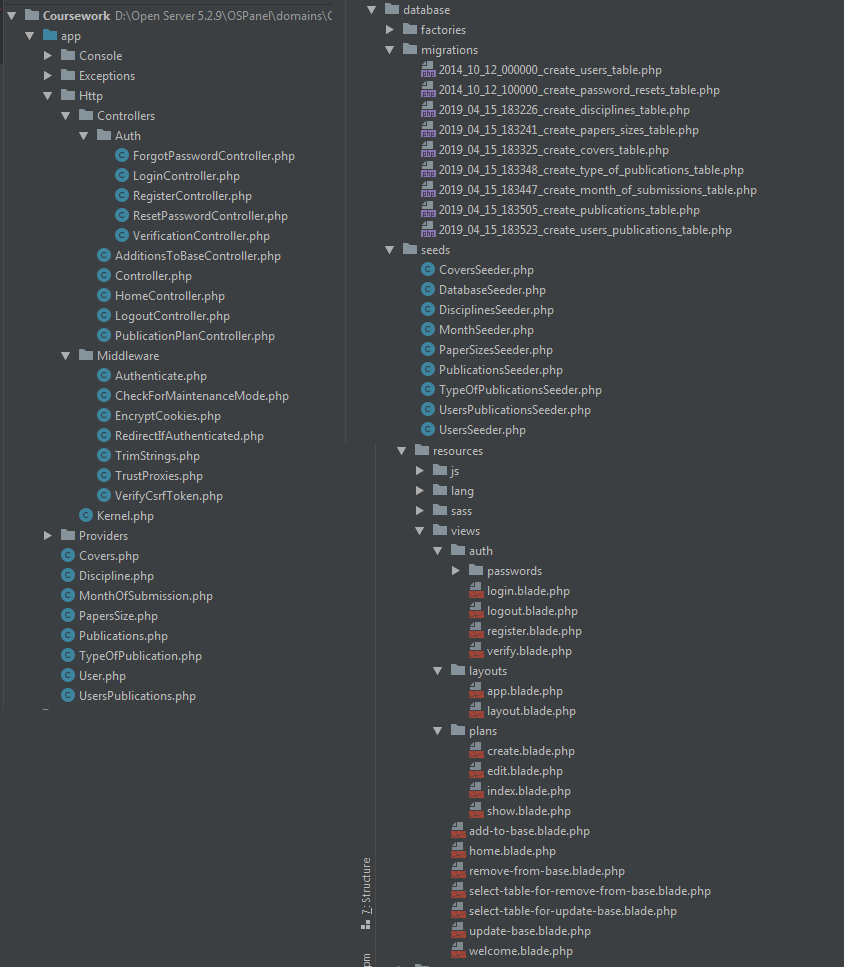


Рисунок 4.35 – Общая структура динамического сайта

# 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 веб-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.

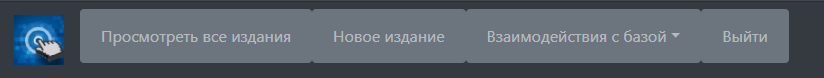


Рисунок 5.1 – Навигационное меню динамического сайта

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.

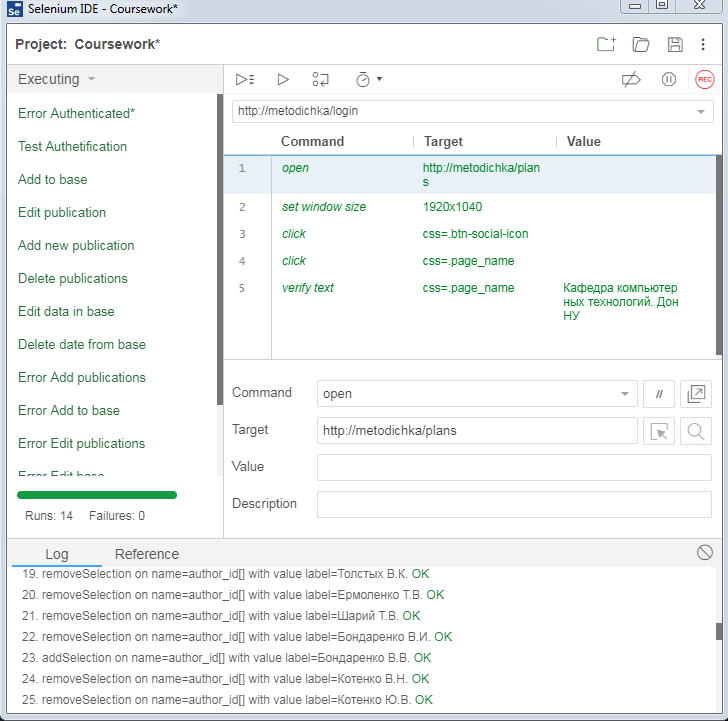


Рисунок 5.2 – Результаты тестирования

## 5.3 Описание установки динамического сайта.

### *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:

* HTTP ­­– Apache-PHP-7.2;
* PHP – PHP-7.2;
* 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 динамического сайта*

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 трем категориям «Год выпуска», «Дисциплины», «Авторы» 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.

В подвале сайта расположена ссылка на информацию о кафедре на сайте ДОННУ и ссылка на группу Вконтакте.

# ЗАКЛЮЧЕНИЕ

В результате, разработан динамический сайт для учета методической деятельности кафедры компьютерных технологий.

Разработанный динамический сайт позволяет выполнять учет методических изданий кафедры компьютерных технологий, а именно аутентифицировать пользователя, добавлять, редактировать и удалять методические издания, фильтровать по категориям, добавлять, редактировать и удалять данные о кафедре компьютерных технологий.

Динамический сайт для учета методической деятельности кафедры компьютерных технологий может использоваться преподавателем, который ведет учет методических изданий кафедры компьютерных технологий.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Котеров Д. В. PHP 5. 2 изд. – БХВ-Петербург, 2012. – 1062 с.
2. Коробко А. И. Методическая работа в вузе: цели, задачи, пути и формы ее ведения //Вестник Московского государственного лингвистического университета. – 2012. – №. 16 (649).
3. Актуальность методической работы в общеобразовательном учреждении // Научный журнал «Молодой ученый»

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

(дата обращения: 10.04.2019)

1. Что такое PHP? // Официальный сайт PHP

URL: https://www.php.net/manual/ru/intro-whatis.php

(дата обращения: 11.04.2019)

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

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

(дата обращения: 09.04.2019)

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

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

(дата обращения: 09.04.2019)

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: 12.04.2019)

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: 12.04.2019)

1. Карпова Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. ­– СПб: Питер, 2001. – 304 с.
2. Фреймворк Bootstrap – что это такое? // Блог о разработке и продвижении сайта

URL: https://zyubin.ru/frameworks/bootstrap/frejmvork-bootstrap-chto-eto-takoe.html (дата обращения: 13.04.2019)

1. Eloquent ORM // Laravel по-русски

URL: https://laravel.ru/docs/v5/eloquent (дата обращения: 14.04.2019)

1. Миграции // Laravel по-русски

URL: https://laravel.ru/posts/273 (дата обращения: 14.04.2019)

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

URL: http://laravel.su/docs/5.2/seeding (дата обращения: 15.04.2019)

# ПРИЛОЖЕНИЕ А

**Экранные формы**

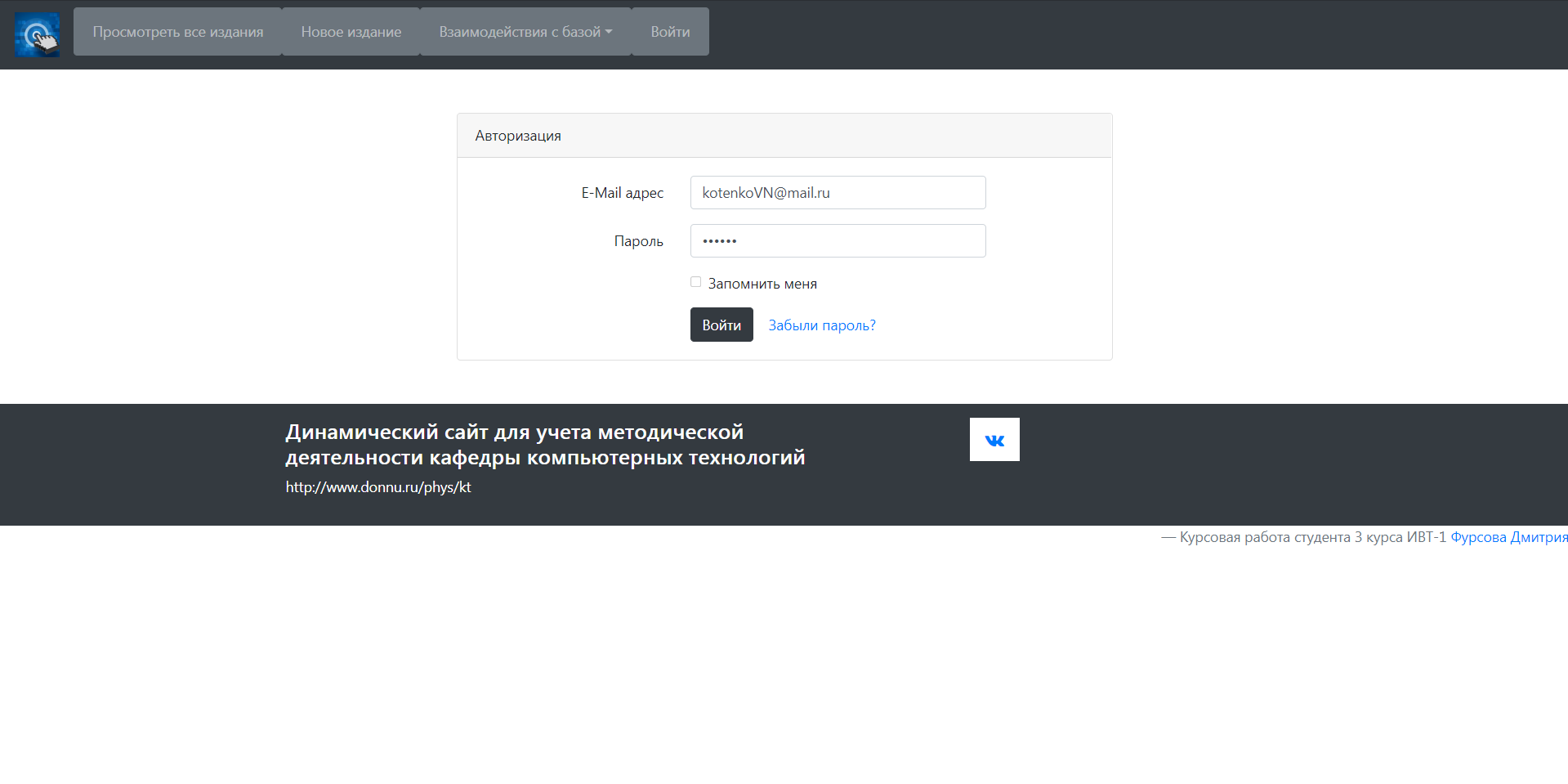


Рисунок А.1 – Страница с формой для добавления нового материала преподавателя

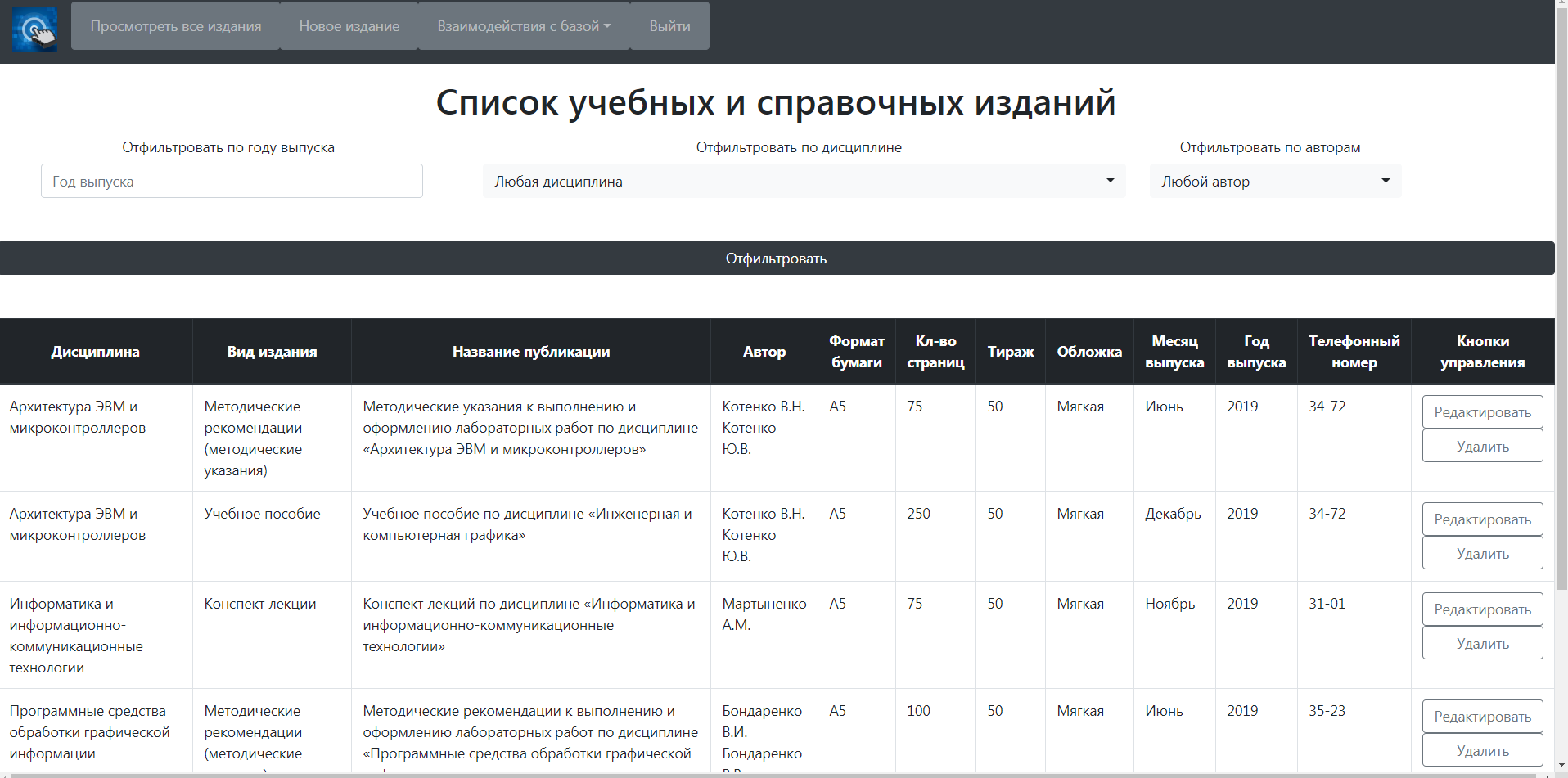


Рисунок А.2 – Страница с формой для добавления нового материала преподавателя

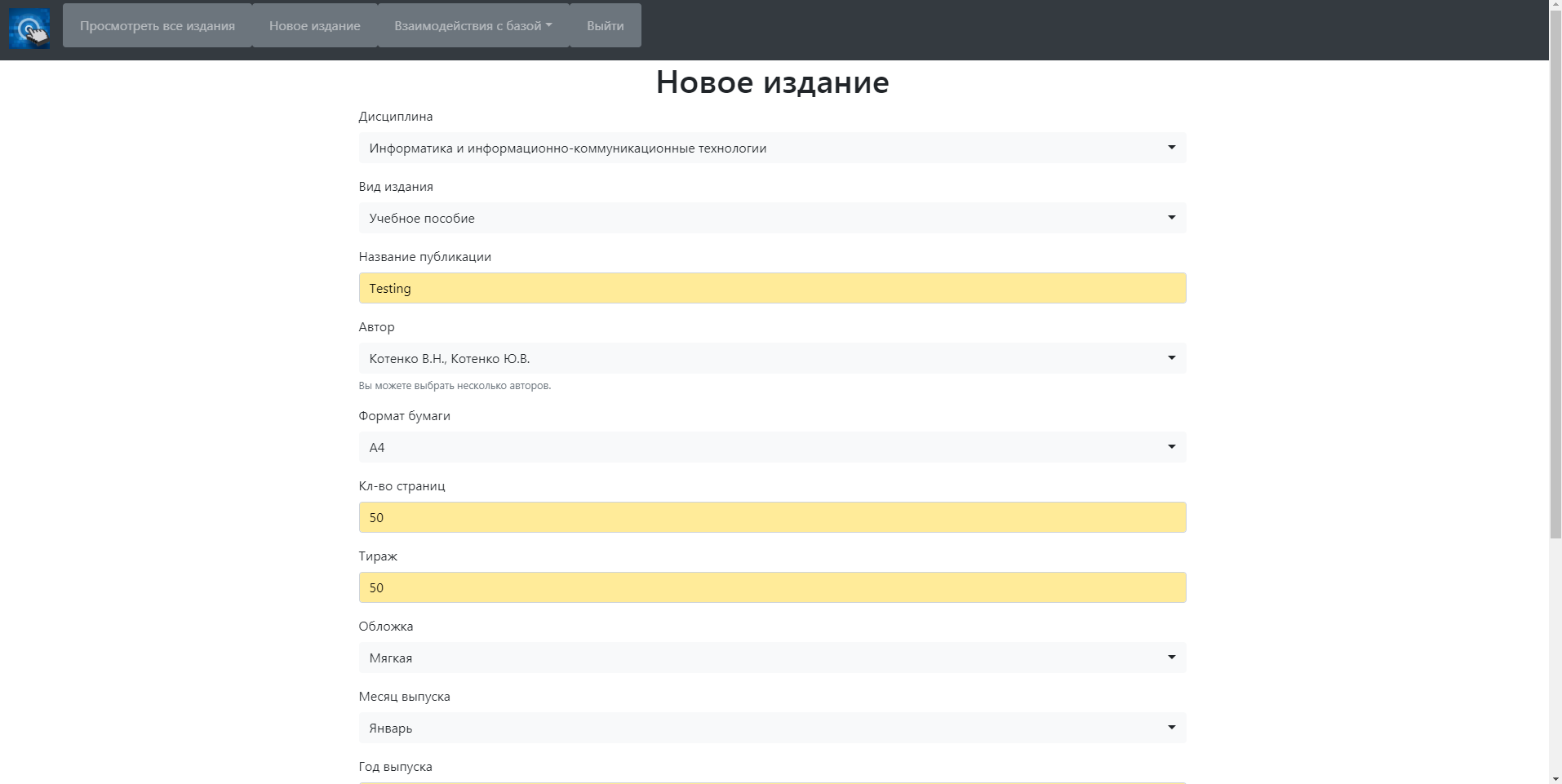


Рисунок А.3 – Страница с формой для добавления нового материала преподавателя

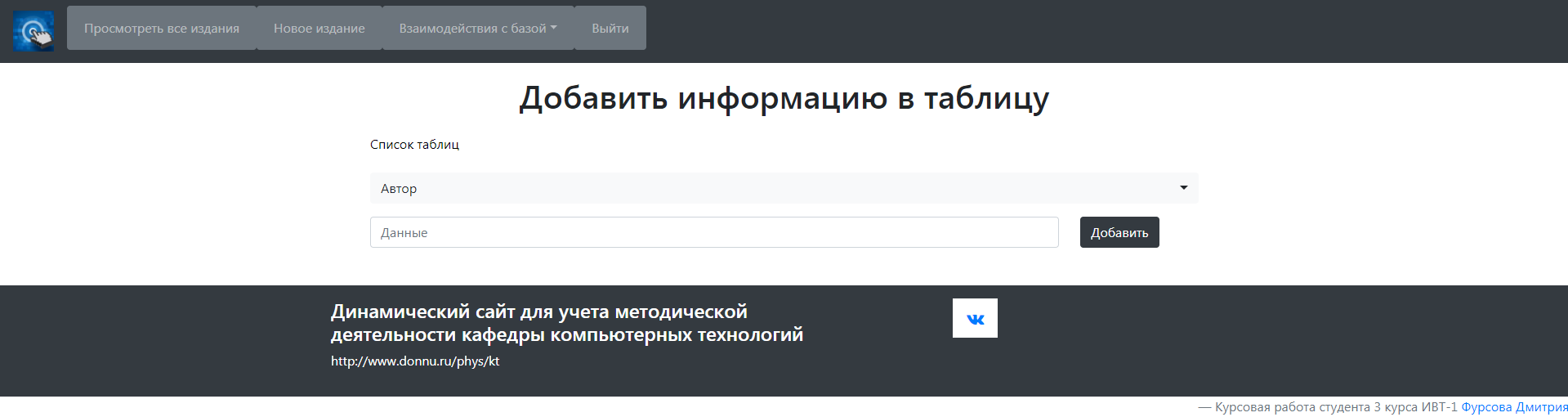


Рисунок А.4 – Страница с формой для добавления нового материала преподавателя

# ПРИЛОЖЕНИЕ Б

**Фрагменты листинга**

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

public function \_\_construct()

{

$this->middleware('auth');

}

public function index(Request $request)

{

$disciplines\_table = Discipline::all();

$autors\_table = User::all();

$select\_author = $request->input('select\_author');

$users\_in\_plans = DB::table('users\_publications')

->join('users', 'users\_publications.user\_id', '=', 'users.id')

->join('publications', 'publications.id', '=', 'users\_publications.plan\_id')

->select('users.id', 'users.name', 'users\_publications.plan\_id')

->get();

if ($select\_author != null && $select\_author != -1)

{

// get all the plans with foreign key and filter author

$users = Db::table('users')

->join('users\_publications', 'users.id', '=', 'users\_publications.user\_id')

->join('publications', 'publications.id', '=', 'users\_publications.plan\_id')

->where('user\_id', '=', $request->input('select\_author'))

->join('disciplines', 'publications.discipline\_id', '=', 'disciplines.id')

->join('type\_of\_publication', 'publications.type\_publication\_id', '=', 'type\_of\_publication.id')

->join('papers\_sizes', 'publications.paper\_size\_id', '=', 'papers\_sizes.id')

->join('covers', 'publications.cover\_id', '=', 'covers.id')

->join('month\_of\_submissions', 'publications.month\_of\_submission\_id', '=', 'month\_of\_submissions.id')

->select('publications.id', 'users\_publications.user\_id', 'publications.discipline\_id', 'disciplines.name\_of\_discipline',

'type\_of\_publication.type\_publication\_name', 'publications.name\_of\_publication',

'papers\_sizes.format\_name' , 'number\_of\_pages', 'number\_of\_copies','covers.cover\_type', 'publications.year\_of\_publication',

'month\_of\_submissions.month\_name', 'phone\_number')

->get();

}

else

{

// get all the plans with foreign key

$users = DB::table('publications')

->join('disciplines', 'publications.discipline\_id', '=', 'disciplines.id')

->join('type\_of\_publication', 'publications.type\_publication\_id', '=', 'type\_of\_publication.id')

->join('papers\_sizes', 'publications.paper\_size\_id', '=', 'papers\_sizes.id')

->join('covers', 'publications.cover\_id', '=', 'covers.id')

->join('month\_of\_submissions', 'publications.month\_of\_submission\_id', '=', 'month\_of\_submissions.id')

->select('publications.id', 'publications.discipline\_id', 'disciplines.name\_of\_discipline', 'type\_of\_publication.type\_publication\_name', 'publications.name\_of\_publication',

'papers\_sizes.format\_name' , 'number\_of\_pages', 'number\_of\_copies','covers.cover\_type', 'publications.year\_of\_publication',

'month\_of\_submissions.month\_name', 'phone\_number')

->get();

}

// load the view and pass the plans

return view('plans.index')->with([

'plans' => $users,

'users' => $users\_in\_plans,

'disciplines' => $disciplines\_table,

'autors' => $autors\_table,

'select\_year' => $request->input('select\_year'),

'select\_discipline' => $request->input('select\_discipline'),

'select\_author' => $select\_author,

]);

}

public function create()

{

$papers\_size = PapersSize::all();

$months = MonthOfSubmission::all();

$cover = Covers::all();

$type\_publication = TypeOfPublication::all();

$disciplines = Discipline::all();

$users = User::all();

// load the create form (app/views/plans/create.blade.php)

return view('plans.create',[

'papers\_size' => $papers\_size,

'months' => $months,

'cover' => $cover,

'type\_publication' => $type\_publication,

'disciplines' => $disciplines,

'users' => $users

]);

}

public function store(Request $request)

{

// validate

$request->validate([

'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',

]);

// store

$plan = new Publications();

$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->save();

$newid = Publications::latest()->first()->id;

$array\_users = $request->input('author\_id.\*');

if(count($array\_users) > 0) {

for ($i=0; $i<count($array\_users); $i++) {

$plan\_user = new UsersPublications();

$plan\_user->plan\_id = $newid;

$plan\_user->user\_id = $array\_users[$i];

$plan\_user->save();

}

}

// redirect

Session::flash('message', 'Новое издание создано!');

return redirect('/plans');

}

public function show($id)

{

//

}

public function edit($id)

{

// get the plan

$plan = Publications::find($id); //selected plan

$papers\_size = PapersSize::all();

$months = MonthOfSubmission::all();

$cover = Covers::all();

$type\_publication = TypeOfPublication::all();

$disciplines = Discipline::all();

$users = User::all();

$plan\_users = DB::table('users\_publications')->where('plan\_id', '=', $id)->get();

$selected\_users = collect();

$unselected\_users = collect();

for ($i = 0; $i < count($users); $i++)

{

for ($j = 0; $j < count($plan\_users); $j++)

{

if($users[$i]->id == $plan\_users[$j]->user\_id)

{

$selected\_users->push($users->slice($i,1));

}

}

}

for ($i = 0; $i < count($users); $i++)

{

for ($j = 0, $selected = false; $j < count($plan\_users); $j++)

{

if($users[$i]->id == $plan\_users[$j]->user\_id)

{

$selected = true;

}

}

if($selected == false)

{

$unselected\_users->push($users->slice($i,1));

}

}

return view('plans.edit',[

'plan' => $plan,

'papers\_size' => $papers\_size,

'months' => $months,

'cover' => $cover,

'type\_publication' => $type\_publication,

'disciplines' => $disciplines,

'users' => $users,

'plan\_users' => $plan\_users,

'selected\_users' => $selected\_users,

'unselected\_users' => $unselected\_users,

]);

}

public function update(Request $request, $id)

{

// validate

// read more on validation at http://laravel.com/docs/validation

$request->validate([

'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',

]);

// store

DB::table('users\_publications')->where('plan\_id', '=', $id)->delete();

$array\_users = $request->input('author\_id.\*');

if(count($array\_users) > 0) {

for ($i=0; $i<count($array\_users); $i++) {

$plan\_user = new UsersPublications();

$plan\_user->plan\_id = $id;

$plan\_user->user\_id = $array\_users[$i];

$plan\_user->save();

}

}

$plan = Publications::find($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->save();

// redirect

Session::flash('message', 'Издание было изменено!');

return redirect('/plans');

}

public function destroy($id)

{

// delete foreign key

$deletedRows = UsersPublications::where('plan\_id', '=', $id)->delete();

// delete

$plan = Publications::find($id);

$plan->delete();

// redirect

Session::flash('message', 'Издание было удалено!');

return redirect('/plans');

}

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

public function \_\_construct()

{

$this->middleware('auth');

}

public function create()

{

return view("add-to-base");

}

public function store(Request $request)

{

//what table we need

$select\_table = $request->input('table');

switch ($select\_table) {

case 'discipline':

$discipline = new Discipline();

$discipline->name\_of\_discipline = $request->input('data');

$discipline->save();

break;

case 'type\_publication':

$type = new TypeOfPublication();

$type->type\_of\_publication = $request->input('data');

$type->save();

break;

case 'name':

$user = new User();

$user->name = $request->input('data');

$user->save();

break;

default:

Session::flash('message', 'Что-то пошло не так. Данные не сохранены!');

return redirect('/add-to-base');

break;

}

// redirect

Session::flash('message', 'Новые данные добавлены!');

return redirect('/add-to-base');

}

public function remove()

{

return view("select-table-for-remove-from-base");

}

public function select\_table\_remove(Request $request)

{

$select\_table = $request->input('table');

$collection = array();

switch ($select\_table) {

case 'discipline':

$table = Discipline::all();

$number\_table = 1;

foreach ($table as $value) {

$collection[$value->id] = $value->name\_of\_discipline;

}

break;

case 'type\_publication':

$table = TypeOfPublication::all();

$number\_table = 2;

foreach ($table as $value) {

$collection[$value->id] = $value->type\_publication\_name;

}

break;

case 'name':

$table = User::all();

$number\_table = 3;

foreach ($table as $value) {

$collection[$value->id] = $value->name;

}

break;

default:

Session::flash('message', 'Что-то пошло не так. Таблицы не существует!');

return redirect('/select-table-for-remove-from-base');

break;

}

return view("remove-from-base")->with([

'values' => $collection,

'select\_table' => $number\_table,

]);

}

public function destroy(Request $request)

{

$elements = $request->input('elements.\*');

if (count($elements) > 0) {

for ($i = 0; $i < count($elements); $i++) {

$element = substr($elements[$i], 0, 1);

switch ($element) {

case 1:

$id = substr($elements[$i], 1);

$plans = Publications::all()->where('discipline\_id', '=', $id);

foreach ($plans as $key => $value) {

$deletedRows = Users\_Publications::where('plan\_id', '=', $value->id)->delete();

$value->delete();

}

$discipline = Discipline::find($id);

$name\_element = $discipline->name\_of\_discipline;

$discipline->delete();

break;

case 2:

$id = substr($elements[$i], 1);

$plans = Publications::all()->where('type\_publication\_id', '=', $id);

foreach ($plans as $key => $value) {

$deletedRows = Users\_Publications::where('plan\_id', '=', $value->id)->delete();

$value->delete();

}

$type = TypeOfPublication::find($id);

$name\_element = $type->type\_publication\_name;

$type->delete();

break;

case 3:

$id = substr($elements[$i], 1);

$user\_publications = Users\_Publications::all()->where('user\_id', '=', $id);

$deletedRows = Users\_Publications::where('user\_id', '=', $id)->delete();

foreach ($user\_publications as $key => $value) {

$count\_publication = Users\_Publications::all()->where('plan\_id', '=', $value->plan\_id)->count();

if ($count\_publication == 0) {

$deletedRows = Publications::where('id', '=', $value->plan\_id)->delete();

}

}

$autor = User::find($id);

$name\_element = $autor->name;

$autor->delete();

break;

}

}

Session::flash('message', 'Данные об ' . $name\_element . ' и все связанные с ним(-и) данные были удалены!');

return redirect('/select-table-for-remove-from-base');

}

}

public function change()

{

return view("select-table-for-update-base");

}

public function select\_table\_for\_update(Request $request)

{

$select\_table = $request->input('table');

$collection = array();

switch ($select\_table) {

case 'discipline':

$table = Discipline::all();

$number\_table = 1;

foreach ($table as $value) {

$collection[$value->id] = $value->name\_of\_discipline;

}

break;

case 'type\_publication':

$table = TypeOfPublication::all();

$number\_table = 2;

foreach ($table as $value) {

$collection[$value->id] = $value->type\_publication\_name;

}

break;

case 'name':

$table = User::all();

$number\_table = 3;

foreach ($table as $value) {

$collection[$value->id] = $value->name;

}

break;

default:

Session::flash('message', 'Что-то пошло не так. Таблицы не существует!');

return redirect('/select-table-for-update-base');

break;

}

return view("update-base")->with([

'values' => $collection,

'select\_table' => $number\_table,

]);

}

public function update(Request $request)

{

$id\_table = substr($request->input('element'), 0, 1);

$data = $request->input('data');

$id = substr($request->input('element'), 1);

switch ($id\_table) {

case 1:

$discipline = Discipline::find($id);

$old\_name = $discipline->name\_of\_discipline;

$discipline->name\_of\_discipline = $data;

$new\_name = $discipline->name\_of\_discipline;

$discipline->save();

break;

case 2:

$type = TypeOfPublication::find($id);

$old\_name = $type->type\_publication\_name;

$type->type\_publication\_name = $data;

$new\_name = $type->type\_publication\_name;

$type->save();

break;

case 3:

$autor = User::find($id);

$old\_name = $autor->name;

$autor->name = $data;

$new\_name = $autor->name;

$autor->save();

break;

}

Session::flash('message', $old\_name . ' изменён на ' . $new\_name);

return redirect('/select-table-for-update-base');

}