**Министерство образования Иркутской области**

Государственное бюджетное профессиональное образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
| ДП.09.02.03.21.173.06.ПЗ | УТВЕРЖДАЮ  Зам. директора по УР, к.т.н.  \_\_\_\_\_\_\_\_\_\_\_Е.А. Коробкова |

ИНФОРМАЦИОННАЯ СИСТЕМА «АВТОСЕРВИС»

|  |  |  |
| --- | --- | --- |
| Нормоконтролер: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (О.Ю. Безносова) |
| Консультант по экономической части: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.А. Белова) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (С.А. Удальцов) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (В.И. Корнецкий) |

Иркутск 2021

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc71239671)

[1 Предпроектное исследование 5](#_Toc71239672)

[1.1 Обзор аналогов программного обеспечения 5](#_Toc71239673)

[1.2 Анализ программных продуктов 9](#_Toc71239674)

[1.3 Обоснование выбора программного продукта 11](#_Toc71239675)

[2 Проектирование 13](#_Toc71239676)

[2.1 Архитектура программного обеспечения 13](#_Toc71239677)

[2.2 Функциональное проектирование 14](#_Toc71239678)

[2.3 Проектирование базы данных 17](#_Toc71239679)

[2.4 Проектирование пользовательского интерфейса 22](#_Toc71239680)

[3 Реализация программного обеспечения 26](#_Toc71239681)

[3.1 Кодирование программного обеспечения 26](#_Toc71239682)

[4 Внедрение и сопровождение программного обеспечения 32](#_Toc71239683)

[5 Документирование программного обеспечения 33](#_Toc71239684)

[5.1 Руководство по установке программного обеспечения 33](#_Toc71239685)

[5.2 Руководство пользователя программного обеспечения 33](#_Toc71239686)

[5.3 Руководство системного администратора программного обеспечения 43](#_Toc71239696)

[6 Стоимость разработки и внедрения программного продукта 45](#_Toc71239697)

[6.1 Организационно-экономическое обоснование проекта 45](#_Toc71239698)

[6.2 Расчет затрат на разработку программного продукта 45](#_Toc71239699)

[6.3 Расчет затрат на внедрение программного продукта 49](#_Toc71239700)

[6.4 Основные выводы 49](#_Toc71239701)

[ЗАКЛЮЧЕНИЕ 50](#_Toc71239702)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 51](#_Toc71239703)

[Приложение А - Листинг маршрутов 52](#_Toc71239704)

ВВЕДЕНИЕ

В современных условиях повышения эффективности управления информацией на предприятии является одной из ключевых задач, стоящих перед руководством. Предприятие, имеющее крупный документооборот, пытается найти решение, позволяющее упростить данную операцию.

Основной задачей, стоящих перед предприятием напрямую зависит от тех инструментов и возможностей, которые необходимы для создания и формирования отчетов. Данную функциональность в современном обществе обеспечивает информационная система, с помощью которой управление базами данных и построение отчетов становиться легче.

В связи с этим поставлена задача в разработке информационной системы. Система будет иметь доступ к базе данных, организует построение отчетов и упростит часть бухгалтерского учёта.

Цель дипломного проекта – разработать информационную системы «Автосервис». Цель разработки – упрощение процесса оказания услуг, накопление клиентской базы, введение учёта затрат на расходные материалы, составление заказ-наряда.

Предполагается использования системы на предприятиях, связанных с оказанием услуг в области автосервиса.

База данных должна быть спроектирована так, чтобы обеспечить хранение всех необходимых данных, имея при этом максимально упрощенную структуру. Структура базы данных должна обеспечивать целостность и корректность информации.

1. Предпроектное исследование
   1. Обзор аналогов программного обеспечения

Перед тем как приступить к разработке программного продукта, необходимо выполнить обзор существующих аналогов.

В результате обзора и анализа выявлены следующие аналоги будущей программной разработки соответствующей предметной области:

1. LogicStar-Avto – это платная многофункциональная система автоматизации предприятий сферы автобизнеса, специализирующихся в области продажи и обслуживании автотранспорта. Фрагмент пользовательского интерфейса LogicStar-Avto можно увидеть на рисунке 1.

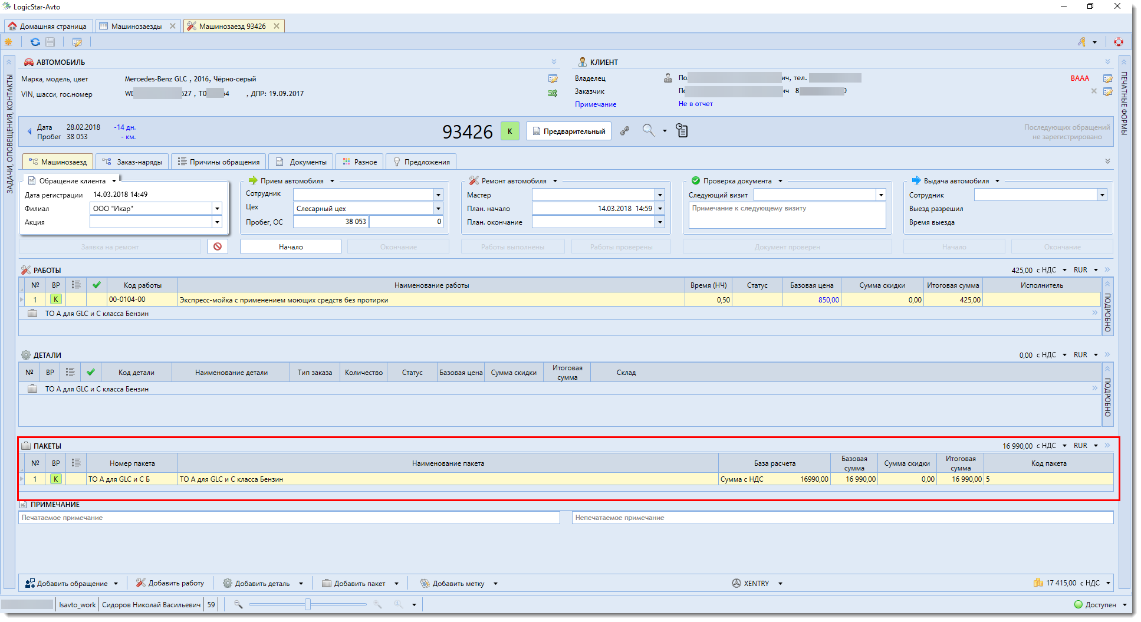


Рисунок 1 – LogicStar-Avto

LogicStar-Avto автоматизирует бизнес-процессы автоцентра в едином интегрированном решении и поддерживает:

* работу со многими брендами - для каждого из брендов могут настраиваться справочники, нормативы, шаблоны первичных документов, поддерживается отчетность;
* интеграцию с внешними системами подбора работ и деталей, Ewa Net, TIS (Opel), Suzuki EPC, Mitsubishi EPC, Iveco Power, Chevrolet Parts Imager, AUDATEX;
* интеграцию с системами автоматического отпуска ГСМ в цехе, к примеру, MDS2000 Fluid Management, позволяющая производить списание ГСМ на Заказ-наряды в реальном времени;
* внешний доступ к информации через различные варианты WEB- интерфейсов как для удаленного просмотра менеджерами результатов деятельности компании, так и предоставление информации для клиентов (о текущем наличии деталей, предварительной записи, ведении и истории ремонта, обязательных сервисных мероприятиях).

LogicStar-Avto можно считать отличным программным продуктом по реализованному функционалу, но его стоимость и техническое обслуживание отталкивает некоторых потенциальных потребителей. Сравнительная таблица версий LogicStar-Avto демонстрируется на рисунке 2.



Рисунок 2 – Сравнительная таблица LogicStar-Avto

1. 1C: Альфа-Авто — платный программный комплекс, разработанный на платформе «1С:Предприятие 8» обеспечивает комплексную поддержку всех бизнес-процессов в тех-центрах, руководство может оперативно получать и использовать данные о различных сторонах деятельности компании. Фрагмент пользовательского интерфейса 1C: Альфа-Авто на рисунке 3.

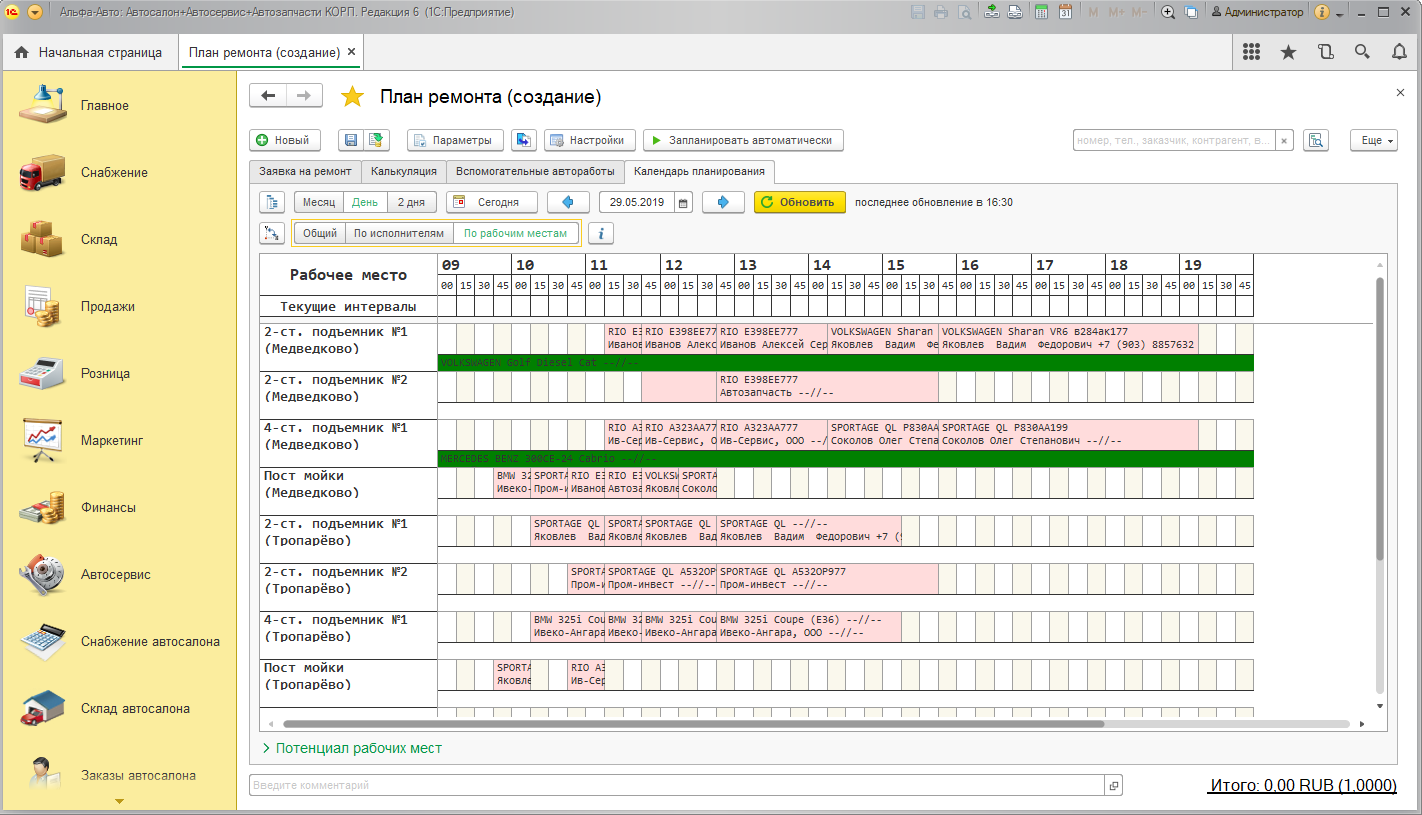


Рисунок 3 – 1C: Альфа-Авто

Основные достоинства:

* повышение эффективности управления автосервисом: отражение всей деятельности автосервиса в едином информационном пространстве - от записи клиента до анализа рентабельности бизнеса;
* встроенные функции управления взаимоотношениями с клиентами (CRM);
* управление автосервисом: планирование загрузки, учет выполненных ремонтных работ, анализ эффективности работы.

1C: Альфа-Авто подойдёт предприятиям, которые используют 1С: Предприятие, как средство документооборота. Также 1С: Альфа-Авто имеет высокую стоимость равную 96000 рублей.

1. TurboService - гибкая система, четко настраиваемая под схему работы предприятия. Обеспечивает единое информационное пространство для всех специалистов автоцентра.

Фрагмент пользовательского интерфейса TurboService на рисунке 4.

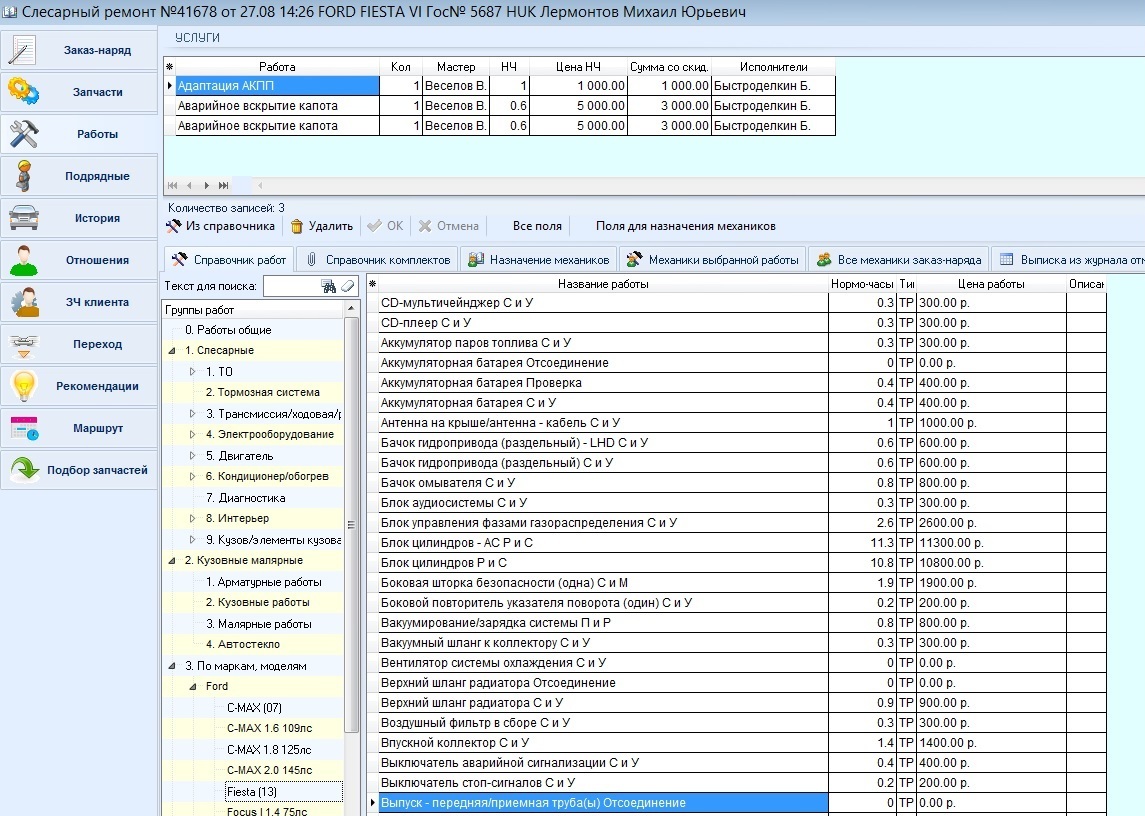


Рисунок 4 – TurboService

Система обеспечивает проведение всех необходимых для комплексной автоматизации производства операций. Вот лишь некоторые из ее возможностей:

* полноценный складской учет;
* оформление и вывод документов - заказ-нарядов, счетов, накладных, актов приемки-сдачи работ и т.д.;
* учет выполнения работ и выработки механиков;
* поддержка справочников работ и автомобилей;
* многоскладность, многофирменность, многовалютность;
* различные методы списания в холдинге;
* перенос данных в популярные бухгалтерские системы;
* формирование и печать разносторонних отчетов о деятельности предприятия.

TurboService отличное решение для бизнеса и поставляется в множестве решений средняя стоимость, которых 5000 рублей в месяц.

Проанализировав аналоги можно прийти к выводу, что в разрабатываемой информационной системе необходимо реализовать накопление справочных данных.

* 1. Анализ программных продуктов

Информационная система «Автосервис» будет реализована при помощи веб-технологий, так как это позволит использовать информационную систему без установки какого-либо дополнительного программного обеспечения.

Для работы с БД рассмотрены следующие программные средства:

1. MySQL;
2. SQLite;
3. PostgreSQL.

Для создания СУБД рассмотрены следующие программные средства:

1. PHPMyAdmin;
2. Adminer;
3. MariaDB.

Для разработки информационной системы рассмотрены следующие программные средства:

1. PHPStorm
2. Notepad++
3. WebStorm

MySQL – свободная реляционная система управления базами данных для сервера SQL. Это простая, эффективная и надежная система управления информацией, использующая традиционный язык SQL. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей.

SQLite - система управления базами данных. SQLite является решением для малых и средних приложений. Обычно SQLite используется в качестве сервера в автономных программах.

PostgreSQL – свободная объектно-реляционная система управления базами данных. Существует в реализациях для множества UNIX-подобных платформ, Linux, Mac OS X, Solaris/OpenSolaris, Tru64, QNX, и даже для Microsoft Windows. PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта языка программирования SQL.

PostgreSQL считаются высокопроизводительной и надёжной СУБД. В ней имеются механизмы транзакций и репликация, а также легкая расширяемость программы позволяет импортировать различные языки программирования запросов.

PHPMyAdmin – веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. PHPMyAdmin позволяет через браузер, осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных. Приложение пользуется большой популярностью у веб-разработчиков, так как позволяет управлять СУБД MySQL без непосредственного ввода SQL команд, предоставляя дружественный интерфейс.

Adminer (бывший phpMinAdmin) – это легковесный инструмент администрирования MySQL, PostgreSQL, SQLite, MS SQL и Oracle. Проект родился как «облегчённый» вариант phpMyAdmin. Распространяется на условиях Apache license в форме одиночного PHP-файла размером около 380 килобайт, который является результатом компиляции исходных php- и js-файлов с помощью специального PHP-скрипта.

MariaDB – ответвление от системы управления базами данных MySQL. MariaDB намерен поддерживать высокую совместимость с MySQL, обеспечивая точное соответствие с API и командами MySQL.

PhpStorm – это интегрированная среда разработки на PHP с интеллектуальным редактором, которая глубоко понимает код, поддерживает PHP 5.3-7.2 для современных и классических проектов, обеспечивает лучшее в индустрии автодополнение кода, предотвращение ошибок налету и поддерживает смешивание языков.

Notepad++ – свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса большого количества языков программирования и разметки.

JetBrains WebStorm – интегрированная среда разработки на JavaScript, CSS & HTML от компании JetBrains.

WebStorm обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями.

* 1. Обоснование выбора программного продукта

Для разработки модуля ИС было решено использовать PHPStorm, PHPMyAdmin и MySQL.

Основные преимущества PHPStorm:

1. автодополнение кода;
2. предотвращение ошибок;
3. выполнение запросов в среде разработки;
4. глубокий анализ кода JavaScript, его популярных библиотек и фреймворков, CSS, HTML, PHP и его популярные библиотеки и фреймворки.

Основные преимущества PHPMyAdmin:

1. простой интерфейс;
2. выполнение всех функций манипуляций с таблицами;
3. настройка прав доступа, конфигураций серверов.

Основные преимущества MySQL:

1. простота;
2. эффективность;
3. быстрота;
4. безопасность;
5. гибкость.

Таким образом, для реализации цели дипломного проекта было решено использовать следующие средства разработки:

* на этапе проектирования, для создания ER-модели использовалось CASE средства – draw io;
* для создания структурных схем использовалось CASE средства – Microsoft Visio;
* для создания контекстной и диаграмм декомпозиции использовался графический редактор – Microsoft Visio;
* на этапе разработки информационной системы, использовались WEB-технологии – HTML, PHP, JavaScript, CSS, JQuery, Bootstrap;
* для работы с базой данных использовался метод языка программирования PHP - PDO.

1. Проектирование

Перед проектированием информационной системы было разработано техническое задание, на основе которого производилось проектирование и информационной системы «Автосервис». Техническое задание приложено к пояснительной записке.

* 1. Архитектура программного обеспечения

Архитектура программного обеспечения – совокупность важнейших решений об организации программной системы. Архитектура включает:

1. выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
2. соединение выбранных элементов структуры и поведения во всё более крупные системы;
3. архитектурный стиль, который направляет всю организацию – все элементы, их интерфейсы, их сотрудничество и их соединение.

Информационная система использует клиент-серверную архитектуру, что говорит о том, что все действия на клиентской части пользователя над данными обрабатываются сервером на серверной части.

Клиент-серверная архитектура позволяет значительно снизить нагрузку у пользователя, что позволяет запускать приложение практически на любом устройстве.

На рисунке 5 демонстрируется архитектура веб-приложений, которая соответствует разрабатываемой информационной системе.

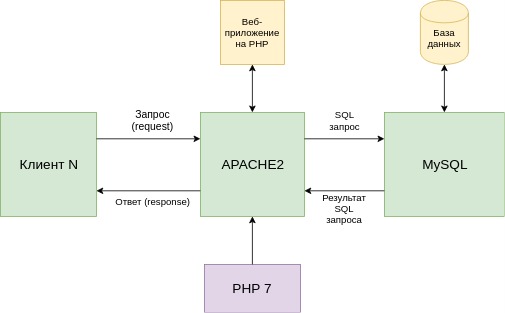


Рисунок 5 – Клиент-серверная архитектура веб-приложения

Браузер клиента отправляет запрос на подключение к странице веб-сервер. Веб-сервер отправляет запрос в веб-приложение, которое запрашивает обработку PHP. PHP может запросить подключение к MySQL, на что Apache отправляет запрос к MySQL, который по запросу берет данные из БД и отправляет обратно на веб-сервер, после чего PHP продолжает обработки. Когда обработка закончит, Apache генерирует необходимую URL и отправляет ответ клиенту, где браузер выполняет скрипты.

* 1. Функциональное проектирование

Функциональное проектирование – это подход к проектированию, который нацелен, прежде всего, на создание эффективно работающего объекта. Функциональное проектирование позволяет раскрыть все функции системы.

Выполнение требуемой функции – главная цель и основа разработки объекта. Во внимание принимаются, прежде всего, функциональные показатели качества и показатели надёжности.

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

На контекстной диаграмме на рисунке 6 показаны входные данные, управление механизм, выходные данные и функция.

В центре диаграммы показана информационная система «Автосервис». Левая стрелка это входные данные, т.е. «Данные с веб-страниц» и «Данные из БД». Верхняя стрелка — это управление «Алгоритм обработчика событий». Нижняя стрелка – это механизм «Пользователь». А правая - выходные данные «Обработанные данные».



Рисунок 6 – Контекстная диаграмма модуля

Для демонстрации деталей в процессе необходимо спроектировать диаграмму декомпозиции.

Диаграмма декомпозиции – это разбиение функции, т.е. информационной системы «Автосервис».

На рисунке 7 показана диаграмма декомпозиции, которая расписывает работу информационной система из контекстной диаграммы. На этой диаграмме показаны 11 функций — это «Аутентификация», «Авторизация», «Заявка», «Изменение БД», «Печать», «Манипулирование данными клиентов», «Манипулирование данными сотрудников», «Манипулирование данными материалов», «Манипулирование данными ролей», «Вывод данных», «Манипулирование данными пользователей». А также на этой диаграмме присутствуют входные данные, выходные данные, управления и механизмы.

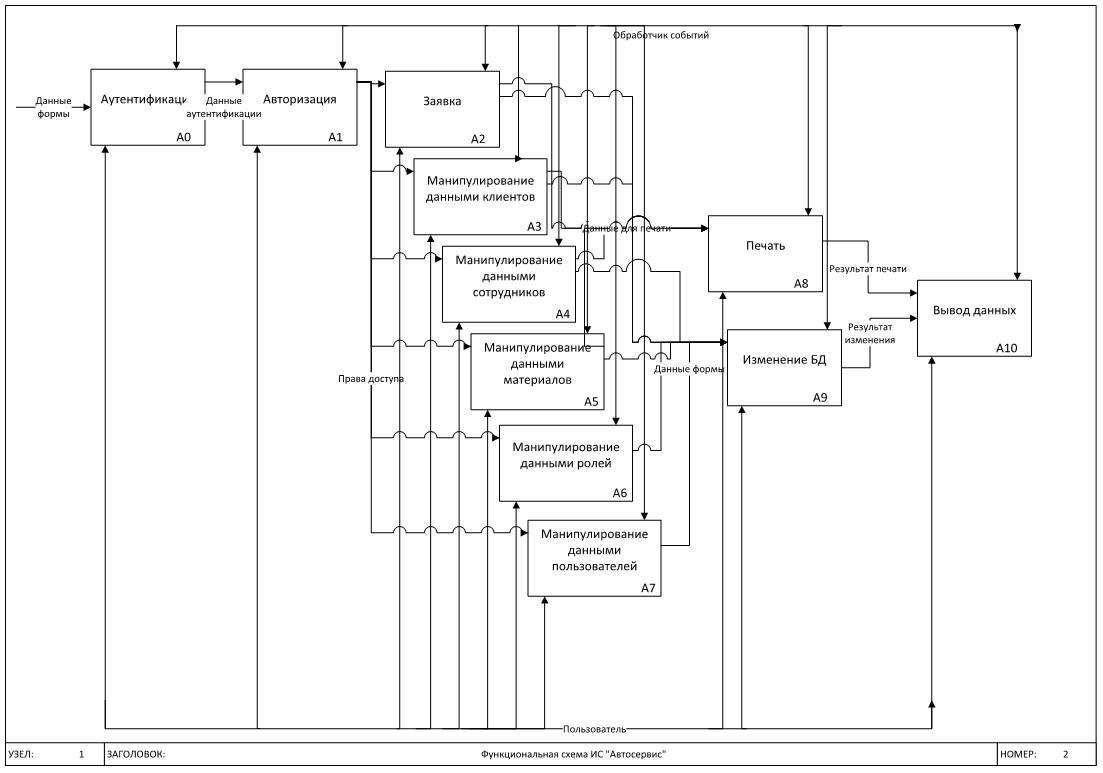


Рисунок 7 – Диаграмма декомпозиций ИС

На рисунке 8 показана диаграмма IDEF3, которая расписывает работу информационной системы из рисунка 6.

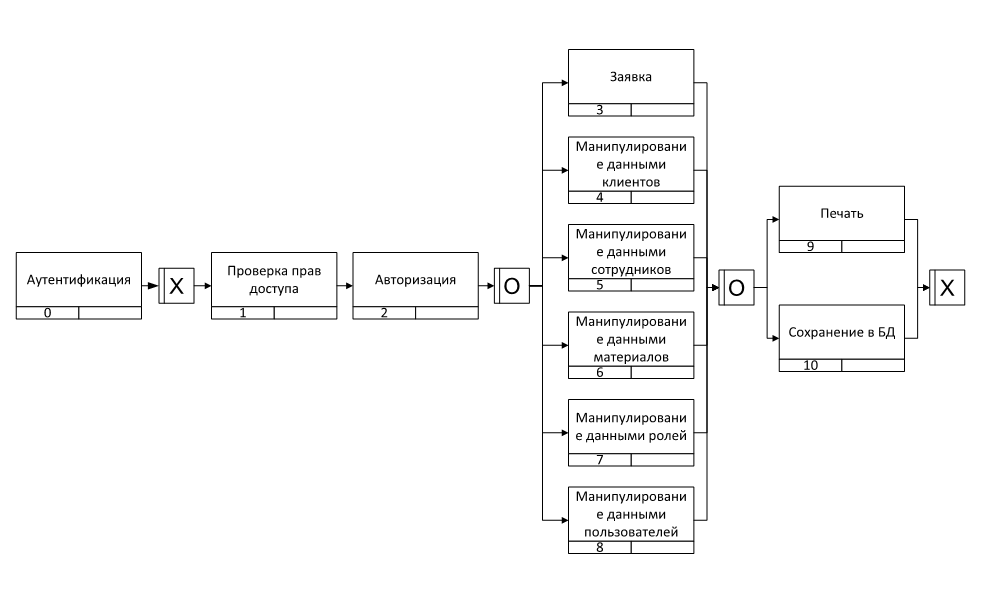


Рисунок 8 –IDEF3 диаграмма

На рисунке 9 показана диаграмма потоков данных, которая описывает передачу данных между разными функция и объектами информационной системы.

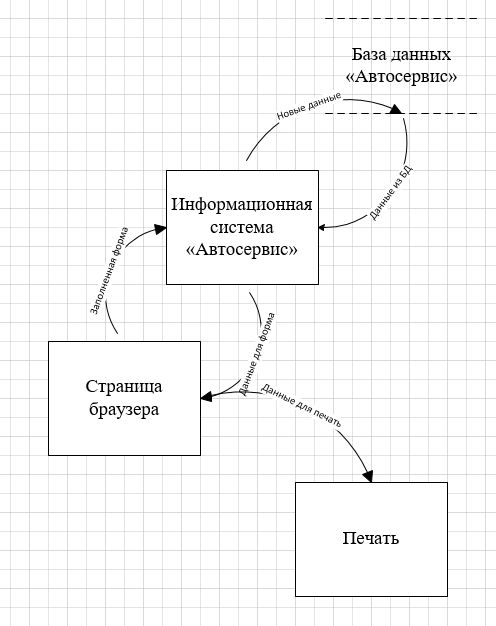


Рисунок 9 – Диаграмма потоков данных

Подробное функциональное проектирование позволяет глубже исследовать проектную область и цель проекта, что может позволить сэкономить время на разработке информационной системы, а также позволит избежать перенаписание кода.

* 1. Проектирование базы данных

Проектирование информационной системы происходит при помощи CASE средств, которые позволяют быстро создавать схемы и реляционные модели программ.

Модель сущность-связь или схема баз данных – модель данных, позволяющая описывать концептуальные схемы предметной области.

На рисунке 10 изображена схема базы данных информационной системы «Автосервис».

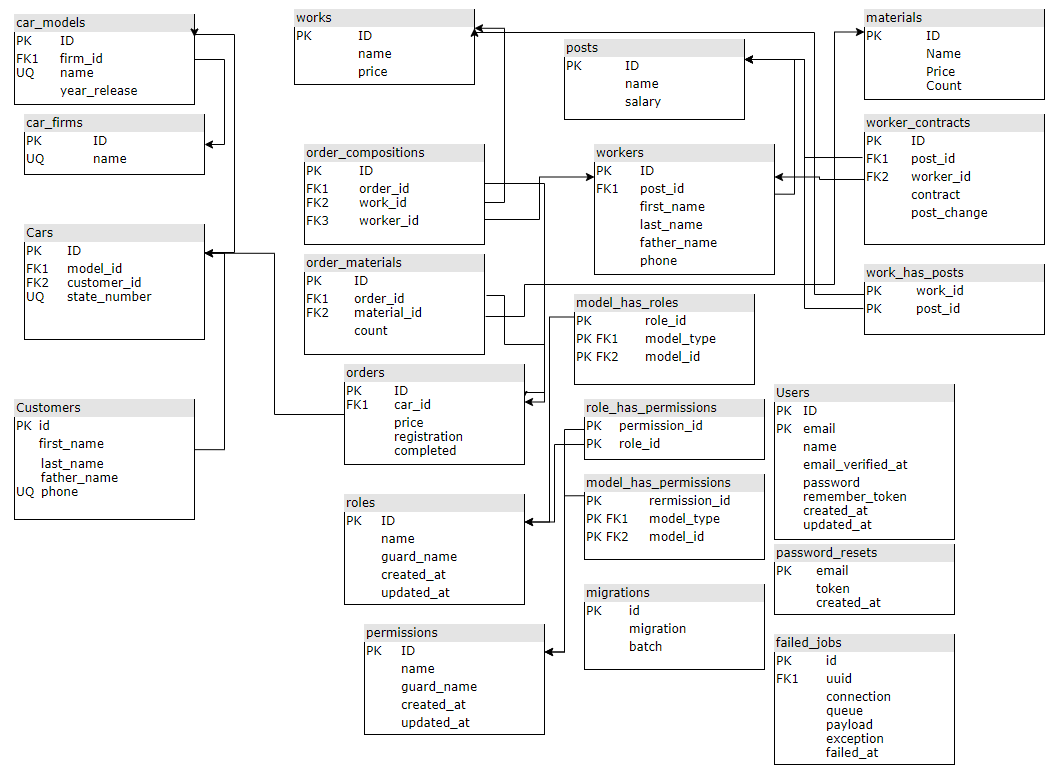


Рисунок 10 – Схема базы данных

База данных представляет собой 22 связанных таблиц. Для создания связей между таблицами используются внешние ключи.

Разработка ER-модели выполнена с помощью программы case-средств draw io. Были выявлены следующие сущности и их атрибуты:

1. Автомобили:

* модель автомобиля;
* владелец автомобиля;
* государственный регистрационный номер автомобиля.

1. Модели автомобилей:

* фирма автомобиля;
* год выпуска автомобиля;
* название модели автомобиля.

1. Фирмы автомобилей:

* название фирмы.

1. Клиенты:

* номер телефона клиента;
* имя клиента;
* фамилия клиента;
* отчество клиента.

1. Виды работ:

* наименование работы;
* стоимость работы.

1. Должность:

* наименование должности;
* оклад должности.

1. Должностные работы:

* должность;
* работа.

1. Сотрудники:

* должность сотрудника;
* номер телефона сотрудника;
* имя сотрудника;
* фамилия сотрудника;
* отчество сотрудника.

1. Документы, подтверждающие смену должности:

* новая должность;
* сотрудник;
* путь до документа;
* сменяемая должность.

1. Состав заказа:

* номер заказа;
* работа, выполняемая в заказе;
* исполнитель работы в заказе.

1. Заказы:

* автомобиль;
* стоимость заказа;
* дата принятия заказа;
* дата окончания заказа.

1. Расходные материалы заказа:

* номер заказа;
* материал;
* количество материала.

1. Пользователи:

* никнейм;
* почта;
* пароль;
* верификация почты.

1. Восстановление пароля:

* почта;
* дата изменения.

1. Миграции:

* миграция;
* количество.

1. Ошибки:

* соединение;
* очередь;
* дата ошибки;
* нагрузка;
* исключения.

1. Роли пользователя:

* название роли;
* дата создания роли.

1. Права доступа:

* название права;
* дата создания права.

1. Модели ролей:

* роль;
* тип модели.

1. Роли прав доступа:

* право доступа;
* роль.

1. Модели прав доступа:

* право доступа;
* тип модели.

1. Материалы:

* наименование материала;
* стоимость материала;
* количество материала.

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Третья нормальная форма, которую придерживается база данных, содержит в себе следующее требование:

– Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Все таблицы содержат минимальный набор независимых данных. Большинство таблиц не содержат больше 5 столбцов.

Исходя из всего выше сказанного, уровень нормализации базы данных равен 3.

* 1. Проектирование пользовательского интерфейса

Интерфейс информационной системы – это одна из важнейших частей проектирования программного обеспечения.

Интерфейсом информационной системы «Автосервис» является меню навигации, содержащее разделы информационной системы, панели управления и рабочей области.

На рисунке 11 показан прототип раздела «Вход», который контролирует вход только зарегистрированных пользователей.

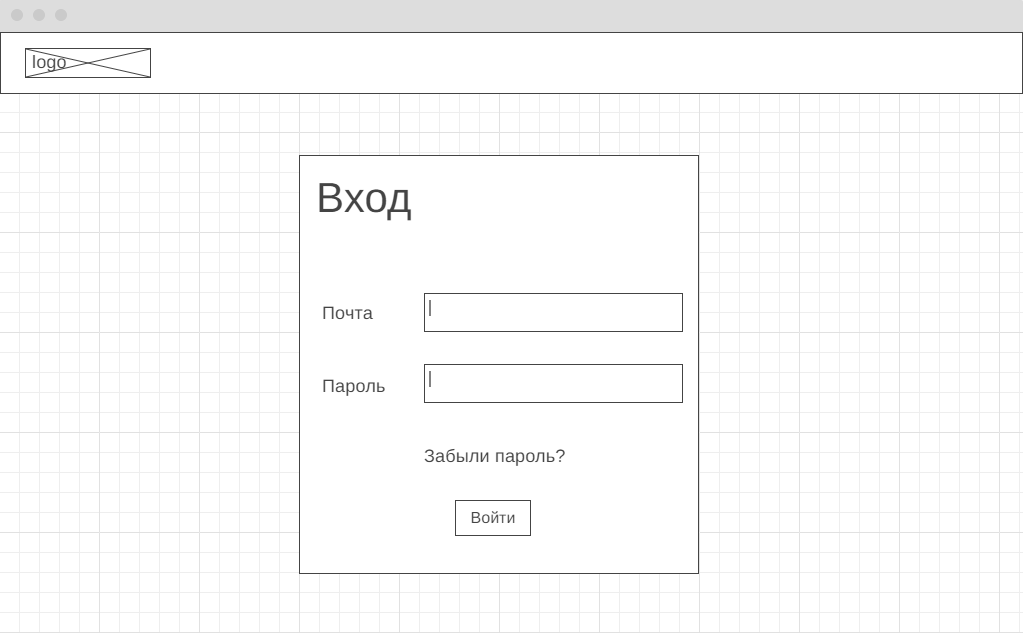


Рисунок 11 – Прототип раздела «Вход»

На рисунке 12 демонстрируется прототип основной страницы «Заказы», на которой позволяет составить заказ и распечатать заказ-наряд по заполненной форме.

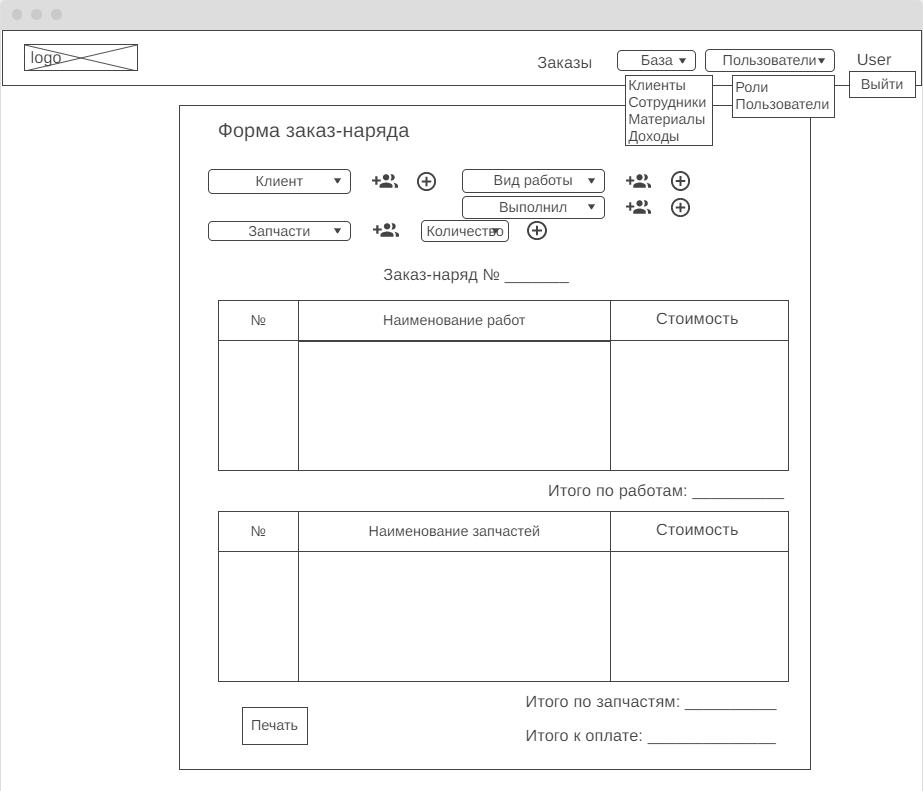


Рисунок 12 – Прототип страницы «Заказы»

На рисунке 13 демонстрируется прототип страницы «Клиенты». На рабочем пространстве размещена таблица с данными базы о клиентах, которые может отредактировать и сохранить, только пользователь с достаточными правами доступа или соответствующей ролью. Также демонстрируется модальное окно, которое позволяет изменить данные автомобилей клиента.

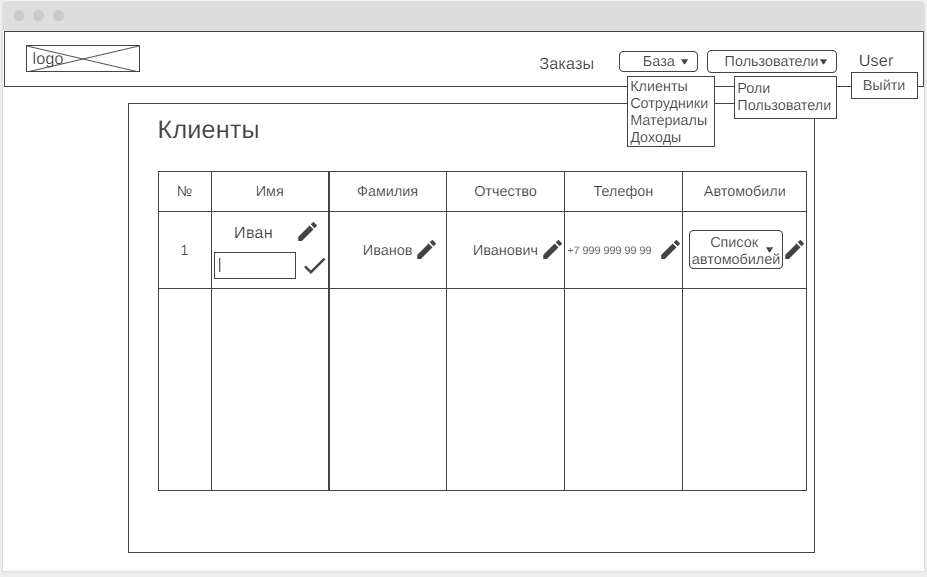


Рисунок 13 – Прототип страницы «Клиенты»

На рисунке 14 демонстрируется прототип страницы «Сотрудники». На рабочем пространстве размещена таблица с данными о сотрудниках, в которой имеется возможность отредактировать данные сотрудника и сменить его должность при отправке соответствующего документа.

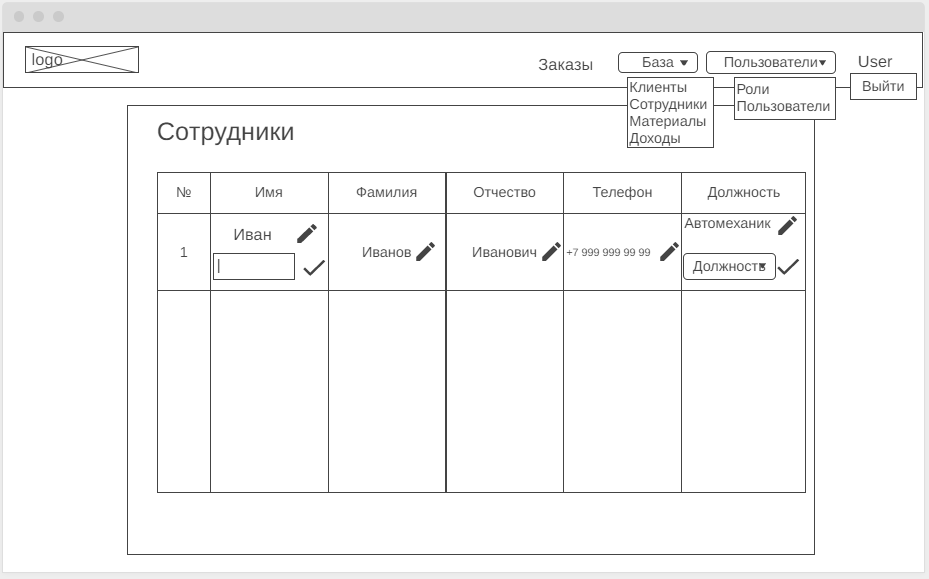


Рисунок 14 – Прототип страницы «Сотрудники»

На рисунке 15 демонстрируется прототип страницы «Регистрация», которая позволяет зарегистрировать нового пользователя системы пользователю с соответствующими правами.

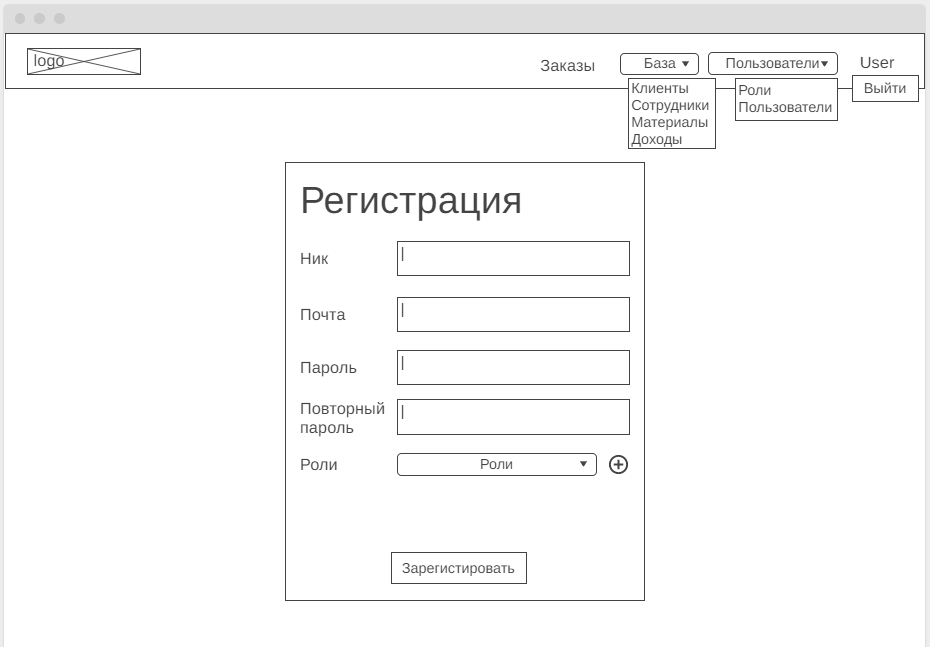


Рисунок 15 – Прототип страницы «Регистрация»

На рисунке 16 демонстрируется прототип страницы «Материалы». На рабочем пространстве размещена таблица с данными о расходных материалах, в которой имеется возможность отредактировать данные материалов, контролировать количество материалов.

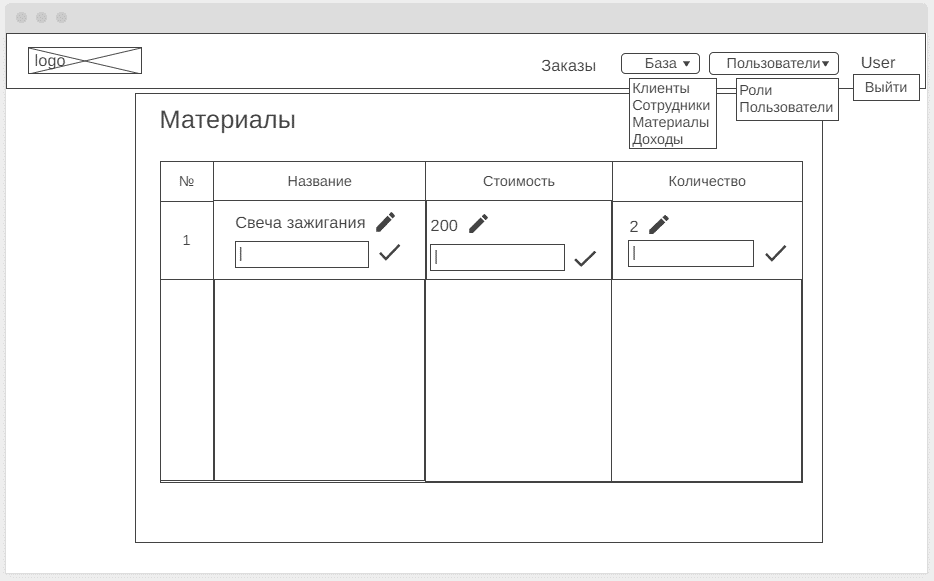


Рисунок 16 – Прототип страницы «Материалы»

Проектирование интерфейса пользователя позволяет определиться с расположение объектов и данных на рабочем пространстве. Также позволяет определиться с количеством страниц и избежать конфликтов с заказчиков.

1. Реализация программного обеспечения
   1. Кодирование программного обеспечения

Суть реализации программного обеспечения заключается в кодировании – процесс написания программного кода. В связи с тем, что разрабатываемый модуль основан на веб-технологиях, в ходе разработки приложения было принято решение использовать:

* фреймворк Bootstrap - для клиентской части;
* фреймворк Vanilla JS - для скриптового языка JavaScript;
* фреймворк Laravel - для серверной составляющей модуля.

Благодаря фреймворку Bootstrap информационная система «Автосервис» имеет простой, но стильный внешний вид, который продемонстрирован на рисунке 17.

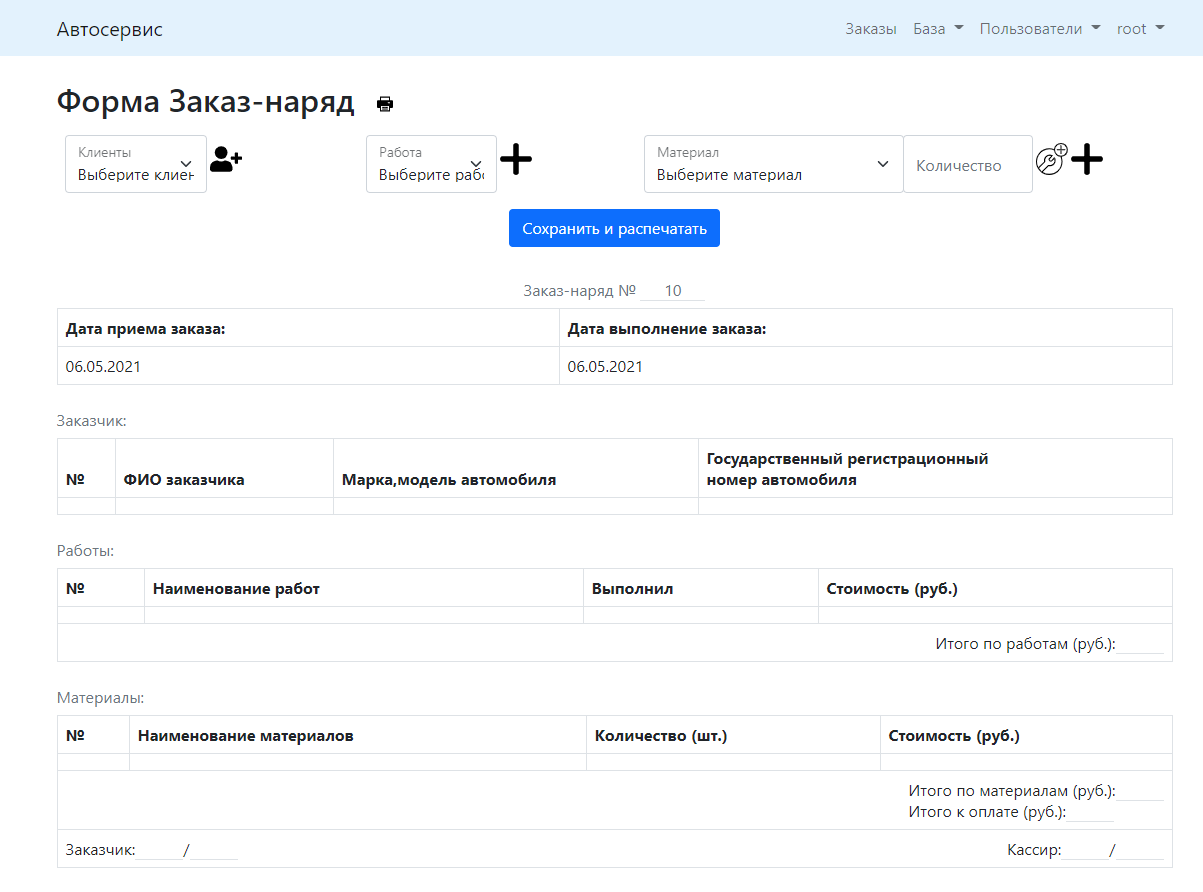


Рисунок 17 – Вкладка «Заказы»

Фреймворк Vanilla JS позволил организовать динамическое получение данных, что позволило реализовать поиск в разделах информационной системы.

Фреймворк Laravel, в данной разработке, позволяет осуществить защиту передачи данных и облегчить работу с БД. Фреймворк использует архитектурную модель MVC (Model View Controller) представленную на рисунке 18. Принцип работы, которой необходимо создавать файл контроллера, модель и представления.

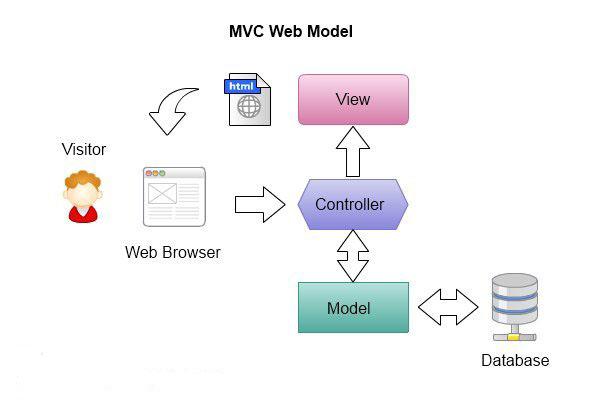


Рисунок 18 – Архитектурная модель MVC

Контроллеры отвечают за обработку запросов пользователя и вызов соответствующих представлений. Контроллер представляет собой класс, который содержит методы обработки функций. Все контроллеры проекта находятся в папке /app/http/Controllers. Самый крупный контроллер это OrderController. Он отвечает за обработку данных раздела «Заказы» и «Доходы». На рисунке 19 представлен код функции search из контроллера OrderController.



Рисунок 19 – Фрагмент контроллера OrderController

Представления отвечают за показ результатов запроса пользователю. Каждое представление – это отдельная страница, содержащая html и javascript код, а также php вставки. Все представления находятся в папке /resource/views. Представление страницы «Материалы» является одной из крупнейших представлений информационной системы. В ней данные из контроллера представляются в виде таблицы. На рисунке 20 изображен фрагмент кода этого представления.



Рисунок 20 – Фрагмент представления страницы «Материалы»

В работе приложения не малую роль играет система маршрутов, которые связывают файл представления с нужным ему контроллером. Все маршруты проекта располагаются в папке routes. Все маршруты проекта можно просмотреть в приложении А Листинг маршрутов.

Модуль содержит несколько крупных функций. Одна из самых крупных функций — это функция обновления таблиц (фрагмент кода на рисунке 21).

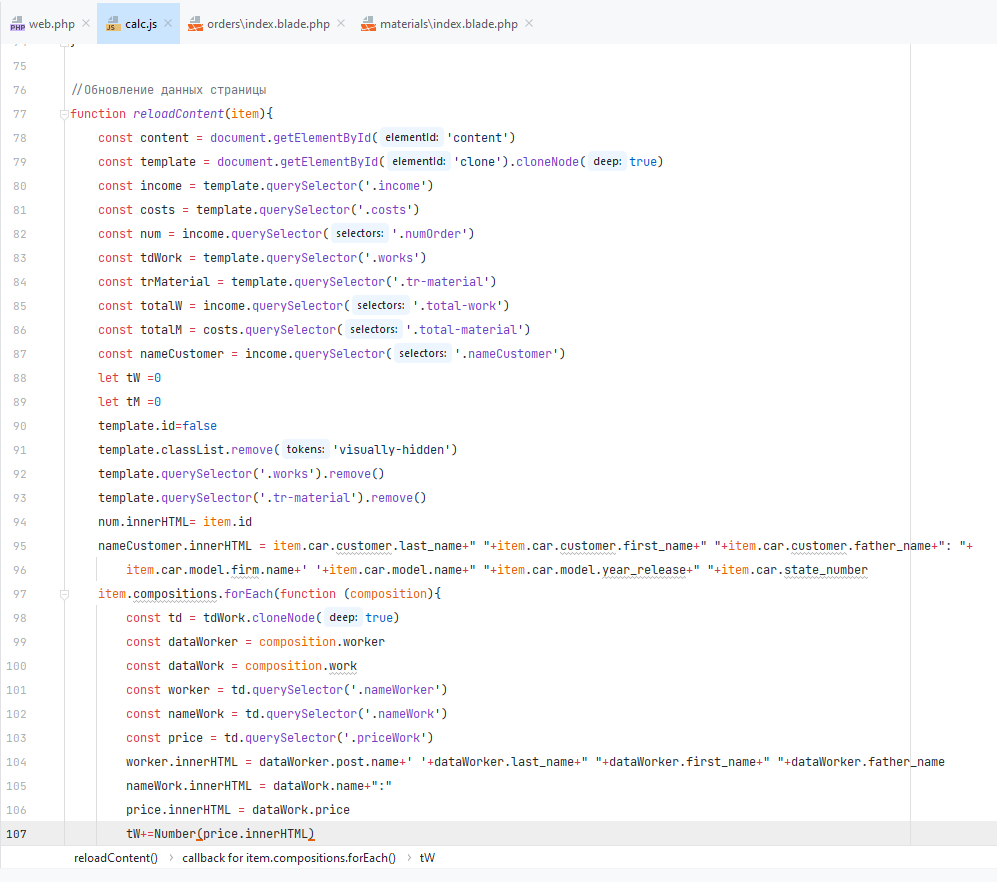


Рисунок 21 – Фрагмент функции обновления таблиц

Эта функция обновляет данные на странице «Доходы», учитывая заполненность соответствующих полей. Функция написана на JavaScript.

В разделах «Клиенты», «Сотрудники» и «Материалы» имеются функции вставки, обновления и удаления данных. Эти функции описаны в соответствующих контроллерах разделов.

Функция вставки данных в базу раздела «Клиенты» демонстрируется на рисунке 22.

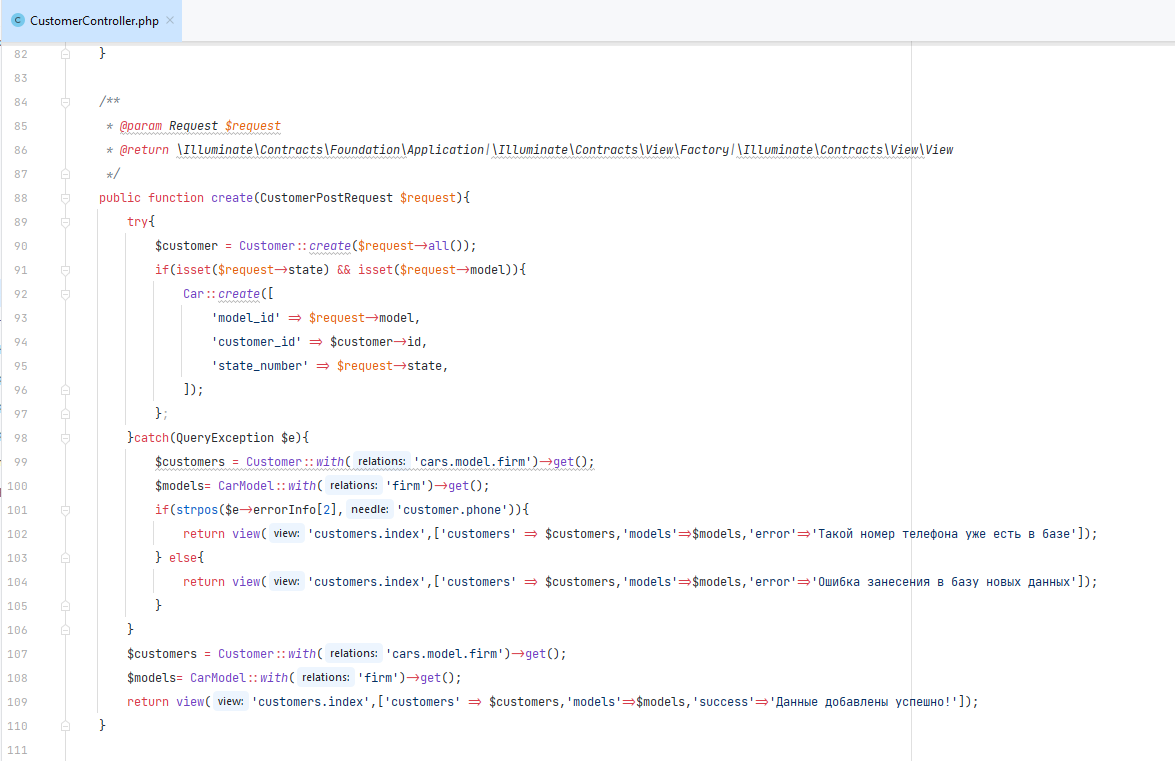


Рисунок 22 – Функция create контроллера CustomerController

Данная функция – это транзакция, которая проверяет корректность заполнения формы страницы и вставляет в базу данных, прошедшие проверку данные, если же в форме были ошибки, то произойдет откат и выведется соответствующее сообщение. Сборка и отправка запросов происходит с помощью конструктора запросов фреймворка Laravel. Он облегчает работу с базой данных и позволяет избежать ошибок в написании запросов.

Функция обновления данных базы страницы «Сотрудники» демонстрируется на рисунке 23.

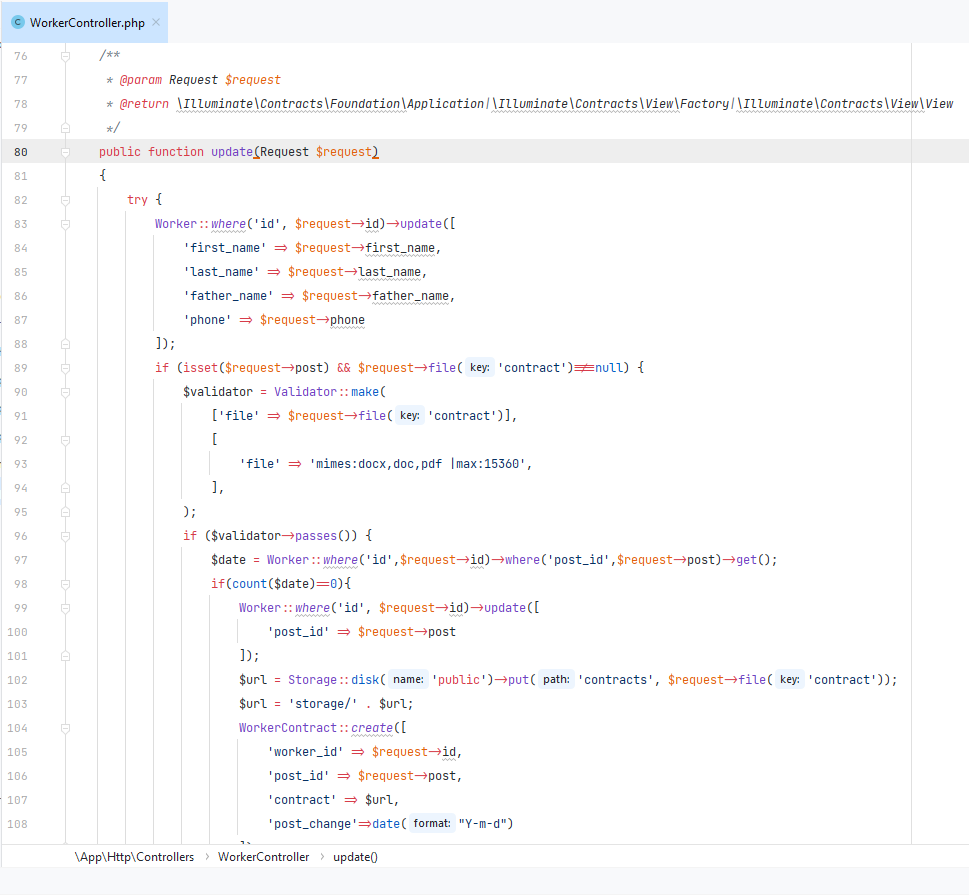


Рисунок 23 – Фрагмент функции update контроллера WorkerController

Данная функция аналогична функции вставки. Она также проверяет корректность заполнения формы и при совершении ошибки происходит откат данных и выводится соответствующее сообщение. Функция комбинирует запросы insert и update для MySQL сервера.

1. Внедрение и сопровождение программного обеспечения

Внедрение информационной системы будет происходить на основе заказа от какого-либо предприятия на соответствующее действие. В процессе внедрения будут предоставлены: информационная система и вся необходимая программная документация.

Сопровождение информационной системы будет происходить по договоренности разработчиков и предприятия.

1. Документирование программного обеспечения
   1. Руководство по установке программного обеспечения

Руководство по установке информационной системы написано для последующих пользователей для облегчения взаимодействия с информационной системой.

Установка информационной системы «Автосервис» производится путем копирования репозитория информационной системы на персональный компьютер или сервер.

Для установки необходимо проделать следующие шаги:

1. Вставьте компакт-диск информационной системы в дисковод или flash-накопитель, содержащий информационную систему.
2. Скопируйте репозиторий по необходимому пути.

Нажмите правой кнопкой мыши по папке AutoService и выберите «Копировать» (или левой кнопкой мыши и сочетание клавиш Ctrl+C). После чего перейдите в папку, где будет хранится информационная система (в примере Program Files) и в свободно месте нажмите правую кнопку мыши и выберите «Вставить» (или сочетание клавиш Ctrl+V). Дождитесь окончания копирования.

Установка информационной системы завершена успешна.

* 1. Руководство пользователя программного обеспечения
     1. Описание системы

Информационная система содержит разделы с разным уровнем доступа. Информационной системой могут пользоваться только зарегистрированные пользователи. Для первоначального запуска существует стандартная учетная запись, которую в последующем рекомендуется удалить. Работа в разделе зависит от уровня доступа учетной записи авторизированного пользователя. Подробнее смотрите в «Страница входа».

* + 1. Страницы входа и учетные записи

Страница входа позволяет войти в одну из учетных записей информационной системы. Страница представляет собой форму, которая демонстрируется на рисунке 24.

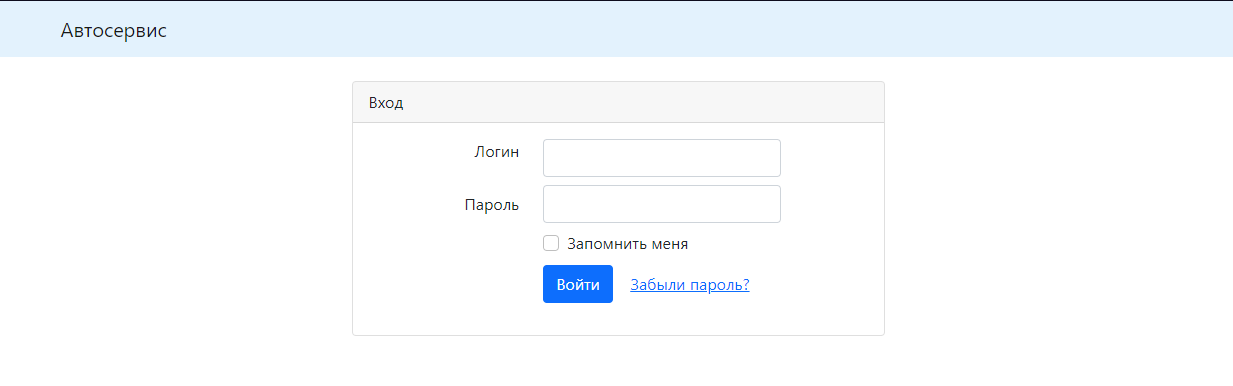


Рисунок 24 – Страница входа

В информационной системе существует гибкая система прав доступа к функциям разделов системы.

При первой установки информационной системы существует только одна учетная запись со всеми правами доступа. Ее данные смотрите в таблице 2. После входа рекомендуется создать новую учетную запись с таким уровнем доступа, войти в нее и удалить стандартную учетную запись, так как её данные доступны любому прочитавшему это руководство пользователя.

Таблица 1 – Стандартная учётная запись

|  |  |
| --- | --- |
| Поле | Данные |
| Имя пользователя | root |
| Пароль | root |

* + 1. Раздел «Заказы»

Раздел «Заказы» позволяет заполнить Заказ-наряд для оказания услуг автосервиса.

Раздел разделен на 2 логические части: форма для заполнения (рисунок 25) и предпросмотр заполняемого заказ-наряда (рисунок 26). При заполнении формы изменяется и заказ-наряд.

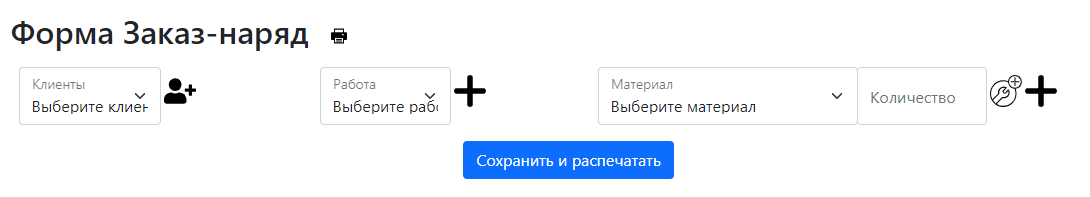


Рисунок 25 – Форма для заполнения заказ-наряда

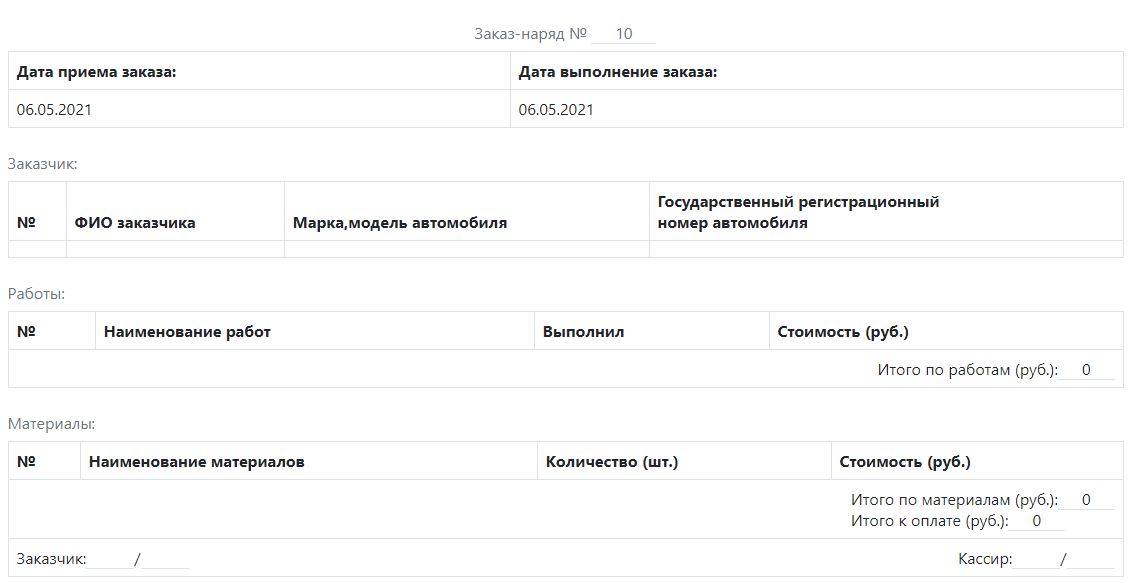


Рисунок 26 – Заполняемый заказ-наряд

Иконки в виде «плюсов» позволяют добавить в заказ-наряд ещё одну работу или материал (рисунок 27).

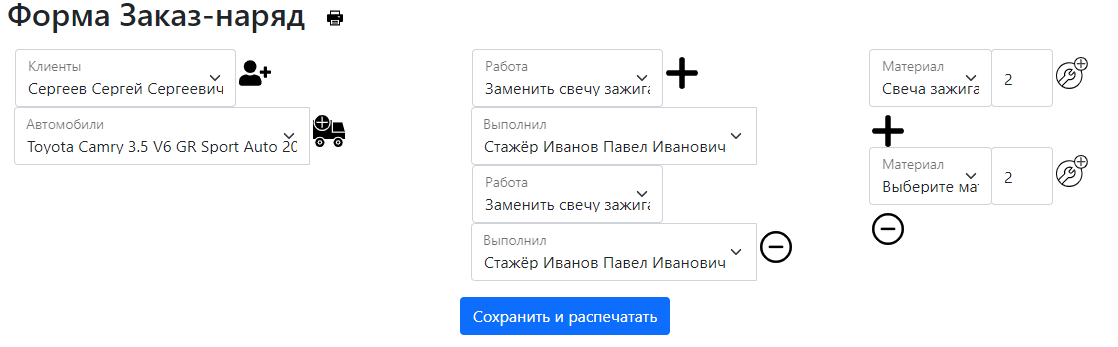


Рисунок 27 – Заполненная форма с несколькими работами и материалами

Иконки в виде «человечка с плюсом» позволяет добавить в базу нового клиента и его автомобиль. При нажатии на эту иконку откроется модальное окно, которое и позволит выполнить её функцию (рисунок 28).

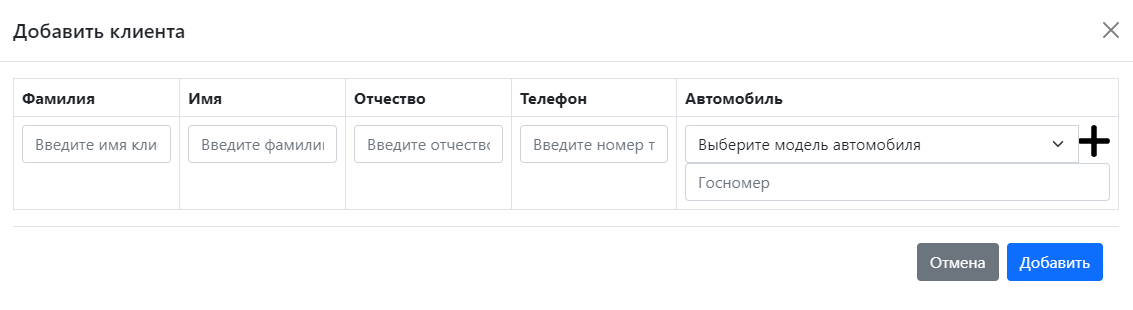


Рисунок 28 – Модальное окно нового клиента

Также имеется возможность добавить автомобиль к уже существующему клиент.

Иконка вида «гаечного ключа с плюсом» позволяет добавить новый расходный материал в базу данных. По нажатию вылазит соответствующее модальное окно, которые необходимо заполнить (рисунок 29).

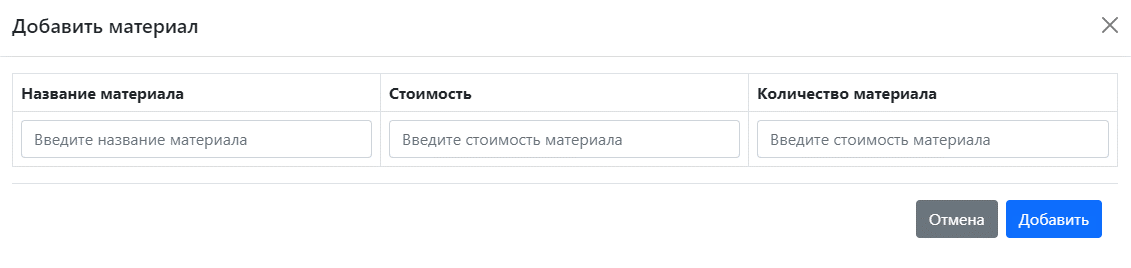


Рисунок 29 – Модальное окно нового материала

* + 1. Раздел «Клиенты»

Раздел «Клиенты» позволяет манипулировать данными клиента. С соответствующим правом доступа, имеется возможность изменить, удалить или добавить нового клиента. Поиск позволяет по введенным данным найти необходимого клиента, если известен хотя бы фрагмент какой-либо информации о нём. Внешний вид раздела представлен на рисунке 30.

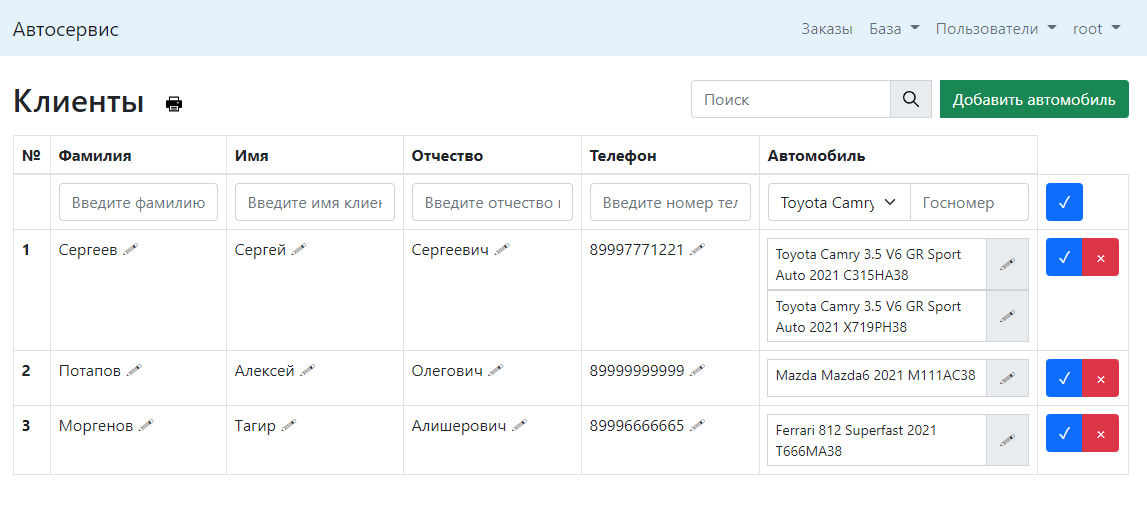


Рисунок 30 – Раздел «Клиенты»

Кнопка «Добавить автомобиль» откроет страницу «Автомобили», которое позволит добавить новую фирму, модель автомобиля, а также изменить уже существующие (Рисунок 31).

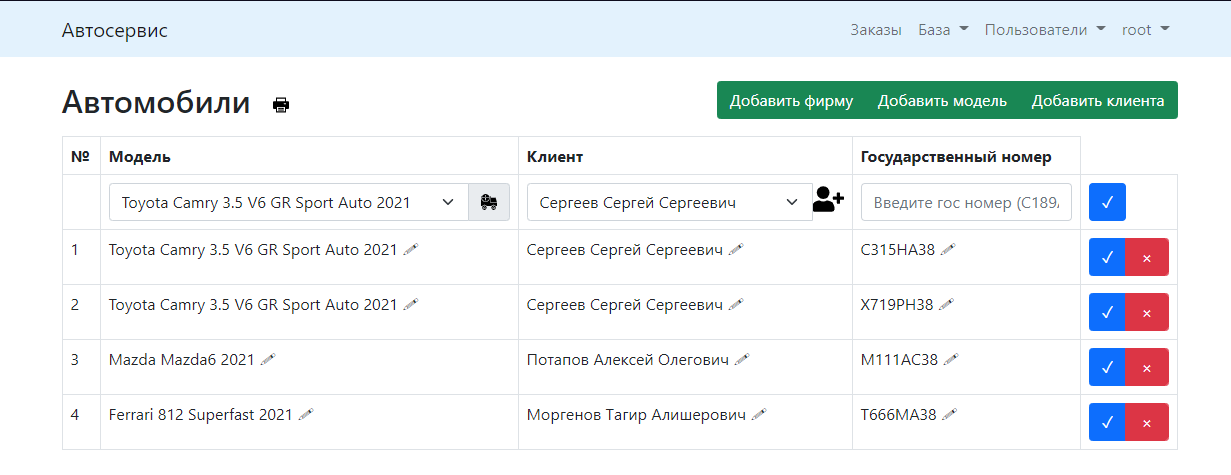


Рисунок 31 – Страница «Автомобили» раздела «Клиенты»

Нажав на кнопки «Добавить фирму», «Добавить модель» отобразиться соответствующее модальное окно, которые позволят добавить новые данные в базу.

На этой странице также можно привязать и изменить автомобиль у существующего клиента в базе (рисунок 32).

Изменить название модели или фирмы можно, нажав на значок «карандаша» в столбце «Модели» (рисунок 32).

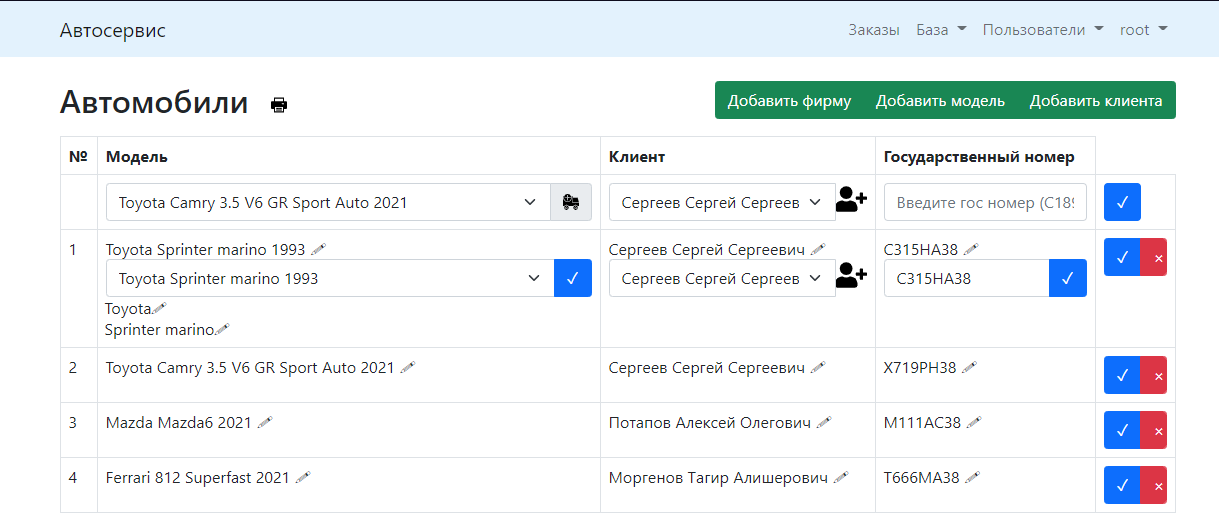


Рисунок 32 – Изменить название модели или фирмы в разделе «Клиенты»

* + 1. Раздел «Сотрудники»

Раздел «Сотрудники» позволяет манипулировать данными сотрудников. В этом разделе имеется возможность изменить личные данные сотрудника, увидеть занимаемую должность, изменить её, скачать документы подтверждающие смену должности, а также его последние заказы, если такие имеются. Внешний вид продемонстрирован на рисунке 33.

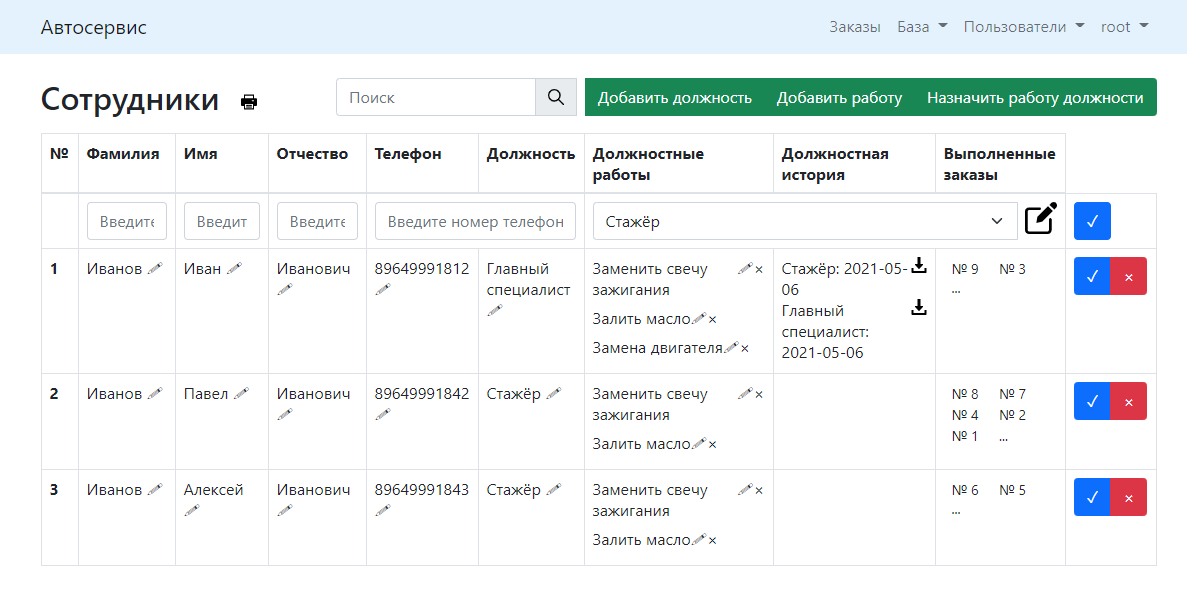


Рисунок 33 –Раздел «Сотрудники»

Нажав на кнопки «Добавить должность», «Добавить работу», «Назначить работу должности» отобразиться соответствующее модальное окно, которые позволят добавить новые данные в базу.

Также в этом разделе имеется возможность манипулировать работами должности (редактировать название, удалить, добавить) для этого необходимо нажать соответствующие знаки. Для добавления работы необходимо нажать кнопку «Назначить работу должности». Для редактирования названия работы необходимо нажать значок «карандаша» возле названия работы. Для удаления работы необходимо нажать значок «крестика» возле названия работы (рисунок 34).

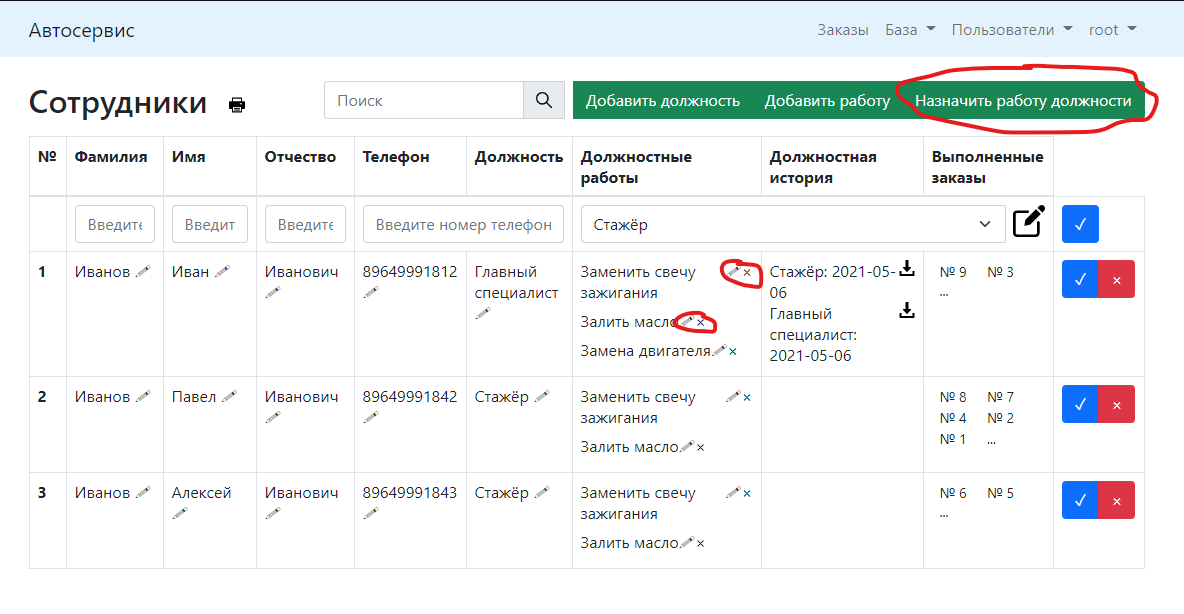


Рисунок 34 – Манипулирование работами в разделе «Сотрудники»

* + 1. Раздел «Материалы»

Раздел «Материалы» позволяет манипулировать данными расходных материалов. Также в этом разделе визуально демонстрируется (подсветкой) недостаток того или иного расходного материала. Внешний вид демонстрируется на рисунке 35.

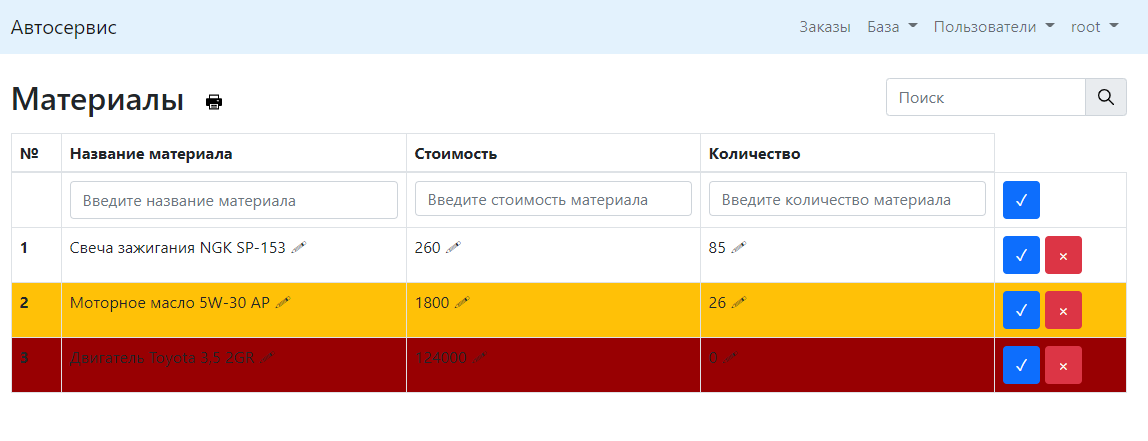


Рисунок 35 – Раздел «Материалы»

Изменение в этом разделе производятся аналогично другим разделам.

* + 1. Раздел «Доходы»

Раздел «Доходы» позволяет наглядно увидеть заказ-наряды в определенный период времени, а также общую сумму этих заказ-нарядов.

Этот раздел имеет 2 логические части: форма с датами и заказ-наряды, которого при открытии изначально нет (рисунок 36).

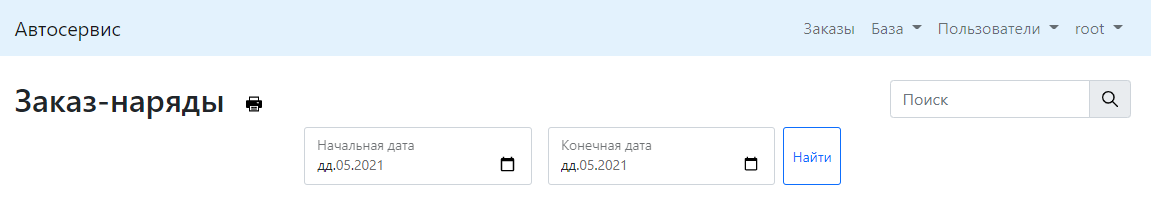


Рисунок 36 – Раздел «Доходы»

После того как период будет выбран и нажата кнопка «Найти», в разделе «Доходы» появится заказ-наряды, которые были зарегистрированы в этот период времени (рисунок 37).

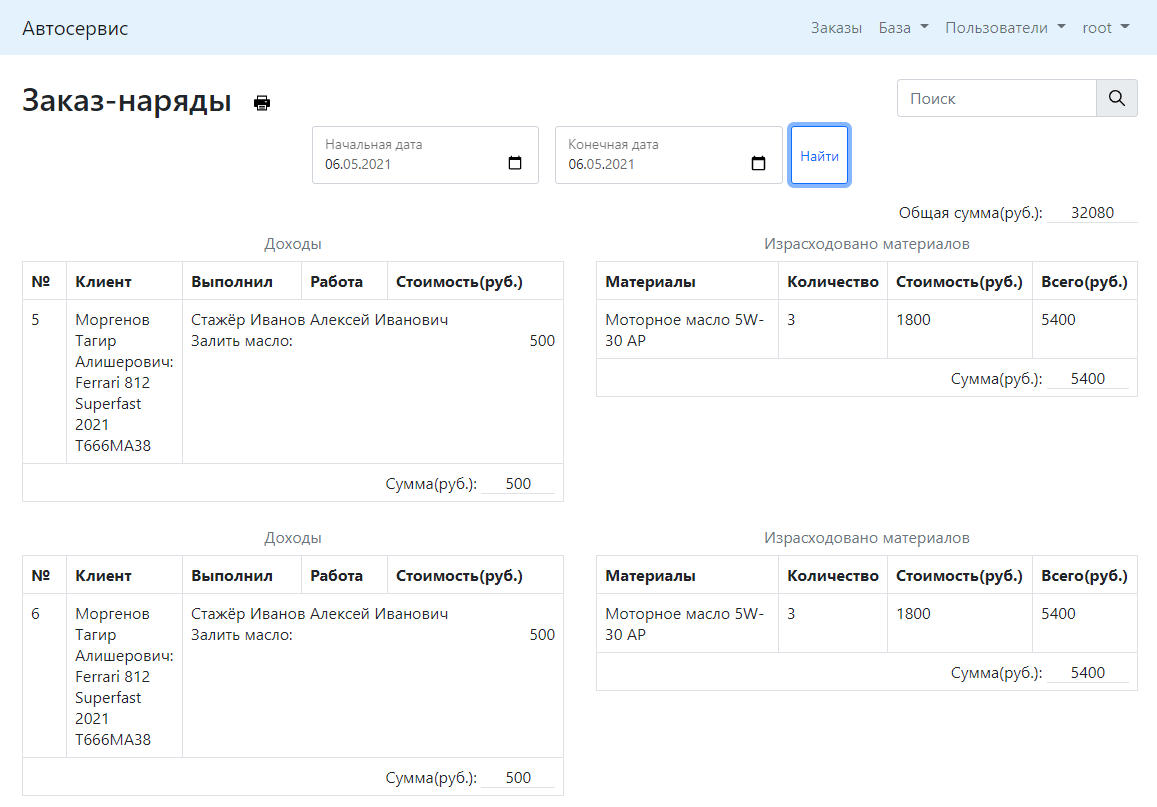


Рисунок 37 – Заказ-наряды раздела «Доходы»

* + 1. Раздел «Роли»

Раздел «Роли» позволяет манипулировать ролями информационной системы. Имеется возможность просмотра прав доступа роли, редактировать права доступа роли, удалить роль, а также создать новую. Внешний вид продемонстрирован на рисунке 38.

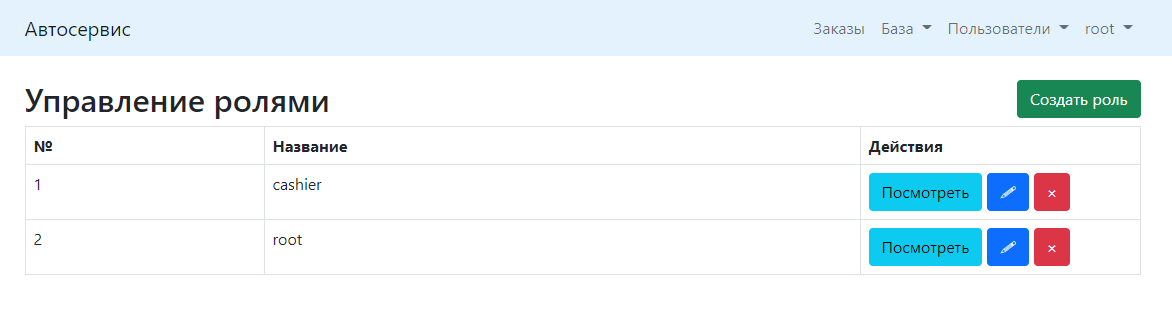


Рисунок 38 – Раздел «Роли»

Редактирование данных роли происходит после перехода на соответствующую страницу раздела. Для этого необходимо нажать на синюю кнопку со значком «карандаша». На странице редактирования имеется возможность изменить название роли, а также переназначить права доступа. Внешний вид продемонстрирован на рисунке 39.

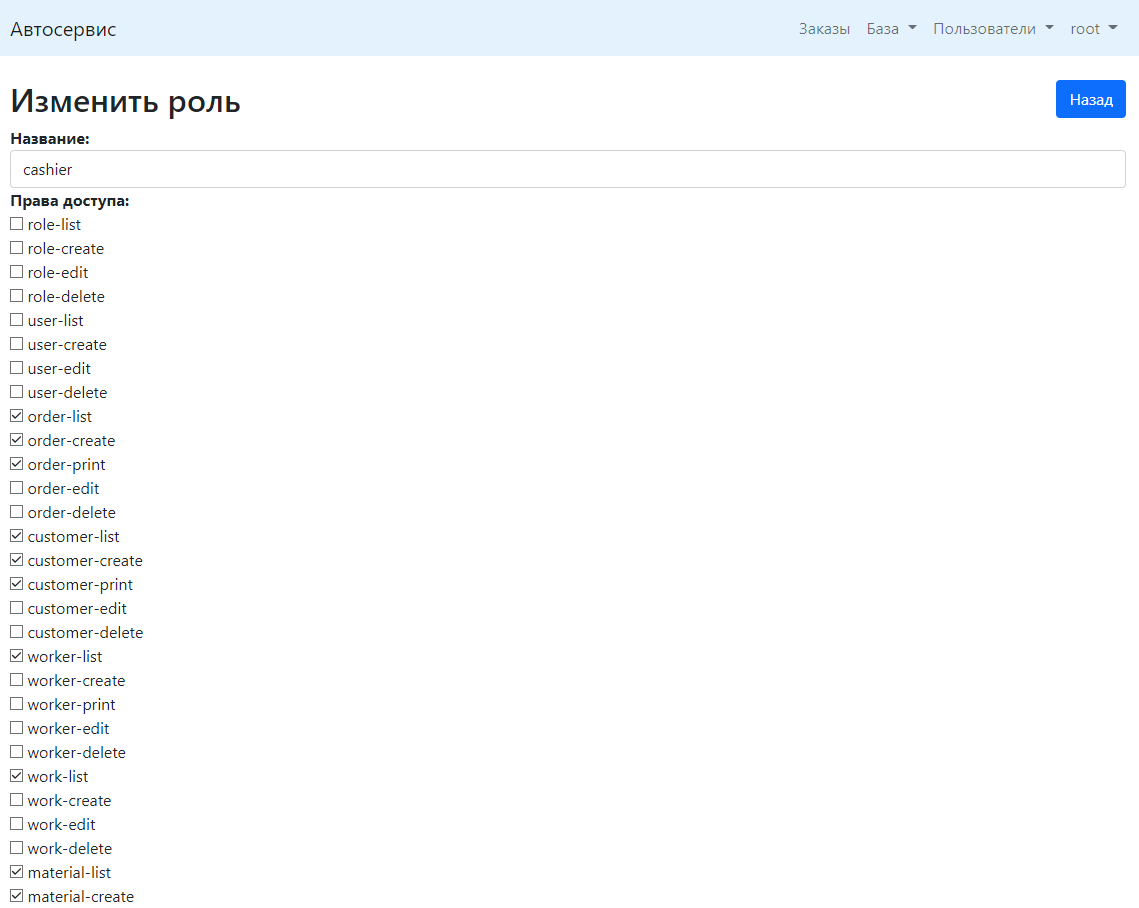


Рисунок 39 – Страница редактирования раздела «Роли»

* + 1. Раздел «Пользователи»

Раздел «Пользователи» позволяет манипулировать данными учетных записей информационной системы. Имеется возможность просмотра ролей учетной записи, изменить пароль, почту, логин, роль учётной записи, удалить учётную запись, а также зарегистрировать новую. Все действия производятся аналогично разделу «Роли». Внешний вид продемонстрирован на рисунке 40.

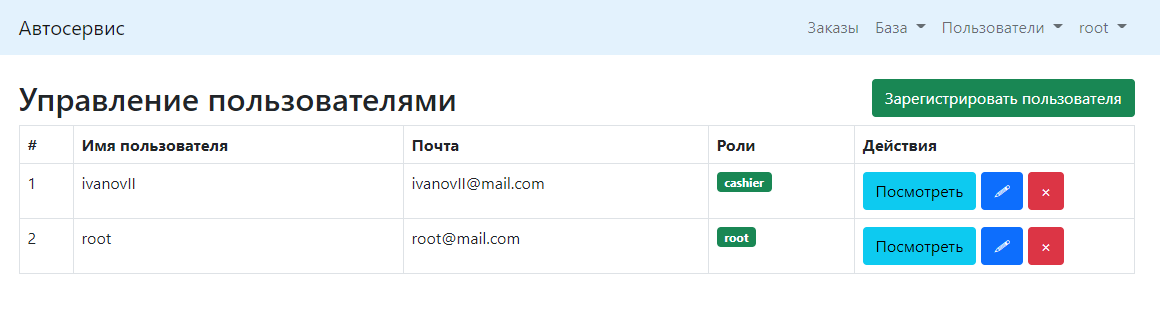


Рисунок 40 – Раздел «Пользователи»

* 1. Руководство системного администратора программного обеспечения

Информационная система «Автосервис» - это клиент-серверное приложение, которому необходимо иметь веб-сервер.

Дальше будет описываться как настроить веб-сервер при помощи программного обеспечения Xampp.

Из предоставленного вам носителя с информационной системой «Автосервис» скопируйте всё в корень диска (пример D:/).

Если название диска D, то производить какие-либо настройки не нужно. В противном случае, необходимо перейти к файлу с настройкой веб-сервера (пример пути до файла: D:\xampp\apache\conf\extra\httpd-vhosts.conf).

В этом файле в строках 46,49,59 и 60 измените диск D букву своего диска (рисунок 41).

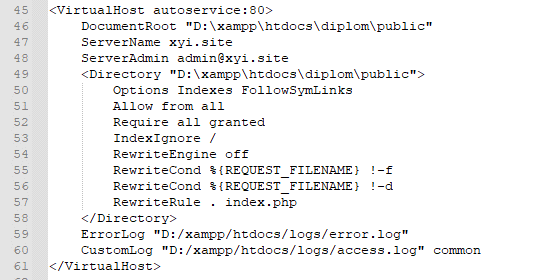


Рисунок 41 – Настройка веб-сервера

После чего запустите xampp\_start.exe папке xampp (рисунок 42). В открывшемся окне запустите apache и mysql (рисунок 43).

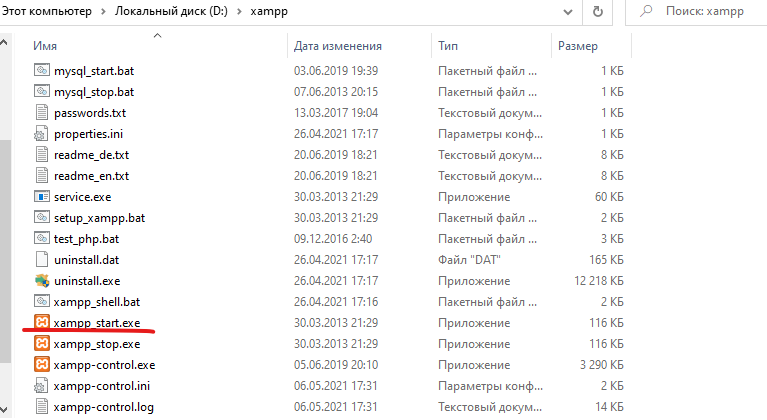


Рисунок 42 – Запуск xampp

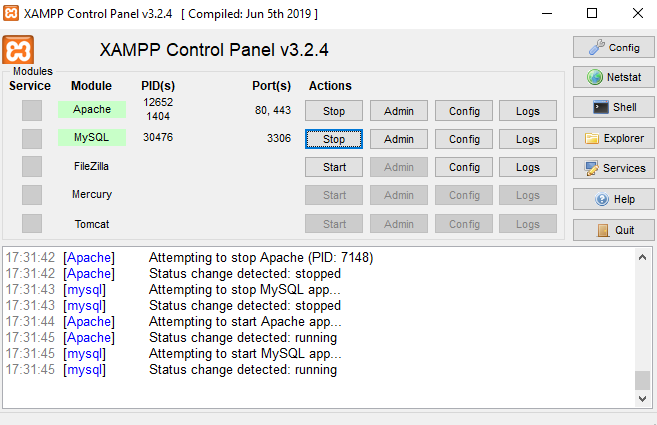


Рисунок 43 – Запуск веб-сервера apache и сервера mysql

После чего в браузере по адресу http://autoservice запустите информационную систему «Автосервис».

1. Стоимость разработки и внедрения программного продукта
   1. Организационно-экономическое обоснование проекта

Разрабатываемое приложение будет использоваться для упрощения процесса оказания услуг, накопления клиентской базы предприятия, связанного c автосервисом.

Средняя стоимость аналогов проводится по формуле 6.1:

(6.1)

,

где Ца – среднее стоимость аналогов,

n – количество рассмотренных аналогов,

Зi – стоимость аналога i.

Стоимости рассмотренных аналогов демонстрируется в таблице 3.

Таблица 3 – Стоимость рассмотренных аналогов

|  |  |
| --- | --- |
| Аналог | Стоимость, руб. |
| LogicStar-Avto | 50000 |
| 1C: Альфа-Авто | 96000 |
| TurboService | 70000 |

Средняя стоимость рассмотренных аналогов равна 72000 рублей.

Данное приложение разрабатывается для предприятия, связанного с оказанием услуг автосервиса.

* 1. Расчет затрат на разработку программного продукта

Расчет полных затрат на разработку проектного решения (КРПР) осуществляется по формуле 6.2:

(6.2)

КРПР=ЗОТР+ЗЭВМ+ЗСПП+ЗХОН+Е+А,

гдеЗОТР – 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 4). 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 6.3:

(6.3)

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

Таблица 4 - 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 | 16 |
| Анализ и уточнение требований | 3 | 24 | 8 |
| Разработка технического задания | 4 | 32 | 32 |
| Проектирование структуры | 5 | 40 | 40 |

Продолжение таблицы 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Программная реализация | 58 | 464 | 450 |
| Тестирование программного продукта | 5 | 40 | 40 |
| Отладка программного продукта | 5 | 40 | 40 |
| Разработка описания | 5 | 40 | 40 |
| ИТОГО | 88 | 704 | 666 |

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 6.4):

(6.4)

Зотр=Ст1\*Фвр,

где Ст1 - часовой тарифной ставки (принимаем равной 20 руб./час);

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

Разработка программного продукта проводилась бесплатно, так как связывающий договор не требует оплаты труда.

Затраты, связанные с использованием вычислительной и оргтехники, рассчитываются по формуле 6.5:

,

(6.5)

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=6руб.

Затраты на электроэнергию рассчитываются по следующей формуле 6.6:

Е=W×t×T,

(6.6)

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

t – время работы ПК, час;

T – тариф электроэнергии, руб.

Тариф 1,07 рубля за киловатт.

Амортизация ПК рассчитывается по следующей формуле 6.7:

(6.7)

где – первоначальная стоимость ПК, руб;

– процент амортизации в год.

При разработке программного продукта использовались следующие свободно распространяемые специальные продукты: Xampp, Bootstrap, PhpStorm, Laravel.

Затраты на хозяйственно-организационные нужды приводятся в таблице 5 и вычисляются по формуле 6.8:

,

(6.8)

где Цτ – цена τ-го товара, руб.;

Кτ – количество τ-го товара.

Таблица 5 - Затраты на хозяйственно-организационные нужды

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу (руб.) | Кол-во (шт.) | Всего (руб.) |
| Диск CD-RWDigitex | 25 | 1 | 25 |
| Бумага | 0,5 | 110 | 55 |
| Чернила для принтера | 700 | 2 | 1400 |
| Итого | | | 1480 |

Результаты выполненных расчетов сводятся в общей таблице (таблица 6).

Таблица 6 – Затраты на разработку

|  |  |  |
| --- | --- | --- |
| Наименование затрат | Условное обозначение | Значение |
| Оплата труда разработчика программного продукта |  | 14080 |
| Затраты, связанные с эксплуатацией техники |  | 3796,2 |
| Затраты на электроэнергию | Е | 564,77 |
| Затраты на амортизацию ПК | А | 1250 |
| Затраты на хозяйственно-операционные нужды |  | 1480 |
| Итого затрат на разработку |  | 21170,97 |

* 1. Расчет затрат на внедрение программного продукта

Внедрение информационной системы в работу предприятия не требует дополнительных затрат, так как все внедряемые программные продукты свободно распространяются.

* 1. Основные выводы

Исходя из расчетов стоимости разработки, можно прийти к выводу, что информационная система обойдется предприятию (без учета оплаты труда) в размере 7090,97 рублей. Такие затраты значительно меньше стоимости рассматриваемых аналогов.

# ЗАКЛЮЧЕНИЕ

В ходе дипломного проекта была разработана информационная система «Автосервис», в которой реализовано хранение информации о клиентах, сотрудниках, выполняемых работах, добавление и редактирование данных, элементы бухгалтерского учёта, возможность подведения итогов за выбранный период времени по различным параметрам, печать отчётов, формирование и печать договоров на оказания услуг и заказ-нарядов.

Информационная система «Автосервис» позволяет упростить процесс оказания услуг, накапливать клиентскую базы, ввести учёт затрат на расходные материалы, составлять заказ-наряды. Это ускорит процесс оказания услуг, что позволит оказывать большее количество услуг, что приведет к росту предприятия.

Все поставленные задачи, а именно: хранение информации о клиентах, сотрудниках, выполняемых работах, добавление и редактирование данных, элементы бухгалтерского учёта, возможность подведения итогов за выбранный период времени по различным параметрам, печать отчётов, формирование и печать договоров на оказания услуг и заказ-нарядов.

Все цели, поставленные для разработки, были достигнуты.

В информационной системе были реализованы следующие функции:

* построение заказ-наряда;
* вставка и обновление данных в базе;
* разграничение прав доступа.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bootstrap [Электронный ресурс] – Режим доступа: http://getbootstrap.ru – Bootstrap Самый популярный в мире front-end фреймворк. (Дата обращения: 10.05.21).
2. CyberForum [Электронный ресурс] – Режим доступа: http://www.cyberforum.ru – (Дата обращения: 08.05.21).
3. Laravel по-русски [Электронный ресурс] – Режим доступа: <https://laravel.ru/docs/v5/queries> – Конструктор запросов (Дата обращения: 10.05.21).
4. PHP [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/PHP – PHP (Дата обращения: 10.05.21).
5. PHP-MyAdmin.ru [Электронный ресурс] – Режим доступа: https://php-myadmin.ru/doc/ability.html – Руководство. Возможности PhpMyAdmin (Дата обращения: 03.05.21).
6. PHPStorm [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/PHPStorm – PHPStorm (Дата обращения: 10.05.21).
7. SQL. Полное руководство. Грофф Дж. Р., Вайнберг П.Н., Оппелъ Э. Дж. 2015 – 959 с.
8. Что такое PHP? [Электронный ресурс] – Режим доступа: https://www.php.net/manual/ru/intro-whatis.php – Что такое PHP? (Дата обращения: 06.05.21).

# Приложение А - Листинг маршрутов

<?php

Route::get('/', function () {return view('auth/login');});

Auth::routes();

Route::group(['middleware' => ['auth']], function() {

Route::get('/',function (){return view('welcome');});

Route::group(['middleware'=>['permission:order-list']],function(){

Route::get('orders', [OrderController::class, 'index'])->name('orders.index');

Route::post('orders', [OrderController::class, 'create']);

Route::get('orders/{id}', [OrderController::class, 'show'])->name('orders.show');

Route::group(['middleware'=>['permission:order-edit']],function(){

Route::get('order/calc', [OrderController::class, 'calc'])->name('orders.calc');

Route::post('order/calc', [OrderController::class, 'getCalc']);

Route::post('order/search', [OrderController::class, 'search']);});

Route::post('orders/cars', [CustomerController::class, 'getCar']);

Route::post('customer/create', [CustomerController::class, 'jsonCreate']);

Route::post('car/create', [CarController::class, 'jsonCreate']);

Route::post('orders/workers', [WorkController::class, 'getWorker']);

Route::post('material/create', [MaterialController::class, 'createOrder']);});

Route::group(['middleware'=>['permission:material-edit']],function(){

Route::get('materials', [MaterialController::class, 'index'])->name('materials.index');

Route::post('materials', [MaterialController::class, 'create']);

Route::post('materials/search', [MaterialController::class, 'search']);

Route::patch('materials', [MaterialController::class, 'update']);

Route::delete('materials', [MaterialController::class, 'destroy']);});

Route::group(['middleware'=>['permission:customer-edit']],function(){

Route::get('customers',[CustomerController::class,'index'])->name('customers.index');

Route::post('customers', [CustomerController::class,'create']);

Route::post('customers/search', [CustomerController::class,'search']);

Route::patch('customers', [CustomerController::class,'update']);

Route::delete('customers', [CustomerController::class,'destroy']);

Route::get('cars', [CarController::class,'index'])->name('cars.index');

Route::post('cars', [CarController::class,'create']);

Route::post('models', [CarController::class,'createModel']);

Route::patch('models', [CarController::class,'updateModel']);

Route::post('firms', [CarController::class,'createFirm']);

Route::patch('firms', [CarController::class,'updateFirm']);

Route::patch('cars', [CarController::class,'update']);

Route::delete('cars', [CarController::class,'destroy']);});

Route::group(['middleware'=>['permission:worker-edit']],function(){

Route::get('workers', [WorkerController::class,'index'])->name('workers.index');

Route::get('workers/{id}', [WorkerController::class,'show'])->name('workers.show');

Route::post('workers/search', [WorkerController::class,'search']);

Route::put('workers', [WorkerController::class,'download']);

Route::post('workers', [WorkerController::class,'create']);

Route::patch('workers', [WorkerController::class,'update']);

Route::delete('workers', [WorkerController::class,'destroy']);

Route::group(['middleware'=>['permission:work-edit']],function(){

Route::get('works', [WorkController::class,'index'])->name('workers.works');

Route::post('works-posts', [WorkController::class,'createWorkPost']);

Route::patch('works-posts', [WorkController::class,'updateWorkPost']);

Route::post('works', [WorkController::class,'create']);

Route::patch('works', [WorkController::class,'update']);

Route::delete('works', [WorkController::class,'destroy']);

Route::post('posts', [PostController::class,'create']);

Route::delete('posts', [PostController::class,'destroy']);});});

Route::resource('roles', RoleController::class);

Route::resource('users', UserController::class);});