**Министерство образования Иркутской области**

Государственное бюджетное профессиональное образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
| **ДП.09.02.03.22.191.01.ПЗ** | УТВЕРЖДАЮ  Зам. директора по УР, к.т.н.  \_\_\_\_\_\_\_\_\_\_\_Е.А. Коробкова |

**ВЕБ-ПРИЛОЖЕНИЕ**

**«Учёт журналов «Сибирячок»»**

|  |  |  |
| --- | --- | --- |
| Нормконтролер: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (В.А. Пролыгина) |
| Консультант по экономической части: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (М.А. Рачкова) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (П.Н. Чернигов) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Д.Ю. Алёнин) |

Иркутск 2023

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc132146343)

[1. Предпроектное исследование 9](#_Toc132146344)

[1.1. Описание предметной области 9](#_Toc132146345)

[1.2. Анализ инструментальных средств реализации 11](#_Toc132146346)

[2. Техническое задание на разработку программного продукта 16](#_Toc132146347)

[3. Проектирование 17](#_Toc132146348)

[3.1. Архитектура программного обеспечения 17](#_Toc132146349)

[3.2. Функциональное проектирование 19](#_Toc132146350)

[3.3. Проектирование базы данных 27](#_Toc132146351)

[3.4. Проектирование пользовательского интерфейса 31](#_Toc132146352)

[4. Реализация программного обеспечения 33](#_Toc132146353)

[4.1. Разработка интерфейса 33](#_Toc132146354)

[4.2. Разработка базы данных 36](#_Toc132146355)

[4.3. Разработка информационной системы 38](#_Toc132146356)

[5. Документирование программного обеспечения 42](#_Toc132146357)

[5.1.Руководство пользователя 42](#_Toc132146358)

[6. Стоимость разработки и внедрения программного продукта 45](#_Toc132146359)

[6.1. Расчет затрат на разработку программного продукта 45](#_Toc132146360)

[6.2. Расчет затрат на внедрение программного продукта 51](#_Toc132146361)

[6.3. Основные выводы 51](#_Toc132146362)

[Заключение 52](#_Toc132146363)

[Список использованных источников 53](#_Toc132146364)

[Приложение А Техническое задание 55](#_Toc132146365)

# **ВВЕДЕНИЕ**

Вопрос учёта товара является неотъемлемой частью жизни организации, благодаря учёту товаров, появляется возможность делать график продаж и тем самым в какой-то степени те же самые продажи поднимать.

Данная тема была выбрана по той причине, что решает проблему универсальным способом, позволяет использовать разрабатываемый программный комплекс не только в выбранном заведении, но и в других учреждениях.

Товарный учёт — это отражение операций с продукцией: покупка или производство, хранение, реализация, возврат. А также информация:

* когда товар поступил по накладной от поставщика
* сколько заплатили поставщику
* как перемещали между складами и магазинами
* когда продали на кассе
* какую выручку получили.

Целью дипломной работы является разработка информационной системы «учёт журналов «Сибирячок»», основная задача которой, состоит в поиске, обработке и хранении информации, личном кабинете сотрудников для облегчения работы персонала, по учёту журналов, для облегчения работы с помощью экспорта данных из базы данных в excel файл.

Задачи дипломной работы будут следующие:

1. исследование предметной области;
2. выборка среды разработки приложения;
3. создание веб – приложения, используя различные средства веб – разработки;
4. описание руководство пользователя;
5. подготовка технической документации;
6. проектирование программного обеспечения;
7. разработка технической документации;
8. разработка и отладка программного продукта.

# **1. Предпроектное исследование**

* 1. Описание предметной области

В качестве предметной области курсового проекта была выбрана работа областного государственного бюджетного учреждения культуры (далее – ОГБУК) редакция журнала «Сибирячок».

Редакция занимается, приобщением детей к культурному наследию народов Сибири и Дальнего Востока, просвещение в области истории, этнографии и краеведения. Созданием условий для выявления, поддержки и развития одаренных детей в области литературы и искусства. Популяризацией чтения, формирования государственного отношения к чтению как одному из основных источников духовности.

Осуществляет такие виды деятельности как, редакторская, издательская, полиграфическая, просветительная, творческая и литературная.

Разработка журнала начинается с написания, нахождения и утверждения текста, после художники рисуют рисунки, после происходит вёрстка журнала и после вёрстки журнал утверждается редакционной коллегией и печатается.

ОГБУК редакция журнала «Сибирячок» было взято за основу проекта. Оно имеет штат сотрудников и читателей, занимается редакцией, выпуском и продажей журнала.

В данной предметной области документооборот включает в себя информацию о следующих компонентах:

* сотрудниках;
* читателях;
* журналах;
* продажах.

В соответствии с предметной областью веб приложения «учёт журналов Сибирячок» можно выделить базовые сущности проектируемой ИС.

Учёт журнала, основные документы редакции, информацию о сотрудниках, каталог журналов, информацию о складе, информация о списании и продажах.

1.2. Анализ инструментальных средств реализации

Для разработки данного продукта было рассмотрено несколько инструментальных средств разработки программного обеспечения. Такие как: Python, JavaScript, PHP, PHP Storm, WebStorm, Sql, Sqlite, Postgresql.

Python – это высокоуровневый язык программирования общего назначения. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами.

Javascript – динамический скриптовый язык программирования высокого уровня. Он отличается мультипарадигменностью. Речь идет о поддержке функционального, императивного, событийно-ориентированного стилей. Чаще всего язык используется для создания интерактивных веб-страниц и приложений. Востребованность, сферы применения. Неизменно высокий интерес к Javascript подтверждают специальные международные рейтинги.

PHP – C-подобный скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Теперь сравним эти языки программирования между собой (Таблица 1).

Таблица 1 – Сравнение языков программирования

|  |  |  |  |
| --- | --- | --- | --- |
| Название языка программирования | PHP | JavaScript | Python |
| Наличие библиотек | + | + | + |
| Инструменты для работы с БД | + | + | + |
| Объектно-ориентированные возможности | + | + | + |
| Динамическая типизация | + | + | + |
| Возможность разработки как клиентской части, так и серверной | - | + | - |

Исходя из данной таблицы, было принято решение остановится на языке программирования PHP.

Продолжение таблицы 1 – Сравнение языков программирования

Разберем среды разработки. Выбор встал между PHP Storm и WebStorm.

PHPStorm – это удобная среда программирования, которая выделяется своей функциональностью. Она предусматривает следующие возможности:

* интеллектуальный редактор кодификации в файле предусматривает подсветку синтаксиса, автодополнение кода расширенными параметрами форматирования;
* есть генераторы, сопрограммы, а также иные улучшения в плане синтаксиса;
* работа с детектором дублируемого кода, PHP рефакторинг;
* контактирование с Docker, Composter, встроенным REST клиентом;
* есть SSG консоль;
* визуальный отладчик для PHP контента, валидация конфигурации отладчика;
* поддержка CSS, HTML, JS;
* набор инструментария для проведения фронтенд разработки проекта;
* поддержка стилей кода, а также наличие встроенных стилей.

WebStorm – интегрированная среда разработки на JavaScript, CSS и HTML от компании JetBrains. Хорошо понимает структуру проектов и поможет вам с любыми аспектами написания кода. Автодополнение кода, безопасный рефакторинг, постоянный поиск потенциальных проблем и подсказки по их исправлению всегда будут у вас под рукой. Поиск потенциальных проблем и подсказки по их исправлению всегда будут у вас под рукой.

Для выбора среды разработки была составлена Таблица 2.

Таблица 2 – Сравнение сред разработки

|  |  |  |
| --- | --- | --- |
| Название IDE | PhpStorm | WebStorm |
| Автосохранение | + | + |
| Автодополнение | + | + |
| Интеграция с системами управления версиями (git) | + | + |
| Поиск по коду | + | + |
| Встроенные языки программирования | PHP, HTML, JavaScript, CSS, SQL | HTML, JavaScript, CSS, SQL |

После сравнения были сделаны выводы, что для выбранной дипломной работы лучшее всего подойдет PhpStorm.

Для выбора средств работы с базой данных, был проведен анализ между двумя более известными СУБД, такими как MySQL и PostgreSQL.

MySQL достаточно легко инсталлируется, а наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных. Обширный функционал. Система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта.

Преимущества:

* простота в использовании. MySQL достаточно легко инсталлируется, а наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных;
* обширный функционал. Система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта;
* безопасность. Система изначально создана таким образом, что множество встроенных функций безопасности в ней работают по умолчанию;
* масштабируемость. Являясь весьма универсальной СУБД, MySQL в равной степени легко может быть использована для работы и с малыми, и с большими объемами данных;
* скорость. Высокая производительность системы обеспечивается за счет упрощения некоторых используемых в ней стандартов.

PostgreSQL – это самая продвинутая СУБД, ориентирующаяся в первую очередь на полное соответствие стандартам и расширяемость. PostgreSQL, или Postgres, пытается полностью соответствовать SQL-стандартам ANSI/ISO.

PostgreSQL отличается от других РСУБД тем, что обладает объектно-ориентированным функционалом, в том числе полной поддержкой концепта ACID (Atomicity, Consistency, Isolation, Durability).

Преимущества:

* полная SQL-совместимость;
* сообщество: PostgreSQL поддерживается опытным сообществом 24/7;
* поддержка сторонними организациями: несмотря на очень продвинутые функции, PostgreSQL используется в многих инструментах, связанных с РСУБД;
* расширяемость: PostgreSQL можно программно расширить за счёт хранимых процедур;
* объектно-ориентированность: PostgreSQL – не только реляционная, но и объектно-ориентированная СУБД.

Сравним СУБД, чтобы определиться, какая больше подойдет для данного курсового проекта (Таблица 1).

Таблица 3 – Сравнение средств реализации базы данных

|  |  |  |
| --- | --- | --- |
| Название БД | MySQL | PostgreSQL |
| Большое количество типов данных | + | + |
| Популярность | + | - |
| Отказоустойчивость | - | + |
| Не требует удаленного сервера | - | - |
| Простота использования | + | - |
| Портативность | + | - |

При сравнении СУБД была выбрана MySQL, в связи с простотой использования, а также более известной в ходе обучения.

Для проектирования интерфейса и диаграмм выбрана среда разработки Draw io, поскольку является простой и удобной в использовании.

БУДЕТ ДОБОВЛЕНО СРАВНЕНИЕ ФРЕЙМВОРКОВ

## **2. Техническое задание на разработку программного продукта**

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602-89.

Согласно ГОСТ 34.602-89 техническое задание должно включать следующие разделы:

1. общие сведения;
2. назначение и цели создания системы;
3. требования к системе в целом;
   1. требования к структуре и функционированию системы;
   2. требования к надежности;
   3. требования к безопасности;
   4. требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
4. требования к документированию;
5. состав и содержание работ по созданию системы.

Техническое задание на разработку приложения представлено в приложении А.

3. Проектирование

3.1. Архитектура программного обеспечения

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает: выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов.

Веб-приложение «учёт журналов «Сибирячок»» использует клиент-серверную архитектуру (рисунок 3.1). **Архитектура «клиент-сервер» предусматривает разделение процессов предоставление услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняют свои задачи независимо от других.**

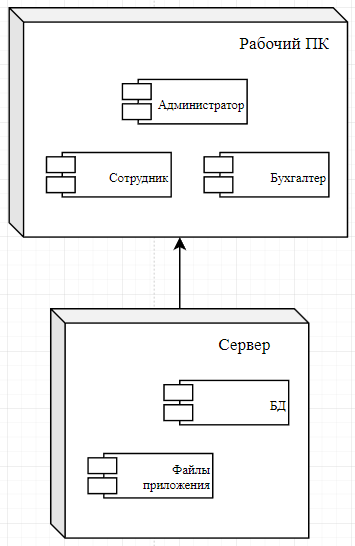


Рисунок 3.1.1 – Архитектура информационной системы «Учёт журналов Сибирячок»

Термин «клиент-серверная архитектура» – сборное понятие, состоящее из двух взаимодополняющих компонентов: сервера и, собственно, клиента.

Клиент – локальный компьютер на стороне виртуального пользователя, который выполняет отправку запроса к серверу для возможности предоставления данных или выполнения определенной группы системных действий.

Сервер – мощный компьютер или специальное системное оборудование, которое предназначается для разрешения определенного круга задач по процессу выполнения программных кодов. Он выполняет работы сервисного обслуживания по клиентским запросам, предоставляет пользователям доступ к определенным системным ресурсам, сохраняет данные или БД.

Особенности такой модели заключаются в том, что пользователь отправляет определенный запрос на сервер, где тот системно обрабатывается и конечный результат отсылается клиенту. В возможности сервера входит одновременное обслуживание сразу нескольких клиентов.

Если одновременно поступает более одного запроса, то такие запросы устанавливаются в определенную очередь и сервером выполняются по очереди. Порой запросы могут иметь свои собственные приоритеты. Часть запросов с более высокими приоритетами будут постоянно выполняться в первоочередном порядке.

Параметры, которые могут реализоваться на стороне сервера:

* хранение, защита и доступ к данным;
* работа с поступающими клиентскими запросами;
* процесс отправки ответа клиенту.

Параметры, которые могут реализоваться на стороне клиента:

* площадка по предоставлению пользовательского графического интерфейса;
* формулировка запроса к серверу и его последующая отправка;
* получение итогов запроса и отправка дополнительной группы команд (запросы на добавление, обновление информации, удаление группы данных).

Архитектура системы клиент-сервер формулирует принципы виртуального общения между локальными компьютерами, а правила и принципы взаимодействия находятся внутри протокола.

Сетевой протокол – это набор правил, на основании которого выполняется точное взаимодействие между компьютерами внутри виртуальной сети.

Таким образом, архитектура системы клиент-сервер формулирует принципы виртуального общения между локальными компьютерами.

3.2. Функциональное проектирование

Проектирование информационной системы происходит при помощи CASE средств, которые помогают обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов.

На рисунке 3.2 изображена диаграмма прецедентов, которая показывает структурную схему информационной системы «Учёт журналов «Сибирячок»» для пользователя «Администратор», «Сотрудник», «Бухгалтер».

Данная схема отображает действия, выполняемые администратором, сотрудником и бухгалтером.

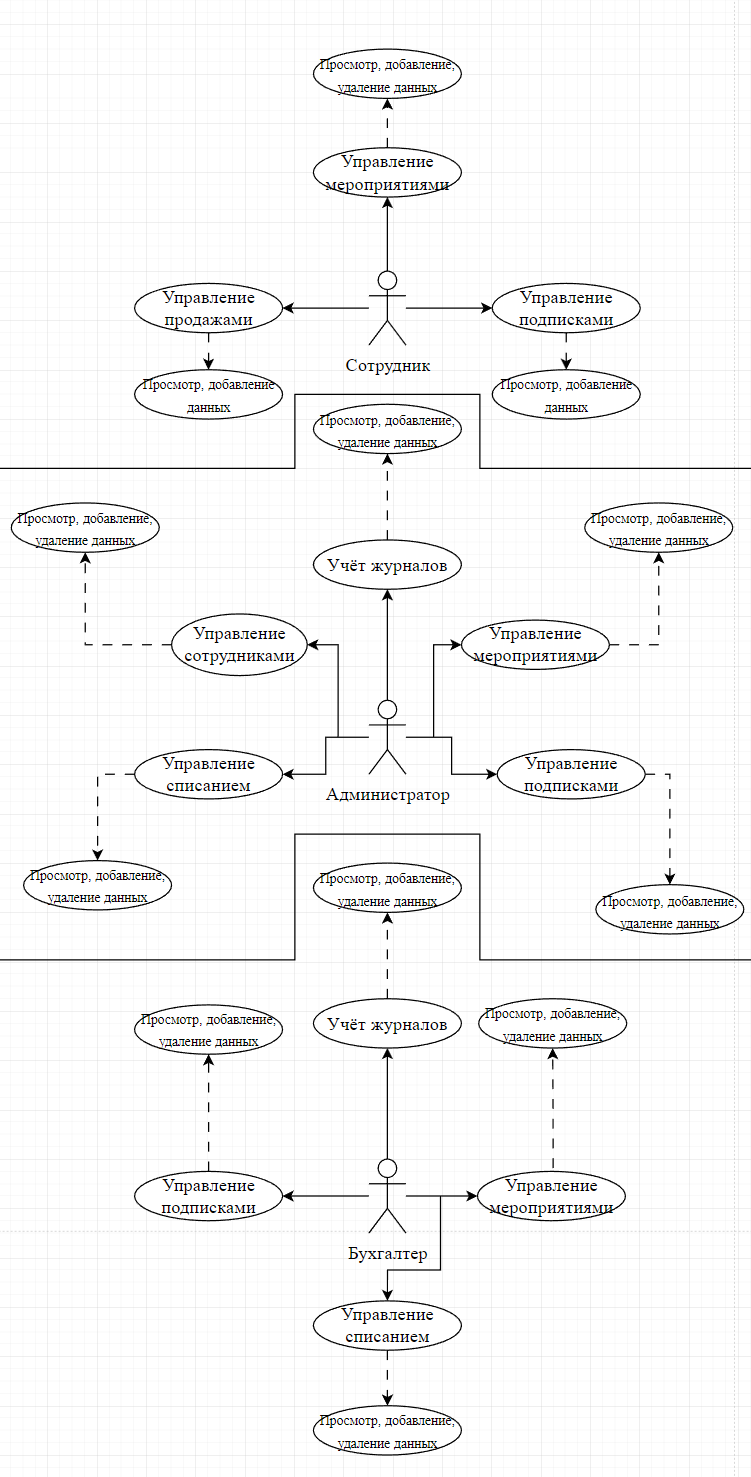


Рисунок 3.2 – Диаграмма вариантов использования пользователей.

Таким образом, представленные диаграммы демонстрируют взаимодействие основных объектов информационной системы и их действия.

Диаграмма вариантов использования – это графическое изображение возможных взаимодействий пользователя с системой. Диаграмма вариантов использования показывает различные варианты использования и различные типы пользователей, которые есть в системе, и часто сопровождается диаграммами других типов. Варианты использования представлены либо кругами, либо эллипсами.

Прецедент – возможность моделируемой системы, благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.

Диаграммы деятельности – частный случай диаграмм состояний. Позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних действий и деятельности. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. При этом каждое состояние может являться выполнением операции определенного класса либо ее части, позволяя использовать диаграммы деятельности для описания реакций на внутренние события системы.

Рисунок 3.3 представляет собой диаграмму деятельности, которая поэтапно описывает все процессы. На данной диаграмме будут описаны все действия, которые совершат актеры. В нашем случае, актерами являются «администратор», «сотрудник» и «бухгалтер».

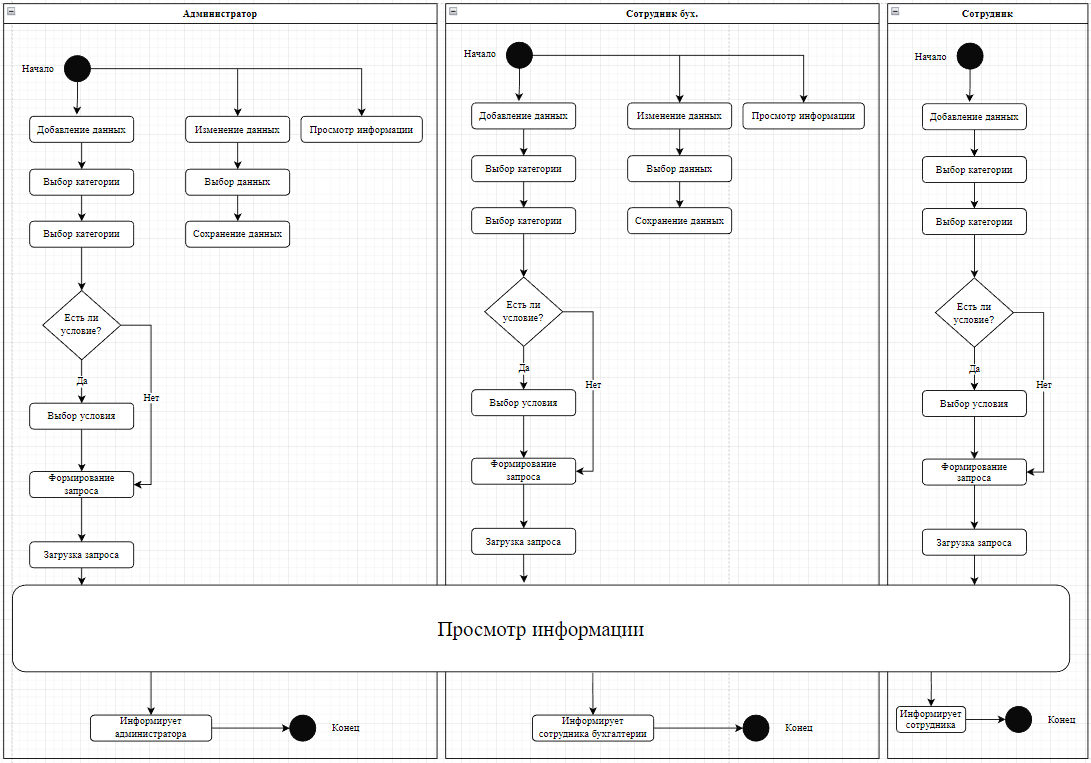


Рисунок 3.3 – Диаграмма деятельности информационной системы

Компонент – физически существующая часть системы, которая обеспечивает реализацию классов и отношений, а также функционального поведения моделируемой программной системы.

Компонент предназначен для представления физической организации ассоциированных с ним элементов модели. Дополнительно компонент может иметь текстовый стереотип и помеченные значения, а некоторые компоненты – собственное графическое представление. Компонентом может быть исполняемый код отдельного модуля, командные файлы или файлы, содержащие интерпретируемые скрипты (рисунок 3.4).

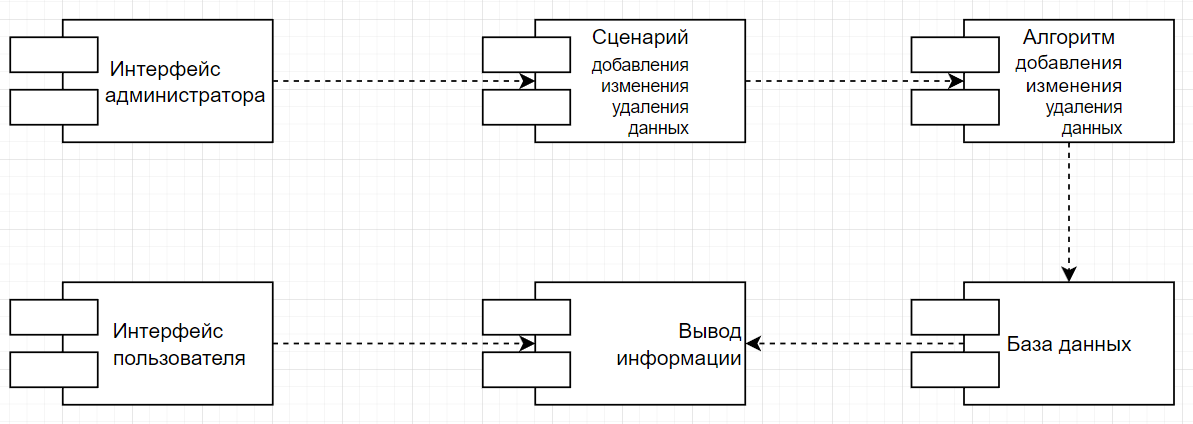


Рисунок 3.4 – Диаграмма компонентов

На рисунке 3.5 представлена диаграмма развертывания – это тип диаграммы, которая определяет физическое оборудование, на котором будет работать программная система. Он также определяет способ развертывания программного обеспечения на базовом оборудовании. Он отображает программные части системы на устройство, которое будет выполнять его.

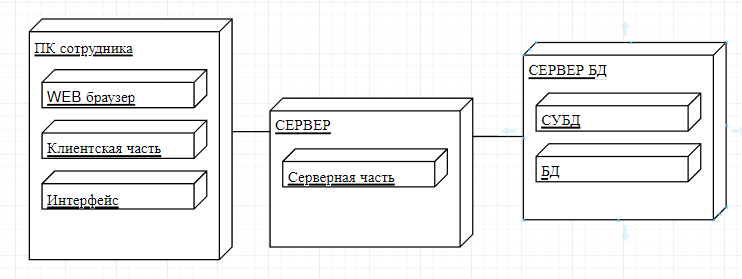


Рисунок 3.5 – Диаграмма развертывания

Контекстная диаграмма – это модель, представляющая систему как набор иерархических действий, в которой каждое действие преобразует некоторый объект или набор объектов.

На рисунке 3.6 показаны входные данные, управление, механизм, выходные данные и функция.

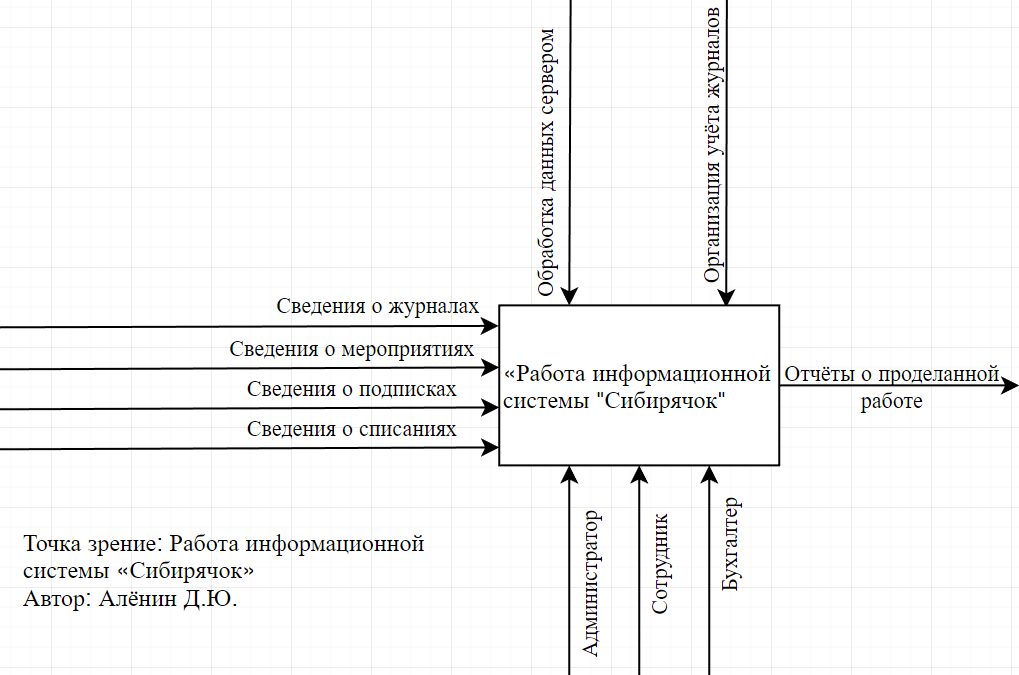


Рисунок 3.6 – Контекстная диаграмма

Диаграмма декомпозиции – предназначена для детализации работы.

На рисунке 3.7 показана диаграмма декомпозиций А1, которая расписывает функции программы. На этой диаграмме показаны такие функции, как учет домов, сотрудников, клиентов, планов и категорий.

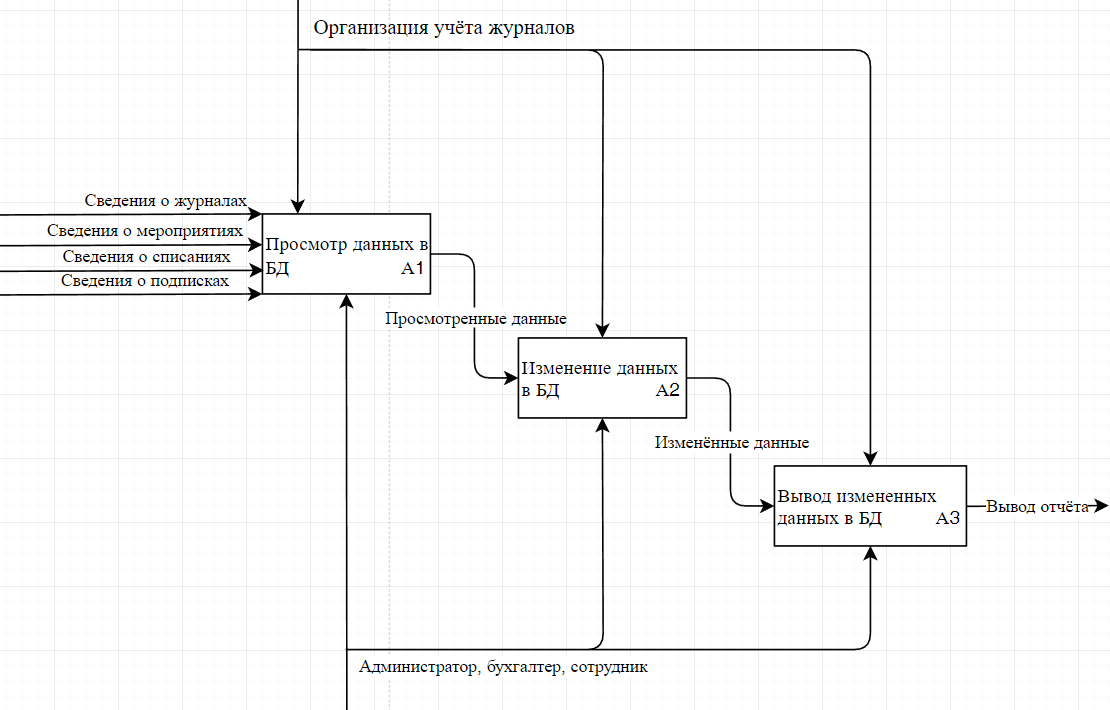


Рисунок 3.7 – Диаграмма декомпозиции А1

На рисунке 3.8 показана диаграмма декомпозиций А2, которая более подробно описывает диаграмму декомпозиции А1.

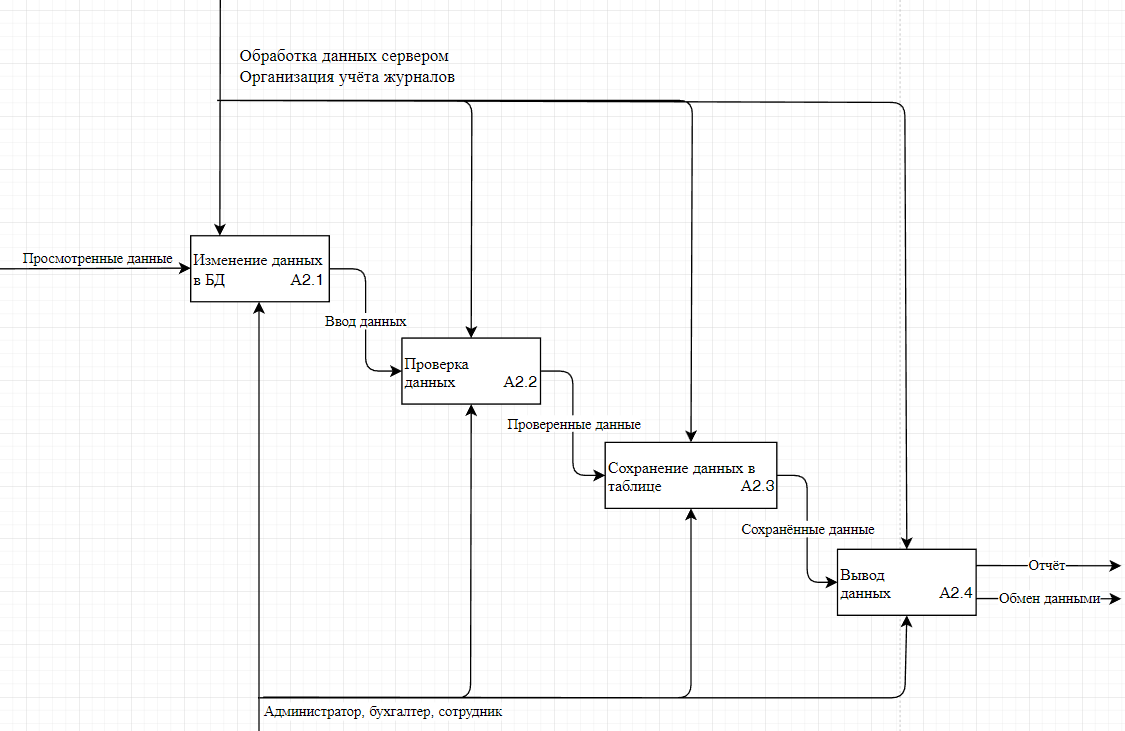


Рисунок 3.8 – Диаграмма декомпозиции А2

Также в ходе работы была разработана диаграмма классов, приведенная на рисунке 3.9, которая отображает внутреннюю работу веб приложения.

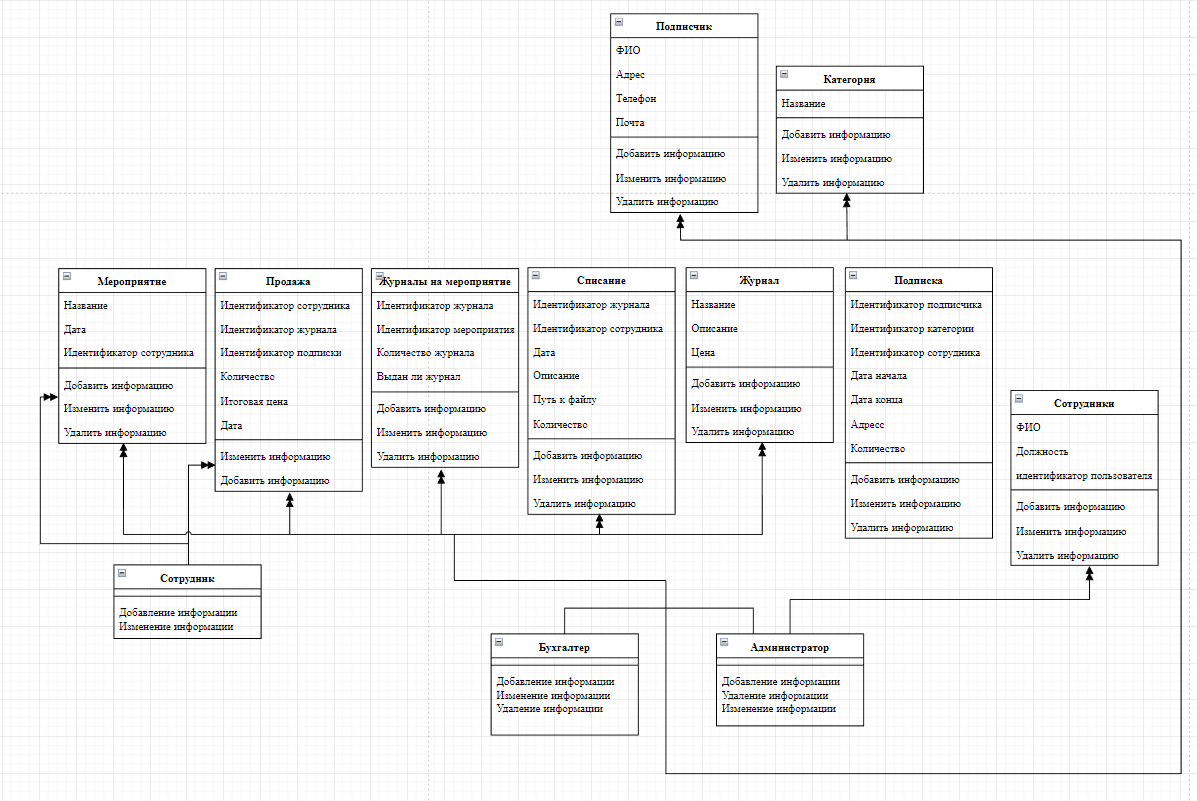


Рисунок 3.9 – Диаграмма классов

Диаграмма потоков данных, представленная на рисунке 3.10. Так называется методология графического структурного анализа, описывающая внешние по отношению к системе источники, и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

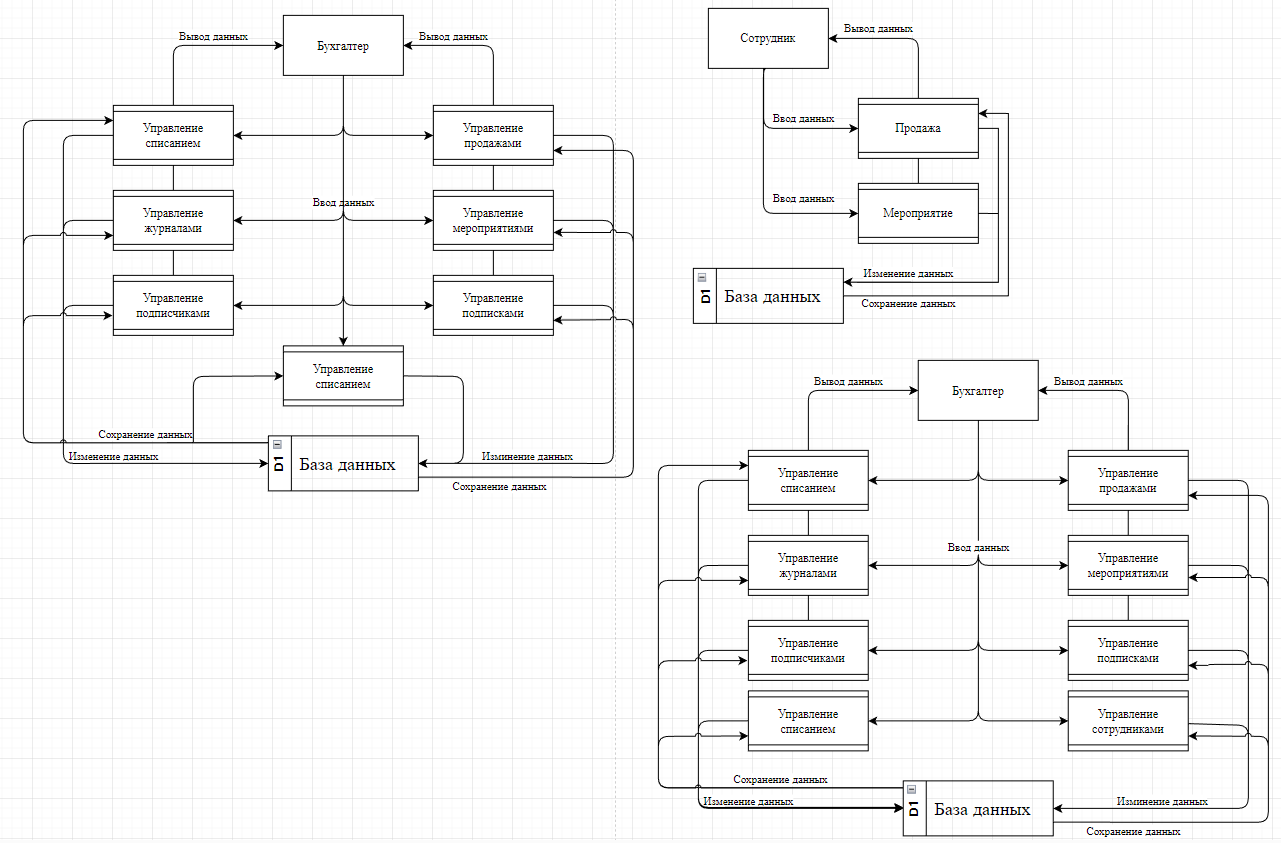


Рисунок 3.10 – Диаграмма потоков данных

Функциональное проектирование представляет наиболее общий подход к описанию систем. Определяются граничные условия и желательные входы, и выходы, составляется подробный перечень функций или операций, которые должны выполняться. При функциональном проектировании осуществляется синтез структуры и определяются основные параметры объекта и его составных частей (элементов), оцениваются показатели эффективности и качества процессов функционирования.

3.3. Проектирование базы данных

Перед разработкой базы данных есть необходимость в инфологическом моделировании. Результатом такого моделирования является инфологическая модель базы данных, изображенная на рисунке 3.11, которая наглядно показывает объект, его свойства и отношения между другими объектами. Объектами в данном случае являются: клиенты, сотрудники, дома, классы, планы. Свойствами объектов на схеме выступают овалы, а отношения между объектами – ромбы.

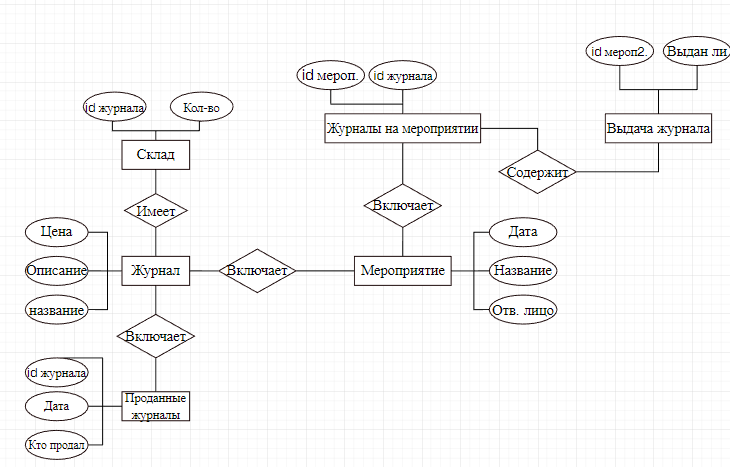


Рисунок 3.11 – Инфологическая модель БД

Датологическая модель данных – это модель, ориентированная на реализацию БД в конкретной СУБД, т.е. это инфологическая модель, трансформированная с учетом требований и ограничений конкретной СУБД Датологическая модель данных – это модель, ориентированная на реализацию БД в конкретной СУБД, т.е. это инфологическая модель, трансформированная с учетом требований и ограничений конкретной СУБД (рисунок 3.12).

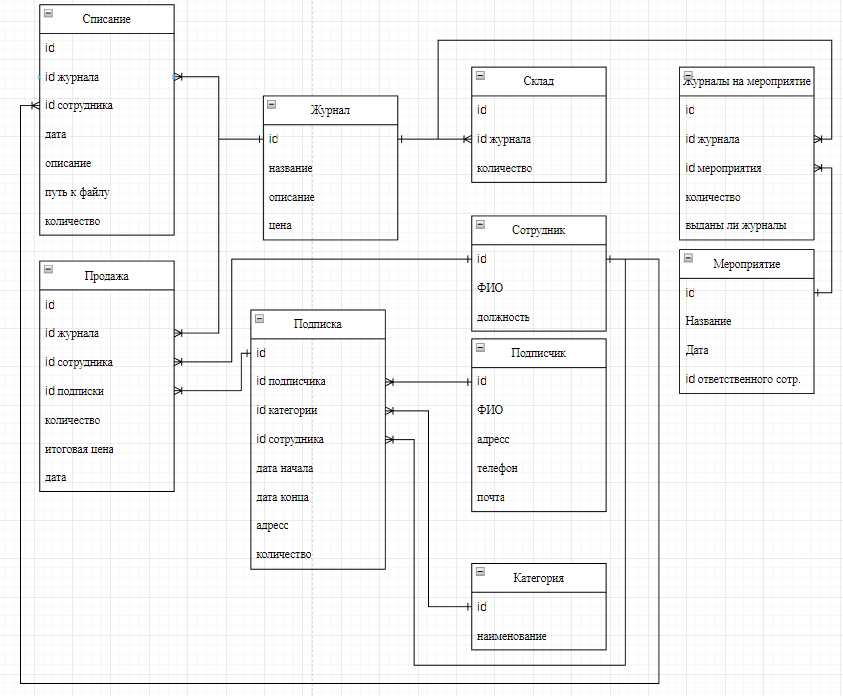


Рисунок 3.12 – Даталогическая модель

ER-модель – это семантическая модель данных, которая предназначена для упрощения процесса проектирования базы данных. Из ER-модели могут быть порождены все виды баз данных: реляционные, иерархические, сетевые, объектные. В основе ER-модели лежат понятия «сущность», «связь» и «атрибут».

Для больших баз данных построение ER-модели позволяет избежать ошибок проектирования, которые чрезвычайно сложно исправлять, в особенности, если база данных уже эксплуатируется или на стадии тестирования.

На рисунке 3.13 представлена ER–модель базы данных. Представлены таблицы, связи между ними и типы данных.

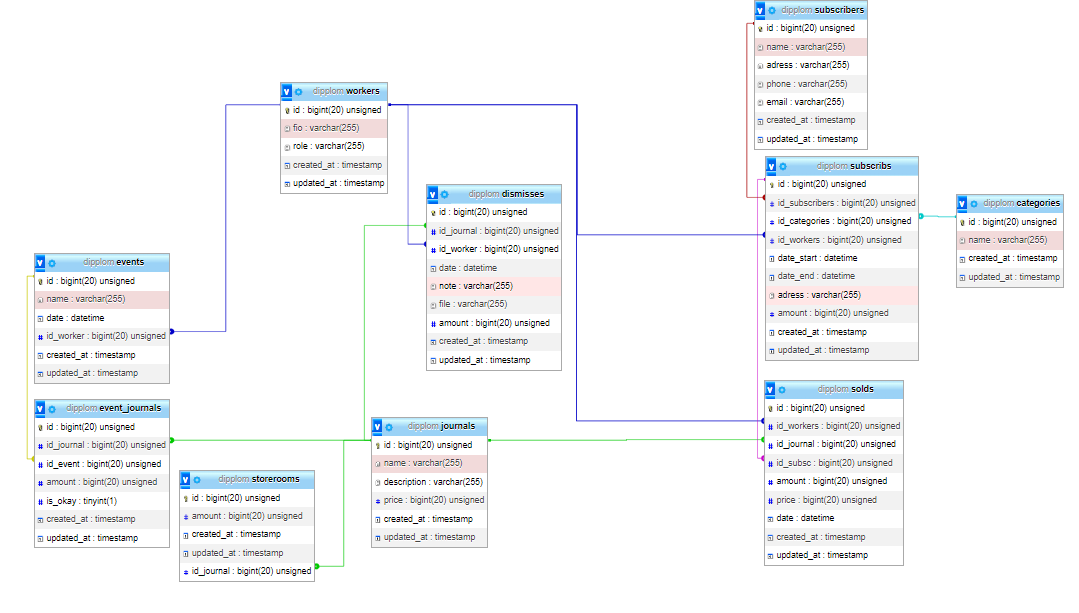


Рисунок 3.13 – ER-модель базы данных

Далее представим независимую сущность таблиц базы данных.

Данная модель содержит 10 таблицы:

Для разработки даталогической был использован инструмент Draw.io.

Для разработки ER-модели был использован PHPmyAdmin.

После завершения разработки прототипа базы данных и таблиц, находящихся в ней, можно перейти к проектированию интерфейса системы.

3.4. Проектирование пользовательского интерфейса

На рисунке 3.14 изображён прототип главной страницы дипломного проекта по теме «Учёт журналов «Сибирячок»». На данном прототипе можно увидеть информацию о проданных журналах в розницу, по подписке и те которые имеются на складе.

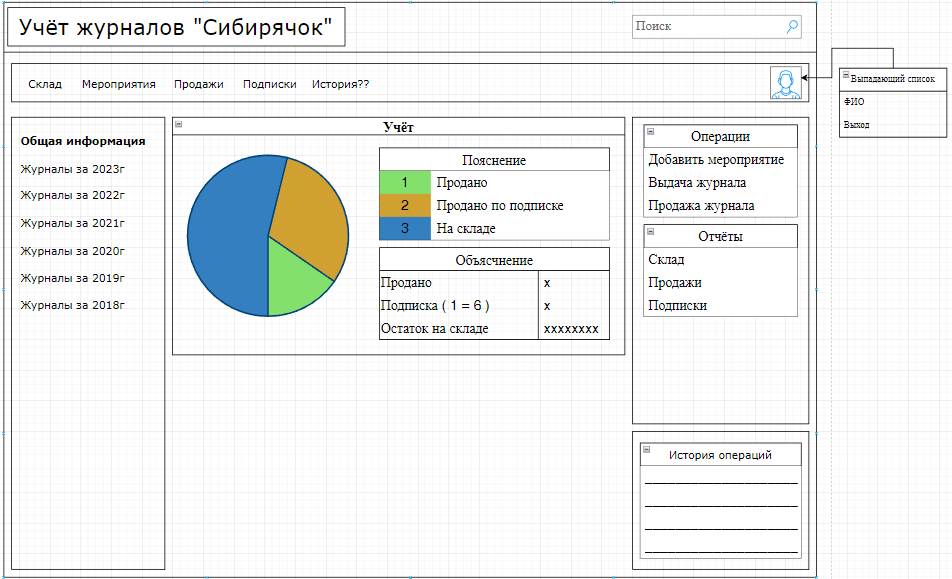


Рисунок 3.14 – Прототип страницы общей информации учёта товаров

На рисунке 3.15 изображен прототип страницы администратора. На данном прототипе можно будет все данные из БД и провести манипуляцию с теми или иными данными.

В верхней части страницы можно осуществить поиск информации.

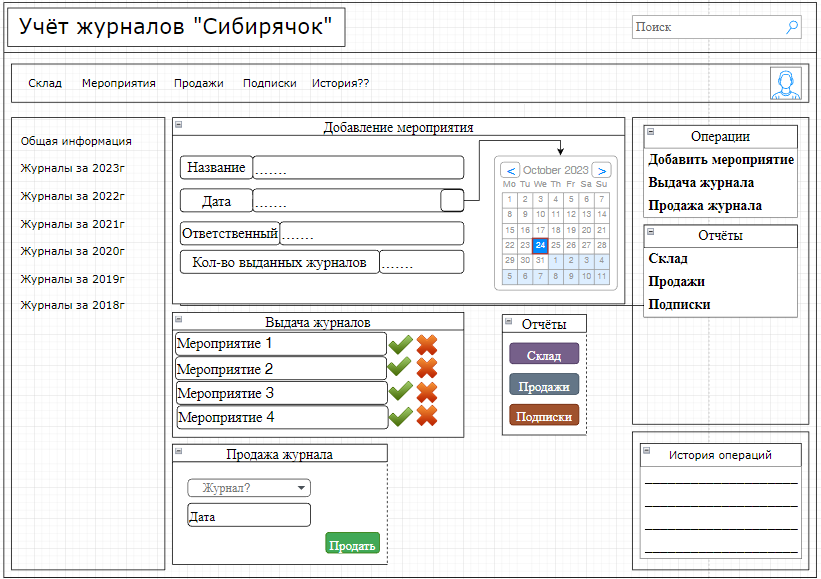


Рисунок 3.15 – Прототип главной страницы

На рисунке 3.16 изображен прототип страницы продажи и подписок. На данном прототипе можно будет просмотреть таблицу продаж, которые уже совершены, а также таблицу подписок.

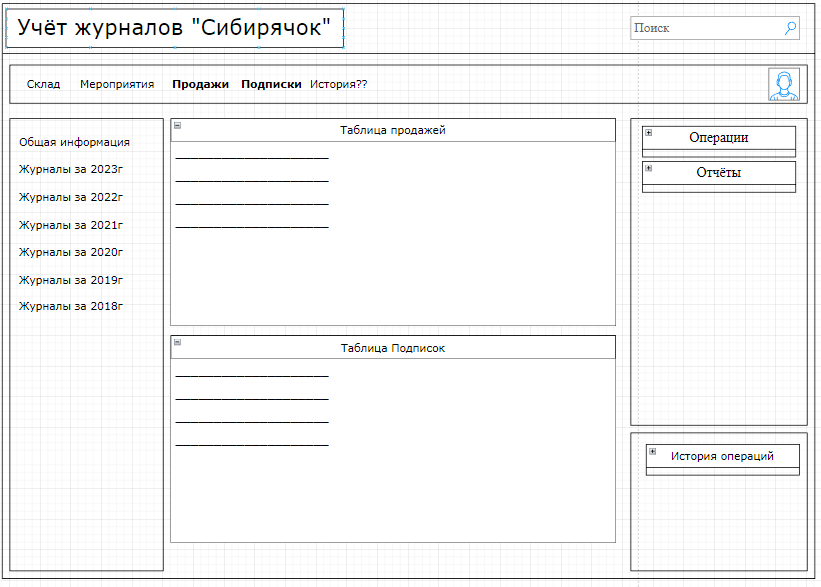


Рисунок 3.16 – Прототип страницы продажи и подписок

На рисунке 3.17 изображен прототип страницы истории операций, на котором изображены данные того, какие манипуляции проводились с данной информационной системой пользователем.

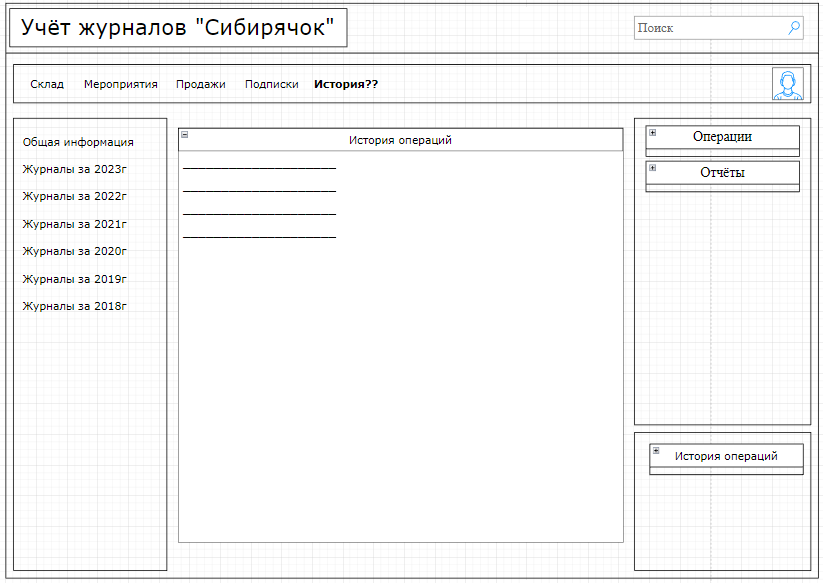


Рисунок 3.17 – Прототип страницы истории операций

На рисунке 3.18 изображен прототип страницы мероприятий, на котором изображены данные о мероприятиях, снизу этих данных изображена диаграмма, которая даёт понятие о том каких журналов на складе больше всего.

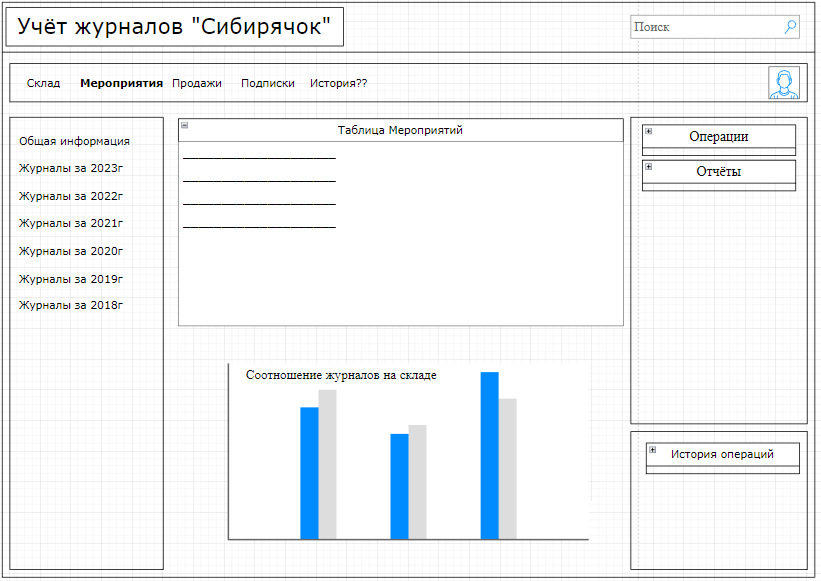


Рисунок 3.18 – Прототип страницы мероприятий

На рисунке 3.19 изображен прототип страницы проданных журналов, на котором изображены данные о продажах на определённый год.

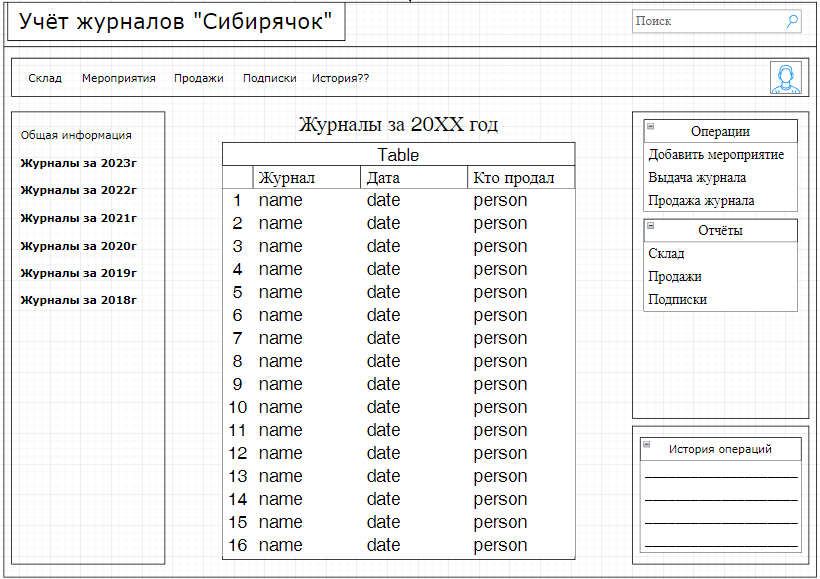


Рисунок 3.19 – Прототип страницы проданных журналов

На данном этапе спроектирован дизайн всех страниц данного информационной системы, после чего начинается этап создания самого информационной системы.

4. Реализация программного обеспечения

4.1. Разработка интерфейса

Работа над веб-приложением по теме «Учёт журналов «Сибирячок»» началась с установки веб интерфейс для администрирования СУБД «PhpMyAdmin» и установки пакетного менеджера уровней приложения для языка программирования PHP, который предоставляет средства по управлению зависимостями в PHP приложении.

Для того, чтобы развернуть проект, необходимо в меню PhpMyAdmin, найти кнопки «apache» и «MySQL» и нажать на эти кнопки, как показано на рисунке 4.1.

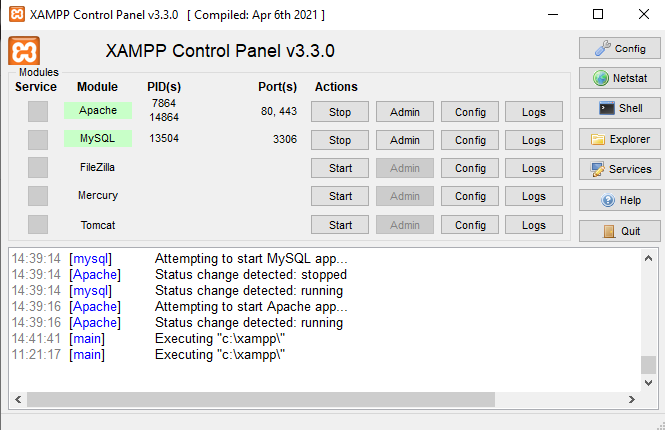


Рисунок 4.1 – интерфейс XAMPP

После чего скачать composer по адресу: «https://getcomposer.org/» и нажать на кнопку «dowloand» и пролистав немного вниз на раздел «Установка – Windows» и нажать на ссылку «Сomposer-Setup.exe» или же установить вручную как показано в следующем разделе (рисунок 4.2).

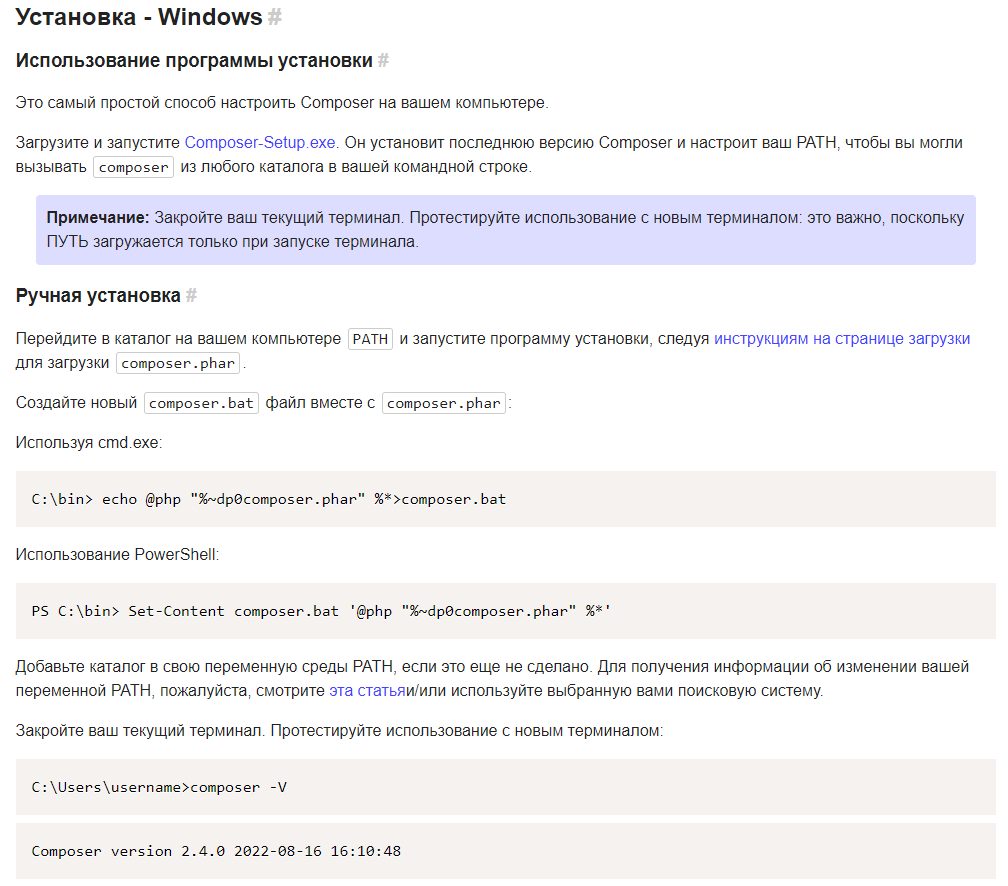


Рисунок 4.2 – установка composer.

Далее необходимо установить фреймворк Laravel, для более простого способа, необходимо перейти в документацию данного фреймворка по адресу: «https://laravel.com/docs/10.x» и в разделе «Your First Laravel Project» найти необходимые команды (рисунок 4.3). Далее перейти в необходимую директорию и вставить определённую команду в терминал, скопировав её или же написав вручную как показано на рисунке 4.4.

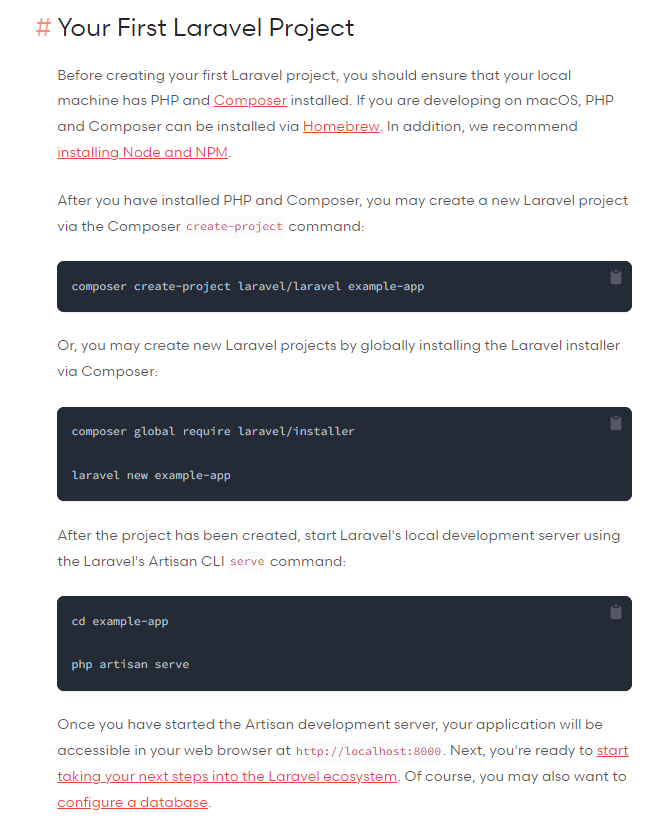


Рисунок 4.3 – документация Laravel.

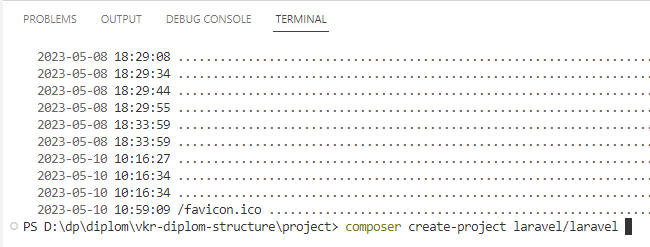


Рисунок 4.4 – команда для установки Laravel.

После этого, в указанной директории создастся проект Laravel (рисунок 4.5). Для запуска проекта, необходимо прописать команду «php artisan serve» в терминал.

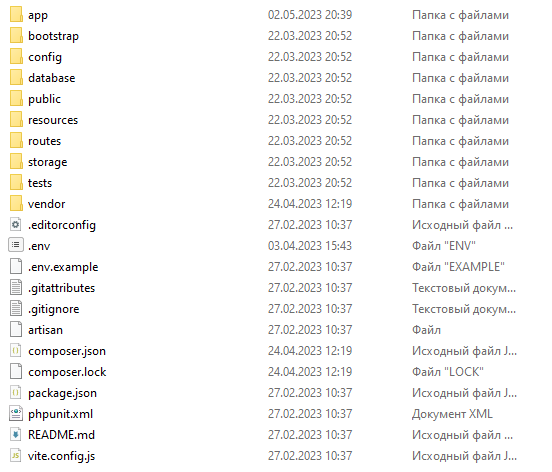


Рисунок 4.5 – Директория в которой создан проект

Разработка удобного пользовательского интерфейса – это один из важнейших этапов в процессе создания информационной системы. Пользователи должны легко и свободно пользоваться интерфейсом

Создание интерфейса началось с шапки информационной системы. Для того, чтобы не дублировать код в каждом файле создадим отдельный «layout» в папке: «PS:\dp\diplom\vkr-diplom-structure\project\resources\views\layouts» чтобы подключить его один раз в файле.

На рисунке 4.6 представлен результат создания шапки, на данной страницу находится выпадающий список, в котором находятся кнопки для авторизации или регистрации, и меню навигации по страницам информационной системы.

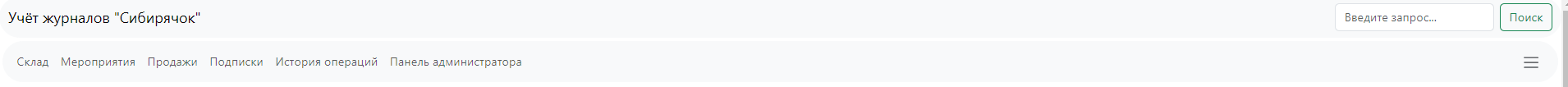


Рисунок 4.6 – Шапка страницы

Результат создания страницы «панель администратора» представлен на рисунке 4.7, на данной странице находятся все таблицы, справа от данных находятся кнопки удалить или же изменить те или иные данные, кнопка для добавления данных находится внизу каждой таблицы.

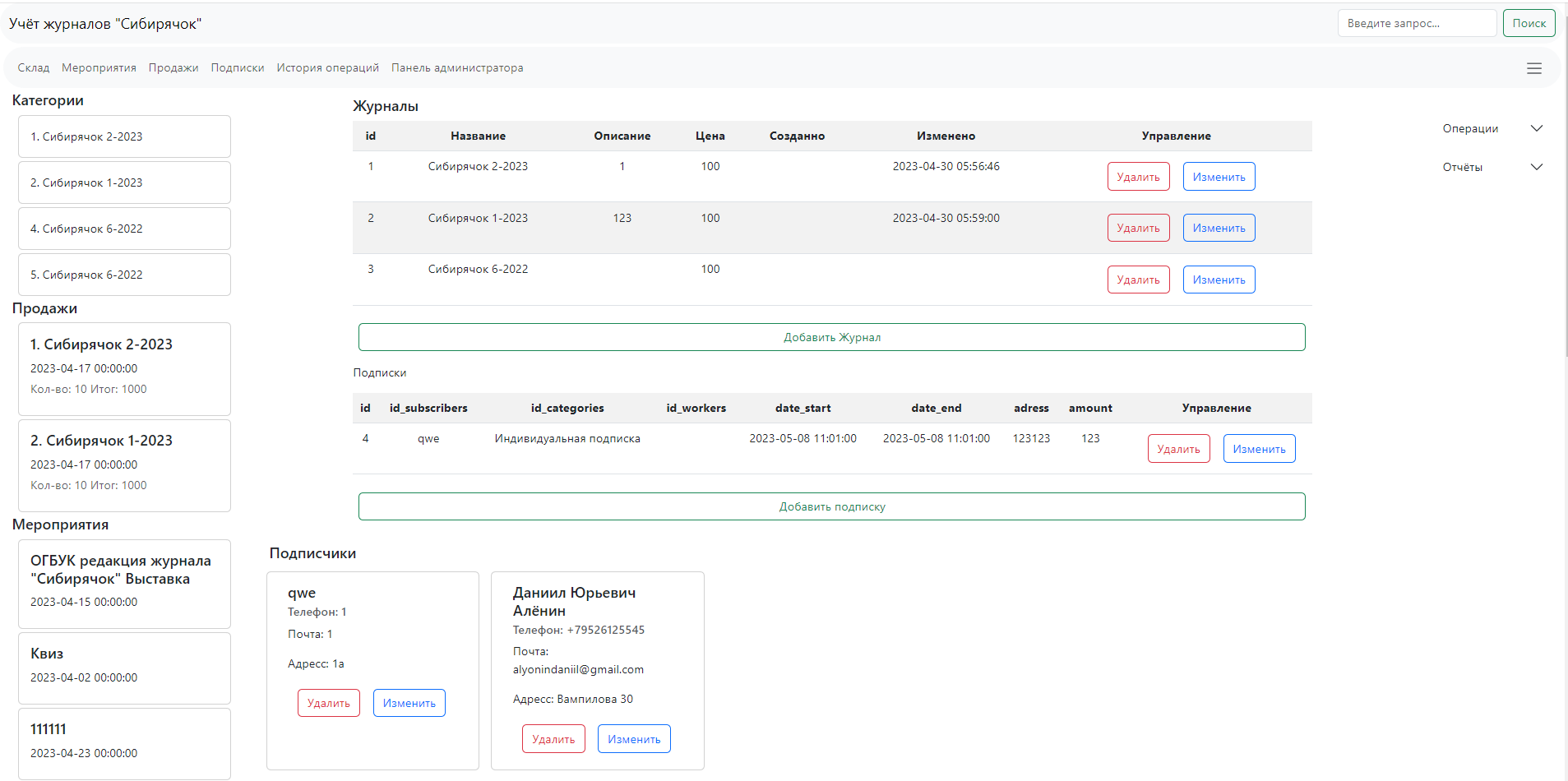


Рисунок 4.7 – Страница «панель администратора»

Результат создания страницы для сотрудников представлен на рисунке 4.8, на данной странице в середине находится таблицы продаж и мероприятий. Справа от таблиц находятся кнопки для скачивания отчётов, и добавления данных в БД.

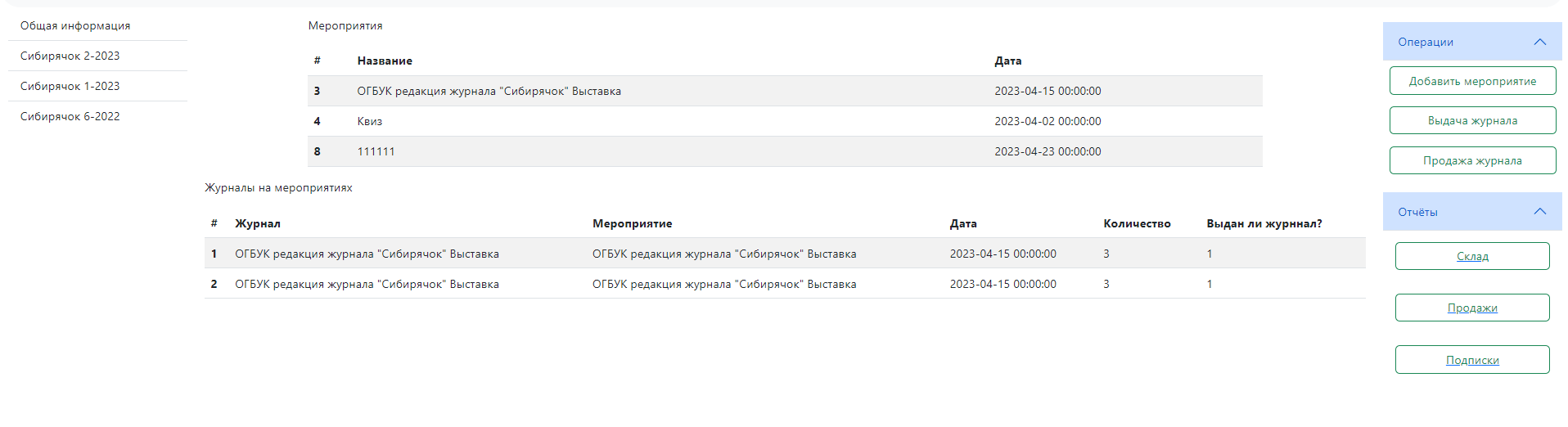


Рисунок 4.8 – Страница для сотрудников

Модальное окно для добавления мероприятия изображено на рисунке 4.9, на данном модальном окне, находятся два поля для записи данных.

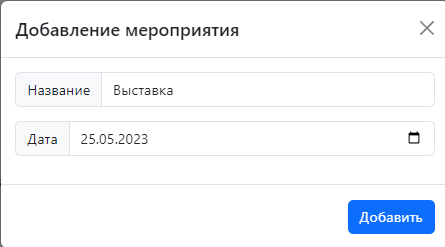


Рисунок 4.9 – Модальное окно добавления мероприятия

Страница с регистрацией и авторизацией сделана понятной и удобной для пользователей. Страница с авторизацией и регистрацией представлена на рисунке 4.10.

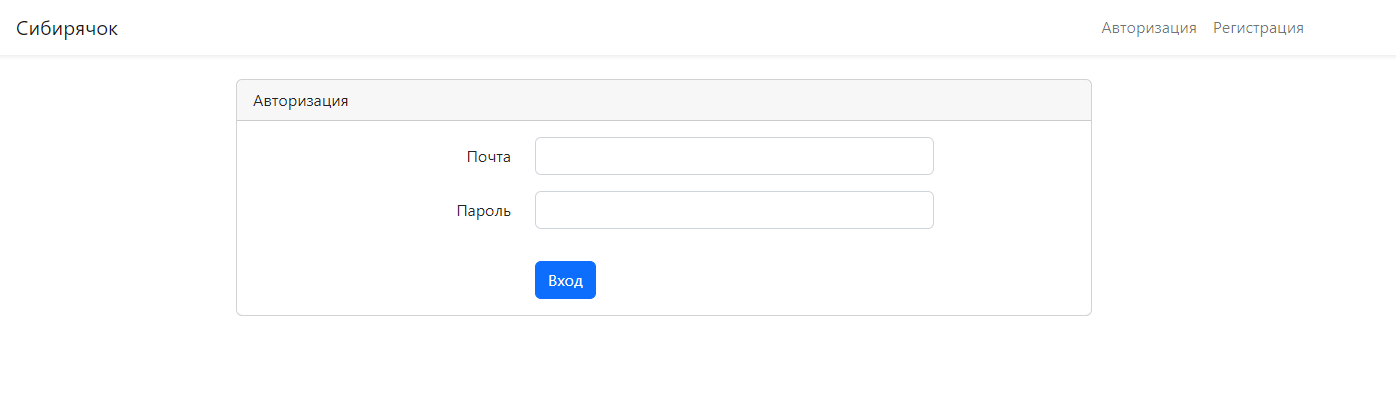


Рисунок 4.10 – Страница авторизации

Данный этап заканчивается на создании интерфейса, после чего начинается разработка базы данных для данной информационной системы.

4.2. Разработка базы данных

База данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Разработка базы данных информационной системы «Учёт журналов «Сибирячок»» реализовывалась в PhpMyAdmin (рисунок 4.11) посредством миграций. База данных информационной системы «Учёт журналов «Сибирячок»» состоит из 10 таблиц.

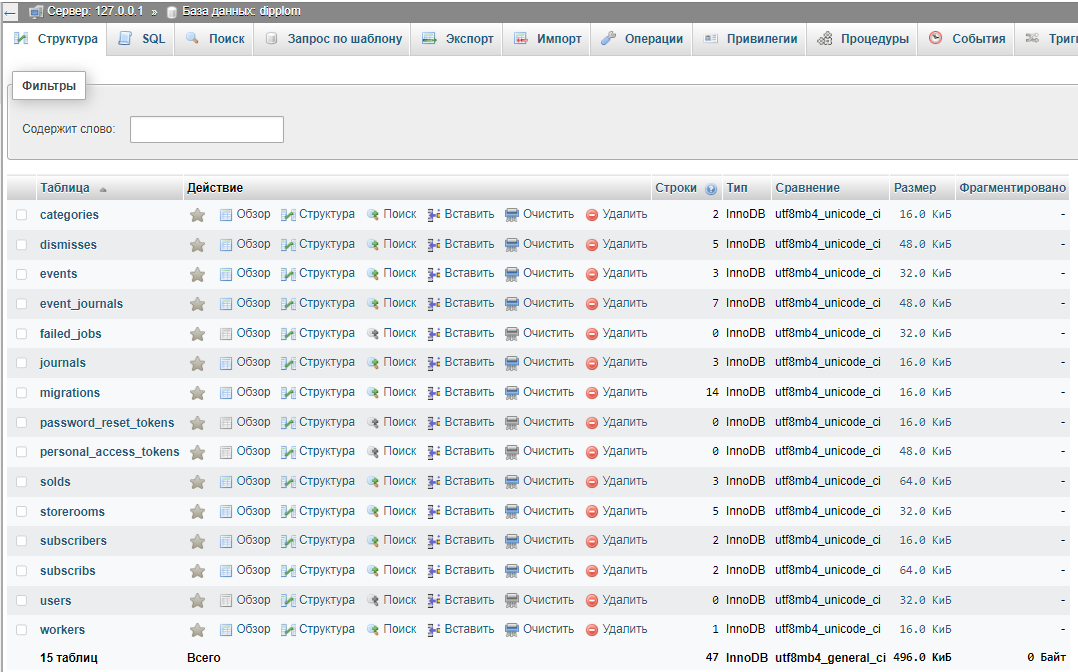


Рисунок 4.11 – СУБД phpMyAdmin и БД

На рисунке 4.12 представлена таблица категорий, в которой хранятся названия категорий. Таблица включает в себя только названия категорий

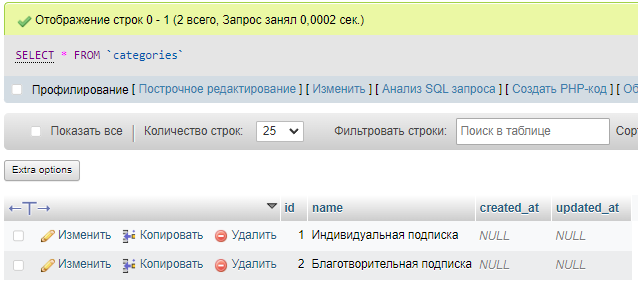


Рисунок 4.12 – Таблица «Категории»

На рисунке 4.13 представлена таблица списания. Таблица включает в себя списании какого-либо журнала, сотрудника, который зарегистрировал, данное списание, дату, описание, путь к файлу и количество списываемых журналов

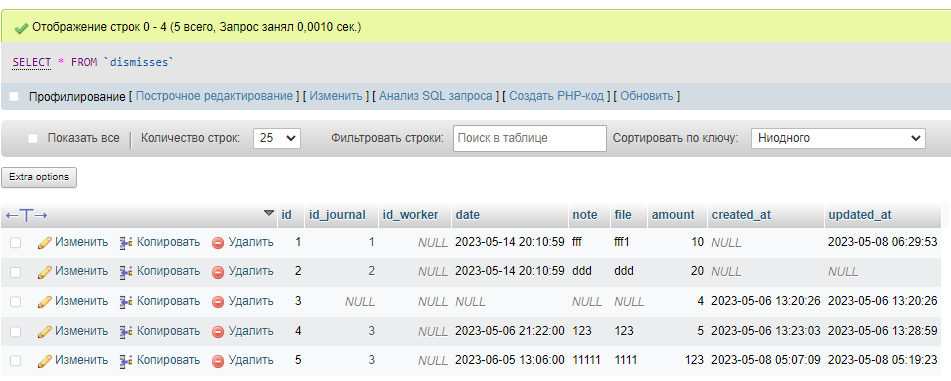


Рисунок 4.13 – Таблица «Списание»

На рисунке 4.14 представлена таблица с мероприятиями, которая включает в себя название, дату и ответственного сотрудника на данное мероприятие.

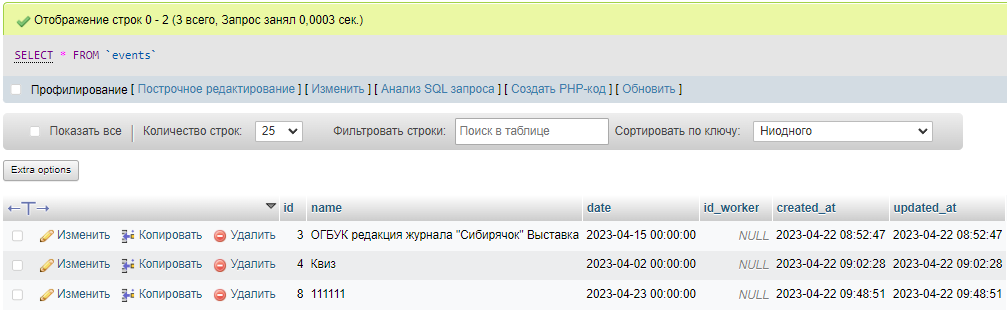


Рисунок 4.14 – Таблица «мероприятия»

На рисунке 4.15 представлена таблица с журналами на мероприятия, которая включает в себя идентификатор журнала, идентификатор мероприятия, количество, и факт выданы ли журналы.

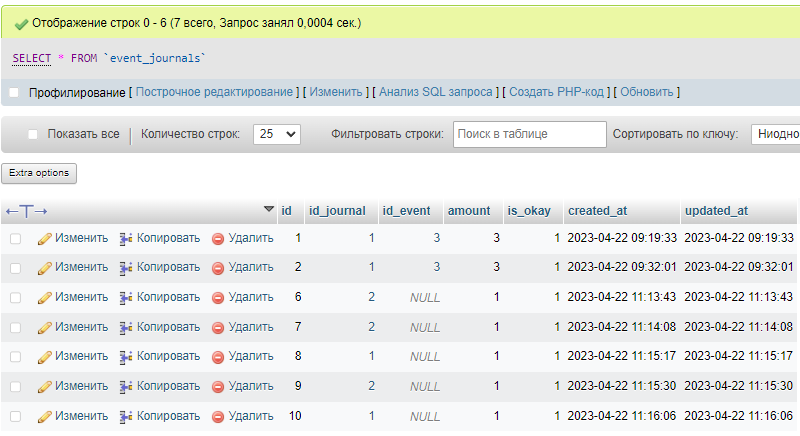


Рисунок 4.15 – Таблица «журналы на мероприятии»

На рисунке 4.16 представлена таблица с журналами, которая включает в себя название журнала, его краткое описание, и цена на данный журнал.

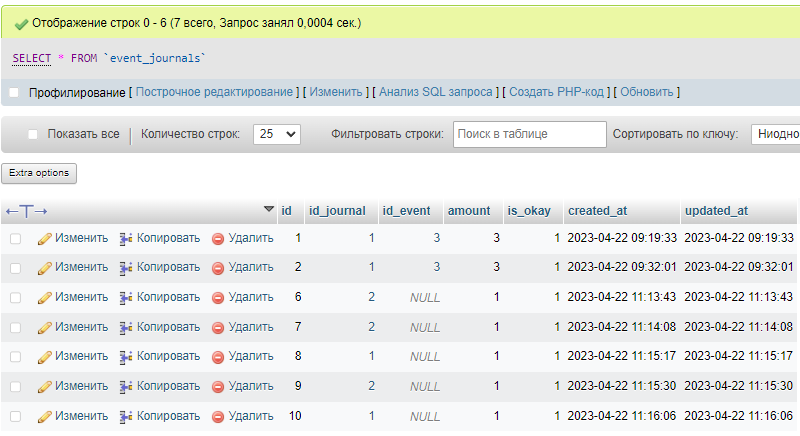


Рисунок 4.16 – Таблица «журналы»

На рисунке 4.17 представлена таблица с продажами, которая включает в себя идентификатор журнала, идентификатор сотрудника, который записал данную продажу, идентификатор подписки, дату, количество и итоговую стоимость.

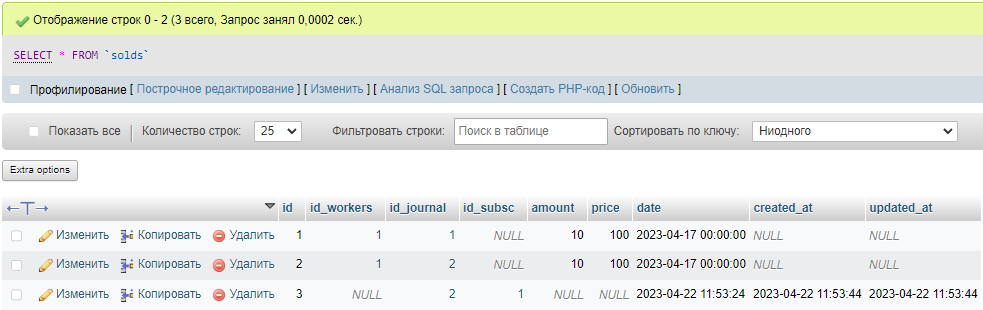


Рисунок 4.17 – Таблица «продажи»

На рисунке 4.18 представлена таблица склада, которая включает в себя идентификатор журнала и количество.

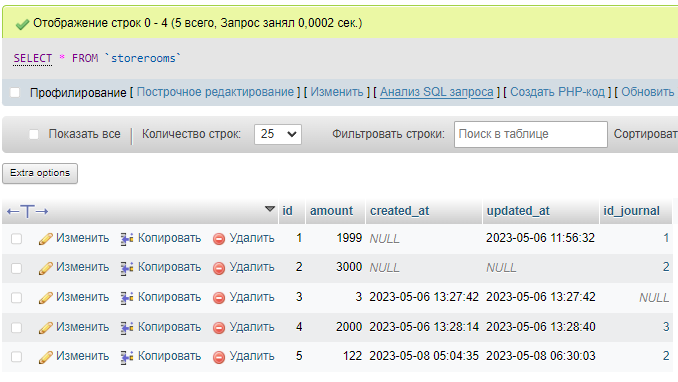


Рисунок 4.18 – Таблица «склад»

На рисунке 4.20 представлена таблица с подписчиками, которая включает в себя ФИО человека или наименование организации, адрес, телефон и электронную почту.

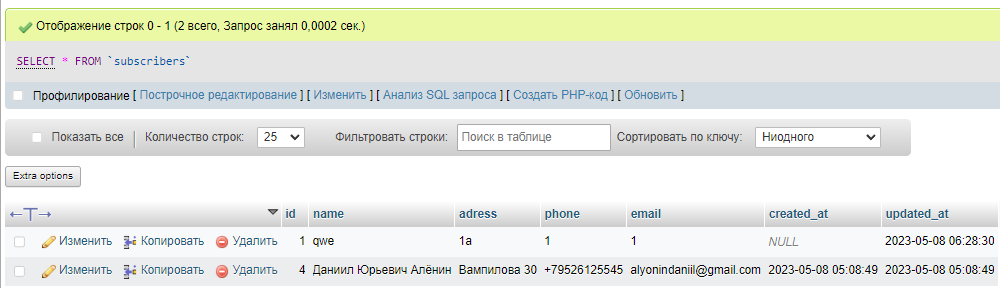


Рисунок 4.20 – Таблица «подписчики»

На рисунке 4.21 представлена таблица с подписками, которая включает в себя, идентификатор подписчика, идентификатор категории, идентификатор сотрудника, который добавил данную запись, дату начала, дату конца подписки адрес, и количество.

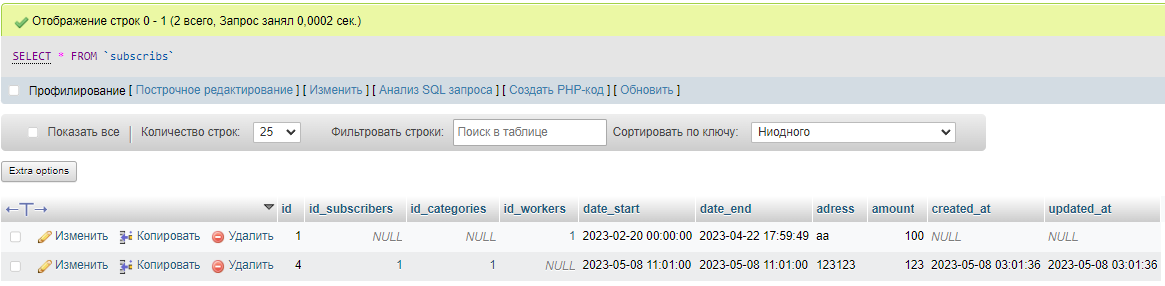


Рисунок 4.21 – Таблица «подписки»

На рисунке 4.22 представлена таблица с сотрудниками, которая включает в себя ФИО сотрудника и должность.

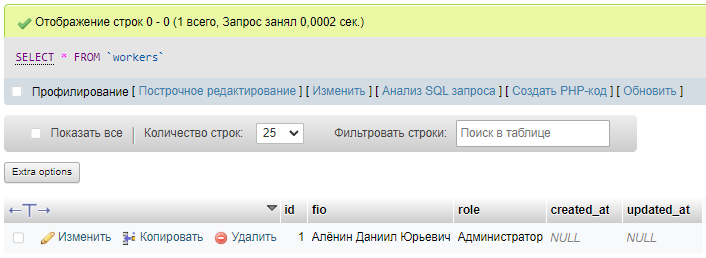


Рисунок 4.22 – Таблица «сотрудники»

Разработка функционала информационной системы «Учёт журналов «Сибирячок»» было реализовано на клиентской части, написанной на «Php» и серверной части, написанной на «Php». Обмен данными будет происходить по способу API (Application Programming Interface).

На рисунке 4.23 представлена директория серверной части.

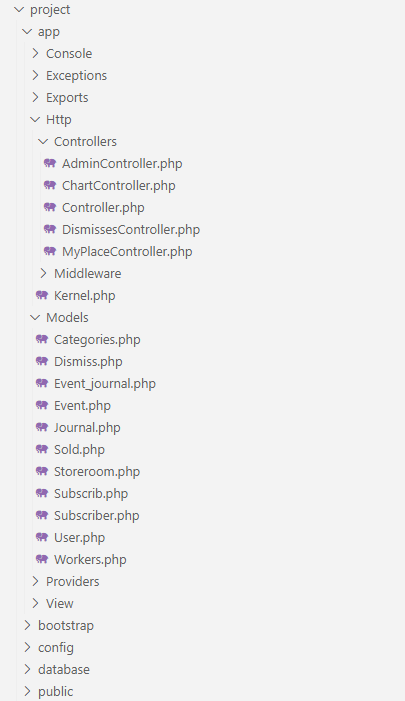


Рисунок 4.23 – структура файлов серверной части информационной системы

База данных и интерфейс разработаны, теперь необходимо начать этап кодирования программного продукта для работы информационной системы.

4.3. Кодирование программного продукта

На рисунке 4.24 изображен код шапки страницы с ссылками на разные страницы и стилями, привязанными к ним.



Рисунок 4.24 – PHP код для создания шапки

После чего необходимо подключить его к данных MySQL (рисунок 4.25) Поменяв конфигурационный файл «.env».

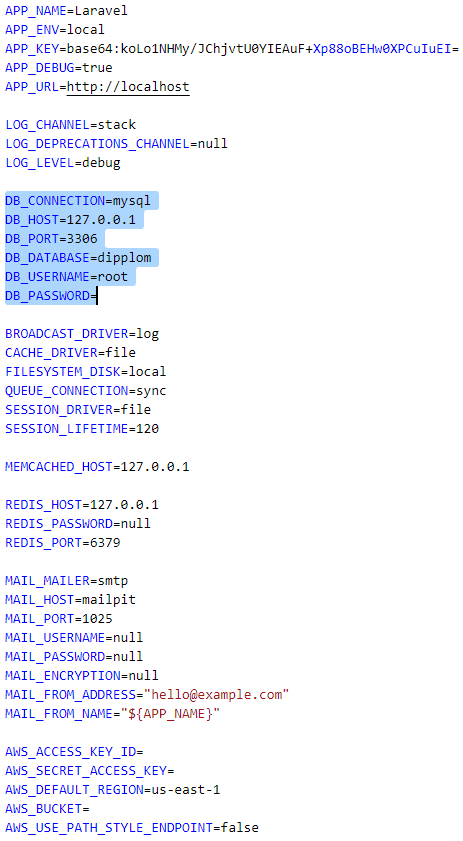


Рисунок 4.25 – Подключение шапки к родительскому компоненту

На рисунке 4.26 изображён код экспорта данных.

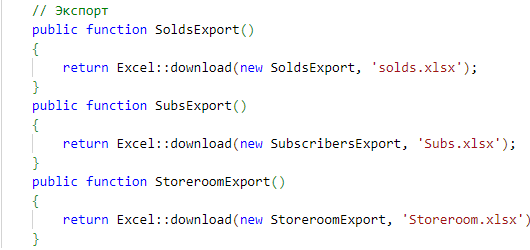


Рисунок 4.26 – Код функций, экспорта данных в файл excel

На рисунке 4.27 изображён код функции в контроллере, которая передаёт данные из БД на страницу администратора.



Рисунок 4.27 – Код функции, передачи данных из БД на страницу администратора

На рисунке 4.28 изображён код миграции таблицы подписок.



Рисунок 4.28 – Код миграции таблицы подписок

На рисунке 4.29 представлен код модель данных подписок.

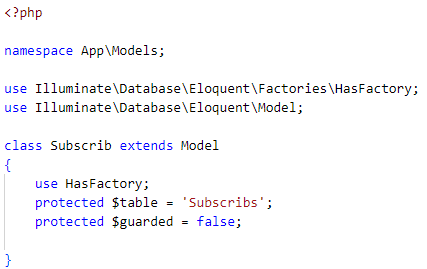


Рисунок 4.26 – Код модели таблицы подписок

После написания программного кода и подключения баз данных, веб-приложение готово к использованию.

5. Документирование программного обеспечения

Процесс написания документов программного обеспечения – важный этап в процессе создания и эксплуатации программного обеспечения. Именно с руководства пользователя начинает свое знакомство с программным продуктом.

## **Руководство пользователя**

При запуске информационной системы откроется страница авторизации, которая изображена на рисунке 5.1.

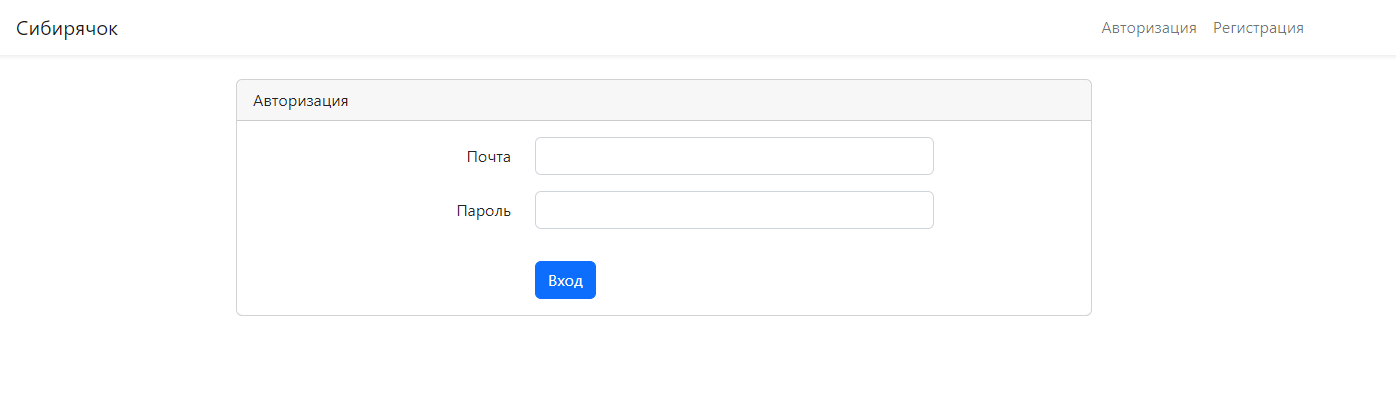


Рисунок 5.1 – Страница авторизации

После авторизации, пользователю с ролью администратора, выдаст страницу администратор (рисунок 5.2).

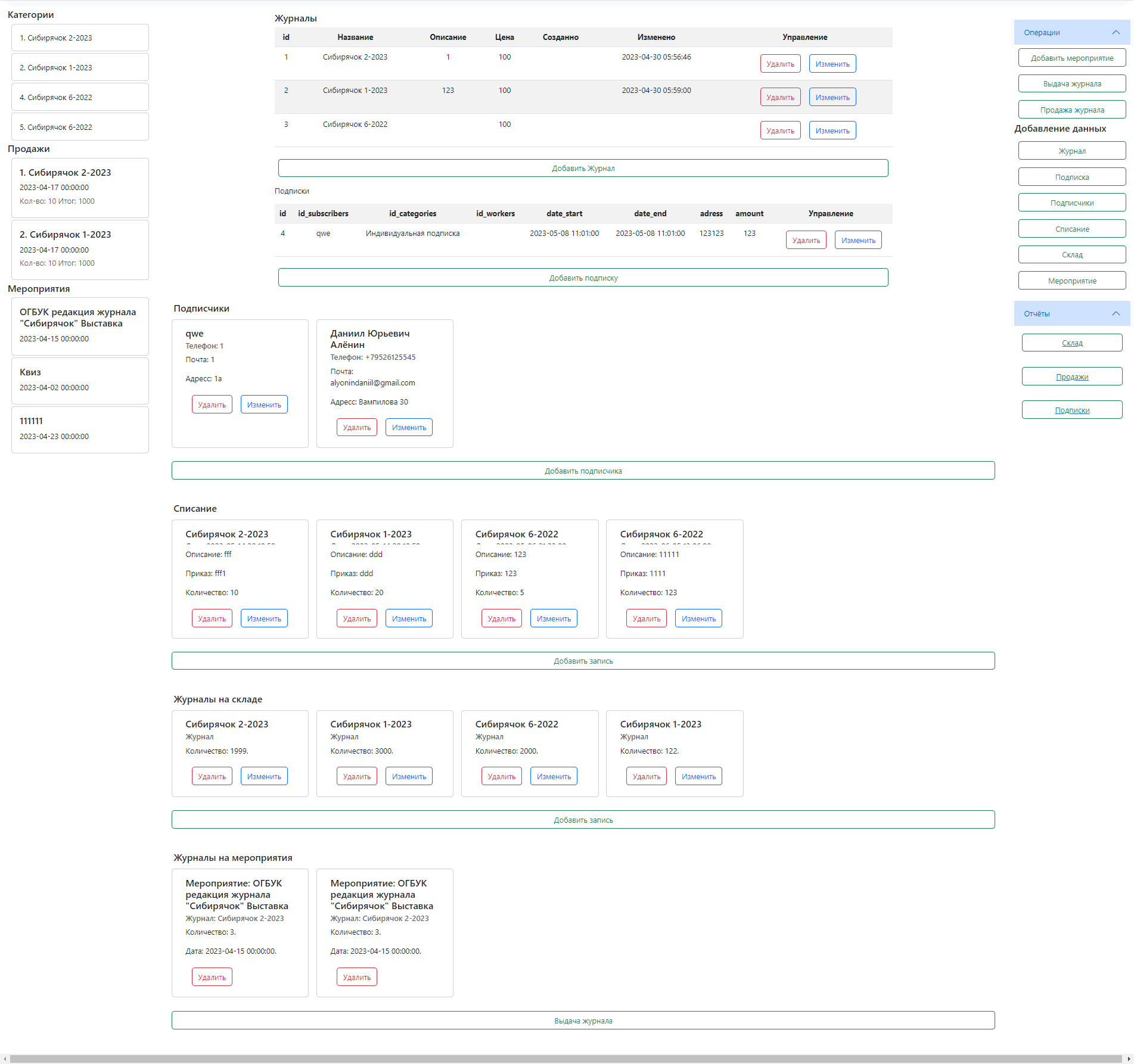


Рисунок 5.2 – страница администратора

Пользователя с ролью бухгалтера или сотрудника перенаправит на страницу общей информации о продажах (рисунок 5.3).

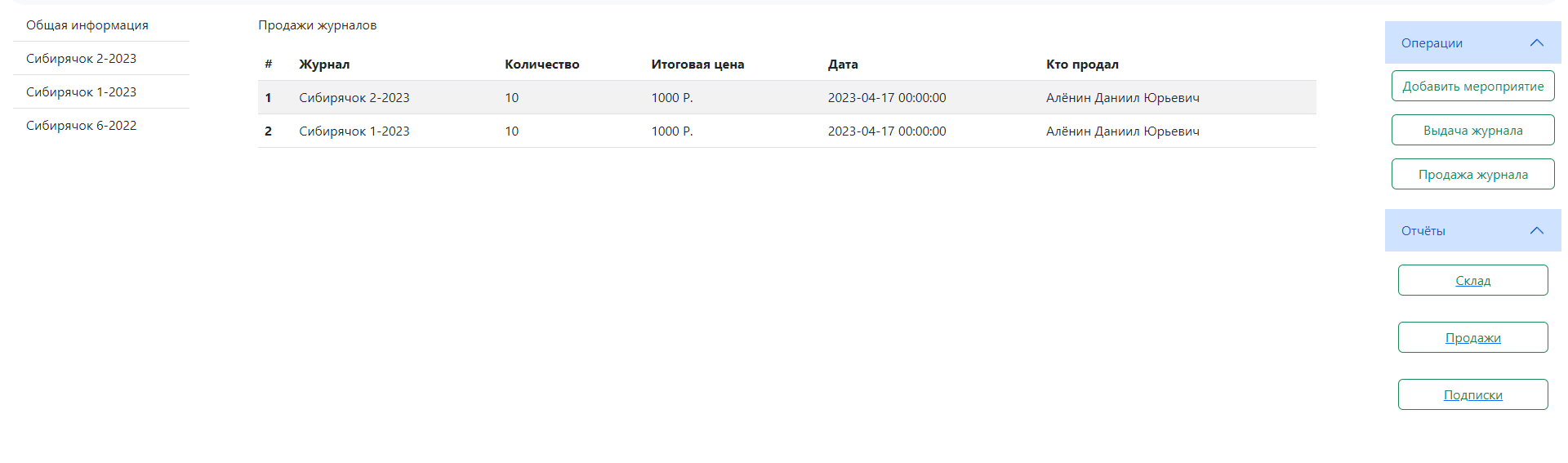


Рисунок 5.3 – Страница продажи журналов

Для продажи журнала, пользователю необходимо нажать на соответствующую кнопку в боковой панели справа, после нажатия откроется модальное окно (рисунок 5.4)

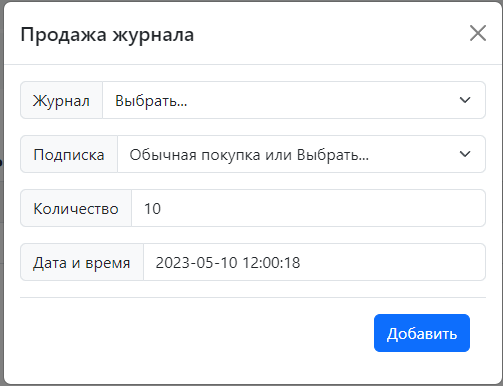


Рисунок 5.4 – Модальное окно продажи журнала

На странице администратор, для добавления, изменения или удаления данных, необходимо перейти по определённой кнопке.

Страница для изменения данных изображена на рисунке 5.5.



Рисунок 5.5 – Страница изменения данных

Перейдя на страницу «Склад», на страницу будет изображена диаграмма, которая показывает сколько журналов имеется на складе (рисунок 5.6).

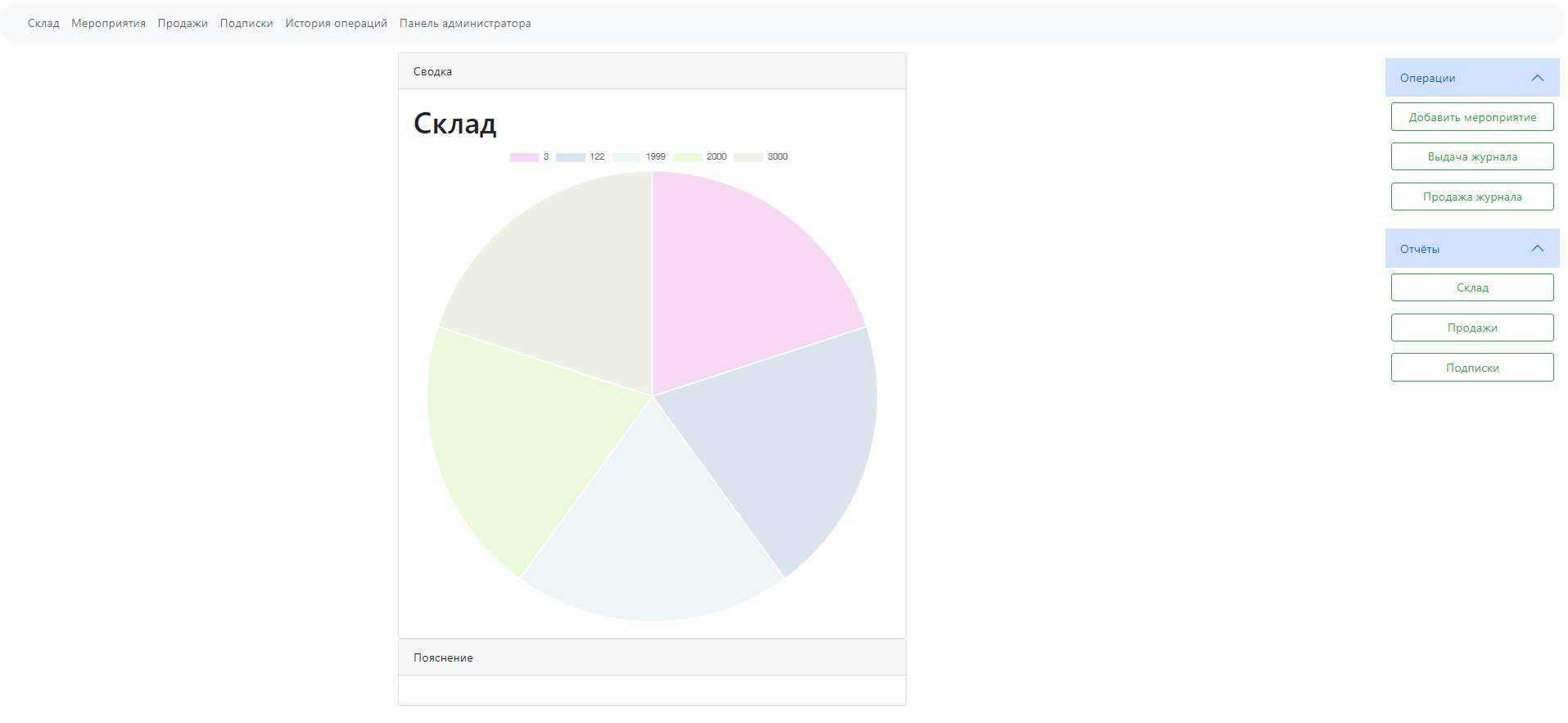


Рисунок 5.6 – Страница «Склад».

6. Стоимость разработки и внедрения программного продукта

6.1. Расчет затрат на разработку программного продукта

Расчет полных затрат на разработку проектного решения (КРПР) осуществляется по формуле:

КРПР=ЗОТР+ЗЭВМ+ЗСПП+ЗХОН+Е+А,

где ЗОТР – сумма оплаты труда разработчика/разработчиков ПП;

ЗЭВМ – затраты, связанные с эксплуатацией техники;

ЗСПП – затраты на специальные программные продукты, необходимые для разработки ПП;

ЗХОН – затраты на хозяйственно-операционные нужды (бумага, литература, носители информации и т.п.);

Е – затраты на электроэнергию, руб.;

А – амортизация ПК, руб.

Для подсчета фонда оплаты труда разработчика необходимо определить общее время разработки (таблица 9). Время, затрачиваемое на разработку проектного решения j-м разработчиком, определяется методом экспертных оценок или хронометража. Итоговое значение рассчитывается на основании приведенных исходных данных по формуле:



где tβ– время β-го этапа разработки проектного решения, дн.

Таблица 9 - Затраты времени на создание программного продукта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Этап создания | | Затраты времени  (в днях) | Затраты времени (в часах) | Машинное время работы над ПП  (в часах) |
| Разработка | Обследование объекта  Автоматизации | 3 | 24 | 24 |
| Анализ и уточнение требований | 2 | 16 | 16 |
| Разработка технического задания | 4 | 32 | 32 |
| Проектирование структуры | 10 | 80 | 80 |
|  | Программная реализация | 15 | 120 | 120 |
| Тестирование программного продукта | 3 | 24 | 24 |
| Отладка программного продукта | 3 | 24 | 24 |
| Разработка описания | 2 | 16 | 16 |
| ИТОГО | 42 | 336 | 336 |

Рабочий день принимается равным 8 часам.

Сумму оплаты труда разработчика за время работы над программным продуктом рассчитаем исходя из часовой тарифной ставки и фонда фактического времени, затраченного на разработку программного продукта по формуле:

Зотр=Ст1\*Фвр,

где Ст1 - часовой тарифной ставки (принимаем равной 90 руб./час);

Фвр – фонд фактического времени, затраченного на разработку программного продукта, час.

Начисления на заработную плату рассчитываются в таблице 10.

Таблица 10 – Начисление на заработную плату

|  |  |  |
| --- | --- | --- |
| Начисление на заработную плату | Процент,% | Сумма,руб. |
| Пенсионный фонд(ПФ):  -страховая часть  -накопительная часть | 16  6 | 9 676,8  3 628,8 |
| Фонд социального страхования(ФСС) | 2,9 | 1 753,92 |
| Федеральный фонд обязательного медицинского страхования(ФФОМС) | 5,1 | 3 084,48 |
| Итого | 30 | 18 144 |

Затраты, связанные с использованием вычислительной и оргтехники (формула 6.4):

,

(6.4)

где ТМРПР – машинное время работы над программным продуктом, час;

kГ– коэффициент готовности ЭВМ,  ;

n – количество единиц техники, равно 1;

СМ-Ч – себестоимость машино-часа, СМ-Ч =10руб.

9 576

Затраты на электроэнергию рассчитываются по следующей формуле 6.5:

(6.5)

Е=W×t×T,

где W – мощность, потребляемая ПК, кВт/час;

t – время работы ПК, час;

T – тариф электроэнергии, руб.

Тариф 1,23 рубля за киловатт.

Амортизация ПК рассчитывается по следующей формуле 6.6:

(6.6)

где – первоначальная стоимость ПК, руб;

– процент амортизации в год.

Для разработки программного продукта не произведено затрат на специальные программные продукты. Отсюда следует, что ЗСПП = 0.

Затраты на хозяйственно-организационные нужды приводятся в таблице 11 и вычисляются по формуле 6.7:

,

(6.7)

где Цτ – цена τ-го товара, руб.;

Кτ – количество τ-го товара.

Таблица 11 – Затраты на хозяйственно-организационные нужды

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу (руб.) | Кол-во (шт.) | Всего (руб.) |
| Диск CD-RW | 80 | 1 | 60 |
| Упаковка Бумага | 500 | 1 | 500 |
| ручка | 20 | 1 | 20 |
| Итого | | | 580 |

Результаты выполненных расчетов сводятся в общей таблице (таблица 12).

Таблица 12 – Затраты на разработку

|  |  |  |
| --- | --- | --- |
| Наименование затрат | Условное обозначение | Значение |
| Оплата труда разработчика программного продукта |  | 60 480 |
| Затраты, связанные с эксплуатацией техники |  | 9 576 |
| Затраты на электроэнергию | Е | 289,29 |
| Затраты на амортизацию ПК | А | 2 900 |
| Затраты на хозяйственно-операционные нужды |  | 580 |
| Затраты на специальные программные продукты | ЗСПП | 0 |
| Итого затрат на разработку |  | 73 765,29 |

Продолжение таблицы 12 – Затраты на разработку

6.2. Расчет затрат на внедрение программного продукта

6.3. Основные выводы

Итоговая стоимость затрат на разработку программного продукта, в которую включена оплата труда разработчика, затраты на эксплуатацию техники, в том числе и амортизации ПК, так де затраты на электроэнергию и затраты на хозяйственно-операционные нужды, составила 73 765,29рублей.

Из проведенных расчетов можно сделать вывод, что разработка информационной системы для магазина является не дорогостоящим процессом.

ЗАКЛЮЧЕНИЕ

Сфера периодического издания, не из самых популярных и распространенных. Очень много создается различных журналов и книг, так как данный вид деятельности повышает культуру в районе не только для детей, но также и взрослых.

Данная информационная система предназначена для компании, поэтому вход в систему выполняется с стороны: администратора. Для пользователя данной системы предусмотрен свой набор привилегий. Администратор может переходить в панель администратора. Обычный сотрудник имеет минимальный доступ, самое главное для них, это регистрация продажи журнала.

Интерфейс для пользователей создавался удобным и практичным, чтобы не возникало сложностей при пользовании сайтом. При возникновении каких-либо вопросов, всегда можно обратиться к администратору, который все подробно объяснит. Для сотрудников предусмотрен только просмотр информации.

Права администратора позволяют управлять информацией, выложенной на сайте, то есть создавать посты и сотрудников. Также в права администратора входит управление данными: обновление и удаление данных. Главная обязанность администратора – помогать пользователям увидеть информацию о журнале.

Информационная система «Сибирячок» поможет клиентам не тратить много времени на посещение редакции для просмотра журнала, и даст понять, кто работает в редакции и какие журналы он может купить.

Целью данного дипломного проекта было создание информационной системы «Сибирячок». В результате работы была создана информационная система с использованием языка разметки HTML, Php–фреймворка «Laravel»с использованием формального языка описания внешнего вида документа CSS. Для реализации цели были выполнены следующие задачи:

1. изучение предметной области;

2. анализ и обоснование выбора аппаратных средств;

3. анализ и обоснование выбора программных средств;

4. анализ и обоснование выбора языка программирования;

5. анализ и обоснование выбора среды разработки;

6. проектирование ИС;

7. проектирование базы данных;

8. проектирование интерфейса;

9. разработка интерфейса;

10. разработка базы данных;

11. разработка ИС;

12. выявить системно-технические требования;

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Drach: Сравнение современных СУБД. – URL: <https://drach.pro/blog/hi-tech/item/145-db-comparison> (дата обращения: 08.10.2023). – Текст: электронный.
2. Справочник от Автор24: Сфера строительства. – URL: <https://spravochnick.ru/arhitektura_i_stroitelstvo/razvitie_stroitelstva> (дата обращения: 09.02.23). – Текст: электронный.
3. Timeweb: лучшие IDE. – URL: https://timeweb.com/ru/community/articles/5-luchshih-ide-1 (дата обращения: 12.10.2022). – Текст: электронный.
4. 1 Laravel: Документация Laravel. – URL: [https://](https://www.codexlaw.ru)laravel.com (дата обращения: 09.03.23). – Текст: электронный.
5. Azone-it: Защита баз данных. – URL: <https://www.azone-it.ru/zashchita-baz-dannyh> (дата обращения 16.03.23). – Текст: электронный.
6. Model Querying – Basics – Sequelize // Simple SELECT queries – URL: https://sequelize.org/docs/v6/core-concepts/model-querying-basics/ (дата обращения 16.03.2023) – Текст электронный
7. Википедия: Определения диаграмм. – URL: https://ru.wikipedia.org/wiki (дата обращения: 25.03.2023). – Текст: электронный.
8. Skysmart: Сравнение популярных языков программирования. – URL: https://skysmart.ru (дата обращения: 07.03.2023). – Текст: электронный.
9. Timeweb: лучшие IDE. – URL: https://timeweb.com/ru/community/articles/5-luchshih-ide-1 (дата обращения: 12.03.2023). – Текст: электронный.
10. Типы данных SQL, MySQL, Oracle, Access, Microsoft SQL Server, PostgreSQL – URL: https://sql-language.ru/ (дата обращения: 12.03.2023). – Текст: электронный.
11. HTML Academy: Понятие и особенности языка PHP. – URL: https://htmlacademy.ru (дата обращения: 05.03.2023). – Текст: электронный.
12. htmlbook.ru: Тег <select>. – URL: http://htmlbook.ru/html/select (дата обращения: 20.03.2023). – Текст: электронный.
13. Axios – Введение | Axios Docs //запросы GET, POST, DELETE, PATCH– URL: https://axios-http.com/ru/docs/intro (дата обращения: 12.03.2023) – Текст электронный.
14. Харб – Оценка стоимости разработки программного продукта. – URL: <https://habr.com/ru/post/713998> (дата обращения: 28.03.2023). – Текст: электронный.

# **ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Министерство образования Иркутской области

Государственное бюджетное профессиональное образовательное учреждение

Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
|  |  |

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

ВЕБ-ПРИЛОЖЕНИЕ «УЧЁТ ЖУРНАЛОВ «СИБИРЯЧОК»»

|  |  |  |
| --- | --- | --- |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (П.Н. Чернигов) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Д.Ю. Алёнин) |

Иркутск 2023

**СОДЕРЖАНИЕ**

[1 Общие сведения 3](#_Toc54261854)

[2 Назначение и цели создания системы 3](#_Toc54261855)

[3 Требования к системе в целом 3](#_Toc54261856)

[3.1 Требования к структуре и функционированию системы 3](#_Toc54261857)

[3.2 Требования к надежности 4](#_Toc54261858)

[3.3 Требования к безопасности 4](#_Toc54261859)

[3.4 Требования к эксплуатации, техническому обслуживанию, ремонту и 4](#_Toc54261860)

[хранению компонентов системы 4](#_Toc54261861)

[4 Требования к документированию 4](#_Toc54261862)

[5 Состав и содержание работ по созданию системы 5](#_Toc54261863)

**1 Общие сведения**

Наименование работы: веб-приложение «Учёт журналов «Сибирячок»»

Исполнитель: студент иркутского авиационного техникума, группы ПКС-19-1, Алёнин Д.Ю..

Разработка информационной системы проходит в рамках дипломного проектирования.

Сроки разработки информационной системы с 06.04.2023 по 01.06.2023 года.

**2 Назначение и цели создания системы**

Данное веб-приложение разрабатывается для облегчения учёта продукции редакции журнала «Сибирячок».

**3 Требования к системе в целом**

**3.1 Требования к структуре и функционированию системы**

Функции информационной системы:

1. Раздел «Авторизация»:
   * авторизация пользователей.
2. Раздел «Регистрация»:
   * регистрация пользователей.
3. Раздел «…»:

**3.2 Требования к надежности**

Для обеспечения надежности необходимо проверять корректность получаемых данных и реализовать валидность полей. Входные данные поступают в виде значений c клавиатуры. Эти значения отображаются в отдельных полях таблицы.

**3.3 Требования к безопасности**

Для обеспечения безопасности в веб-приложении, необходимо реализовать разграничение прав доступа.

**3.4 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы**

Минимальные системные требования для сервера:

1. Процессор: Intel Pentium 4 2.0Ghz / AMD XP 2200+;
2. Оперативная память: 4 ГБ;
3. Жёсткий диск: 150ГБ;
4. Операционная система: Windows 7/8/10;
5. Версия MySQL 5.0 и выше;

Минимальные системные требования для рабочей станции:

1. Процессор: Intel Pentium 4 2.0Ghz / AMD XP 2200+;
2. Оперативная память: 4Гб;
3. Жёсткий диск: 150Гб;
4. Операционная система: Windows 7/8/10.

**4 Требования к документированию**

Основным документам, регламентирующими использование веб-приложение является руководство пользователя.

Основным документам, регламентирующими разработку информационной системы является техническое задание.

**5 Состав и содержание работ по созданию системы**

В таблице 1 представлены плановые сроки начала и окончания работы по созданию информационной системы.

Таблица 1 – Этапы выполнения разработки.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 20% | 25% | 35% | 50% | 60% | 70% | 80% | 90% | 100% |
| Провести предпроектное исследование. | 23.04 |  |  |  |  |  |  |  |  |
| Составить техническое задание на разработку в соответствии с ГОСТ. |  | 27.04 |  |  |  |  |  |  |  |
| Провести проектирование программного продукта. |  |  | 04.05 |  |  |  |  |  |  |
| Реализовать программный продукт. |  |  |  | 18.05 |  |  |  |  |  |
| Выполнить тестирование и отладку программного обеспечения. |  |  |  |  | 20.05 |  |  |  |  |
| Представить внедрение и сопровождение программного продукта. |  |  |  |  |  | 25.05 |  |  |  |
| Разработать документы для программного продукта. |  |  |  |  |  |  | 28.05 |  |  |
| Расчитать стоимость разработки и внедрения программного продукта. |  |  |  |  |  |  |  | 30.05 |  |
| Составить пояснитиельную записку. |  |  |  |  |  |  |  |  | 01.06 |
|  | 23.04 | 27.04 | 04.05 | 18.05 | 20.05 | 25.05 | 28.05 | 30.05 | 01.06 |
| Подпись руководителя дипломного проекта |  |  |  |  |  |  |  |  |  |

## **ПРИЛОЖЕНИЕ Б – ЛИСТИНГ admin.blade.php**

@extends('layouts.main')

@section('Сибиряк', 'Page Title')

@section('header')

<!-- @parent -->

@section('content')

<div class="container-fluid mx-1 my-1">

<!-- лефтбар -->

<div class="row">

<div class="col">

<h5>Категории</h5>

@foreach($dismissfull as $post)

<x-Admin.-card-left-bg-start />

<p class="card-text">{{$post->id}}. {{$post->name}}</p>

<x-Admin.-endcard />

@endforeach

<h5>Продажи</h5>

@foreach($inners as $post)

<x-Admin.-card />

<h5 class="card-text">{{$post->id}}. {{$post->name}}</h5>

<p class="card-title">{{$post->date}}</p>

<p class="card-subtitle mb-2 text-body-secondary">Кол-во: {{$post->amount}} Итог: {{$post->amount \* 100}}</p>

<x-Admin.-endcard />

@endforeach

<h5>Мероприятия</h5>

@foreach($event as $post)

<x-Admin.-card />

<h5 class="card-text">{{$post->name}}</h5>

<p class="card-title">{{$post->date}}</p>

<x-Admin.-endcard />

@endforeach

</div>

<div class="col-9">

<!-- Таблица журналов -->

<x-Admin.-table-journals />

@foreach($journals as $post)

<tr class="text-center">

<td>{{$post->id}}</td>

<td>{{$post-> name}}</td>

<td>{{$post-> description}}</td>

<td>{{$post-> price}}</td>

<td>{{$post-> created\_at}}</td>

<td>{{$post-> updated\_at}}</td>

<!-- <td>{{$post->category\_id}}</td> -->

<td class="text-center">

<form class="btn " action="{{route('admin\_journal.delete', $post->id)}}" method="post">

@csrf

@method('DELETE')

<input type="submit" class="btn btn-outline-danger" value="Удалить">

</form>

<a href="{{route('admin.edit-journal', $post->id)}}">

<button class="btn btn-outline-primary">

Изменить

</button>

</a>

</td>

</tr>

@endforeach

<x-Admin.-table-journals-end />

<!-- Подписки -->

<x-Admin.-table-subscribs />

@foreach($subscribsfull as $post)

<tr class="text-center">

<td>{{$post->id}}</td>

<td>{{$post-> sab\_name}}</td>

<td>{{$post-> name}}</td>

<td>{{$post-> id\_workers}}</td>

<td>{{$post-> date\_start}}</td>

<td>{{$post-> date\_end}}</td>

<td>{{$post-> adress}}</td>

<td>{{$post-> amount}}</td>

<td class="text-center">

<form class="btn " action="{{route('admin\_subscribs.delete', $post->id)}}" method="post">

@csrf

@method('DELETE')

<input type="submit" class="btn btn-outline-danger" value="Удалить">

</form>

<a href="{{route('admin.edit-subscribs', $post->id)}}">

<button class="btn btn-outline-primary">

Изменить

</button>

</a>

</td>

</tr>

@endforeach

<x-Admin.-table-subscribs-end />

<x-Admin.Crud />

<!-- Подписчики -->

<div class="row mx-2 my-4">

<h5>Подписчики</h5>

@foreach($subscriber as $post)

<x-Admin.-card />

<h5 class="card-title">{{$post->name}}</h5>

<h6 class="card-subtitle mb-2 text-body-secondary">Телефон: {{$post->phone}}</h6>

<p class="card-text">Почта: {{$post->email}}</p>

<p class="card-text">Адресс: {{$post->adress}}</p>

<form class="btn " action="{{route('admin\_subscriber.delete', $post->id)}}" method="post">

@csrf

@method('DELETE')

<input type="submit" class="btn btn-outline-danger" value="Удалить">

</form>

<a href="{{route('admin.edit-subscriber', $post->id)}}">

<button class="btn btn-outline-primary">

Изменить

</button>

</a>

<x-Admin.-endcard />

@endforeach

<div class="p-2 d-grid gap-2 my-3"> <button class="btn btn-outline-success" type="button" data-bs-toggle="modal" data-bs-target="#staticBackdropCreateSubscriber"> Добавить подписчика</button></div>

</div>

<!-- Списание -->

<div class="row mx-2 my-4">

<h5>Списание</h5>

@foreach($dismissfull as $post)

<x-Admin.-card />

<h5 class="card-title">{{$post->name}}</h5>

<h6 class="card-subtitle mb-2 text-body-secondary">Дата: {{$post->date}}</h6>

<p class="card-text">Описание: {{$post->note}}</p>

<p class="card-text">Приказ: {{$post->file}}</p>

<p class="card-text">Количество: {{$post->amount}}</p>

<form class="btn " action="{{route('admin\_dismissfull.delete', $post->id)}}" method="post">

@csrf

@method('DELETE')

<input type="submit" class="btn btn-outline-danger" value="Удалить">

</form>

<a href="{{route('admin.edit-dismissfull', $post->id)}}">

<button class="btn btn-outline-primary">

Изменить

</button>

</a>

<x-Admin.-endcard />

@endforeach

<div class="p-2 d-grid gap-2 my-3"> <button class="btn btn-outline-success" type="button" data-bs-toggle="modal" data-bs-target="#staticBackdropCreateDismissFull"> Добавить запись</button></div>

</div>

<!-- Журналы на складе -->

<div class="row mx-2 my-4">

<h5>Журналы на складе</h5>

@foreach($RoomJournals as $post)

<x-Admin.-card />

<h5 class="card-title">{{$post->name}}</h5>

<h6 class="card-subtitle mb-2 text-body-secondary">Журнал</h6>

<p class="card-text">Количество: {{$post->amount}}.</p>

<form class="btn " action="{{route('admin\_storerooms.delete', $post->id)}}" method="post">

@csrf

@method('DELETE')

<input type="submit" class="btn btn-outline-danger" value="Удалить">

</form>

<a href="{{route('admin.edit-storerooms', $post->id)}}">

<button class="btn btn-outline-primary">

Изменить

</button>

</a>

<x-Admin.-endcard />

@endforeach

<div class="p-2 d-grid gap-2 my-3"> <button class="btn btn-outline-success" type="button" data-bs-toggle="modal" data-bs-target="#staticBackdropCreateStoreRooms"> Добавить запись</button></div>

</div>

<!-- Журналы на мероприятие -->

<div class="row mx-2 my-4">

<h5>Журналы на мероприятия</h5>

@foreach($event\_journals\_full as $post)

<x-Admin.-card />

<h5 class="card-title">Мероприятие: {{$post->event\_name}}</h5>

<h6 class="card-subtitle mb-2 text-body-secondary">Журнал: {{$post->name}}</h6>

<p class="card-text">Количество: {{$post->amount}}.</p>

<p class="card-text">Дата: {{$post->date}}.</p>

<form class="btn " action="{{route('admin\_event\_journals\_full.delete', $post->id)}}" method="post">

@csrf

@method('DELETE')

<input type="submit" class="btn btn-outline-danger" value="Удалить">

</form>

<x-Admin.-endcard />

@endforeach

<div class="p-2 d-grid gap-2 my-3"> <button class="btn btn-outline-success" type="button" data-bs-toggle="modal" data-bs-target="#staticBackdropEventToJournals">Выдача журнала</button></div>

</div>

</div>

<div class="col my-4">

<x-Admin.-side-bar-right-btn />

<x-Admin.-modalcreateevents />

<!-- Modal создание данных о журнале на складе-->

<div class="modal fade" id="staticBackdropCreateStoreRooms" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="staticBackdropCreateStoreRoomsLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h1 class="modal-title fs-5" id="staticBackdropCreateStoreRoomsLabel">Журналы на складе</h1>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<form action="{{route('admin.store\_Storeroom')}}" method="POST">

@csrf

<div class="input-group mb-3">

<label class="input-group-text" for="inputGroupSelect01">Журнал</label>

<select class="form-select" name="id\_journal" id="id\_journal">

<option selected>Выбрать...</option>

@foreach ($journals as $row)

<option value='{{$row->id}}'>{{$row->name}}</option>

@endforeach

</select>

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Количество</span>

<input type="number" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="amount" id="amount">

</div>

</div>

<div class="modal-footer">

<button type="submit" class="btn btn-primary">Добавить</button>

</div>

</form>

</div>

</div>

</div>

<!-- ggggggggggggggggggggggggggggggggggggggggggggggggg -->

<!-- Modal создание данных о подписчике-->

<div class="modal fade" id="staticBackdropCreateSubscriber" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="staticBackdropCreateSubscriberLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h1 class="modal-title fs-5" id="staticBackdropCreateSubscriberLabel">Подписчки</h1>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<form action="{{route('admin.store\_subscriber')}}" method="POST">

@csrf

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">ФИО</span>

<input type="text" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="name" id="name">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Телефон</span>

<input type="tel" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="phone" id="phone">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Почта</span>

<input type="email" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="email" id="email">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Адресс</span>

<input type="text" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="adress" id="adress">

</div>

</div>

<div class="modal-footer">

<button type="submit" class="btn btn-primary">Добавить</button>

</div>

</form>

</div>

</div>

</div>

<!-- Modal создание данных списании-->

<div class="modal fade" id="staticBackdropCreateDismissFull" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="staticBackdropCreateDismissFullLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h1 class="modal-title fs-5" id="staticBackdropCreateDismissFullLabel">Журналы на складе</h1>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<form action="{{route('admin.store\_dismissfull')}}" method="POST">

@csrf

<div class="input-group mb-3">

<label class="input-group-text" for="inputGroupSelect01">Журнал</label>

<select class="form-select" name="id\_journal" id="id\_journal">

<option selected>Выбрать...</option>

@foreach ($journals as $row)

<option value='{{$row->id}}'>{{$row->name}}</option>

@endforeach

</select>

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Дата</span>

<input type="datetime-local" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="date" id="date">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Описание</span>

<input type="text" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="note" id="note">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Путь к файлу</span>

<input type="text" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="file" id="file">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Количество</span>

<input type="number" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="amount" id="amount">

</div>

</div>

<div class="modal-footer">

<button type="submit" class="btn btn-primary">Добавить</button>

</div>

</form>

</div>

</div>

</div>

<!-- Modal create subs -->

<div class="modal fade" id="staticBackdropCreateSubs" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="staticBackdropCreateSubsLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h1 class="modal-title fs-5" id="staticBackdropCreateSubsLabel">Добавление подписки</h1>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<form action="{{route('admin.store\_Subscrib')}}" method="POST">

@csrf

<div class="input-group mb-3">

<label class="input-group-text" for="inputGroupSelect01">Подписчик</label>

<select class="form-select" name="id\_subscribers" id="id\_subscribers">

<option selected>Выбрать...</option>

@foreach ($subscriber as $row)

<option value='{{$row->id}}'>{{$row->name}}</option>

@endforeach

</select>

</div>

<div class="input-group mb-3">

<label class="input-group-text" for="inputGroupSelect01">Категория</label>

<select class="form-select" name="id\_categories" id="id\_categories">

<option selected>Выбрать...</option>

@foreach ($categories as $row)

<option value='{{$row->id}}'>{{$row->name}}</option>

@endforeach

</select>

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Дата начала</span>

<input type="datetime-local" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="date\_start" id="date\_start">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Дата конца</span>

<input type="datetime-local" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="date\_end" id="date\_end">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Описание</span>

<input type="text" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="adress" id="adress">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Кол-во</span>

<input type="number" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="amount" id="amount">

</div>

</div>

<div class="modal-footer">

<button type="submit" class="btn btn-primary">Добавить</button>

</div>

</form>

</div>

</div>

</div>

<!-- Modal -->

<div class="modal fade" id="staticBackdropEventToJournals" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="staticBackdropEventToJournalsLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h1 class="modal-title fs-5" id="staticBackdropEventToJournalsLabel">Выдача журналов</h1>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<form action="{{route('event\_journal.storeEJ')}}" method="POST">

@csrf

<div class="input-group mb-3">

<label class="input-group-text" for="inputGroupSelect01">Журнал</label>

<select class="form-select" name="id\_journal" id="id\_journal">

<option selected>Выбрать...</option>

@foreach ($journals as $row)

<option value='{{$row->id}}'>{{$row->name}}</option>

@endforeach

</select>

</div>

<div class="input-group mb-3">

<label class="input-group-text" for="inputGroupSelect01">Мероприятие</label>

<select class="form-select" name="id\_event" id="id\_event">

<option selected>Выбрать...</option>

@foreach ($event as $row)

<option value='{{$row->id}}'>{{$row->name}} | {{$row->date}}</option>

@endforeach

</select>

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Количество</span>

<input type="number" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="amount" id="amount">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Выдан ли журнал?</span>

<!-- <input type="text" name="name" id="name"> -->

<input type="number" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="is\_okay" id="is\_okay" min="0" max="1" value="0">

</div>

</div>

<div class="modal-footer">

<button type="submit" class="btn btn-primary">Добавить</button>

</div>

</form>

</div>

</div>

</div>

<!-- Modal -->

<div class="modal fade" id="staticBackdropSoldsJournals" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="staticBackdropSoldsJournalsLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h1 class="modal-title fs-5" id="staticBackdropSoldsJournalsLabel">Продажа журнала</h1>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<form action="{{route('sold.storeS')}}" method="POST">

@csrf

<div class="input-group mb-3">

<label class="input-group-text" for="inputGroupSelect01">Журнал</label>

<select class="form-select" name="id\_journal" id="id\_journal">

<option selected>Выбрать...</option>

@foreach ($journals as $row)

<option value='{{$row->id}}'>{{$row->name}}</option>

@endforeach

</select>

</div>

<div class="input-group mb-3">

<label class="input-group-text" for="inputGroupSelect01">Подписка</label>

<select class="form-select" name="id\_subsc" id="id\_subsc">

<option selected value="">Обычная покупка или Выбрать...</option>

@foreach ($subscribs as $row)

<option value='{{$row->id}}'>{{$row->id\_subscribers}} | {{$row->adress}}</option>

@endforeach

</select>

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Количество</span>

<input type="number" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="amount" id="amount">

</div>

<div class="input-group mb-3">

<span class="input-group-text" id="inputGroup-sizing-default">Дата и время</span>

<input type="text" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default" name="date" id="date" value="<?php echo date("Y-m-d H:i:s"); ?>">

</div>

<!-- </div>datetime-local -->

<div class="modal-footer">

<button type="submit" class="btn btn-primary">Добавить</button>

</div>

</form>

</div>

</div>

</div>

</div>

<x-Admin.-btnexport />

</div>

</div>

</div>

</div>

@endsection

## **ПРИЛОЖЕНИЕ В – ЛИСТИНГ ChartController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use LaravelDaily\LaravelCharts\Classes\LaravelChart;

class ChartController extends Controller

{

public function index()

{

$chart\_options = [

'chart\_title' => 'Склад',

'report\_type' => 'group\_by\_string',

'model' => 'App\Models\Storeroom',

'group\_by\_field' => 'amount',

'chart\_type' => 'pie',

'filter\_field' => 'id',

// 'filter\_period' => 'month', // show users only registered this month

];

$chart2 = new LaravelChart($chart\_options);

return view('storeroom', compact('chart2'));

}

}