**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 3](#_Toc73137411)

[1 Предпроектное исследование 5](#_Toc73137412)

[1.1 Исследование предметной области 5](#_Toc73137413)

[1.2 Анализ инструментов разработки 6](#_Toc73137414)

[1.3 Обоснование выбора программного продукта 8](#_Toc73137415)

[2 Проектирование 11](#_Toc73137416)

[2.1 Архитектура программного обеспечения 11](#_Toc73137417)

[2.2 Функциональное проектирование 13](#_Toc73137418)

[2.3 Проектирование базы данных 16](#_Toc73137419)

[2.4 Проектирование пользовательского интерфейса 21](#_Toc73137420)

[3 Реализация программного обеспечения 25](#_Toc73137421)

[3.1 Кодирование программного обеспечения 25](#_Toc73137422)

[4 Документирование программного обеспечения 30](#_Toc73137423)

[4.1 Руководство пользователя программного обеспечения 30](#_Toc73137424)

[5 Стоимость разработки и внедрения программного продукта 39](#_Toc73137434)

[5.1 Организационно-экономическое обоснование проекта 39](#_Toc73137435)

[5.2 Расчет затрат на разработку программного продукта 39](#_Toc73137436)

[5.3 Расчет затрат на внедрение программного продукта 43](#_Toc73137437)

[5.4 Основные выводы 43](#_Toc73137438)

[ЗАКЛЮЧЕНИЕ 44](#_Toc73137439)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45](#_Toc73137440)

[Приложение А - Листинг маршрутов 47](#_Toc73137441)

ВВЕДЕНИЕ

Темой дипломного проекта является информационная система «Автосервис».

Автотранспорт является наиболее популярное средство перемещения человека. Автотранспорт может быть поврежден и для этого необходимы автосервисы, которые позволяют исправить существующие проблемы автотранспорта.

Самое главное при работе с автотранспортом — это определение необходимых материалов для исправления повреждений автомобиля, а также составления заказ-наряда для сотрудников автосервиса. Все необходимые данные удобно хранить в таблицах. Информация в базе данных (далее БД) тоже представляется в виде таблиц, что позволяет организовать автоматизацию сборки заказ-наряда, накапливать клиентскую базу, вести учёт работы сотрудников и учёт материалов. Качество и надежность результатов вышеназванных действий служит основой эффективной работы автосервиса. Актуальным является сохранять сведения о названных процедурах в специальных модулях программного обеспечения. Данную базу можно применить в любом автосервисе. Главы компании всегда должны отслеживать работу компании и следить за своими сотрудниками. Для этого нужна общая база данных, включающая всю необходимую информацию. И программа, которая будет понятна и автоматизирует работу с базой данных.

Цель: разработать информационную систему «Автосервис».

Задачи:

* + Провести предпроектное исследование.
  + Составить техническое задание на разработку программного продукта в соответствии с ГОСТ.
  + Провести проектирование программного продукта.
  + Реализовать программный продукт.
  + Разработать документы для программного продукта.
  + Рассчитать стоимость разработки и внедрения программного продукта.

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. Исследование предметной области

Предметной областью дипломного проекта является информационная система «Автосервис» (далее ИС).

Автосервис - активно развивающаяся сфера экономики, инфраструктурный элемент транспортной отрасли. Предназначением отрасли автосервиса является удовлетворение потребностей, связанных с поддержанием технического состояния и эксплуатационных характеристик автотранспортных средств.

Автотранспортом называется вид транспорта, который относится к наземному виду транспортного средства, с помощью которого выполняют перевоз грузов и пассажиров.

Техническое обслуживание (ТО) — это комплекс операций или операция по поддержанию исправного состояния колесного транспортного средства (составных частей, систем колесного транспортного средства) в соответствии с инструкциями его изготовителя.

Ремонт - комплекс операций по восстановлению исправного состояния колесного транспортного средства (его составных частей, систем).

Все обращающиеся в автосервис клиенты, желающие оформить заказ на оказание услуг, должны заполнить форму регистрации, в которой необходимо указать свои ФИО, марку автомобиля, номерной знак и ФИО владельца машины, контактный телефон.

Таким образом, в функционирование автосервиса входит:

- регистрация клиентов;

- справочная информация о доступных услугах;

- прием заказа на оказание услуг;

- справочная информация о сделанном заказе;

- выполнение заказа;

- отчет о проделанных работах и расчет стоимости предоставленных услуг.

* 1. Анализ инструментов разработки

Для разработки ИС можно использовать следующие программные продукты: SQL, Django, PyCharm, Python, РНР, Ruby, JavaScript, JetBrains PhpStorm, JetBrains WebStorm, Notepad++, Laravel, NodeJS, MySQL, Oracle, PostgreSQL, Microsoft Visual Studio, MySQL Workbench, PhpMyAdmin, Microsoft Visio.

SQL — «Язык структурированных запросов» декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Python — это высокоуровневый язык программирования, который используется в различных сферах IT, таких как машинное обучение, разработка приложений и другие.

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django.

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования МVС.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств.

РНР: Hypertext Preprocessor — «Инструменты для создания персональных веб-страниц» -серверный язык программирования общего назначения, интенсивно применяемый для разработки веб-приложений.

Notepad++ — свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса большого количества языков программирования и разметки, а также языков описания аппаратуры VНDL и Verilog.

JetBrains PhpStorm — представляет собой интеллектуальный редактор для РНР, НТМL и JavaScript, с возможностями анализа кода на лету, предотвращения ошибок в коде, автоматизированными средствами рефакторинга для РНР и JavaScript.

JetBrains WebStorm — интегрированная среда разработки на JavaScript, CSS & НТМL от компании JetBrains, разработанная на основе платформы IntelliJ IDEA. WebStorm обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями

JavaScript — это полноценный динамический язык программирования, который применяется к НТМL документу, и может обеспечить динамическую интерактивность на веб-сайтах.

Laravel — веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC.

NodeJS — программная платформа, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.j s добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на С++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кoдa.

MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle

PhpMyAdmin — веб-приложение с открытым кодом, написанное на языке РНР и представляющее собой веб-интерфейс для администрирования CYБД MySQL.

PostgreSQL — свободная объектно-реляционная система управления базами данных. Существует в реализациях для множества UNIХ-подобных платформ, включая AIX, различные ВSD-системы, НР-UХ, IRIX, Linux, macOS, Tru64, QNX, а также для Microsoft Windows.

Oracle Database или Oracle RDBMS — объектно-реляционная система управления базами данных компании Oracle

MySQL Workbench — инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL.

* 1. Обоснование выбора программного продукта

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

Для выбора инструментов разработки были произведены сравнения, которые показаны в таблице 1, таблице 2, таблице 3, и таблице 4.

Таблица 1 — Сравнение IDE для разработки программного продукта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название IDE | Visual studio code | Notepad++ | JetBrains PhpStorm | JetBrains  WebStorm |
| Стоимость | Свободное ПО | Свободное ПО | Студенческая лицензия | Студенческая лицензия |
| Автосохранение | + | - | + | + |
| Автодополнение | + | - | + | + |
| Интеграция с системами управления версиями (git) | -  (возможно добавить плагин) | -  (возможно добавить плагин) | + | + |
| Поиск по функции в коде | - | - | + | + |
| Встроенные языки программирования | -  (возможно добавить плагин) | -  (возможно добавить плагин) | PHP, HTML, JavaScript, CSS, SQL | HTML, JavaScript, CSS, SQL |

Таким образом, из представленных IDE можно выделить несколько инструментов, способных облегчить и ускорить разработку программного обеспечения, а именно JetBrains PhpStorm, JetBrains WebStorm, так как программный продукт будет реализован с помощью web - технологий. Исходя из того, что для разработки необходим язык программирования PHP, то в качестве IDE был выбран JetBrains PhpStrorm.

Таблица 2 — Сравнение языков программирования для разработки программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| Название языка программирования | PHP | Ruby | Python |
| Простота обучаемости | + | - | + |
| Читабельность | + | - | + |
| Динамическая типизация | + | + | + |
| Интеграция баз данных | + | + | - |
| Объектно-ориентированные  возможности | + | + | + |

Вывод, из представленных языков программирования можно выделить РНР, так как именно этот язык более перспективный в разработке web -приложения.

Таблица 3 — Сравнение СУБД для разработки программного продукта.

|  |  |  |  |
| --- | --- | --- | --- |
| Название СУБД | Oracle | MySQL | PostgreSQL |
| Язык программирования | С, С++, Java, Ruby, Objective | Delphi, С, С++, Java, Perl, РНР, Pvthon, Ruby, SQL | pgSQL, РНР, Python, Ruby и др. |
| Скорость обработки | Быстрая | Быстрая | Медленная |
| Операционная система Windows | Поддерживается | Поддерживается | Поддерживается |
| Исходный код | Закрытый | Открытый | Открытый |
| Лицензия | Коммерческая | Свободная | Свободная |

В результате сравнения, из представленных СУБД можно выделить MySQL, так как оно имеет свободную лицензию и подходит для создания web – приложения.

Таблица 4 — Сравнение средств проектирования

|  |  |  |  |
| --- | --- | --- | --- |
| Название средства проектирования | Draw.io | Visio | NClass |
| Поток данных диаграмм | + | + | - |
| Er - диаграмма | + | + | - |
| Проверка связей | + | + | - |
| Удобство использования | + | + | - |
| Лицензия | свободная | коммерческая | свободная |

Следовательно, из представленных средств проектирования, было принято решение использовать Draw.io, так как оно имеет свободную лицензию и большое количество UМL средств.

1. Проектирование

Перед проектированием информационной системы было разработано техническое задание, на основе которого производилось проектирование 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 1 демонстрируется архитектура веб-приложений, которая соответствует разрабатываемой информационной системе.

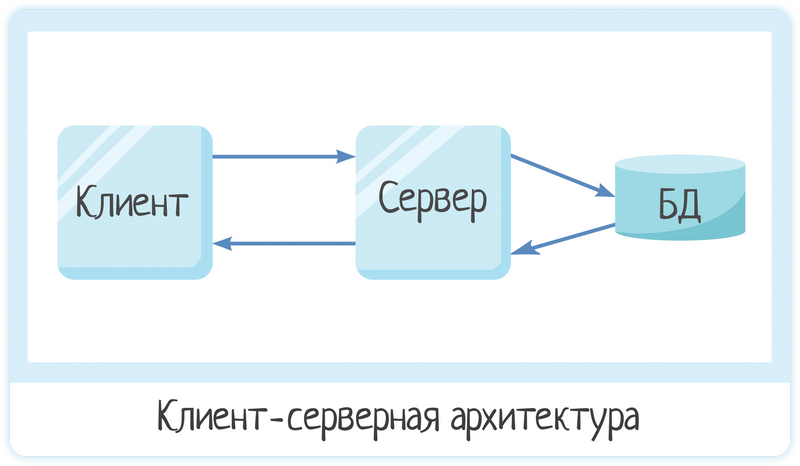


Рисунок 1 – Клиент-серверная архитектура веб-приложения

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.

Фреймворк Laravel использует архитектурную модель MVC (Model View Controller) представленную на рисунке 1.1.

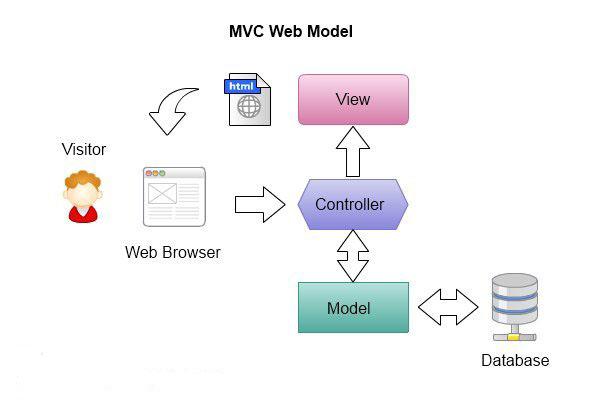


Рисунок 1.1 – Архитектурная модель MVC

* 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 2 показаны входные данные, управление механизм, выходные данные и функция.

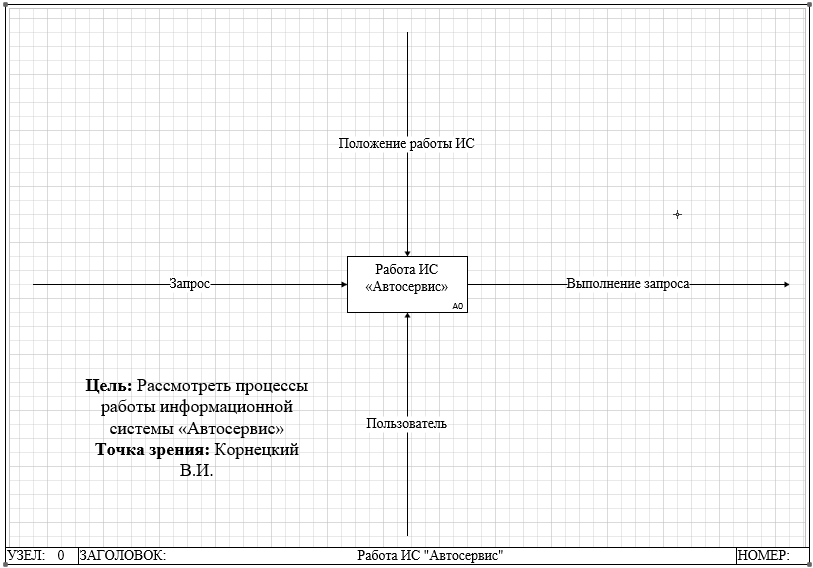


Рисунок 2 – Контекстная диаграмма модуля

Для демонстрации деталей в процессе необходимо спроектировать диаграмму декомпозиции.

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

На рисунке 3 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.

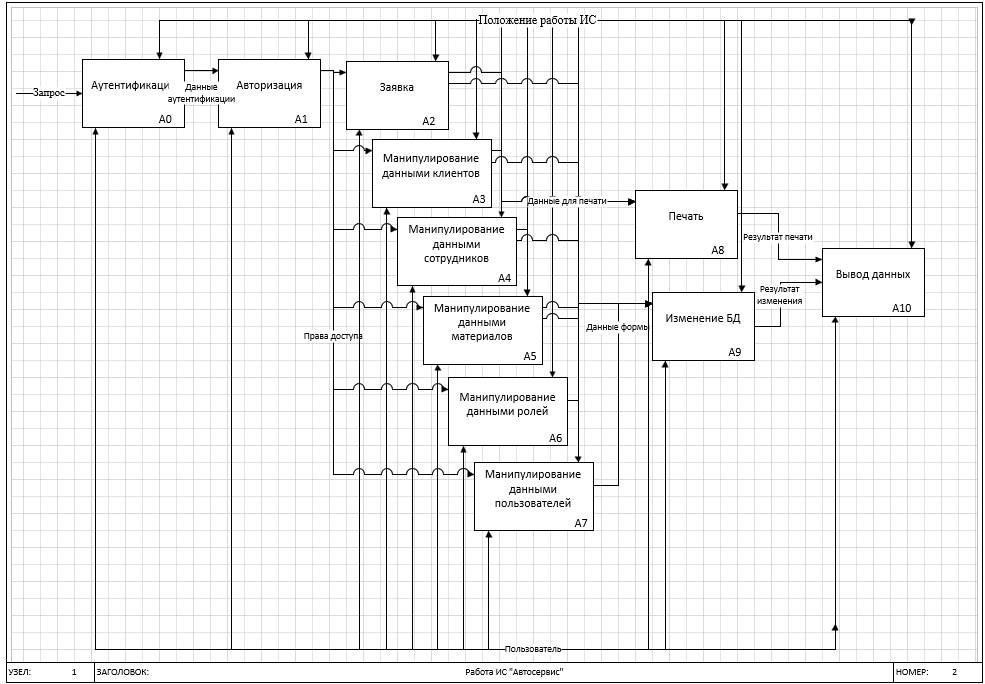


Рисунок 3 – Диаграмма декомпозиций ИС

На рисунке 4 показана диаграмма IDEF3, которая расписывает работу информационной системы из рисунка 2.

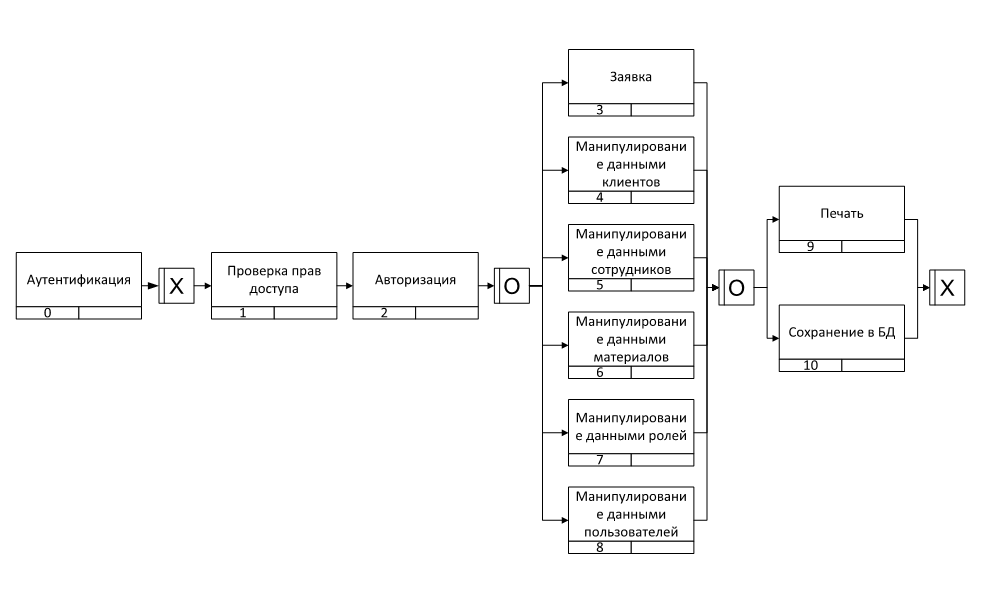


Рисунок 4 –IDEF3 диаграмма

На рисунке 5 показана диаграмма потоков данных, которая описывает передачу данных между разными функция и объектами информационной системы.



Рисунок 5 – 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 6 изображена схема базы данных информационной системы «Автосервис».

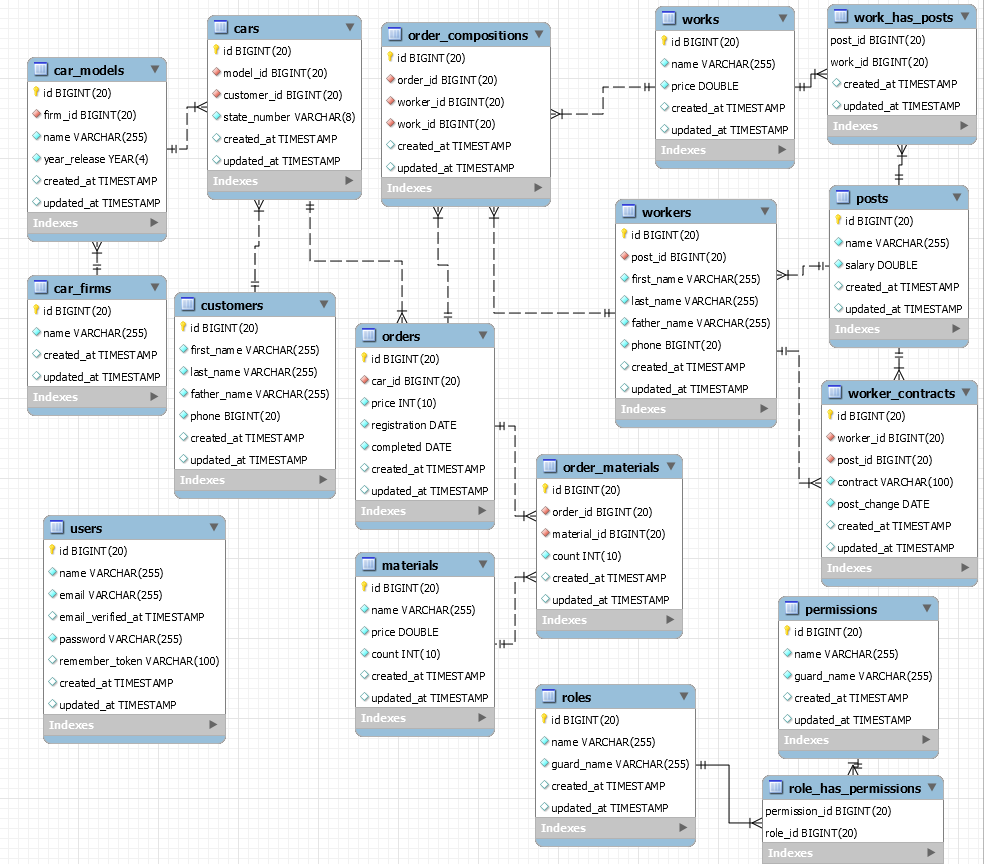


Рисунок 6 – Схема базы данных

База данных состоит из 16 связанных и 1 несвязанной таблицы. Для создания связей между таблицами используются внешние ключи.

Были выявлены следующие сущности и их атрибуты представленные в таблицах от 5 до 21.

Таблица 5 — Атрибуты сущности cars

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор автомобиля |
| model\_id | bigint | Идентификатор модели автомобиля |
| customer\_id | bigint | Идентификатор заказчика |
| state\_number | varchar (8) | Государственный регистрационный номер |

Таблица 6 — Атрибуты сущности car\_models

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор модели автомобиля |
| firm\_id | bigint | Идентификатор фирмы автомобилей |
| name | varchar (255) | Название модели автомобиля |
| year\_release | year (4) | Год выпуска модели автомобиля |

Таблица 7 — Атрибуты сущности car\_firms

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор фирмы автомобилей |
| name | varchar (255) | Название фирмы автомобилей |

Таблица 8 — Атрибуты сущности customers

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор заказчика |
| first\_name | varchar (255) | Имя заказчика |
| last\_name | varchar (255) | Фамилия заказчика |
| father\_name | varchar (255) | Отчество заказчика |
| phone | bigint | Телефон заказчика |

Таблица 9 — Атрибуты сущности works

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор выполняемой работы |
| name | varchar (255) | Название выполняемой работы |
| price | double | Стоимость выполнения работы |

Таблица 10 — Атрибуты сущности posts

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор должности |
| name | varchar (255) | Название должности |
| salary | double | Оклад должности |

Таблица 11 — Атрибуты сущности work\_has\_posts

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| post\_id | bigint | Идентификатор должности |
| work\_id | bigint | Идентификатор выполняемой работы |

Таблица 12 — Атрибуты сущности workers

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор сотрудника |
| post\_id | bigint | Идентификатор должности |
| first\_name | varchar (255) | Имя сотрудника |
| last\_name | varchar (255) | Фамилия сотрудника |
| father\_name | varchar (255) | Отчество сотрудника |
| phone | bigint | Телефон сотрудника |

Таблица 13 — Атрибуты сущности worker\_contracts

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор договора на должность |
| worker\_id | bigint | Идентификатор сотрудника |
| post\_id | bigint | Идентификатор должности |
| contract | varchar (100) | Путь до документа |
| post\_change | date | Дата изменение должности |

Таблица 14 — Атрибуты сущности orders

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор заказа |
| car\_id | bigint | Идентификатор автомобиля |
| price | int | Стоимость заказа |
| registration | date | Дата принятия заказа |
| completed | date | Дата окончания заказа |

Таблица 15 — Атрибуты сущности order\_compositions

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор работ заказов |
| order\_id | bigint | Идентификатор заказа |
| worker\_id | bigint | Идентификатор работника |

Продолжение таблицы 15

|  |  |  |
| --- | --- | --- |
| work\_id | bigint | Идентификатор работы |

Таблица 16 — Атрибуты сущности order\_materials

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор материалов заказов |
| order \_id | bigint | Идентификатор заказа |
| material\_id | bigint | Идентификатор материала |
| count | int | Количество материала |

Таблица 17 — Атрибуты сущности users

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор пользователя |
| name | varchar (255) | Имя пользователя |
| email | varchar (255) | Почта пользователя |
| email\_verified\_at | timestamp | Дата и время верификации почты |
| password | varchar (255) | Пароль пользователя |
| remember\_token | varchar (100) | Токен для запоминания пользователя |

Таблица 18 — Атрибуты сущности roles

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор роли |
| name | varchar (255) | Название роли |
| guard\_name | varchar (255) | Тип роли |

Таблица 19 — Атрибуты сущности permissions

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор права доступа |
| name | varchar (255) | Название права доступа |
| guard\_name | varchar (255) | Тип права доступа |

Таблица 20 — Атрибуты сущности role\_has\_permissions

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| permission\_id | bigint | Идентификатор права доступа |
| role\_id | bigint | Идентификатор роли |

Таблица 21 — Атрибуты сущности materials

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint | Идентификатор материала |
| name | varchar (255) | Название материала |
| price | double | Стоимость материала |
| count | int | Имеющееся 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.

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 7 показан прототип раздела «Вход», который контролирует вход только зарегистрированных пользователей.

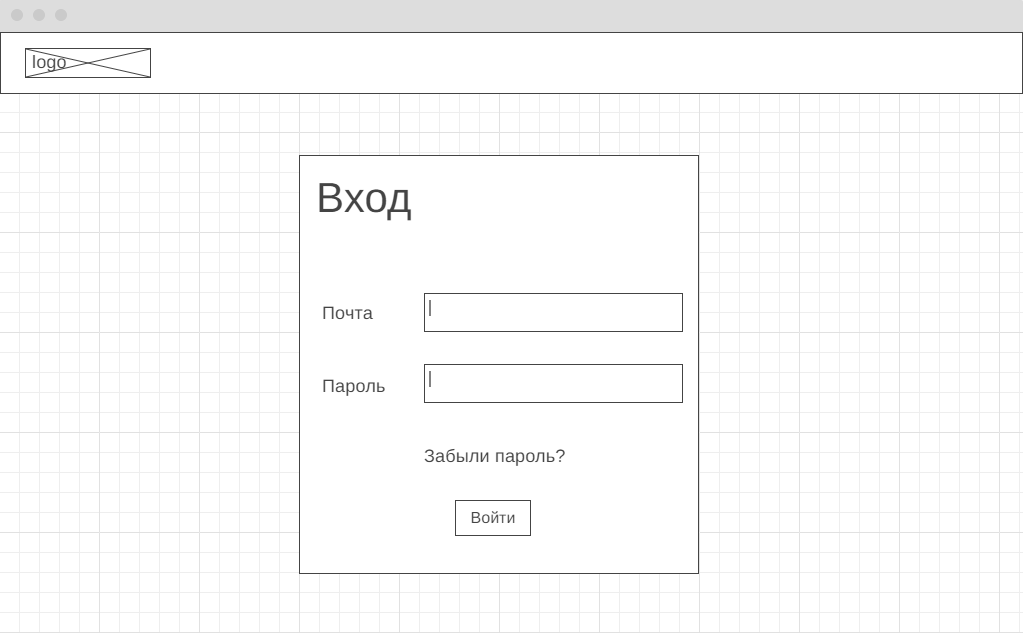


Рисунок 7 – Прототип раздела «Вход»

На рисунке 8 демонстрируется прототип основной страницы «Заказы», на которой позволяет составить заказ и распечатать заказ-наряд по заполненной форме.

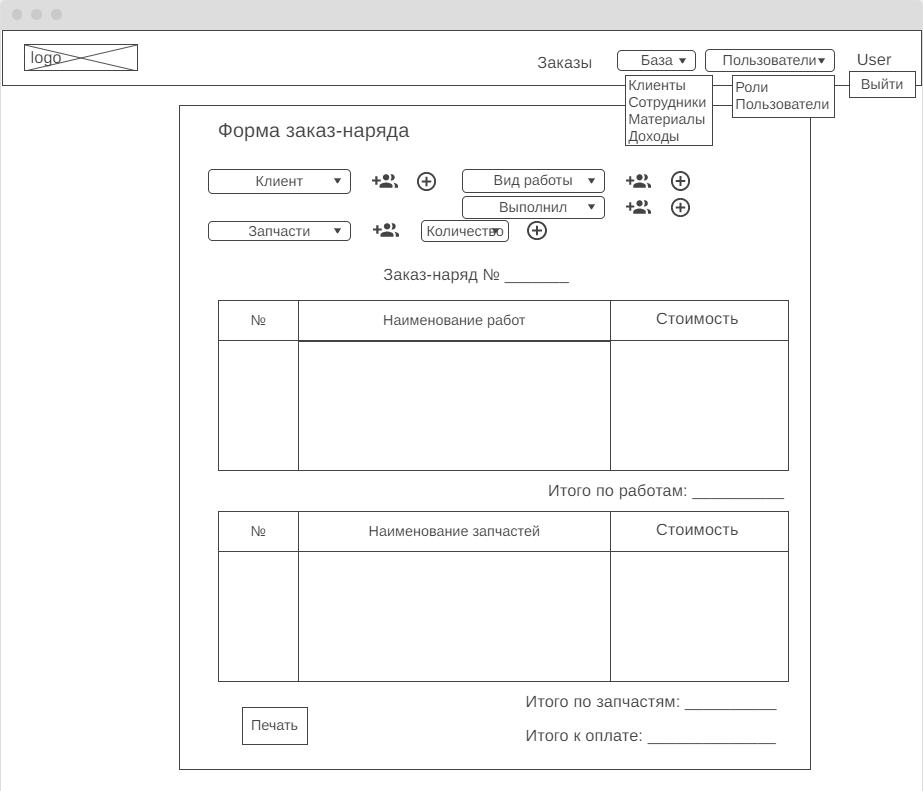


Рисунок 8 – Прототип страницы «Заказы»

На рисунке 9 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.

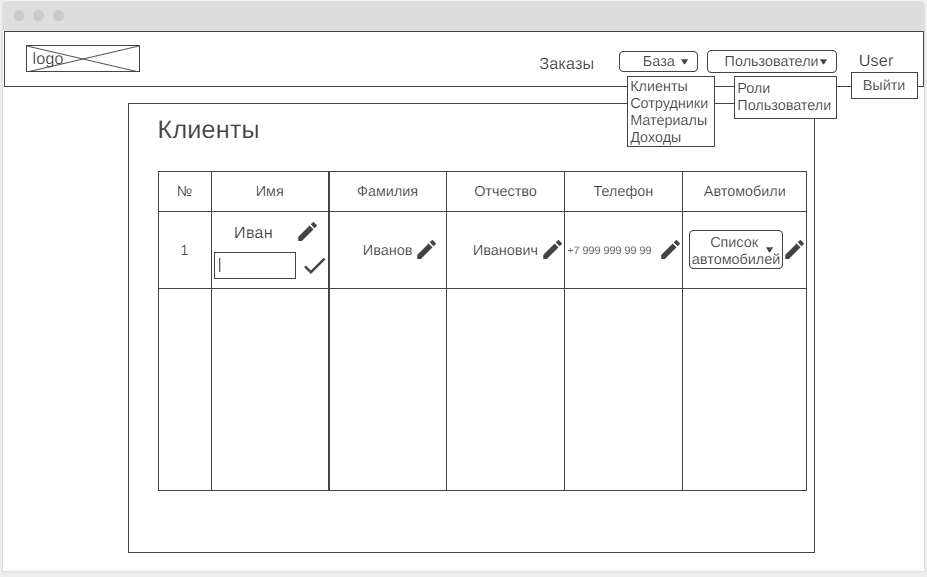


Рисунок 9 – Прототип страницы «Клиенты»

На рисунке 10 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.

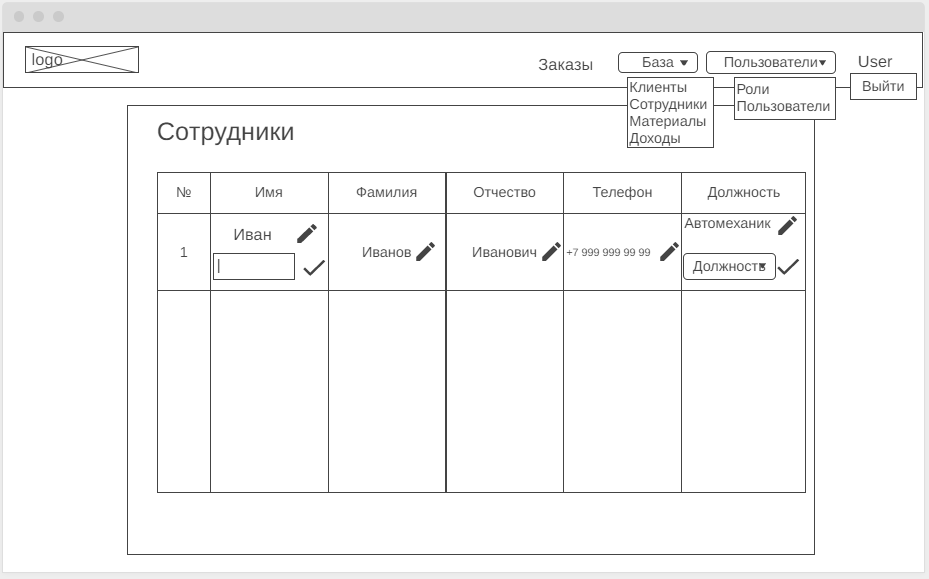


Рисунок 10 – Прототип страницы «Сотрудники»

На рисунке 11 демонстрируется прототип страницы «Регистрация», которая позволяет зарегистрировать нового пользователя системы пользователю с соответствующими правами.

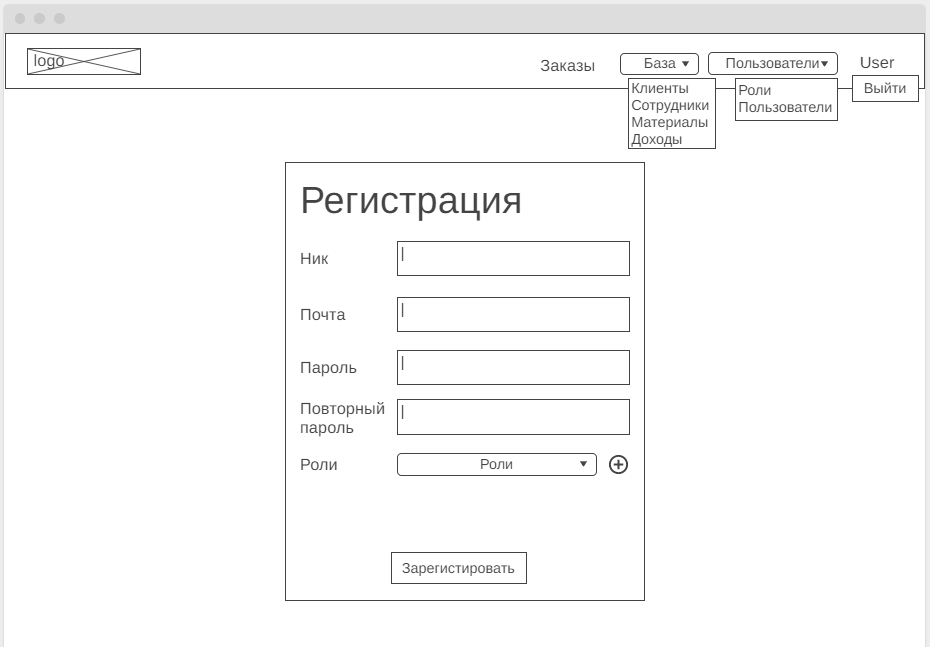


Рисунок 11 – Прототип страницы «Регистрация»

На рисунке 12 демонстрируется прототип страницы «Материалы». На рабочем пространстве размещена таблица с данными о расходных материалах, в которой имеется возможность отредактировать данные материалов, контролировать количество материалов.

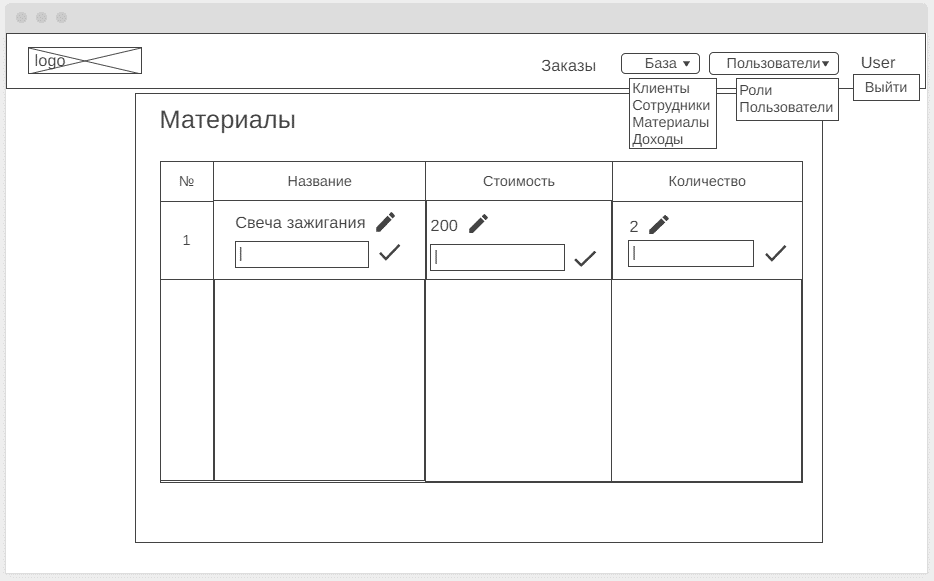


Рисунок 12 – Прототип страницы «Материалы»

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 разрабатываемая информационная система основана на веб-технологиях, в ходе разработки было принято решение использовать:

* фреймворк Bootstrap - для клиентской части;
* скриптовый язык JavaScript – для клиентской части;
* фреймворк Laravel - для серверной составляющей модуля.

Благодаря фреймворку Bootstrap информационная система «Автосервис» имеет простой, но стильный внешний вид, который продемонстрирован на рисунке 13.

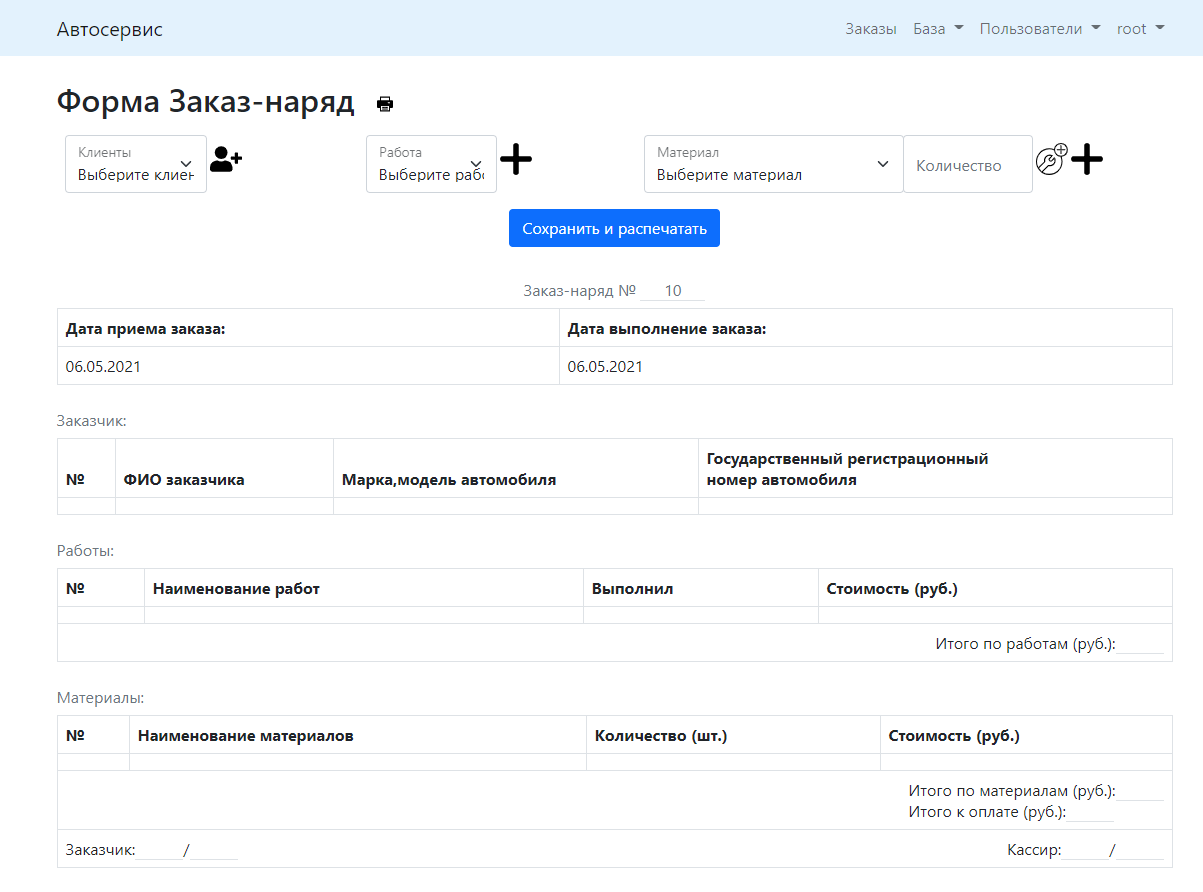


Рисунок 13 – Вкладка «Заказы»

Скриптовый язык JavaScript 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. Принцип работы, которой необходимо создавать файл контроллера, модель и представления.

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 14 представлен код функции search из контроллера OrderController.



Рисунок 14 – 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 15 изображен фрагмент кода этого представления.



Рисунок 15 – 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.

Информационная система содержит несколько крупных функций. Одна из самых крупных функций — это функция обновления таблиц (фрагмент кода на рисунке 16).

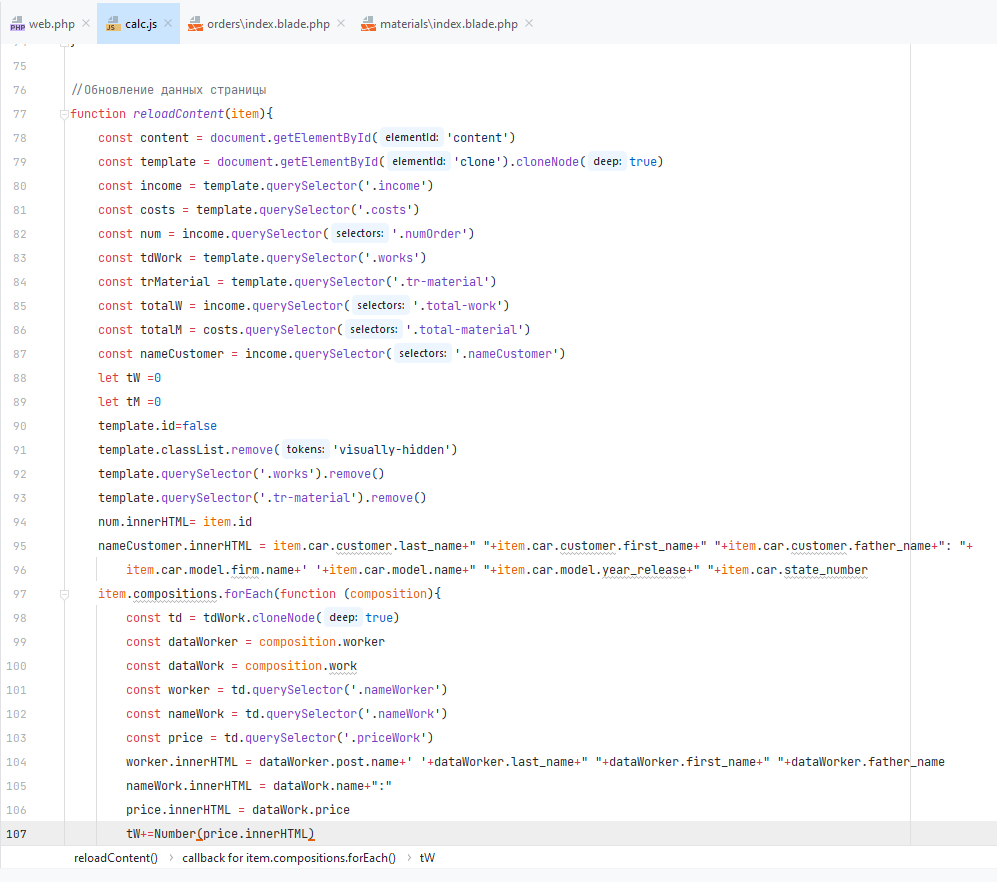


Рисунок 16 – Фрагмент функции обновления таблиц

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 17.

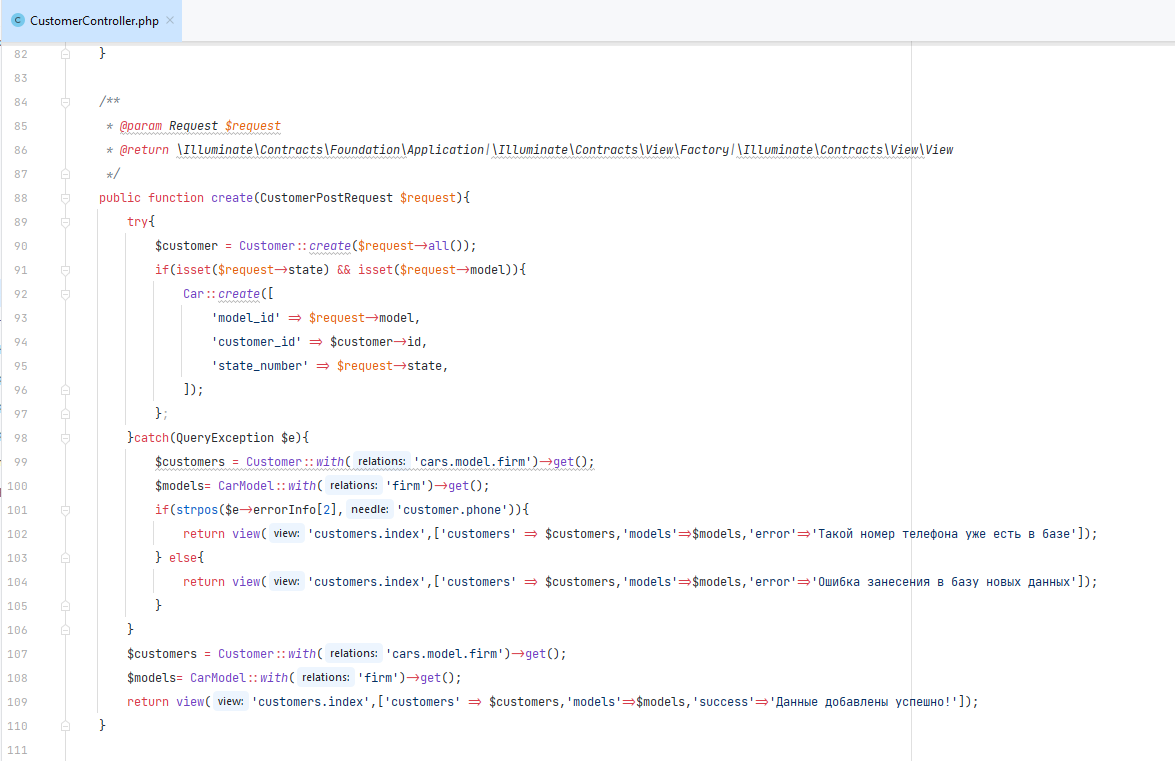


Рисунок 17 – Функция create контроллера CustomerController

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 18.

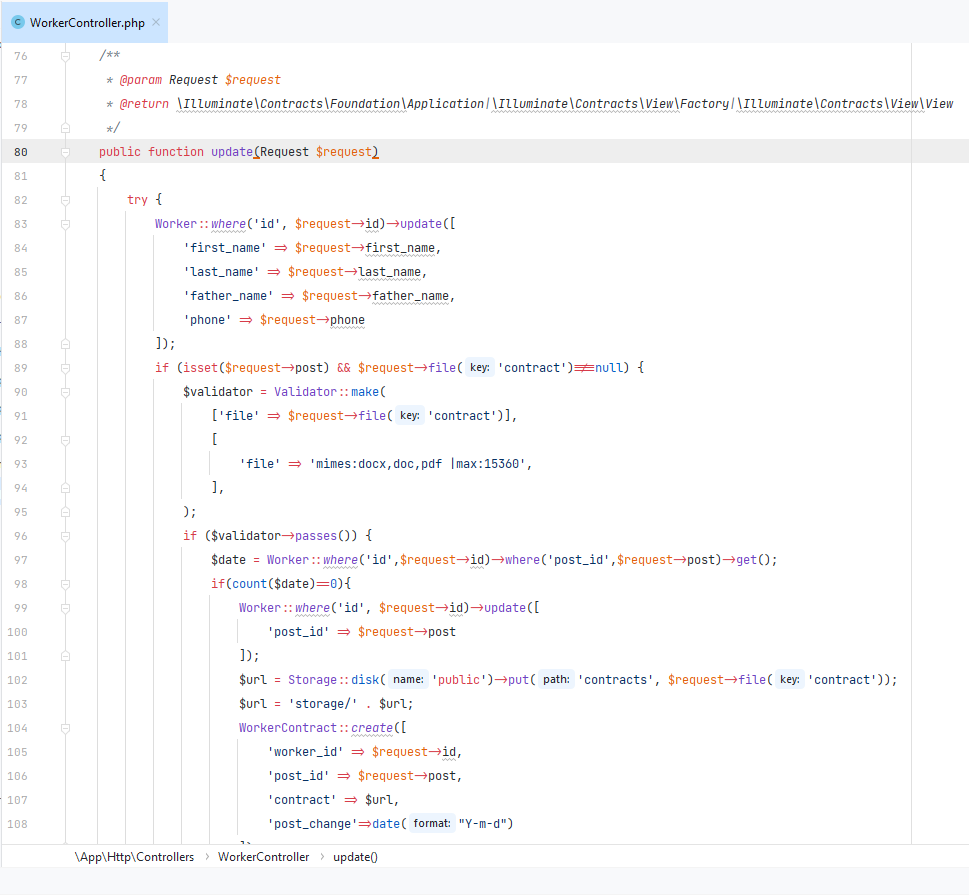


Рисунок 18 – Фрагмент функции update контроллера WorkerController

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. Документирование программного обеспечения
   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 19.

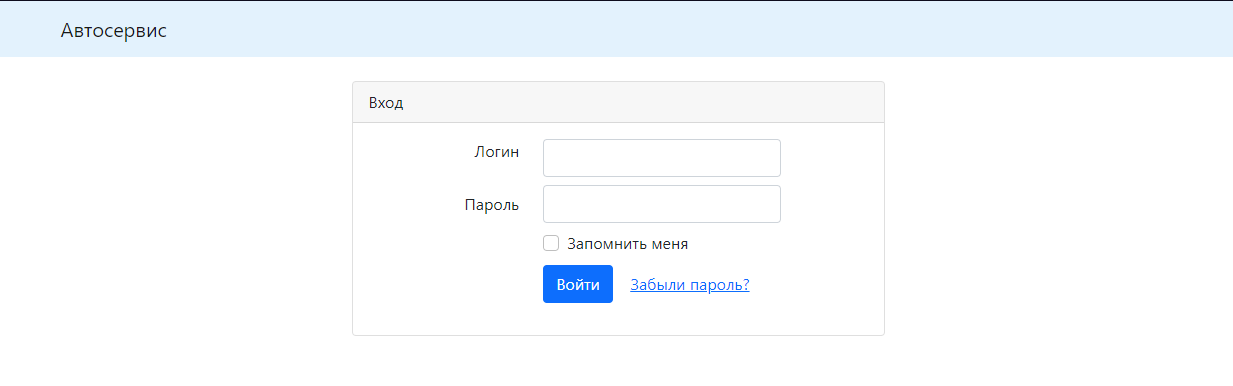


Рисунок 19 – Страница входа

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 22. 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 22 – 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 20) и предпросмотр заполняемого заказ-наряда (рисунок 21). При заполнении формы изменяется и заказ-наряд.

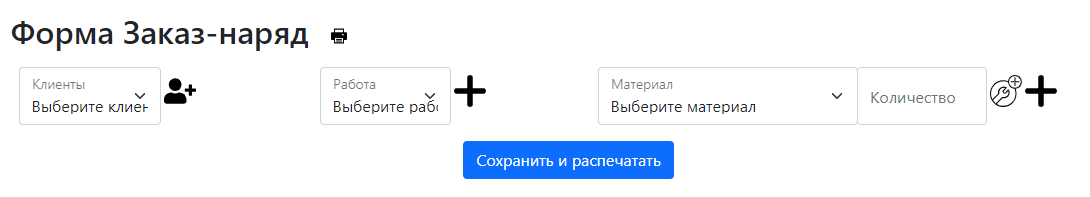


Рисунок 20 – Форма для заполнения заказ-наряда

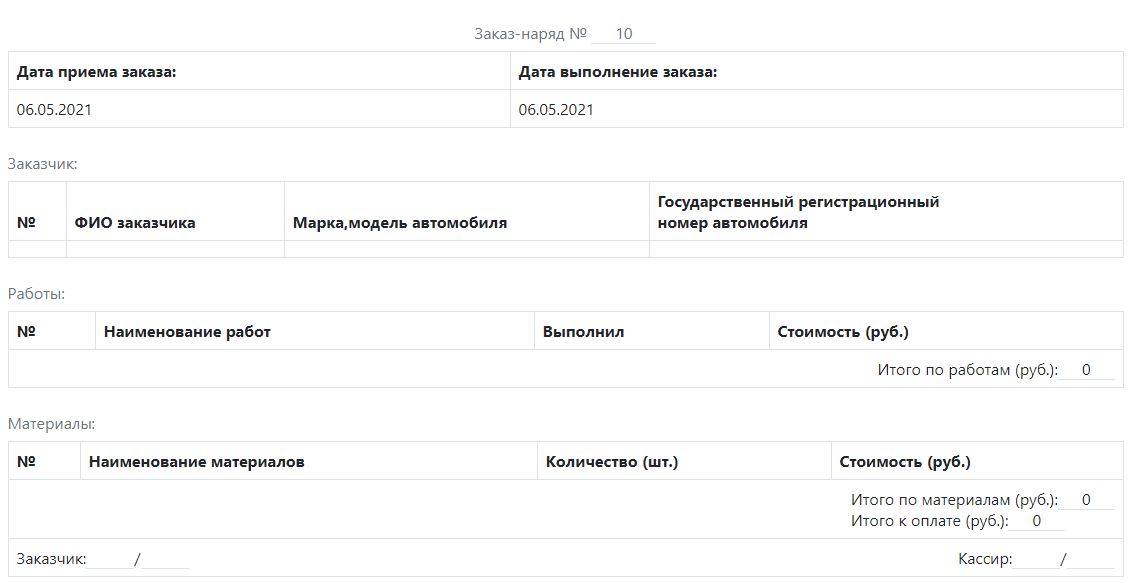


Рисунок 21 – Заполняемый заказ-наряд

Иконки в виде «плюсов» позволяют добавить в заказ-наряд ещё одну работу или материал (рисунок 22).

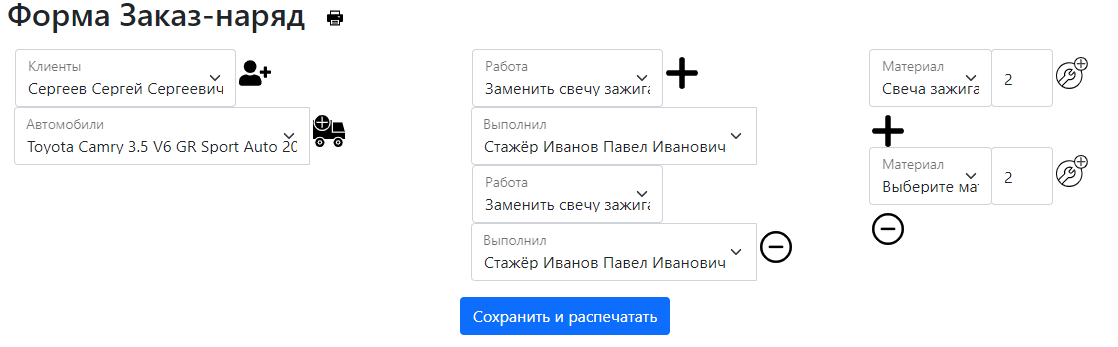


Рисунок 22 – Заполненная форма с несколькими работами и материалами

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 23).

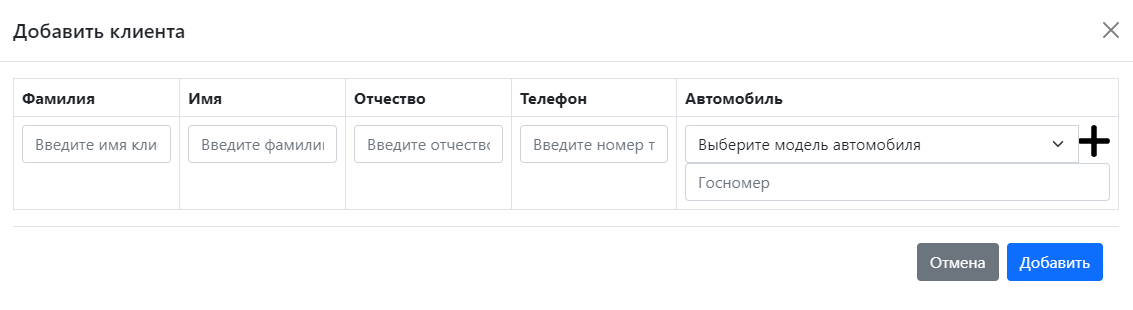


Рисунок 23 – Модальное окно нового клиента

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 24).

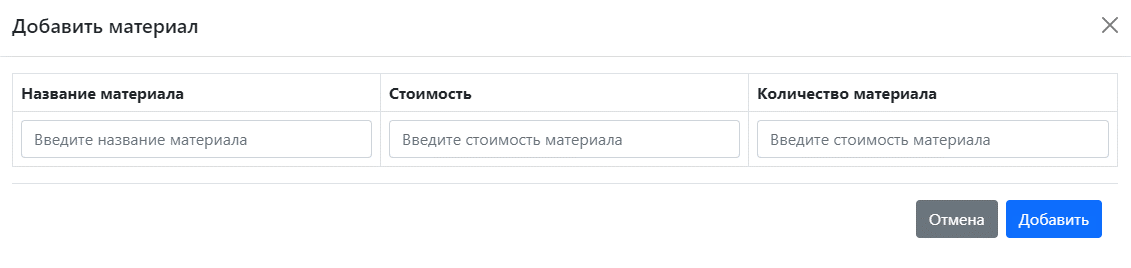


Рисунок 24 – Модальное окно нового материала

* + 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 25.

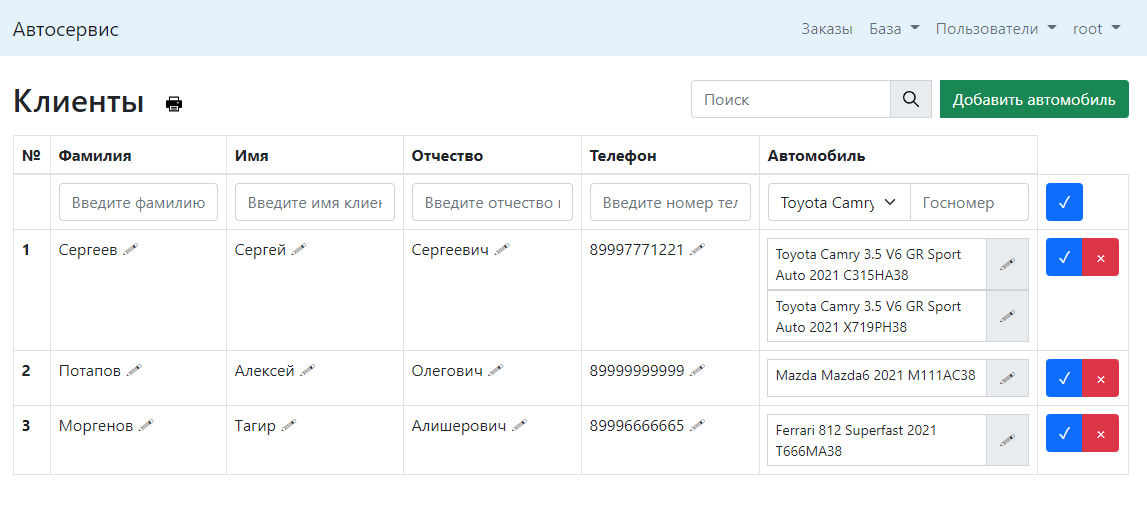


Рисунок 25 – Раздел «Клиенты»

Кнопка «Добавить автомобиль» откроет страницу «Автомобили», которое позволит добавить новую фирму, модель автомобиля, а также изменить уже существующие (Рисунок 26).

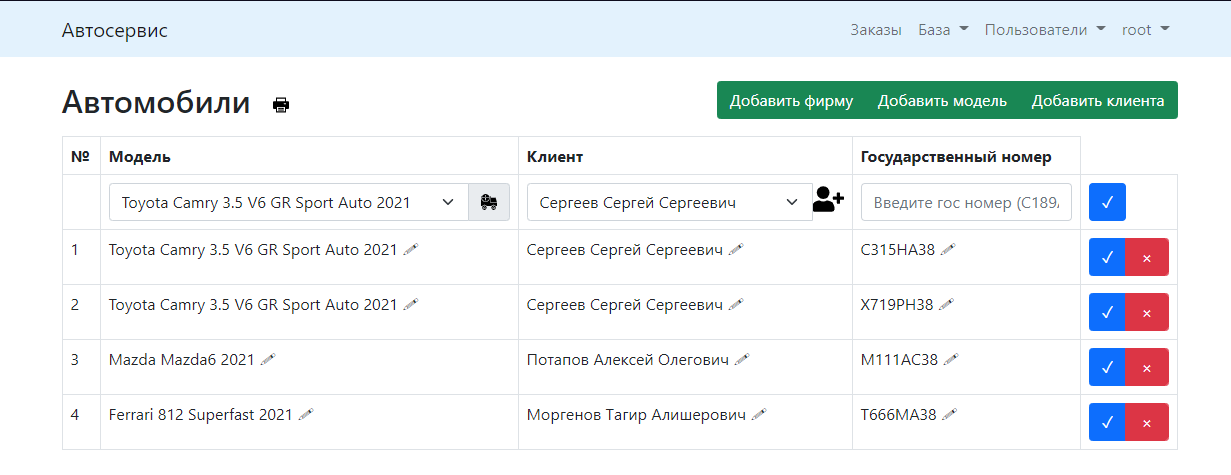


Рисунок 26 – Страница «Автомобили» раздела «Клиенты»

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 27).

Изменить название модели или фирмы можно, нажав на значок «карандаша» в столбце «Модели» (рисунок 27).

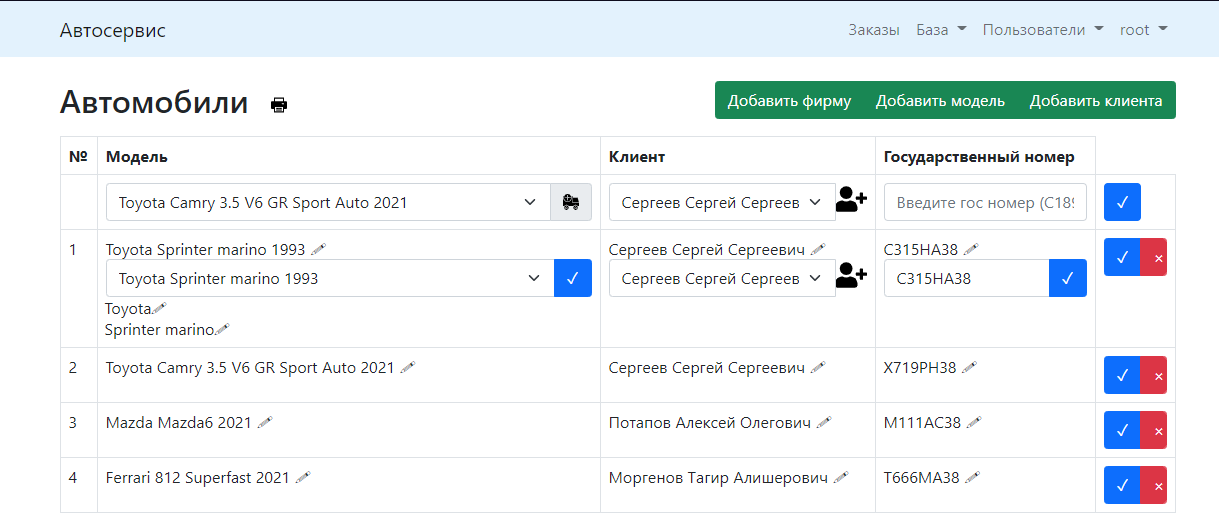


Рисунок 27 – Изменить название модели или фирмы в разделе «Клиенты»

* + 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 28.

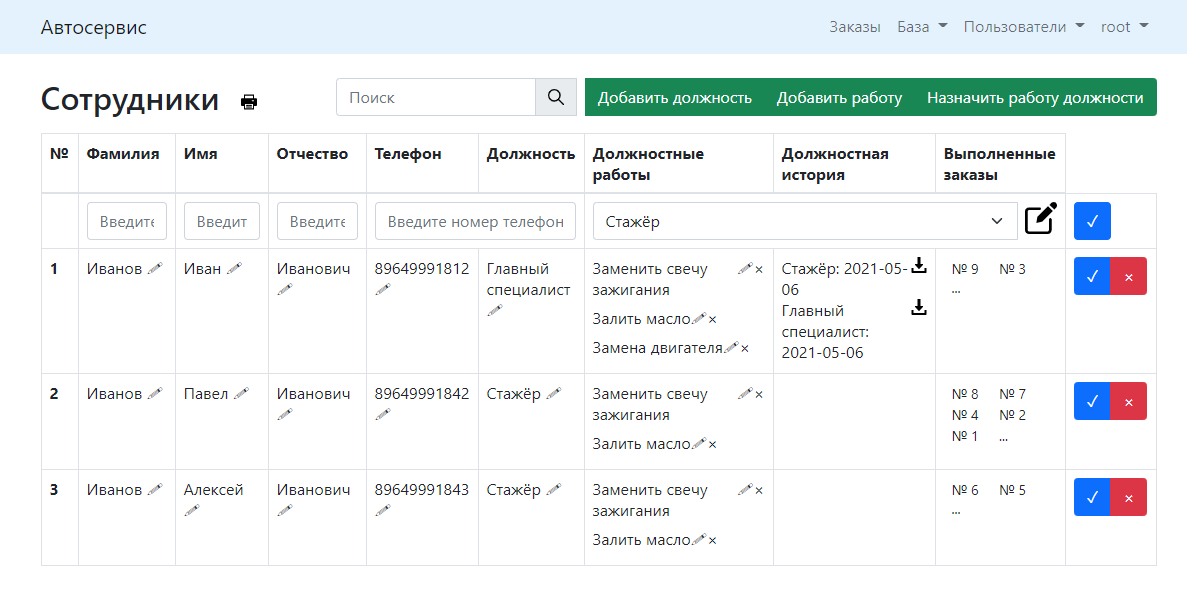


Рисунок 28 –Раздел «Сотрудники»

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 29).

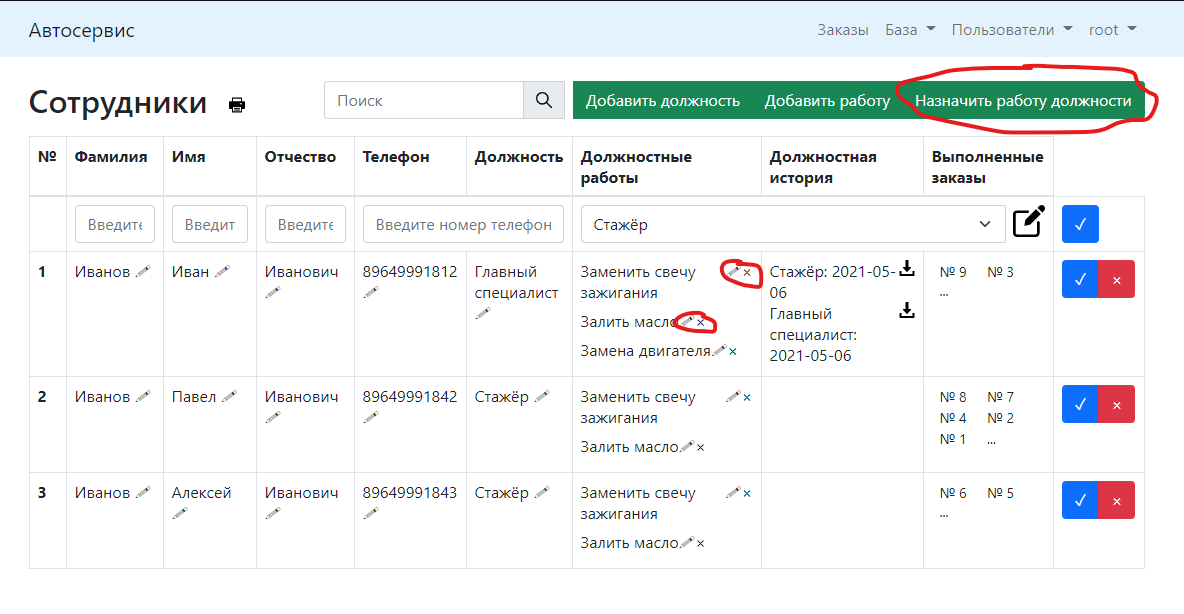


Рисунок 29 – Манипулирование работами в разделе «Сотрудники»

* + 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 30.

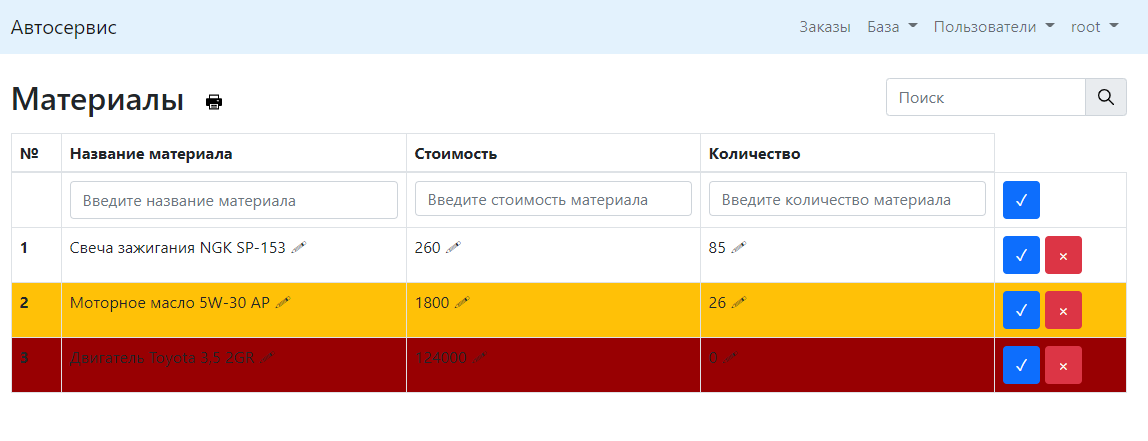


Рисунок 30 – Раздел «Материалы»

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 31).

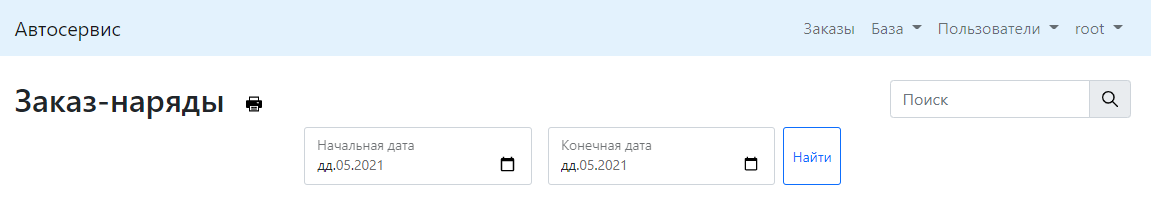


Рисунок 31 – Раздел «Доходы»

После того как период будет выбран и нажата кнопка «Найти», в разделе «Доходы» появится заказ-наряды, которые были зарегистрированы в этот период времени (рисунок 32).

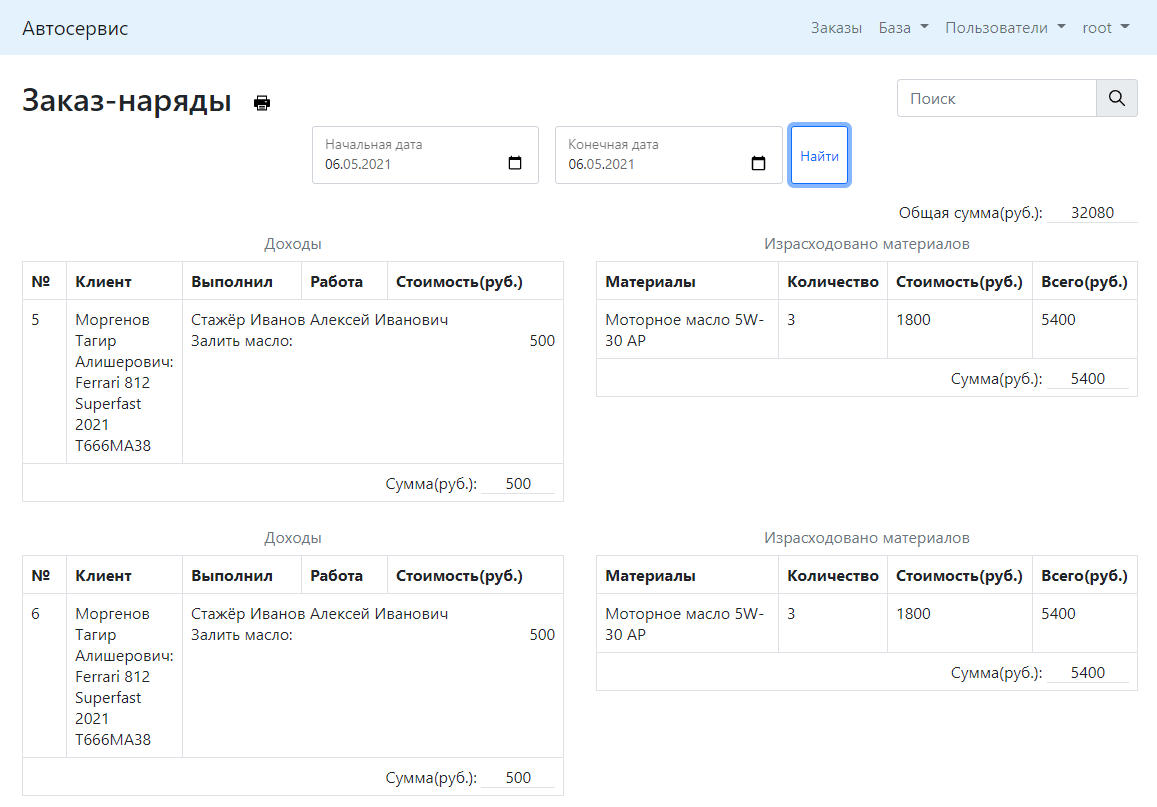


Рисунок 32 – Заказ-наряды раздела «Доходы»

* + 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 33.

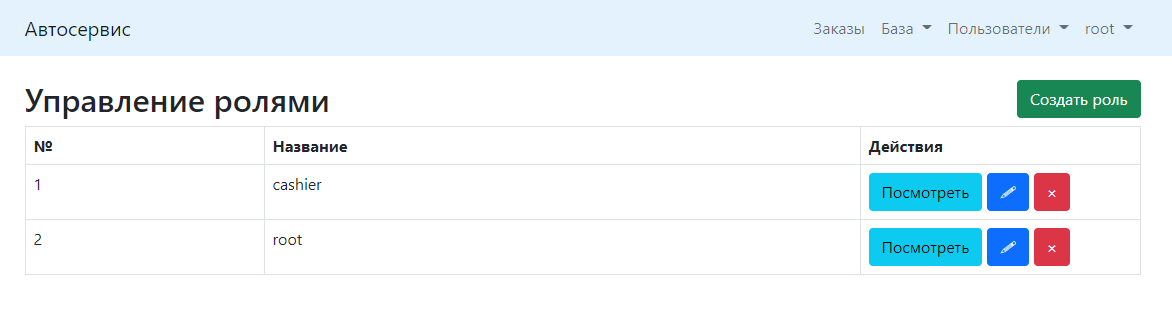


Рисунок 33 – Раздел «Роли»

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 34.

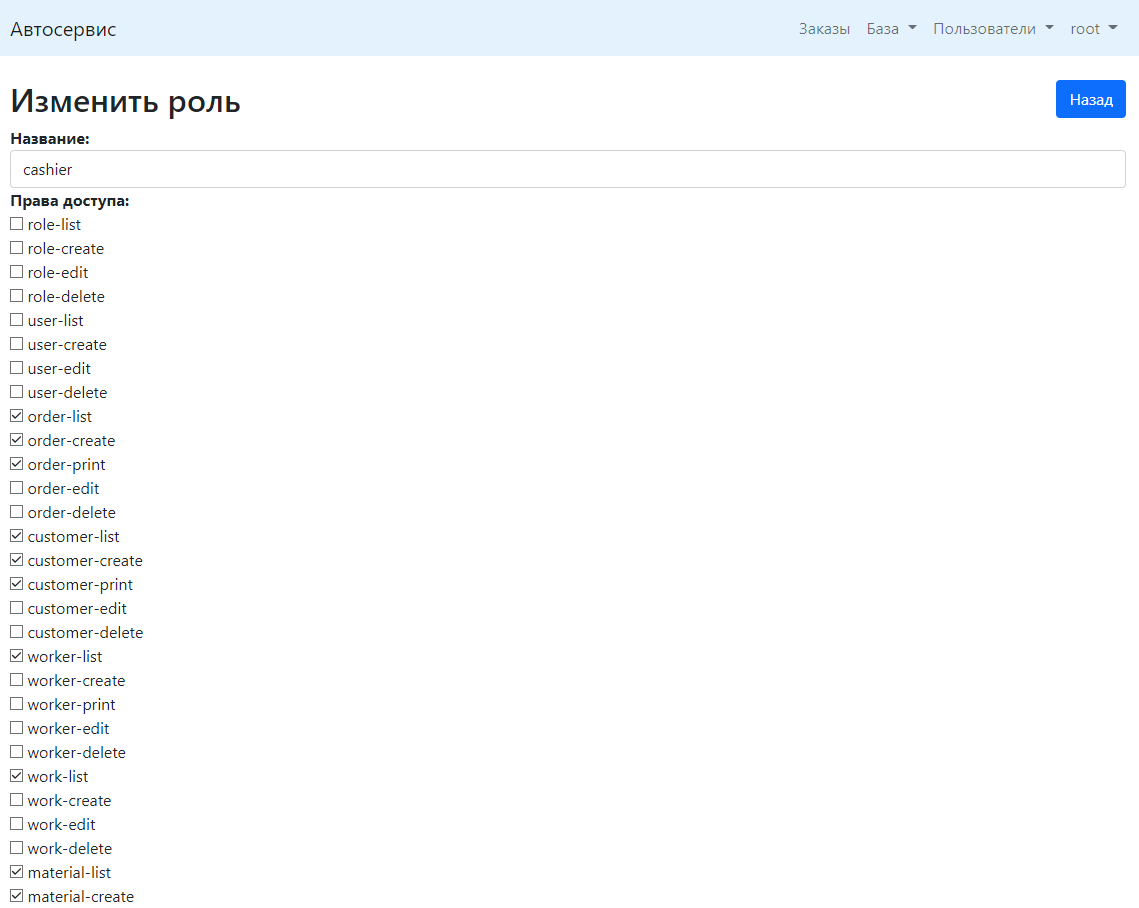


Рисунок 34 – Страница редактирования раздела «Роли»

* + 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 35.

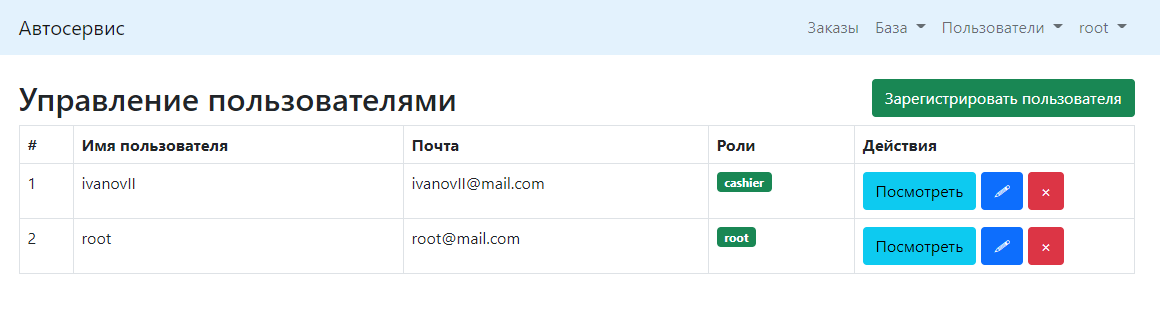


Рисунок 35 – Раздел «Пользователи»

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.

Разработанный программный продукт достаточно распространённый, из-за чего все аналоги находятся в открытом доступе. Стоимость разработки информационной системы составляет от 50 000 рублей, в зависимости от функционала.

* 1. Расчет затрат на разработку программного продукта

Расчет полных затрат на разработку проектного решения (КРПР) осуществляется по формуле 6.1:

(6.1)

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 23). 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 6.2:

(6.2)

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

Таблица 23 - 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 |
|  | 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 6.3):

(6.3)

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 6.4:

,

(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=6руб.

Затраты на электроэнергию рассчитываются по следующей формуле 6.5:

Е=W×t×T,

(6.5)

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 6.6:

(6.6)

где – первоначальная стоимость ПК, руб;

– процент амортизации в год.

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

Затраты на хозяйственно-организационные нужды приводятся в таблице 24 и вычисляются по формуле 6.7:

,

(6.7)

где Цτ – цена τ-го товара, руб.;

Кτ – количество τ-го товара.

Таблица 24 - 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 25).

Таблица 25 – 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 | 623,38 |
| Затраты на амортизацию ПК | А | 1250 |
| Затраты на хозяйственно-операционные нужды |  | 1480 |
| Итого затрат на разработку |  | 21229,58 |

* 1. Расчет затрат на внедрение программного продукта

Информационная система не внедряется в предприятие, поэтому затрат на внедрение нет. Также все используемые программные средства бесплатны и 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 7149,58 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. BizzApps [Электронный ресурс] – Режим доступа: https://bizzapps.ru/p/draw-io/ – Документация (Дата обращения: 26.04.2021).
4. Hostinger [Электронный ресурс] – Режим доступа: https://www.hostinger.ru/rukovodstva/web-server-apache/ – Что такое Apache? (Дата обращения: 06.05.21).
5. Laravel [Электронный ресурс] – Режим доступа: <https://laravel.ru/docs/v5/queries> – Конструктор запросов (Дата обращения: 10.05.21).
6. Wikipedia [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/PHP – PHP (Дата обращения: 10.05.21).
7. PHP.NET [Электронный ресурс] – Режим доступа: https://www.php.net/ – Документация по PHP (Дата обращения: 15.04.2021).
8. PHP-MyAdmin.ru [Электронный ресурс] – Режим доступа: https://php-myadmin.ru/doc/ability.html – Руководство. Возможности PhpMyAdmin (Дата обращения: 03.05.21).
9. Wikipedia [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/PHPStorm – PHPStorm (Дата обращения: 10.05.21).
10. SQL. Полное руководство. Грофф Дж. Р., Вайнберг П.Н., Оппелъ Э. Дж. 2015 – 959 с.
11. Life-prog [Электронный ресурс] – Режим доступа: https://life-prog.ru – Документация (Дата обращения: 26.04.2021).
12. Lucidchart [Электронный ресурс] – Режим доступа: https://www.lucidchart.com/pages/ru/visio-что-это – Что такое Visio? (Дата обращения: 19.04.2021).
13. Computerra [Электронный ресурс] – Режим доступа: https://www.computerra.ru/230618/7-besplatnyh-instrumentov-wireframe-dlya-proektirovaniya-interfejsov-i-protsessov/– 7 бесплатных инструментов wireframe (Дата обращения: 05.05.2021).
14. WebForMyself [Электронный ресурс] – Режим доступа: https://webformyself.com/peredacha-znachenij-peremennyx-iz-javascript-v-php-i-naoborot/ – Документация (Дата обращения: 07.04.2021).
15. Habr [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/ruvds/blog/539098/ – 9 самых популярных php-фреймворков (Дата обращения: 18.04.2021).
16. Javascript [Электронный ресурс] – Режим доступа: https://learn.javascript.ru/manuals-specifications – Справочники и спецификации (Дата обращения: 03.04.2021).

# 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);});