

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем
Дисциплина «Программное обеспечение мобильных систем»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсового проекта
Ассистент кафедры ПИКС
_____._____.2023 К.С. Крез

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему:
«ПРОГРАММНОЕ СРЕДСТВО АВТОСАЛОНА»

БГУИР КП 1-39 02 01 016 ПЗ

Выполнил студент группы 113802
РАЗУМОВ Дмитрий Александрович

(подпись студента)
Курсовой проект представлен на
проверку _____._____.2023

(подпись студента)

Минск 2023

РЕФЕРАТ

БГУИР КП 1-39 02 01 016 ПЗ

Разумов, Д.А. Программное средство автосалона: пояснительная записка к курсовой работе / Д.А. Разумов. – Минск: БГУИР, 2023. – 71 с.

Пояснительная записка 69 с., 64 рис., 0 табл., 30 источников, 3 приложения.

ПРОГРАММНОЕ СРЕДСТВО, УЧЕТ АВТОМОБИЛЕЙ, УЧЕТ ВАЖНЫХ ДАННЫХ О СОТРУДНИКАХ, УЧЕТ ВАЖНЫХ ДАННЫХ О КЛИЕНТАХ, УЧЕТ ЗАКАЗОВ И АВТОМОБИЛЕЙ.

Цель проектирования: разработка программного средства автосалона, предназначенного для упрощения и автоматизации процессов работы автосалона и введения учета о продажах, клиентах, автомобилях и сотрудниках.

Методология проведения работы: в процессе решения поставленных задач использованы принципы системного подхода, был проведен анализ потребностей автосалона, был реализован необходимый функционал.

Результаты работы: выполнен анализ потребностей и требований сотрудников автосалона, рассмотрено общетехническое обоснование разработки программного средства, осуществлена проверка исключительных ситуаций, уделено внимание вопросам корректной работы программы, разработана графическая часть проекта.

Программное средство обеспечивает учет данных о клиентах, сотрудниках, автомобилях и продажах в автосалоне.

Структуру функционала пользователь определяет и задает путем прохождения авторизации. Авторизация предполагает задание прав, которые должны соответствовать регалиям сотрудников согласно их должности.

Область применения результатов: могут быть использованы при проектировании аналогичных программных средств для смежных областей с повышенной надежностью работы.

СОДЕРЖАНИЕ

Введение.....	5
1 Анализ исходных данных на курсовое проектирование	6
1.1 Анализ исходных данных к курсовой работе	6
1.2 Обоснование и описание выбора языка программирования, средств раз работки, используемых технологий и сторонних библиотек	7
2 Проектирование и разработка программного средства.....	12
2.1 Проектирование объектной модели и описание состояний программного средства	12
2.2 Проектирование и разработка графического интерфейса	19
2.3 Описание и реализация используемых в программном средстве алгоритмов	29
3 Эксплуатация программного средства.....	36
3.1 Ввод в эксплуатацию и обоснование минимальных технических требований к оборудованию	36
3.2 Руководство по эксплуатации программного средства	41
Заключение	57
Список использованных источников	58
Приложение А (обязательное) Отчет о проверке на заимствование в системе «Антиплагиат».....	60
Приложение Б (обязательное) Листинги кода с комментариями	61
Приложение В (обязательное) Ведомость курсовой работы.....	69

ВВЕДЕНИЕ

Как всем известно, любой бизнес начинается с продаж. Множество различных товаров, заказов, клиентов, сотрудников и т.д. В связи с этим и было разработано программное средство в рамках данной курсовой работы, направленное на помощь в автоматизации деятельности автосалона.

Данное программное средство было разработано на языке программирования PHP, что позволит беспрепятственно запускать разработанную программу на разных операционных системах: Windows, Linux, Mac OS X и другие.

Целью данной курсовой работы является создание программного средства, которое могло бы помочь автоматизировать деятельность работников автосалона и свести временные затраты на учет автомобилей. Также были поставлены задачи обеспечить разносторонний функционал, при помощи которого начальство сможет взаимодействовать с данными не только о продажах и авто, но и о самих сотрудниках и клиентах.

При выполнении курсового проекта были применены государственные стандарты и нормативные документы, а также СТП БГУИР 01-2017 [1]. В пояснительной записке будут представлены три раздела: анализ исходных данных на курсовое проектирование, проектирование и разработка программного средства, эксплуатация программного средства.

Перечень графического материала данной курсовой работы будет включать в себя четыре диаграммы: схема алгоритма, UML диаграмма классов, диаграмма состояний и структура графического пользовательского интерфейса.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет 92.94%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников».

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

1.1 Анализ исходных данных к курсовой работе

В рамках данной курсовой работы рассматривается предметная область «Программное средство автосалона». В современном мире, в условиях, когда автомобиль является не роскошью, а постоянным средством передвижения, то практически каждый человек нуждается в нем. За частую, если человек не испытывает финансовые трудности, то он обращается в автосалон за покупкой автомобиля. Эта информация обширна и разрознена. Чтобы вести учет всех автомобилей, которые можно купить, в организации имеется потребность в структурировании данных об авто и заказах. Отсутствие такой возможности приводит к проблеме утери данных и большим временным затратам на выборку данных.

Функционирование автосалона без автоматизации процесса достаточно трудоемкая работа. При автоматизации значительно сократиться время и осуществление основных операций автосалона. Автосалон располагает современной материальной базой, включающей вычислительную технику.

Автосалон – важная часть структуры каждого автомобильного бренда. Они представляют собой специальные магазины, офисы, дилеров, осуществляющие посреднические функции для каждого бренда [2]. В большинстве случаев автосалоны являются импортерами каждого бренда автомобиля, в отдельной стране, а также в тех странах, которым принадлежит данный бренд автомобиля, автосалоны являются государственными учреждениями, но при этом имеют большое число посреднических фирм, эффективность деятельности которых очень высока.

Данная категория включает в себя исторически сформировавшийся экономический механизм, который осуществляет определенный комплекс услуг и отношений, содействует установлению и соблюдению баланса интересов между клиентами, автоконцернами (заводами), бизнесом и государством.

Автосалон – юридическое лицо, имеющее зарегистрированный товарный знак, фирменный знак, расчетный и иные счета в рублях и иностранной валюте в учреждениях банков. Автосалон имеет в своем составе администрацию, бухгалтерию, отдел кадров, отдел маркетинга и менеджмента, магазинов.

Автомобильный бренд может быть представлен разными «фигурами» в зависимости от того, в чье имущество входит бренд. Им могут быть: отдельный человек или компания, которая является создателем данного бренда и имеет на него большую часть прав (Tesla Илона Маска), но в большинстве случаев это государственные большие автоконцерны (BMW Германия).

Клиенты – это люди, которые включены в процесс приобретения нового автомобиля по своим личным целям. В свою очередь автосалоны и автоконцерны являются организаторами процесса продажи и производства автомоби-

лей. Клиенты, соответственно, люди разного пола и возраста, с абсолютно разным уровнем заработка, с собственным вкусом и предпочтением, для которых необходим широкий выбор как в ценовом диапазоне, так и в большом выборе моделей каждого бренда. Для каждого покупателя автомобиль имеет разную ценность, поскольку все люди различаются по признакам возраста, пола, личного заработка и уровня жизни, и у каждого эксплуатация автомобиля разная.

Ограничениями каждого автосалона являются:

- продажа авто в кредит или рассрочку людям неспособным выплатить данную сумму в срок;
- продажа авто лицам младше 18 лет;
- оформление авто на людей, не имеющих водительского удостоверения;

Главными мерками успешной покупки автомобиля из салона является:

- высококлассная подготовка, позволяющая менеджеру правильно изложить всю информацию, чтобы облегчить выбор покупателю;
- большой выбор автомобилей;
- минимальное представление, что он ожидает от каждого автомобиля.

В каждом автосалоне развернута локальная вычислительная сеть (ЛВС), которая позволяет осуществлять централизованное хранение и обработку информации. Сеть охватывает все отделы.

1.2 Обоснование и описание выбора языка программирования, средств разработки, используемых технологий и сторонних библиотек

В данной курсовой работе использованы:

- языки программирования PHP и JavaScript;
- фреймворк Laravel;
- язык гипертекстовой разметки текста HTML;
- каскадные таблицы стилей CSS, а также препроцессор SCSS;
- локальный веб-сервер OpenServer;
- база данных MySQL и инструмент взаимодействия PHPMyAdmin;
- система контроля версий Git и онлайн-сервис GitHub;
- сборщик GULP для стилей, скриптов и изображений;
- интегрированная среда разработки VSCode.

PHP – строго типизированный объектно-ориентированный язык программирования общего назначения [3]. Язык программирования PHP спроектировал датский программист Расмус Лердорф в 1995 году как инструмент для создания интерактивных и динамических веб-сайтов. Основная область применения – разработка скриптов, которые работают на стороне сервера.

PHP.net – это официальный веб-сайт языка программирования PHP, который широко используется для создания веб-сайтов и веб-приложений [4]. Этот веб-сайт предоставляет обширную документацию, ресурсы, и информацию о PHP, включая официальную документацию по языку.

PHP является языком программирования, построенным на принципах объектно-ориентированной парадигмы. Это означает, что разработка проекта на PHP будет организована вокруг объектов, что упрощает разработку, обеспечивает повторное использование кода и делает его более подходящим для создания крупных и сложных проектов.

PHP имеет встроенные механизмы безопасности, которые сводят к минимуму риск возникновения уязвимостей и защищают от нежелательных взаимодействий с внешними компонентами. Это особенно важно для проектов, связанных с обработкой и хранением конфиденциальной информации.

Выбор языка программирования PHP для разработки веб-сайта можно обосновать множеством факторов. PHP – это язык программирования, который изначально разрабатывался с учетом платформонезависимости. Это означает, что веб-сайты могут быть запущены на различных операционных системах без изменений в исходном коде. Эта особенность крайне важна при разработке веб-сайтов, так как они могут быть запущены на разных версиях Windows, Linux или macOS.

Ключевые возможности PHP:

- динамические веб-страницы;
- обработка форм;
- работа с базами данных;
- работа с файлами;
- создание сессий;
- создание API;
- интеграция с другими технологиями;
- множество библиотек и фреймворков;
- кросс-платформенность.

Laravel – это бесплатный PHP-фреймворк с открытым исходным кодом, специально разработанный для создания сложных сайтов и веб-приложений [5]. Позволяет упростить аутентификацию, маршрутизацию, сессии, кэширование, архитектуру приложения, работу с базой данных.

Назначение – создание веб-приложений и сайтов на основе MVC [6][7]. Это вариант архитектуры, при котором компоненты программы делятся на три части:

- модель (model) представляет данные и методы работы с ними: запросы в базу данных, проверка на корректность;

- представление (view) показывает пользователю эти данные и изменяется, если меняется модель;
- контроллер (controller) направляет данные от пользователя к системе и наоборот.

Когда пользователь работает с приложением, построенным по схеме MVC, он взаимодействует с представлением и контроллером. Представление – это то, что он видит, например сведения, которые отображены в визуальном интерфейсе. А контроллеру пользователь отдает команды.

У Laravel есть активное сообщество разработчиков и обширная документация, что облегчает получение поддержки и решение возникающих вопросов.

HTML – это язык гипертекстовой разметки текста [8]. Это набор команд, следуя которым, браузеры выводят на монитор различные документы и странички сайтов.

CSS – формальный язык декорирования и описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки [9]. CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS является ограждение и отделение описания логической структуры веб-страницы от описания внешнего вида этой веб-страницы. Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. В данной работе в режиме разработки будет использоваться фреймворк SCSS [10], для более гибкого и быстрого написания стилей, который в итоге при помощи сборщика соберется в один css файл.

JS – это язык программирования, который используется для создания интерактивных веб-страниц и приложений [11]. Он выполняется на стороне клиента (в браузере) и позволяет добавлять динамические элементы на страницу, изменять содержимое и стиль элементов, обрабатывать события пользователя, делать анимации и многое другое.

Так же существует множество мощных интегрированных сред разработки (IDE) [12] для веб-сайтов с помощью PHP, JS, HTML, CSS, таких как VSCode, IntelliJ IDEA, WebStorm и другие. Они предоставляют разработчикам удобные инструменты для создания, отладки и тестирования приложений. При разработке программного средства для данной курсовой работы выбор пал на VSCode.

VSCode предоставляет множество встроенных операций рефакторинга, таких как переименование переменных, извлечение методов, оптимизация импортов и многое другое. Эти инструменты облегчают процесс изменения структуры и улучшения читаемости вашего кода. IDE предлагает контекстно-

зависимые подсказки и авто дополнение кода, что ускоряет написание кода и помогает избежать опечаток и ошибок.

IDEA предоставляет интуитивный интерфейс для создания новых проектов, а также позволяет импортировать существующие проекты. VSCode интегрируется с системами контроля версий, такими как Git, Subversion, и Mercurial, обеспечивая удобное отслеживание изменений в коде.

VSCode не только поддерживает язык программирования PHP, но также имеет мощные функции для разработки других языков программирования, включая Kotlin, JavaScript, Python и многие другие [13]. Это обеспечивает разработчикам универсальный инструмент для работы с различными языками программирования в рамках одной среды разработки.

Также IDEA включает инструменты для работы с базами данных и SQL-запросами. Это дает возможность создавать и редактировать таблицы, выполнять запросы и многое другое, не покидая среду разработки.

Стоит отметить, что VSCode предлагает множество настраиваемых параметров, которые позволяют разработчикам адаптировать среду под свои потребности и предпочтения. Кроме того, с помощью плагинов и расширений разработчики могут расширять возможности VSCode и добавлять дополнительные функции и инструменты по мере необходимости.

Отдельным плюсом данной среды разработки является тот факт, что IDEA поддерживает различные серверы приложений, такие как OpenServer, Lando облегчая развертывание и отладку приложений.

В качестве сервера будет использоваться OpenServer [14]. OpenServer – это программное обеспечение для создания локального веб-сервера на компьютере разработчика. Это позволяет разрабатывать и тестировать веб-приложения на локальном сервере, без необходимости загрузки на удаленный сервер. OpenServer включает в себя все необходимые компоненты для работы веб-сервера, такие как Apache, PHP, MySQL, phpMyAdmin и другие. Он также имеет удобный интерфейс для управления сервером и настройки его параметров. OpenServer очень полезен для веб-разработчиков, которые работают над проектами на PHP или других языках программирования, используемых для создания веб-приложений. Он позволяет им быстро создавать и тестировать приложения на локальном сервере, что ускоряет процесс разработки и улучшает качество кода. Кроме того, OpenServer может использоваться для создания и тестирования локальных копий веб-сайтов, что полезно для тестирования новых функций и обновлений перед их загрузкой на удаленный сервер.

Из баз данных будет использоваться MySQL [15], а в качестве инструмента для управления базой данных PHPMyAdmin [16]. MySQL – это система управления реляционными базами данных, которая используется для хранения и управления данными на веб-сервере. Она позволяет создавать, изменять

и удалять таблицы, а также выполнять запросы к данным. PHPMyAdmin – это инструмент для управления базами данных MySQL через веб-интерфейс. Он позволяет пользователям просматривать, изменять и удалять данные в базе данных, а также выполнять различные операции с таблицами и запросами. MySQL и PHPMyAdmin очень полезны для веб-разработчиков, которые работают с базами данных на своих локальных серверах. Они позволяют легко управлять базами данных и выполнять различные операции, такие как создание таблиц, добавление данных и выполнение запросов. Кроме того, MySQL и PHPMyAdmin могут использоваться для создания и управления базами данных на удаленных серверах, что полезно для работы с большими проектами и хранения большого объема данных.

Git – это система контроля версий, которая позволяет отслеживать изменения в коде и управлять ими [17]. Она используется для хранения и управления кодом проектов, а также для совместной работы над проектами в команде. GitHub – это онлайн-сервис, который предоставляет хостинг для репозиторий Git [18]. Он позволяет пользователям хранить свои репозитории на удаленном сервере, делиться ими с другими пользователями и работать над проектами в команде. Кроме того, Git и GitHub могут использоваться для хранения документации, управления задачами и проектами, а также для автоматизации процесса разработки [19].

Gulp – это инструмент автоматизации задач для разработки веб-приложений и веб-сайтов [20]. Он позволяет автоматически выполнять рутинные задачи, такие как компиляция CSS и JavaScript файлов, оптимизация изображений, сборка проекта и т.д. Он также имеет множество плагинов, которые расширяют его функциональность и позволяют выполнять различные задачи. Gulp может быть использован для ускорения процесса разработки, уменьшения количества ошибок и повышения качества кода.

Из всего вышеперечисленного можно сделать вывод о том, что PHP является очень сильным языком программирования для разработки программного средства с нуля. Наличие ранее описанных мощных и многофункциональных библиотек, взаимодействие с HTML, CSS, JS, а также крайне удобной среды разработки с множеством дополнительных функций, ускоряющих процесс разработки за счет своей производительности, предоставляет огромное количество возможностей для реализации массивного функционала программного средства. Также данное программное средство имеет огромный плюс в виде своей адаптивности под разные операционные системы, что позволит не испытывать проблем с его поддержкой в будущем.

2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

2.1 Проектирование объектной модели и описание состояний программного средства

Проектирование объектной модели является важным этапом разработки программного средства. В данной курсовой работе для разработки серверной части проекта используются различные паттерны, которые могут быть использованы для создания гибкой и расширяемой системы.

Один из ключевых паттернов – это *Dependency Injection (DI)*, который позволяет управлять зависимостями между классами [21]. Вместо того, чтобы создавать объекты вручную и передавать им необходимые зависимости, они автоматически инжектируются при создании объекта. Объекты, которые создаются и конфигурируются контейнером, называются бинами.

По умолчанию бина создается с использованием такого шаблона проектирования как *Singleton* [22]. Управление осуществляется только одним общим экземпляром бина-одиночки, и все обращения из других классов производятся к этому бину. Таким образом, пропадает необходимость каждый раз создавать новый объект класса. Вместо этого можно обратиться к уже существующему бину, который храниться в контексте приложения.

Папочная структура проекта представлена на рисунке 2.1.

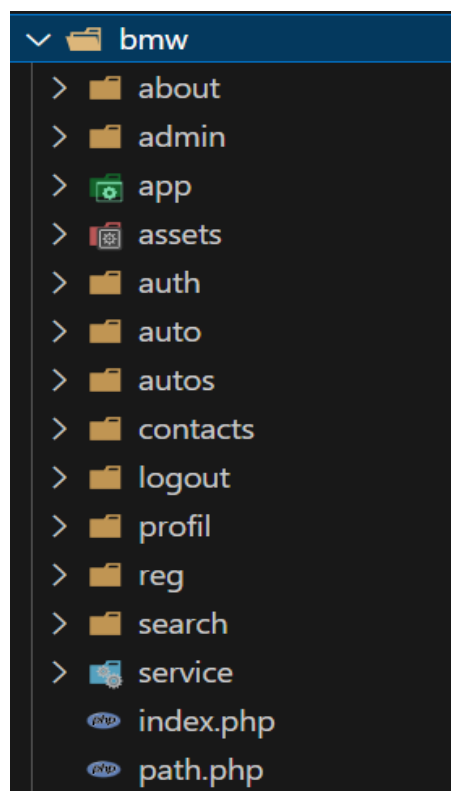


Рисунок 2.1 – Папочная структура проекта

Для создания класса, который будет связан с таблицами в базе данных, используется паттерн *ORM (Object Relationship Mapping)* [23].

ORM позволяет упростить работу с базой данных, представляя данные в виде объектов, а не записей в таблицах. Это позволяет уменьшить количество кода для работы с базой данных и сделать систему более гибкой и легко расширяемой.

ORM обеспечивает отображение между объектами и реляционными базами данных, ведь объекты и таблицы реляционных баз данных по-разному хранят данные приложения. Кроме того, в объектах и таблицах есть различные механизмы структурирования данных.

При создании программного средства автосалона были созданы классы для взаимодействия с базой данных. Перечень классов представлен на рисунке 2.2.

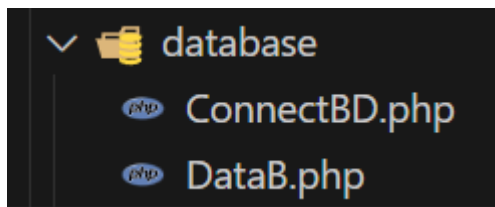


Рисунок 2.2 – Печень классов для взаимодействия с базой данных

Пример создания класса для взаимодействия с базой данных приведен на рисунке 2.3.

```
session_start();
require('ConnectBD.php');

class DataB {
    private function executeSql($sql): object {}
    private function dbCheckErr($query): bool {}
    public function selectAll($table, $params = []) {}
    public function insert($table, $params) {}
    public function selectOne($table, $params = []) {}
    public function update($table, $id, $params) {}
    public function delete($table, $id) {}
    public function selectAutoFromAutosWithModelsOnSingle($table1, $table2,
$id) {}
    public function getCountModel($idModel) {}
    public function getPersonalData($table1, $table2, $table3, $table4, $id)
{}
    public function getOrders($id) {}
    public function getColorsAutos() {}
    public function searchAutos($params, $paramsPrice, $paramsYear) {}
    public function searchAdmin($search, $table) {}
}
```

Рисунок 2.3 – Часть кода класса *DataB*

Кроме того, необходимо учитывать различные типы пользователей системы. Каждый тип пользователя может иметь свои методы и свойства, которые будут соответствовать их ролям в системе. Для того, чтобы разграничивать функционал, используются перечисления с указанием ролей.

Структура перечислений представлена на рисунке 2.4.

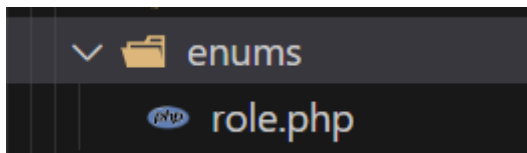


Рисунок 2.4 – Структура перечислений

Так как разрабатываемое программное средство является *RESTful* приложением [24], то требуется предусмотреть возможность принятия поступающих на сервер запросов и отправки ответов.

Контроллеры отвечают за обработку входящих запросов *REST API*, подготовку и возврат данных для ответа на запрос. Сначала запрос получает *REQUEST_METHOD* и передаваемые параметры.

Структура контроллеров приведена на рисунке 2.5.

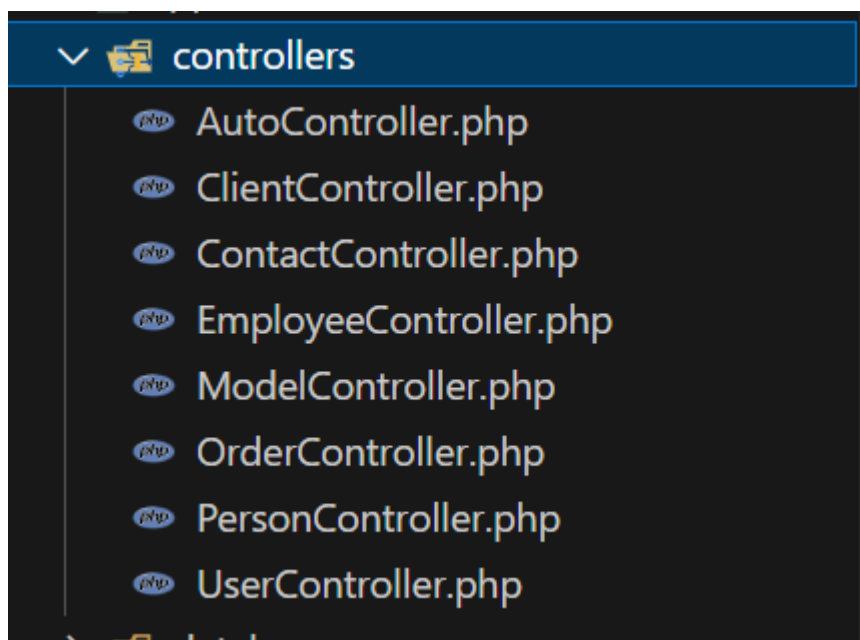


Рисунок 2.5 – Структура контроллеров

Пример реализации контроллера представлен на рисунке 2.6.

```
class AutoController {
    public function addAuto(): void {
        global $autoService;
        if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['auto-
create'])) {
            $autoService -> addAuto();
        }
    }

    public function editAuto(): void {
        global $autoService;
        if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['auto-
edit'])) {
            $autoService -> updateAuto();
        }
    }

    public function deleteAuto(): void {
        global $autoService;
        if($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['del_id'])) {
            $id = $_GET['del_id'];
            $autoService -> deleteAuto($id);
        }
    }
}
```

Рисунок 2.6 – Часть кода контроллера AutoController

В процессе проектирования объектной модели необходимо учитывать особенности бизнес-процессов автосалона, такие как обработка заказов, работа с клиентами и др. Сервисы обеспечивают доступ к данным, хранящимся в базе данных или других источниках данных. Сервисы могут выполнять операции чтения, записи, обновления, удаления данных и т.д..

Структура сервисов представлена на рисунке 2.7.

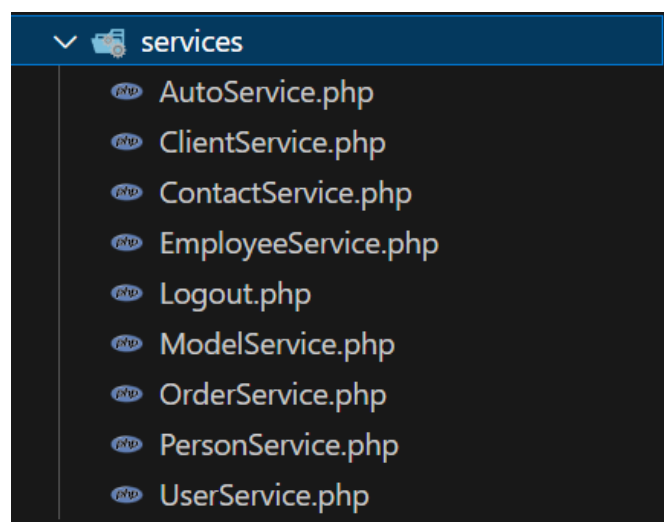


Рисунок 2.7 – Структура сервисов

Пример реализации сервиса представлен на рисунке 2.8.

```
class AutoService {  
  
    private $AVAILABLE = 1;  
    private $NO_AVAILABLE = 0;  
  
    public function updateStatusAuto($id): void {  
        $db = new DataB();  
        $status = $_GET['status'];  
        $autoId = $db->update('auto', $id, ['status' => $status]);  
        header('location:' . ADMIN_URL . '/auto');  
    }  
  
    public function deleteAuto($id): void {  
        $db = new DataB();  
        $db->delete('auto', $id);  
        header('location:' . ADMIN_URL . '/auto');  
    }  
}
```

Рисунок 2.8 – Часть кода сервиса AutoService

Разделение приложений на пакеты и отдельные классы произведено в связи с первым и двумя последними принципами SOLID – принцип единственной ответственности, принцип разделения интерфейсов и принцип инверсии зависимостей.

Принцип единственной ответственности служит для разделения типов поведения, благодаря которому ошибки, вызванные модификациями в одном поведении, не распространялись на прочие, не связанные с ним типы.

Принцип разделения интерфейсов подразумевает то, что не нужно добавлять в сущность функциональность, которая ему не нужна. Сущность должна производить только те операции, которые обеспечивают правильность выполнения его методов.

Принцип инверсии зависимостей подразумевает то, что модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те, и другие должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций. Классы верхнего уровня – классы, которые выполняют операцию при помощи инструмента. Модули или классы нижнего уровня – это инструменты, которые нужны для выполнения операций. Абстракции представляют интерфейс, соединяющий два класса. Согласно данному принципу, класс не должен соединяться с инструментом, который применяет для выполнения операции. Вместо этого он должен быть соединён с интерфейсом, который поможет установить связь между инструментом и классом. Кроме того, принцип гласит, что ни интерфейс, ни класс, не обязаны вникать в специфику работы инструмента. Напротив, это инструмент должен подходить под требования интерфейса.

Использование системы пакетов в архитектуре дает возможность грамотно разделить все классы согласно вышеперечисленным принципам.

Диаграмма состояний представлена в графическом материале. При запуске программного средства веб-сайт находится на главной странице, где есть кнопка авторизации с возможностью авторизации.

Диаграмма состояний этой области представлена на рисунке 2.9.

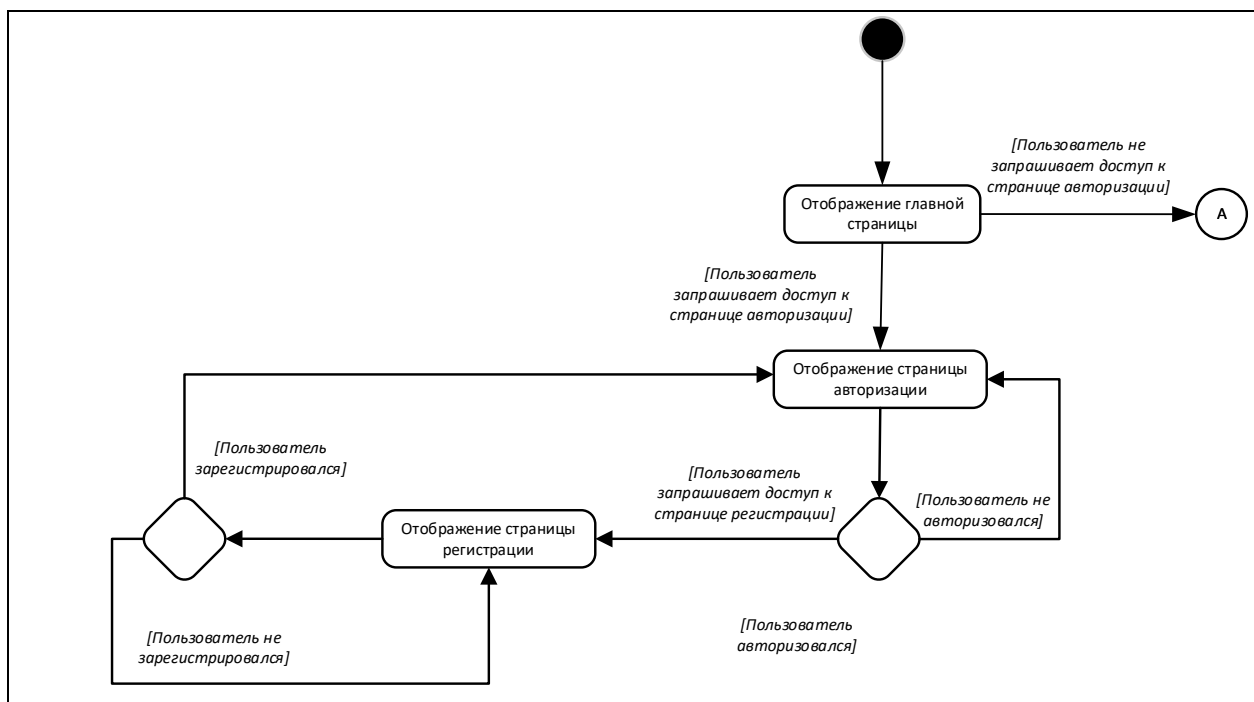


Рисунок 2.9 – Диаграмма состояния авторизации

Если авторизация не была пройдена, предоставляется возможность регистрации, иначе система переходит в другое состояние, в котором есть возможность выбора одного из пунктов меню или выхода из аккаунта.

Фрагмент диаграммы состояния данного случая представлен на рисунке 2.10.

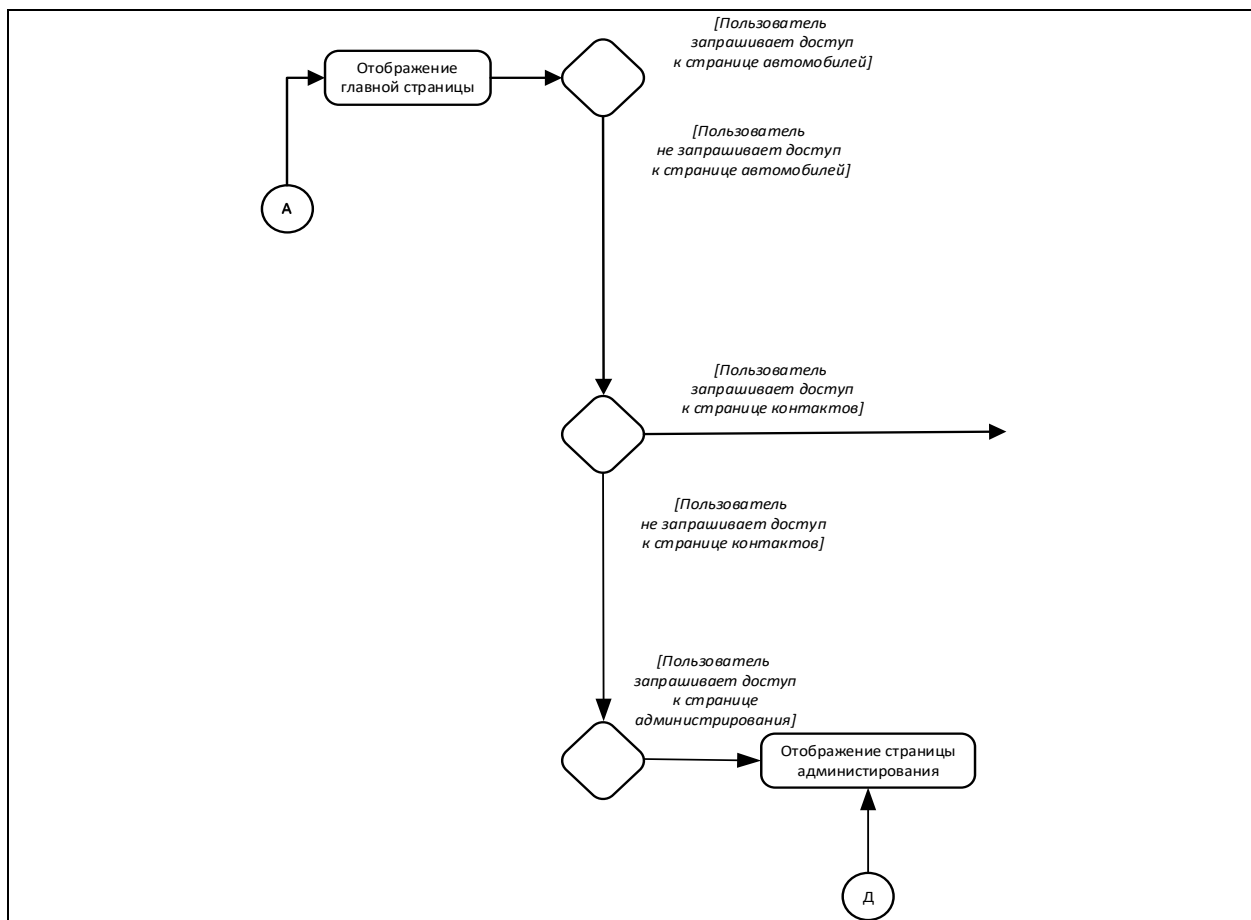


Рисунок 2.10 – Диаграмма состояния главного меню

Рассмотрим состояние системы при выборе раздела клиентов, диаграмма состояний которого представлена на рисунке 2.11.

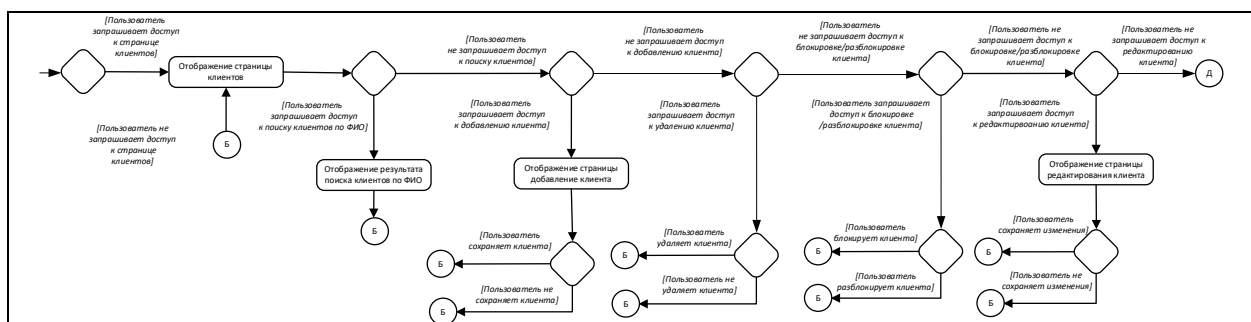


Рисунок 2.11 – Диаграмма состояний в разделе клиентов

В данном случае у пользователя есть возможность добавить нового клиента в базу данных, поиску клиента по ФИО, удалению клиентов из базы данных, заблокировать/разблокировать доступ клиенту.

При запросе поиска и его дальнейшем осуществлении, система предоставляет данные в соответствии с поисковым запросом. В случае запроса на удаление или блокировку/разблокировку пользователя система совершает действие и остается на данной странице. В случае запроса на добавление (редактирование) пользователя система перенаправляет на соответствующую страницу. Затем, после добавления (редактирования) необходимых данных, происходит сохранение изменений. После этого система перенаправляет пользователя на страницу клиентов.

При переходе в другие разделы состояния системы аналогичны.

2.2 Проектирование и разработка графического интерфейса

Проектирование и разработка графического интерфейса – это один из самых важных этапов создания веб-приложения, так как именно интерфейс является основным способом взаимодействия пользователя с приложением.

Перед началом проектирования необходимо определить целевую аудиторию приложения и ее потребности. Это поможет создать удобный и понятный интерфейс, который будет максимально соответствовать требованиям пользователей.

Основными элементами графического интерфейса, которым необходимо уделить особое внимание во время разработки, являются:

1 Цветовая гамма – выбор цветов должен соответствовать тематике приложения и быть приятным для глаз пользователя.

2 Шрифты – необходимо выбрать удобный и читаемый шрифт для текстовых элементов интерфейса.

3 Иконки и изображения – использование иконок и изображений помогает сделать интерфейс более понятным и интуитивно понятным для пользователя.

4 Навигация – необходимо обеспечить удобную навигацию по приложению, чтобы пользователь мог легко находить необходимые функции.

5 Формы и поля ввода – необходимо создать удобные формы и поля ввода, чтобы пользователь мог быстро и легко заполнять необходимые данные. В формах необходимо предусмотреть валидацию, которая бы проверяла введенные значения на корректность.

Для построения грамотного интерфейса нужно соблюдать принципы UI/UX [25].

UI (User Interface) и *UX (User Experience)* – это два термина, связанных с проектированием пользовательского интерфейса приложений или веб-сайтов.

UI (User Interface) описывает то, как пользователь взаимодействует с интерфейсом приложения или веб-сайта. Это включает в себя дизайн элементов интерфейса, таких как кнопки, поля ввода, меню и т.д., а также расположение

и организацию этих элементов. Цель *UI*-дизайна – сделать интерфейс интуитивно понятным, удобным в использовании и привлекательным для пользователя.

UX (User Experience) описывает общий опыт пользователя при использовании приложения или веб-сайта. Это включает в себя все аспекты взаимодействия пользователя с продуктом – от первичного впечатления до навигации, процесса взаимодействия и выполнения задач. Цель *UX*-дизайна – создать удовлетворительный и приятный пользовательский опыт, учитывая потребности и ожидания пользователей.

Таким образом, необходимо создавать простой и интуитивно понятный интерфейс, который будет легко осваиваться пользователями. Отсутствие лишних элементов и явных инструкций также может улучшить юзабилити интерфейса.

Важно учитывать требования к скорости загрузки и производительности интерфейса. Слишком медленный или неотзывчивый интерфейс может отпугнуть пользователей и негативно повлиять на их впечатление от приложения. В случае возникновения непредвиденных ситуаций пользователь должен получать соответствующее сообщение.

Особое внимание стоит уделить обеспечению соответствия графического интерфейса брендингу и стилю компании, чтобы создать единый и узнаваемый образ.

Наконец, необходимо проводить тестирование графического интерфейса на различных устройствах и с разными типами пользователей, чтобы обнаружить и исправить возможные ошибки и недочеты.

Создание графического интерфейса приложения осуществлялось на языке гипертекстовой разметки *HTML* с использованием *RНР* вставки. Данная технология позволяет отделить логику приложения от представления, что упрощает разработку и поддержку кода.

Исходя из назначения файлов проекта была выбрана соответствующая папочная структура. В ней отражается разделение файлов на группы, внешний вид пользовательского интерфейса, хранения текущего состояния страниц и хранение необходимых для отображения на странице статических файлов.

Папочная структура проекта представлена на рисунке 2.12.

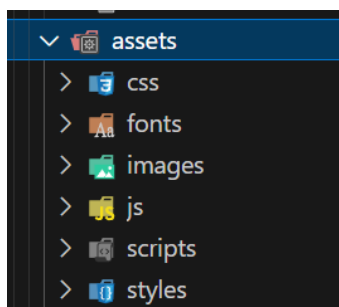


Рисунок 2.12 – Папочная структура фронтенд части

Для описания стилей компонент был выбран подход с созданием *SCSS* модулей. *SCSS* модуль – это *SCSS* файл, в котором все имена классов имеют локальную область видимости по умолчанию. Это позволяет писать более чистый *SCSS* код и не бояться повторения названий классов в разных участках *HTML*. Также модульная структура файлов стилей позволяет избежать монолитных файлов большого размера, которые затруднили бы разработку и возможное переиспользование кода компонент. Все модули собираются в единый минимизированный *CSS* файл, для возможности подключения и использования на странице, а также для оптимизации загрузки.

Папочная структура стилей представлена на рисунке 2.13.

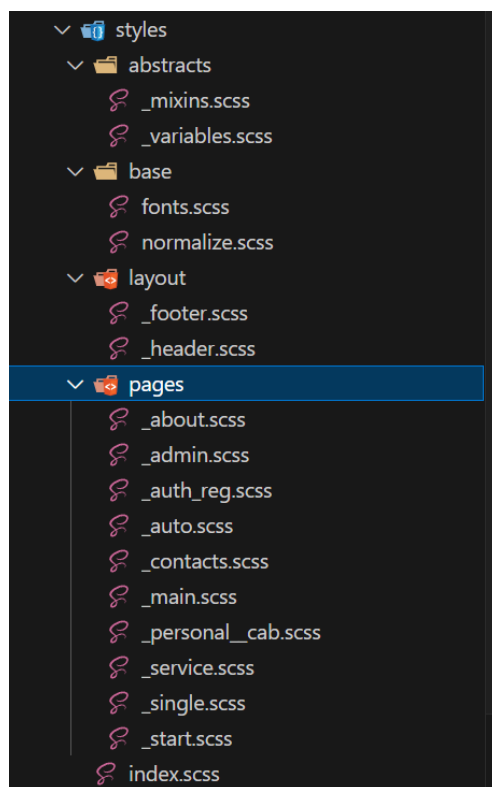


Рисунок 2.13 – Папочная структура стилей

Папка *scripts* содержит файлы скрипта, разбитые по папкам и файлам, которые при помощи *gulp*-сборщика собираются в единый минимизированный файл, находящийся в папке *js*.

Папочная структура скриптов представлена на рисунке 2.14.

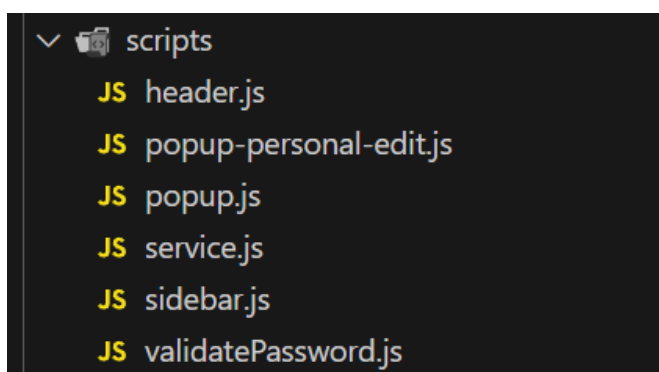


Рисунок 2.14 – Папочная структура скриптов

Папка *fonts* содержит шрифты, которые будут применены для данного веб-сайта. В качестве шрифта выбран – *Montserrat*.

Папка *images* содержит все картинки, иконки, фав-иконки и т.д.

Для упрощения работы с *HTML* кодом некоторые части кода, которые встречаются на каждой странице, вынесены в отдельную папку для более быстрого подключения и дальнейшего упрощения при редактировании.

Папочная структура пере используемых основных частей *HTML* кода представлена на рисунке 2.15.

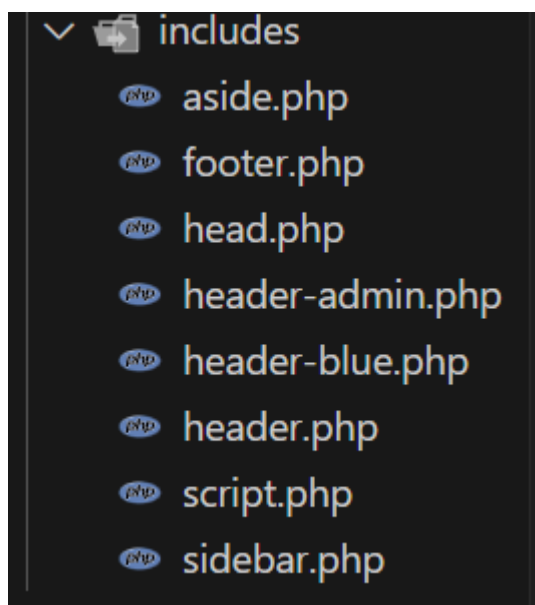


Рисунок 2.15 – Папочная структура пере используемых частей HTML кода

Для отображения динамических и статических страниц используются *php* файлы с возможностью вставки гипертекстовой разметки с внедрением

php кода. Каждая страница имеет папку, характеризующую ее название и лежащий *index.php* файл. Также для отделения пользовательских страниц от страниц администратора, создана специальная папка *admin*, содержащая исходный основной файл и подпапки для внутренних страниц (добавления, редактирования), имеющих также *index.php* файлы для отображения.

Пример папочной структуры страниц представлен на рисунке 2.16.

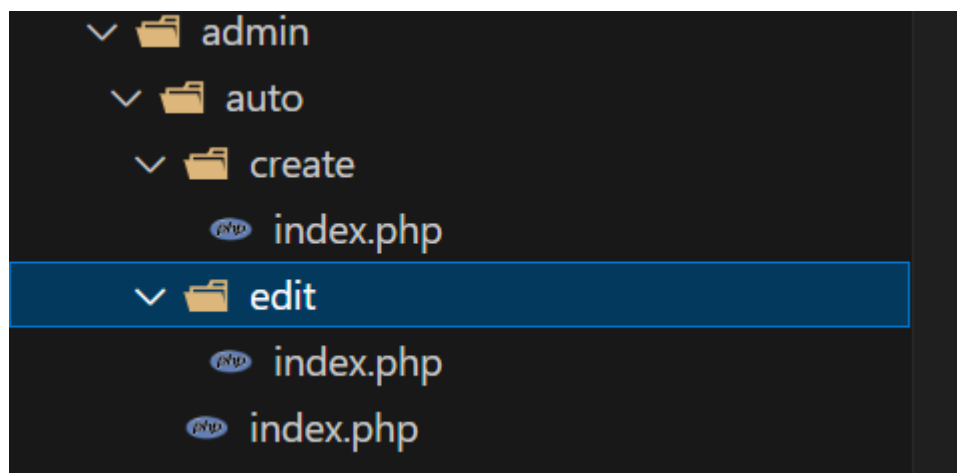


Рисунок 2.16 – Папочная структура страницы

Точкой входа в программу является главная страница. Пользователь имеет возможность ознакомиться с основным функционалом и понятной панелью навигации. Также есть возможность войти в систему или зарегистрироваться. После успешной авторизации система перенаправляет пользователя на главную страницу.

Пример страницы авторизации представлен на рисунке 2.17.

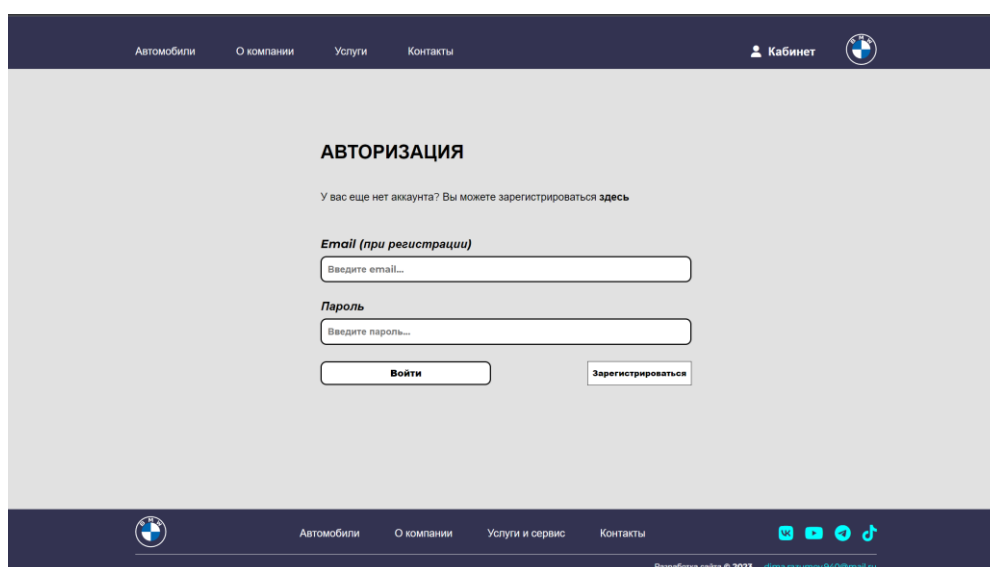


Рисунок 2.17 – Страница авторизации

Пример реализации страницы авторизации представлен на рисунке 2.18.

```
<!DOCTYPE html>
<html lang="ru">

<head>
  <?php include(SITE_ROOT . '/app/includes/head.php') ?>
  <title>Авторизация BMW</title>
</head>

<body>

  <?php include(SITE_ROOT . '/app/includes/header-blue.php') ?>

  <div class="dark-wrapper"></div>

  <main>
    <div class="container">
      <form method="post" action="" class="form-auth">
        <h1 class="form-auth__title">Авторизация</h1>
        <p class="form-auth__desc">У вас еще нет аккаунта? Вы можете зарегистрироваться <a href="<?= BASE_URL ?>/reg">здесь</a></p>
        <div class="error">
          <?php include(SITE_ROOT . "/app/helps/errInfo.php") ?>
        </div>
        <div class="form-auth__item">
          <label for="email">Email (при регистрации)</label>
          <input type="email" value="<?=$email ?>" name="email" id="email"
placeholder="Введите email..." required>
        </div>
        <div class="form-auth__item">
          <label for="password">Пароль</label>
          <input type="password" name="password" id="password"
placeholder="Введите пароль..." required>
        </div>
        <div class="form-auth__item form-auth__buttons">
          <button type="submit" name="button__auth">Войти</button>
          <a href="<?= BASE_URL ?>/reg">Зарегистрироваться</a>
        </div>
      </form>
    </div>
  </main>

  <?php include(SITE_ROOT . '/app/includes/footer.php') ?>
  <?php include(SITE_ROOT . '/app/includes/script.php') ?>

</body>
</html>
```

Рисунок 2.18 – Фрагмент кода страницы авторизации

Пример главной страницы представлен на рисунке 2.19.



Рисунок 2.19 – Часть главной страницы

Пример реализации главной страницы представлен на рисунке 2.20.

```
<body>
  <main>
    <section class="cars">
      <div class="container">
        <div class="cars__container">
          <h2 class="cars__title">Автомобили</h2>
          <div class="cars__items">
            <div class="car__item">
              
              <h3 class="car__item-name">BMW 4 серии Cabrio</h3>
            </div>
            <div class="car__item">
              
              <h3 class="car__item-name">BMW 6 серии GT</h3>
            </div>
            <div class="car__item">
              
              <h3 class="car__item-name">BMW M8 Coupe</h3>
            </div>
            <div class="car__item">
              
              <h3 class="car__item-name">BMW X5</h3>
            </div>
            <div class="car__item">
              
              <h3 class="car__item-name">BMW i7</h3>
            </div>
            <div class="car__item">
              
              <h3 class="car__item-name">BMW 2 серии Coupe</h3>
            </div>
          </div>
          <a href="<?= BASE_URL ?>/autos" class="button cars__button"
title="Узнать подробнее">Подробнее</a>
        </div>
      </div>
    </section>
  </main>
```

Рисунок 2.20 – Фрагмент кода главной страницы

Пример страницы автомобилей в менеджер панели представлен на рисунке 2.21.

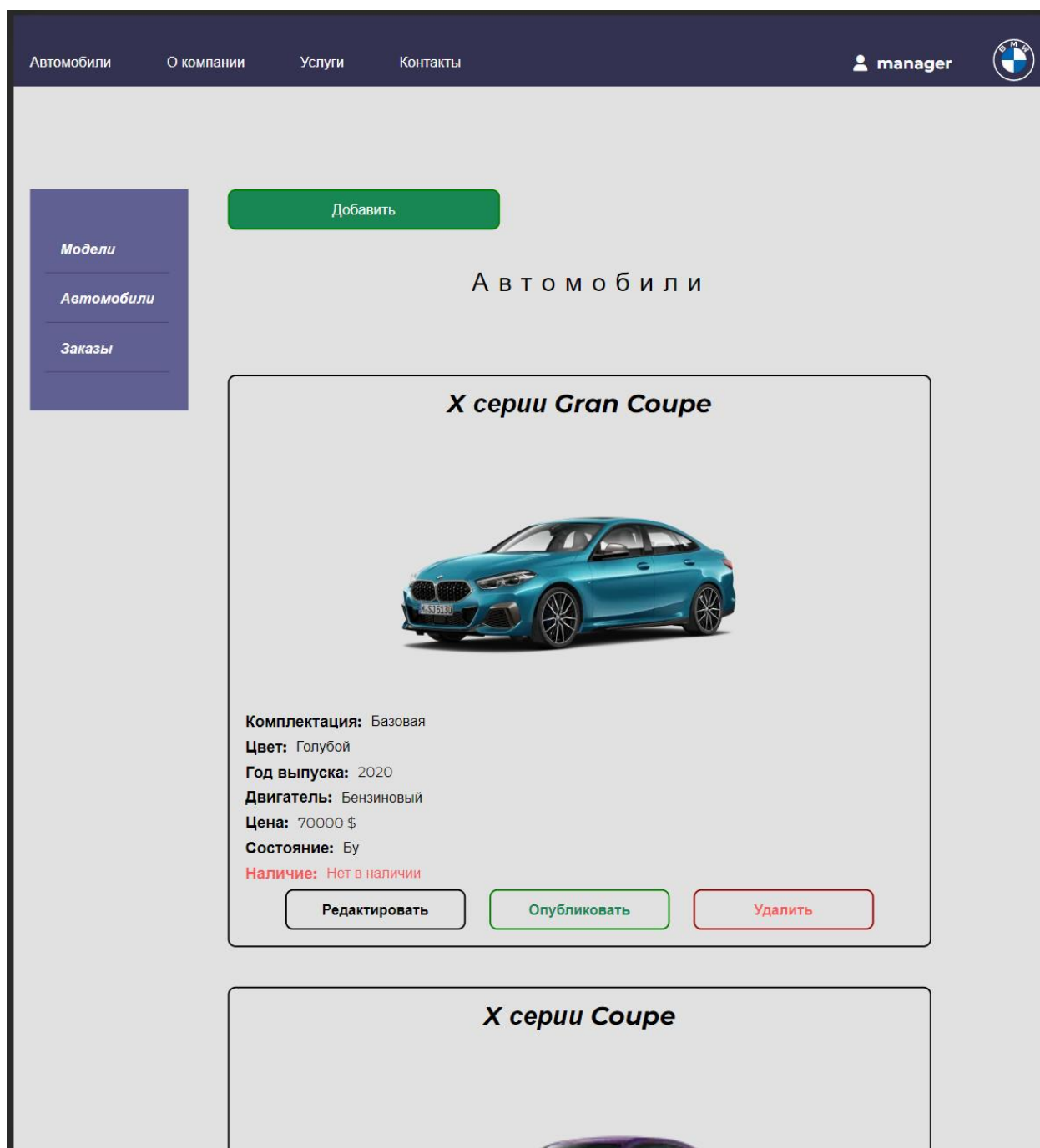


Рисунок 2.21 – Страница автомобилей через менеджер панель

Пример реализации страницы автомобилей в менеджер панели представлен на рисунке 2.22.

```
<div class="container">
  <div class="panel__container">
    <div class="panel__body">
      <a class="button panel__button" href="<?= ADMIN_URL .
"/auto/create/" ?>">Добавить</a>
      <h1 class="title-pages panel__title">Автомобили</h1>
      <?php if(empty($autoModelsName)):?>
        <p class="panel__empty">Автомобили в базе данных отсутствуют.
Но вы можете добавить</p>
      <?php else:?>
        <div class="panel__blocks">
          <div class="error"></div>
          <?php foreach ($autoModelsName as $key => $auto): ?>
            <div class="panel__block model__cars-js" model="<?=
$auto['model']?>">
              <h2 class="panel__subtitle"><?= $auto['model']?> серии
<?= $auto['name']; ?></h2>
              <?php if(!empty($auto['img'])): ?>
                " class="panel__img panel__img-sm">
              <?php endif; ?>
              <div class="panel__buttons">
                <a class="button panel__button-edit" href="<?=
ADMIN_URL ?>/auto/edit?id=<?=$auto['id'];?>">Редактировать</a>
                <?php if($auto['status'] == 0): ?>
                  <a class="button panel__button-publish" href="<?=
ADMIN_URL ?>/auto/edit?status=1&pub_id=<?=$auto['id'];?>">Опубликовать</a>
                <?php else:?>
                  <a class="button panel__button-publish" href="<?=
ADMIN_URL ?>/auto/edit?status=0&pub_id=<?=$auto['id'];?>">Снять с публика-
ции</a>
                <?php endif; ?>
                <a class="button panel__button-red" href="<?=
ADMIN_URL ?>/auto/edit?del_id=<?=$auto['id'];?>">Удалить</a>
              </div>
            </div>
          <?php endforeach; ?>
        <?php endif; ?>
      </div>
    </div>
  </div>
</section>
```

Рисунок 2.22 – Фрагмент кода страницы автомобилей через менеджер панель

Таким образом, проектирование и разработка графического интерфейса приложения – это сложный и ответственный процесс, который требует профессионального подхода и внимания к деталям. Однако, правильно спроектированный интерфейс поможет сделать веб-приложение более удобным и привлекательным для пользователей, что положительно скажется на его популярности на рынке и может стать ключевым фактором успеха веб-приложения.

2.3 Описание и реализация используемых в программном средстве алгоритмов

В мире программирования алгоритмы играют решающую роль, определяя эффективность и функциональность создаваемых программ. Алгоритм – это набор инструкций, предписывающих последовательность шагов для достижения конкретной цели [26]. Он подобен рецепту, по которому готовится блюдо, где каждый шаг определен и направлен на получение определенного результата.

Алгоритмы широко используются в программировании для решения разнообразных задач, начиная от сортировки данных и поиска оптимальных путей до шифрования информации. Важной характеристикой алгоритма является его инструктивность – способность быть понятным и легко интерпретированным человеком.

Взаимодействие алгоритмов с различными структурами данных, такими как массивы, списки и деревья, играет ключевую роль в эффективной обработке и хранении информации. Выбор подходящей структуры данных в сочетании с оптимальным алгоритмом обеспечивает оптимальную производительность программы.

Известные алгоритмы, такие как сортировка QuickSort, поиск в ширину (BFS), шифрование RSA, являются краеугольными камнями мира программирования. Их анализ и понимание обеспечивают необходимый фундамент для разработчиков.

Существуют различные типы алгоритмов [27]:

- линейный алгоритм – последовательность команд (инструкций), выполняемых последовательно во времени;
- разветвляющийся алгоритм – алгоритм, содержащий в качестве результата управления хотя бы одно условие, которое можно разделить на несколько альтернативных ветвей алгоритма;
- циклический алгоритм – алгоритм, в котором одно и то же действие (один и тот же процесс) повторяется несколько раз. Большинство вычислительных методов сводятся к круговым алгоритмам. Цикл программы – последовательность инструкций (серия, тело цикла), которая может выполняться многократно.

Основным способом визуализации последовательности алгоритмов является создание схем алгоритмов. С помощью схемы алгоритма можно проиллюстрировать последовательность действий, операторы и управляющие конструкции, используемые в процессе выполнения алгоритма.

Схема алгоритма позволяет легко визуализировать и понять последовательность выполнения операций, структуру алгоритма и поток управления

программы. Она может быть использована для разработки, исправления и отладки алгоритмов перед переходом к написанию программного кода. Благодаря графическому представлению схема алгоритма удобна для коммуникации и обмена информацией между разработчиками или командами, работающими над одной задачей.

Для того, чтобы начать работу с данными необходимо пройти авторизацию. Для авторизации была создана схема алгоритма. Эта схема представлена на рисунке 2.23.

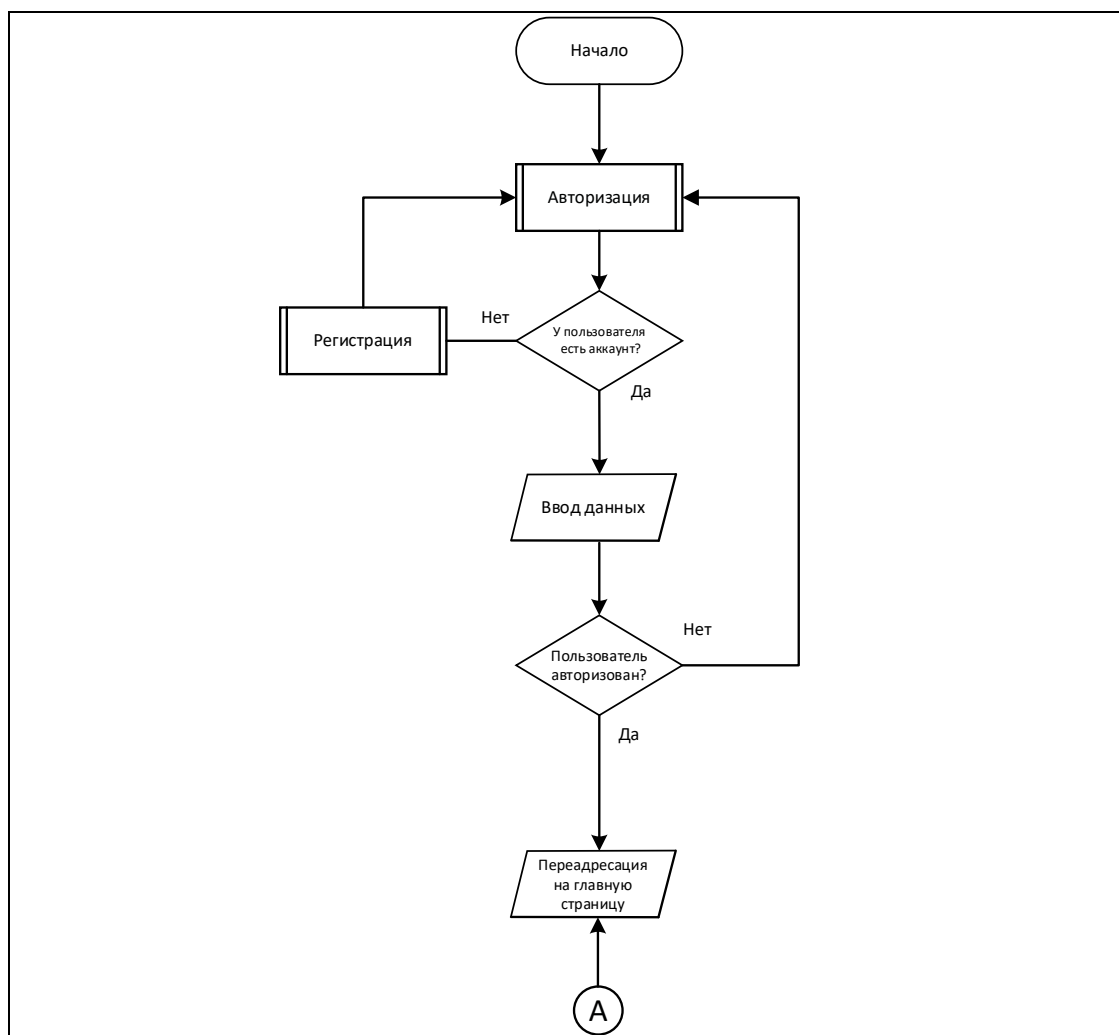


Рисунок 2.23 – Схема алгоритма

В данном случае, когда пользователь запускает программное средство, то ему сразу демонстрируется главная страница с возможностью перехода на страницу авторизации. Если пользователь авторизовался, то ему демонстрируется главный экран приложения. Если нет, то ему будет предложено пройти регистрацию.

После авторизации можно получить роль пользователя, логин и идентификационный номер из глобального объекта `$_SESSION` для дальнейшей работы с данными.

Реализация метода регистрации представлена на рисунке 2.24.

```
public function registration() {
    global $userService;

    if($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['button__reg'])) {

        $lastName = trim($_POST['last_name']);
        $firstName = trim($_POST['first_name']);
        $email = trim($_POST['email']);
        $login = trim($_POST['login']);
        $userService -> registration();

        $arrRes =
        [
            $lastName, $firstName, $email, $login
        ];

        return $arrRes;
    }
}
```

Рисунок 2.24 – Метод регистрации пользователя

Рассмотрим фрагмент из схемы алгоритмов раздела «Сотрудники». Алгоритм работы со страницей «Клиентов» представлен на рисунке 2.25.

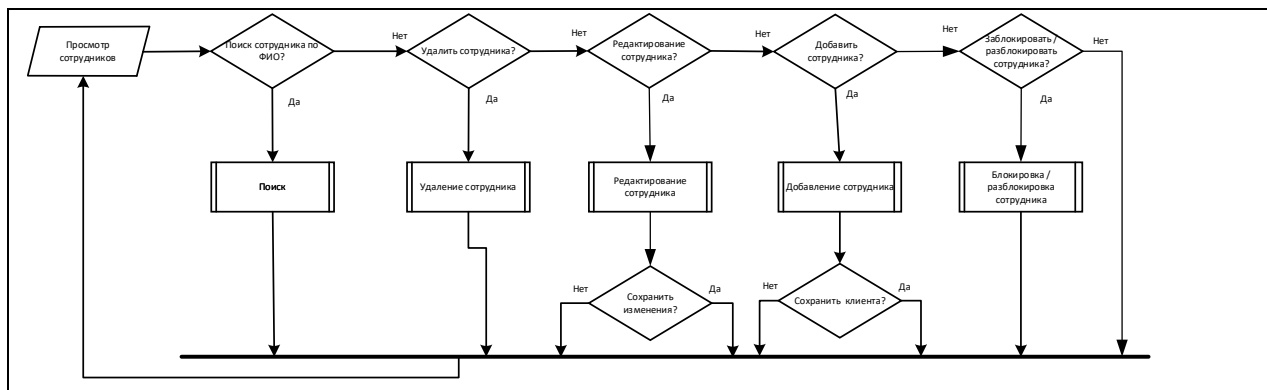


Рисунок 2.25 – Алгоритм работы со страницей «Сотрудники»

Рассмотрим алгоритм поиска сотрудника, реализация которого представлена на рисунке 2.26.

```
public function search(): ?array {
    global $employeeService;

    $db = new DataB();

    if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['search-employee'])) {
        $employees = $db->searchAdmin($_POST['search-employee'], 'employeesView');
        if(empty($employees)) {
            array_push($employeeService -> errMsg, 'По данному поиску ничего не найдено!
Повторите поиск!');
        }
    }

    return $employees;
}
```

Рисунок 2.26 – Код для метода поиска сотрудников

В строке поиска мы вводим ФИО, которое нам необходимо найти, если такой сотрудник присутствует, происходит вывод информации о нем на экран. Если при поиске ничего не найдено, на экран выводится соответствующее сообщение.

Код для метода удаления сотрудника представлен на рисунке 2.27.

```
public function deleteUser($id): void {
    $db = new DataB();

    if($_SESSION['role'] == $this -> EMPLOYEE) {
        $idEmployee = $db->selectOne('employees', ['id_auth' => $id]);
        $db->delete('authorization', $idAuth);
        $db->delete('employees_address', $idAddress);
        $db->delete('employees_passport', $idPas); $db->delete('employees', $id);
    } else {
        $idClients = $db->selectOne('clients', ['id_auth' => $id]);
        $idAuth = $idClients['id_auth'];
        $idAddress = $idClients['id_address']; $idPas = $idClients['id_passport'];
        $db->delete('authorization', $idAuth);
        $db->delete('clients_address', $idAddress);
    }
}
```

Рисунок 2.27 – Код метода удаления сотрудника

Пользователь выбирает сотрудника информацию о котором хочет удалить, реализация которого происходит после нажатия на кнопку удаления.

Реализация метода редактирования информации о сотруднике представлена на рисунке 2.28.

```
public function update(): void {  
    global $employeeService;  
    if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['employee-edit'])) {  
        $employeeService -> update();  
    }  
}
```

Рисунок 2.28 – Код метода редактирования информации

Пользователь может менять существующую информацию, заменяя ее на необходимую, после изменения необходимой информации пользователь сохраняет изменения.

Реализация метода блокировки и разблокировки доступа сотруднику представлена на рисунке 2.29.

```
public function updateStatus($id): void {  
    $db = new DataB();  
    $access = $_GET['access'];  
    $employee = $db->selectOne('employees', ['id' => $id]);  
    $employeeAuth = $db->selectOne('authorization', ['id' => $employee['id_auth']]);  
    $idAuth = $employeeAuth['id'];  
    $db->update('authorization', $idAuth, ['access' => $access]);  
    header('location:' . ADMIN_URL . '/employee');  
}
```

Рисунок 2.29 – Алгоритм блокировки сотрудника

Пользователь может заблокировать или разблокировать доступ существующему сотруднику. Пользователь выбирает сотрудника, которого он хочет разблокировать или заблокировать, реализация которого происходит после нажатия на соответствующую кнопку.

Код для метода добавления сотрудника представлен на рисунке 2.30.

```
public function add() {  
    global $employeeService;  
    if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['employees-create']))  
    {  
        $employeeService -> add();  
        $arrRes =  
        [  
            $lastName, $firstName, $surname, $dateBirth, $phone, $city, $street, $house,  
            $apartment,  
            $series, $number, $issuedBy, $login, $password, $email, $jobTitle  
        ];  
        return $arrRes;  
    }  
}
```

Рисунок 2.30 – Код для метода добавления сотрудника

Раздел аккаунта аналогичен разделу сотрудники в удалении, добавлении, редактировании, но имеет дополнительный функционал в виде изменения пароля.

Алгоритм аккаунта администратора представлен на рисунке 2.31.

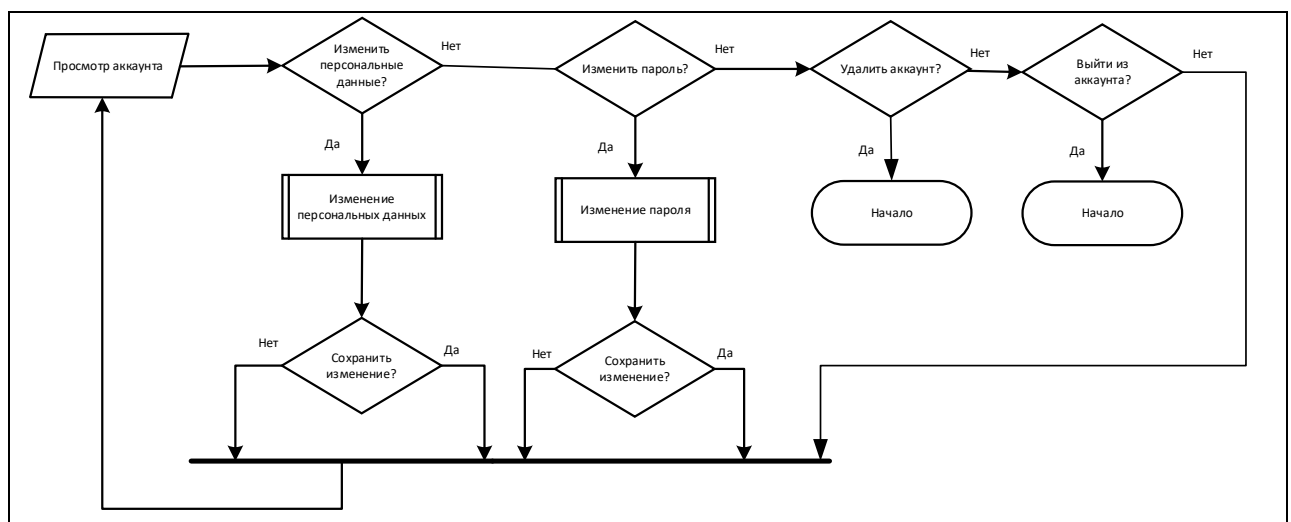


Рисунок 2.31 – Алгоритм работы со страницей аккаунта

Рассмотрим алгоритм изменения пароля, реализация которого представлена на рисунке 2.32.

```
public function editPassword($id): void {  
  
    $db = new DataB();  
  
    $passwordF = $_POST['passF'];  
    $password = password_hash($passwordF, PASSWORD_DEFAULT);  
  
    $data = [  
        'password' => $password,  
        'access' => $this->ACCESS  
    ];  
  
    $db->update('authorization', $id, $data);  
}
```

Рисунок 2.32 – Код метода изменения пароля

3 ЭКСПЛУАТАЦИЯ ПРОГРАММНОГО СРЕДСТВА

3.1 Ввод в эксплуатацию и обоснование минимальных технических требований к оборудованию

Для деплоя веб-сайта нужно использовать какой-либо хостинг [28]. Хостинг – это услуга по предоставлению ресурсов для размещения информации на сервере, постоянно имеющем доступ к сети. Есть большое количество хостингов как платных, так и бесплатных, некоторые из них имеют ограничения по количеству данных, некоторые по времени и занимаемой памяти т.д.

Для данной курсовой работы используется платный хостинг Beget с функцией бесплатного использования для одного сайта, одной базы данных и ограниченного по памяти [29][30].

Проходим регистрацию, указывая номер телефона, на который будет зарегистрирован аккаунт, после успешной регистрации на указанный номер телефона придет смс с данными для авторизации логин и пароль. Авторизовавшись попадаем в панель управления, представленную на рисунке 3.1.

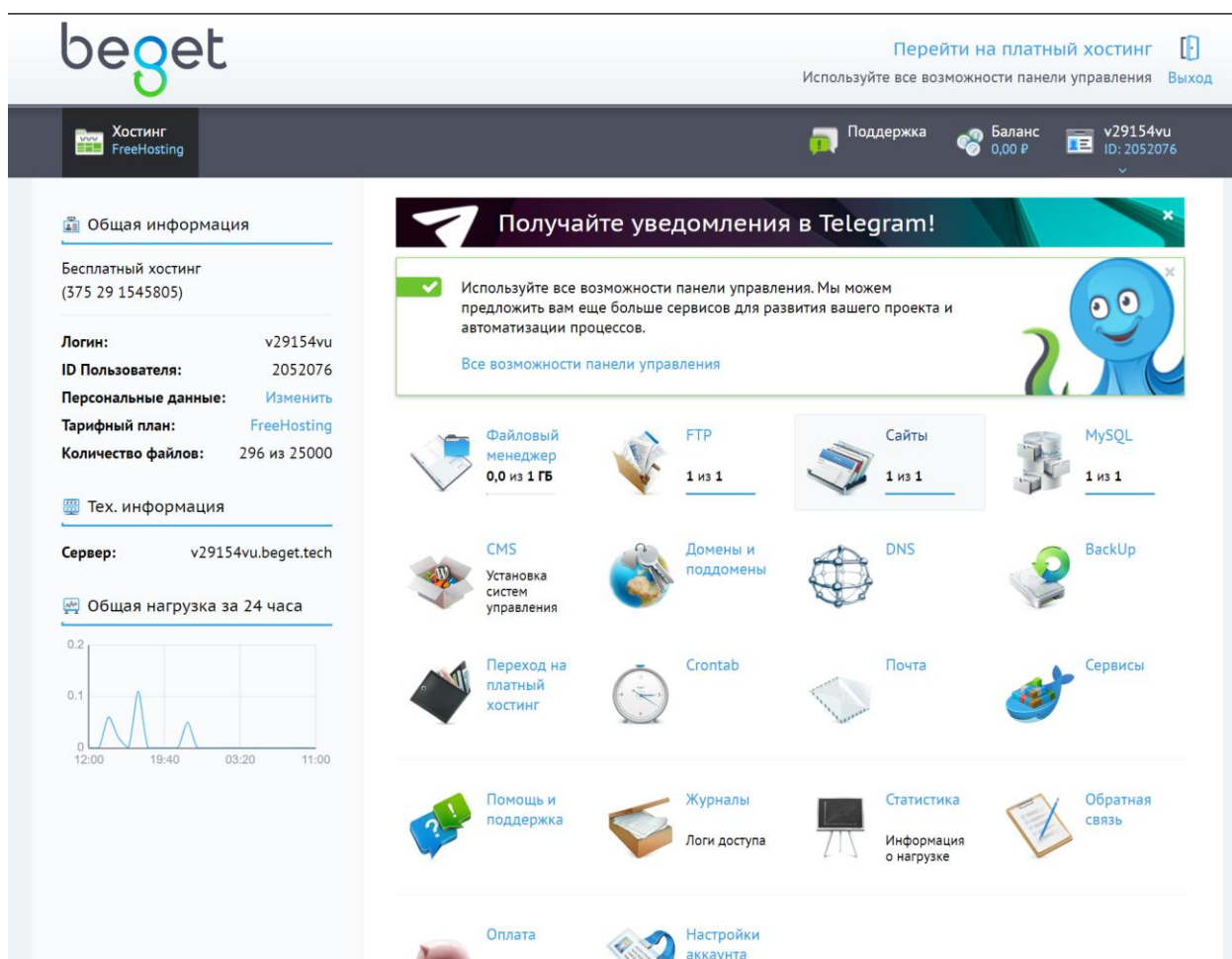


Рисунок 3.1 – Панель управления Beget

На локальном устройстве архивируем все файлы данной курсовой работы и при помощи Файлового менеджера погружаем их на сервер, предварительно удалив присутствующие файлы по умолчанию в нужной папке. Пример заливки файлов через файловый менеджер представлен на рисунке 3.2.

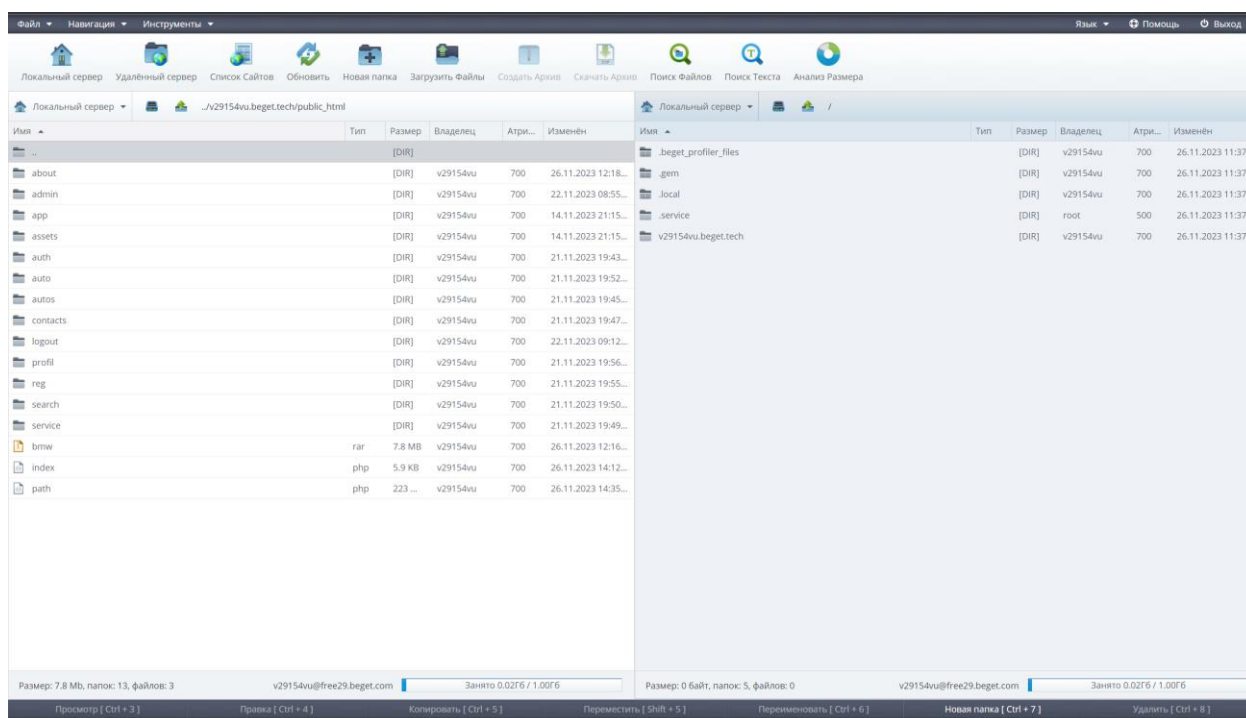


Рисунок 3.2 – Заливка файлов через файловый менеджер на хостинг

Далее нужно указать версии используемых языков программирования на хосте, такие же которые использовались в работе, для совмещения версий.

Настройка и подключение базы данных. Заходим через локальный сервер в базу данных, в данном примере в качестве локального сервера используется OpenPanelServer, а для работы с базой данных PHPMyAdmin и экспортируем базу данных.

Экспорт БД представлен на рисунке 3.3.

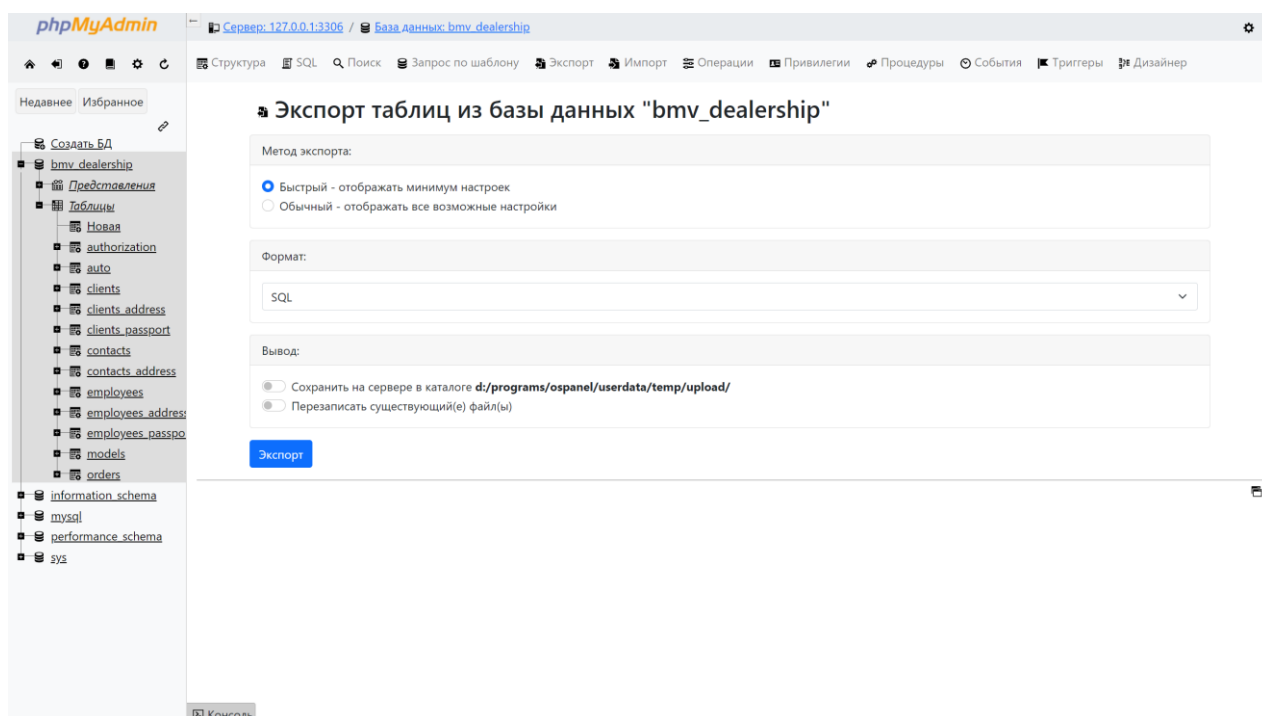


Рисунок 3.3 – Экспорт БД из локального сервера

На хостинге создаем базу данных указывая имя базы данных и пароль, которые нужно будет указать в файле подключения к базе данных и импортируем БД. Пример создания БД на хостинге представлен на рисунке 3.4.

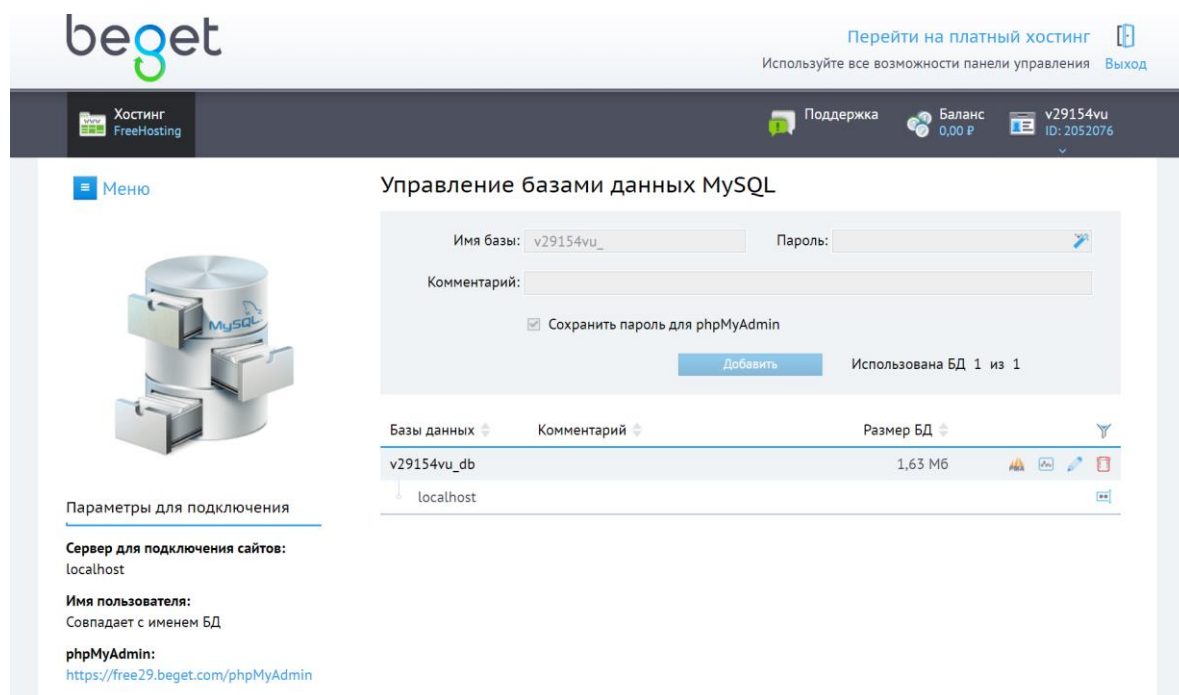


Рисунок 3.4 – Создание БД на хостинге

В случае успешной загрузки всех данных на хостинг веб-сайт откроется по выданной ссылке в левом меню блока.

Для работы и внесения правок на сервере нужно зайти в раздел FTP и создать FTP-аккаунт, к которому можно подключиться через FileZila или другое подобное программное средство для внесения изменений сразу на сервере. Пример создания FTP-аккаунта представлен на рисунке 3.5.

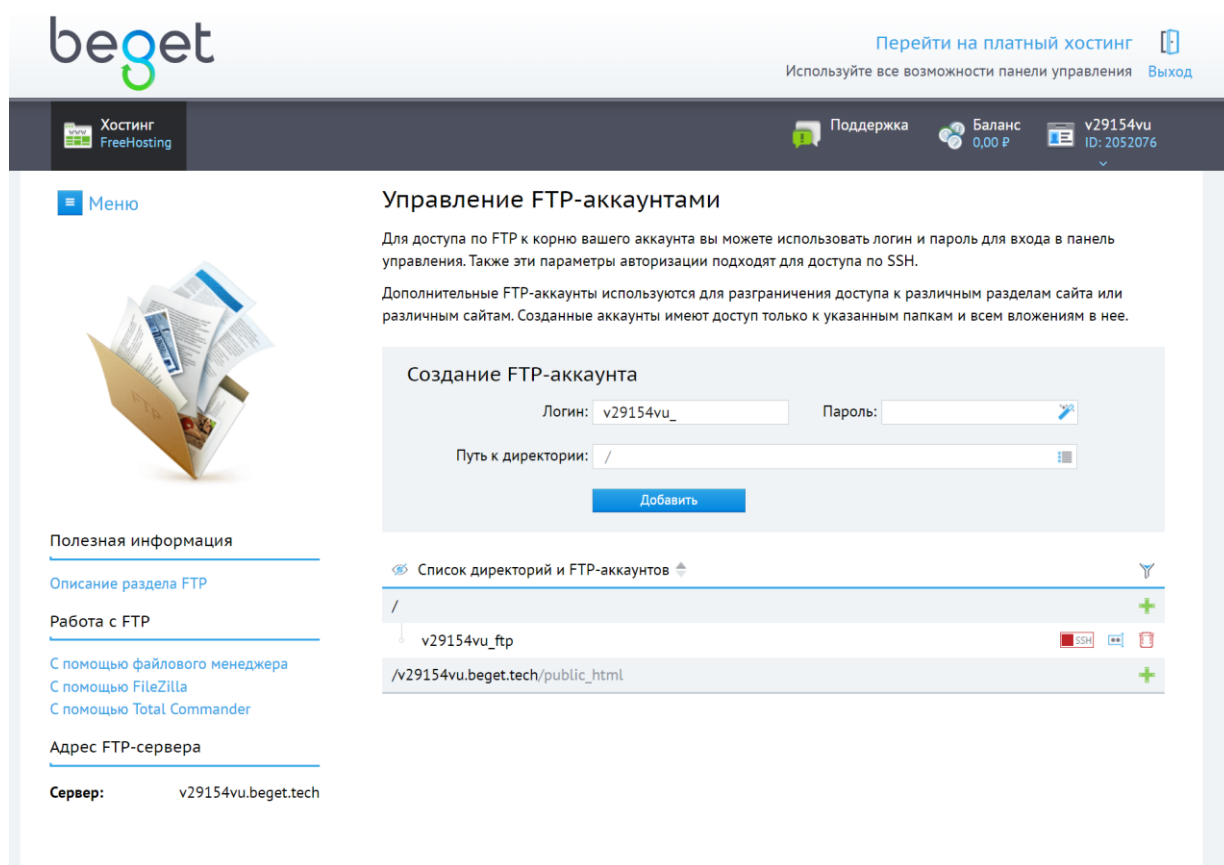


Рисунок 3.5 – Создание FTP-аккаунта

После создания FTP-аккаунта получаем данные для подключения: хост, пароль, имя пользователя. В fileZila создаем подключение и получаем все файлы для изменений. Пример подключения через FileZila представлен на рисунке 3.6

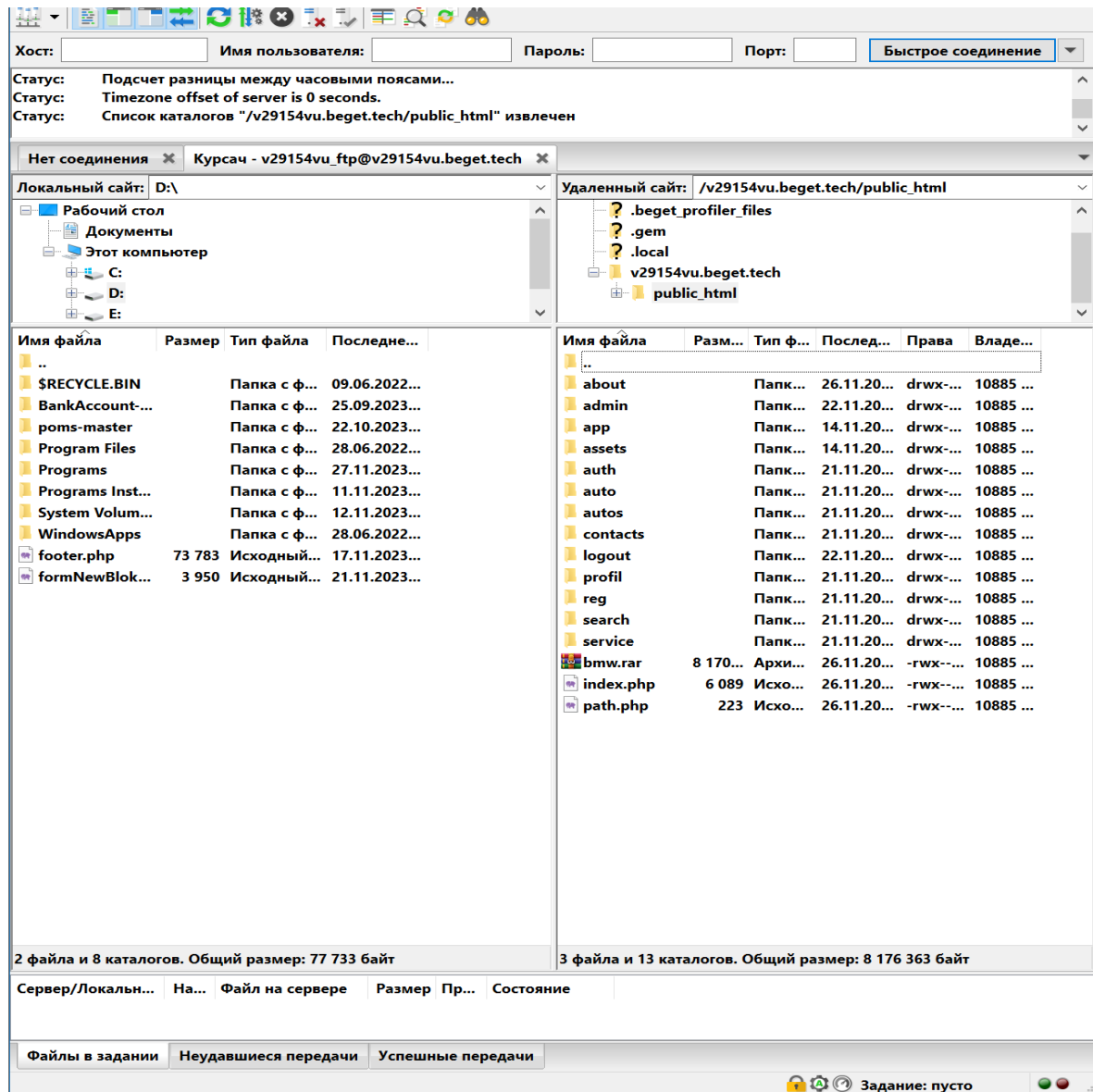
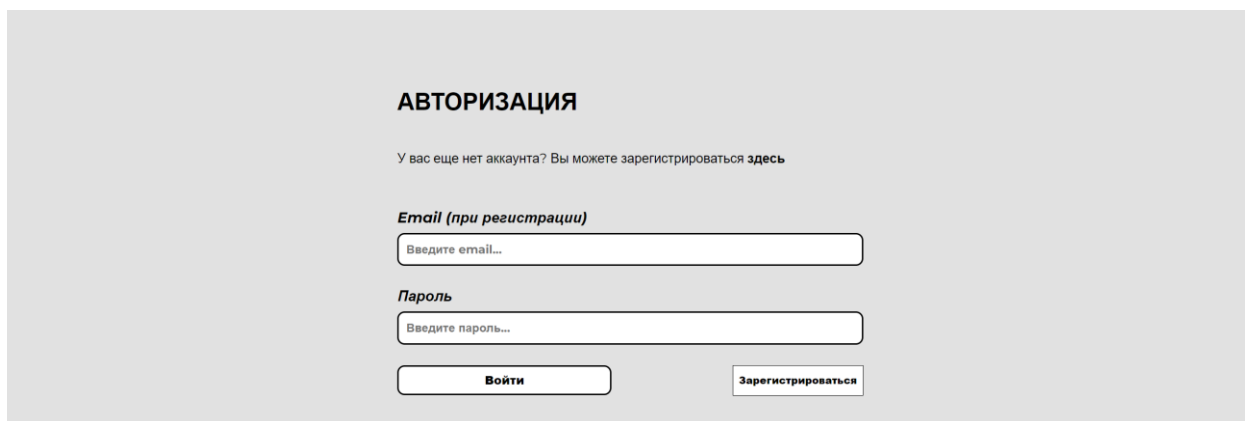


Рисунок 3.6 – Подключение к веб-сайту для изменений через FileZila

Данная курсовая работа доступна в свободном доступе сети Интернет по ссылке, которая доступна и открывается на любом устройстве и любом браузере.

3.2 Руководство по эксплуатации программного средства

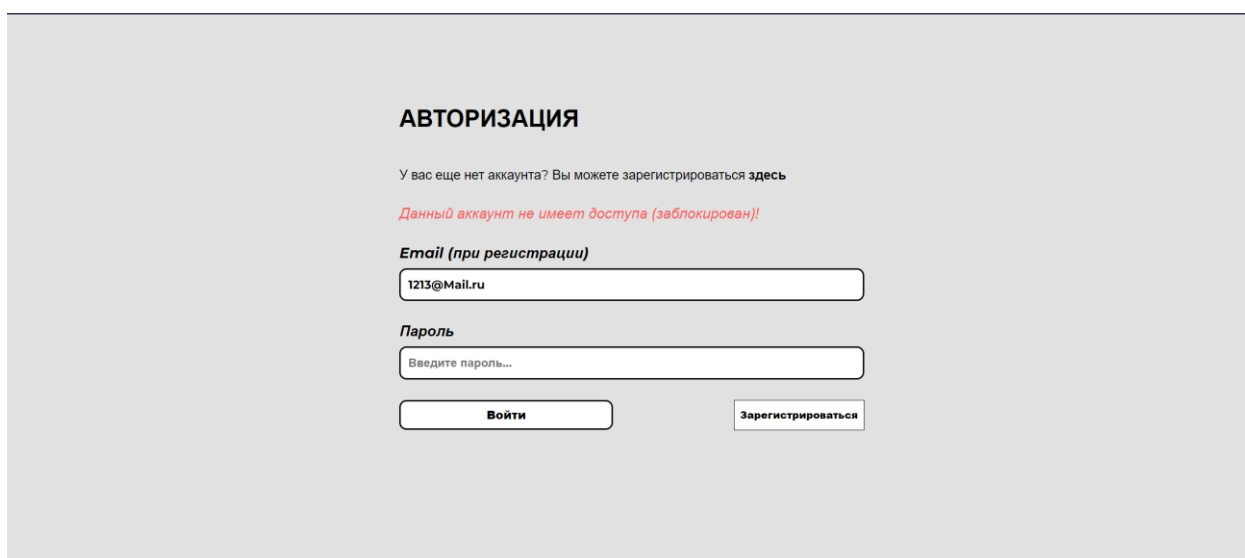
В первую очередь, при проектировании пользовательского интерфейса, нужно создать грамотный экран «входа». Экран «входа» является лицом приложения, это тот экран, с которым пользователь будет взаимодействовать в первую очередь. Поэтому он должен сочетать в себя приятный интерфейс, который сможет зацепить нового пользователя. Он должен быть интуитивно понятен. Пример интерфейса авторизации представлен на рисунке 3.7.



The screenshot shows a login form with the title "АВТОРИЗАЦИЯ". Below the title is a link: "У вас еще нет аккаунта? Вы можете зарегистрироваться [здесь](#)". There are two input fields: "Email (при регистрации)" with placeholder text "Введите email..." and "Пароль" with placeholder text "Введите пароль...". At the bottom are two buttons: "Войти" and "Зарегистрироваться".

Рисунок 3.7 – Интерфейс авторизации

Если пользователь не зарегистрирован либо же не имеет доступа, войти в аккаунт не получится. Пример исключительной ситуации авторизации показан на рисунке 3.8.



The screenshot shows the same login form as in Figure 3.7, but with an error message in red text: "Данный аккаунт не имеет доступа (заблокирован)!". The "Email (при регистрации)" field now contains the text "1213@Mail.ru". The "Пароль" field still has the placeholder "Введите пароль...". The buttons "Войти" and "Зарегистрироваться" are still present at the bottom.

Рисунок 3.8 – Исключительная ситуация при авторизации

Если пользователь не зарегистрирован, ему следует перейти по кнопке/ссылке «Зарегистрироваться» внизу окна авторизации. После перехода на страницу регистрации пользователю предоставляется следующий интерфейс, указанный на рисунке 3.9.

Автомобили О компании Услуги Контакты Кабинет

РЕГИСТРАЦИЯ

У вас уже есть аккаунт? Вы можете авторизоваться [здесь](#)
* - обязательное поле для заполнения

Личные данные
Фамилия*
Введите фамилию...
Имя*
Введите имя...
Email*
Введите почту...

Данные для входа
Логин (более 3-х символов)*
Введите логин...
Пароль*
Введите пароль...
Повторите пароль*
Введите пароль...

Зарегистрироваться Войти

Рисунок 3.9 – Интерфейс регистрации

Исключительная ситуация, если пользователь пытается зарегистрироваться с логином, который уже занят. Пример исключительной ситуации при регистрации с логином, который уже занят показан на рисунке 3.10.

Автомобили О компании Услуги Контакты Кабинет

РЕГИСТРАЦИЯ

У вас уже есть аккаунт? Вы можете авторизоваться [здесь](#)
* - обязательное поле для заполнения

Пользователь с таким логином уже зарегистрирован!

Личные данные
Фамилия*
Разумов
Имя*
Дмитрий
Email*
admin@mail.ru

Данные для входа
Логин (более 3-х символов)*
Введите логин...
Пароль*
Введите пароль...
Повторите пароль*
Введите пароль...

Зарегистрироваться Войти

Рисунок 3.10 – Исключительная ситуация при регистрации

После успешной авторизации и регистрации, пользователь может перейти на персональную страницу. На персональной странице он видит всю информацию о себе, свои заказы, которые может отменить или перейти на страницу автомобилей и оформить новый заказ. Также пользователь может редактировать свои данные, изменить пароль, удалить свой аккаунт или выйти из аккаунта. Пример персональной страницы представлен на рисунке 3.11.



Рисунок 3.11 – Персональная страница пользователя

Интерфейс редактирования профиля, включает редактирование персональных, паспортных и адресных данных. Пример редактирования профиля представлен на рисунке 3.12.

The image shows a 'Редактирование профиля' (Edit Profile) dialog box. At the top, it has a title bar with a close button (X). Below the title, there is a note: '* - обязательное поле для заполнения' (required field for filling). A message states: 'Для того чтобы оформить заказ на авто, у вас должны быть заполнены все поля!' (To place an order for a car, all fields must be filled!). The form is titled 'Личные данные' (Personal data) and contains several input fields: 'Фамилия*' (Surname) with the value 'Островская', 'Имя*' (Name) with the value 'Анастасия', 'Отчество' (Patronymic) with the value 'Александровна', 'Дата рождения' (Date of birth) with the value '11.02.2000' and a calendar icon, and 'Номер телефона' (Phone number) with the value '80447103212'. The background shows a dark sidebar with navigation links like 'компании', 'Услуги', and 'Контакты', and a main content area with some text.

Рисунок 3.12 – Интерфейс редактирования данных пользователя

Пример смены пароля включающий проверку на совпадение паролей показан на рисунке 3.13.

The image shows a 'Изменение пароля' (Change Password) dialog box. It has a title bar with a close button (X). A red error message at the top says 'Пароли должны совпадать!' (Passwords must match!). Below this, there is a note: '* - обязательное поле для заполнения' (required field for filling). The form contains two input fields: 'Введите пароль*' (Enter password) and 'Повторите пароль*' (Repeat password). The first field has a masked password '...', and the second field has a masked password '1'. At the bottom, there is a blue button labeled 'Изменить пароль' (Change password). The background shows a dark sidebar with navigation links like 'авто', '23', '00', and 'Базовая'.

Рисунок 3.13 – Пример изменения пароля

Если пользователь зашел в функционал Менеджера, то у него есть возможность работы с моделями, автомобилями и заказами. Есть возможность просматривать, добавлять, удалять, редактировать. Пример панели управления для менеджера показан на рисунке 3.14.

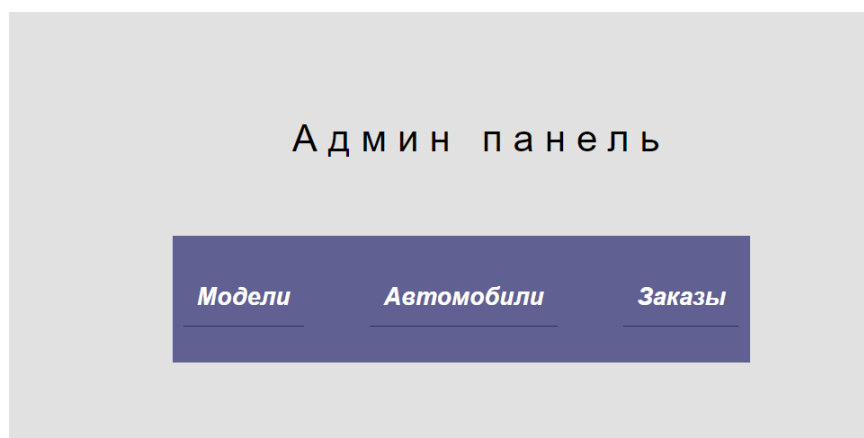


Рисунок 3.14 – Панель управления для менеджера

Если пользователь зашел в функционал Администратора, то у него есть возможность работы с клиентами, сотрудниками, контактами. Есть возможность просматривать, добавлять, удалять, редактировать. Пример панели управления для администратора показан на рисунке 3.15.

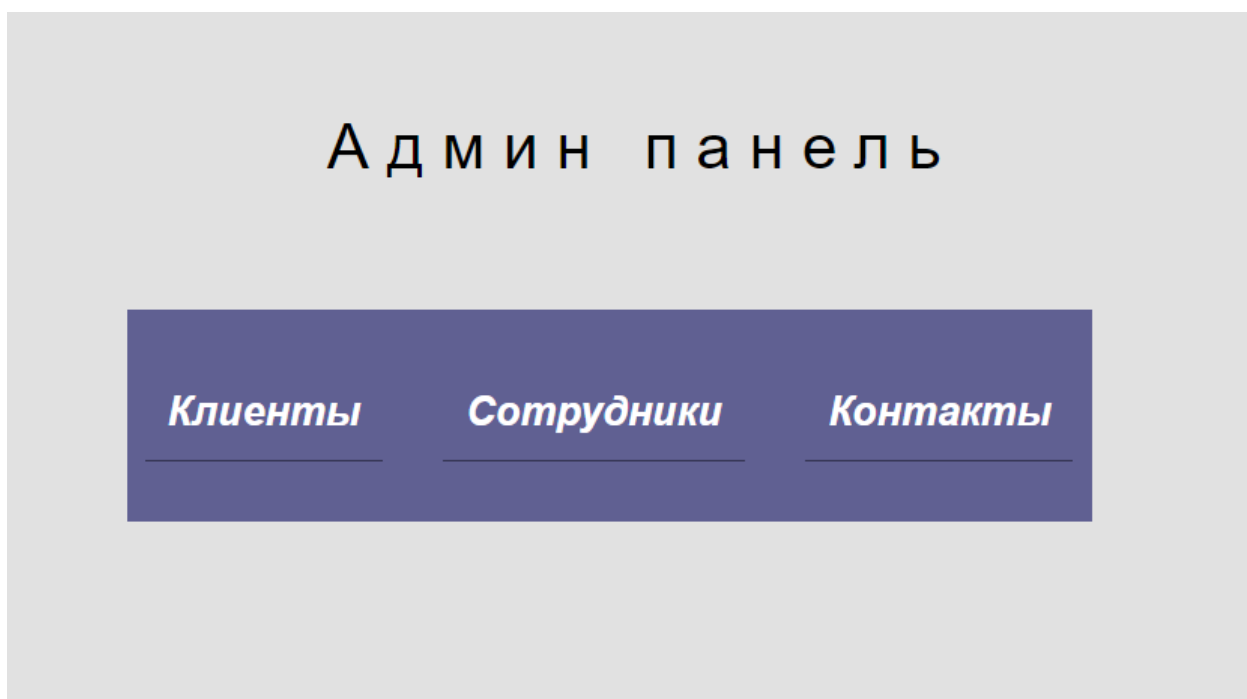


Рисунок 3.15 – Панель управления для администратора

Перейдя на страницу «Автомобили» можно посмотреть все автомобили автосалона, а также перейти на персональную страницу каждого автомобиля. Также можно произвести поиск. Пример страницы показан на рисунке 3.16.

Автомобили О компании Услуги Контакты

ost

BMW

Выберите свой автомобиль

Подберите авто по вашим запросам

Название	Цвет	Комплектация	Наличие	Двигатель	Состояние	Год выпуска
<input type="text"/>	Цвет: <input type="button" value="v"/>	Комплектация: <input type="button" value="v"/>	<input type="checkbox"/> Есть	<input type="checkbox"/> Бензиновый	<input type="checkbox"/> Новое	От <input type="text"/>
				<input type="checkbox"/> Электрический	<input type="checkbox"/> Бу	До <input type="text"/>

Цена

От

До

Найти

ALL X M 8 7 6 5 4 3 2 1 Z4

X

Рисунок 3.16 – Страница «Автомобили» с возможностью поиска и сортировки по названию модели

На странице Автомобили, можно перейти на страницу автомобиля, на которой описаны характеристики данного автомобиля. Также можно оформить авто. Пример страницы с автомобилем представлен на рисунке 3.17.

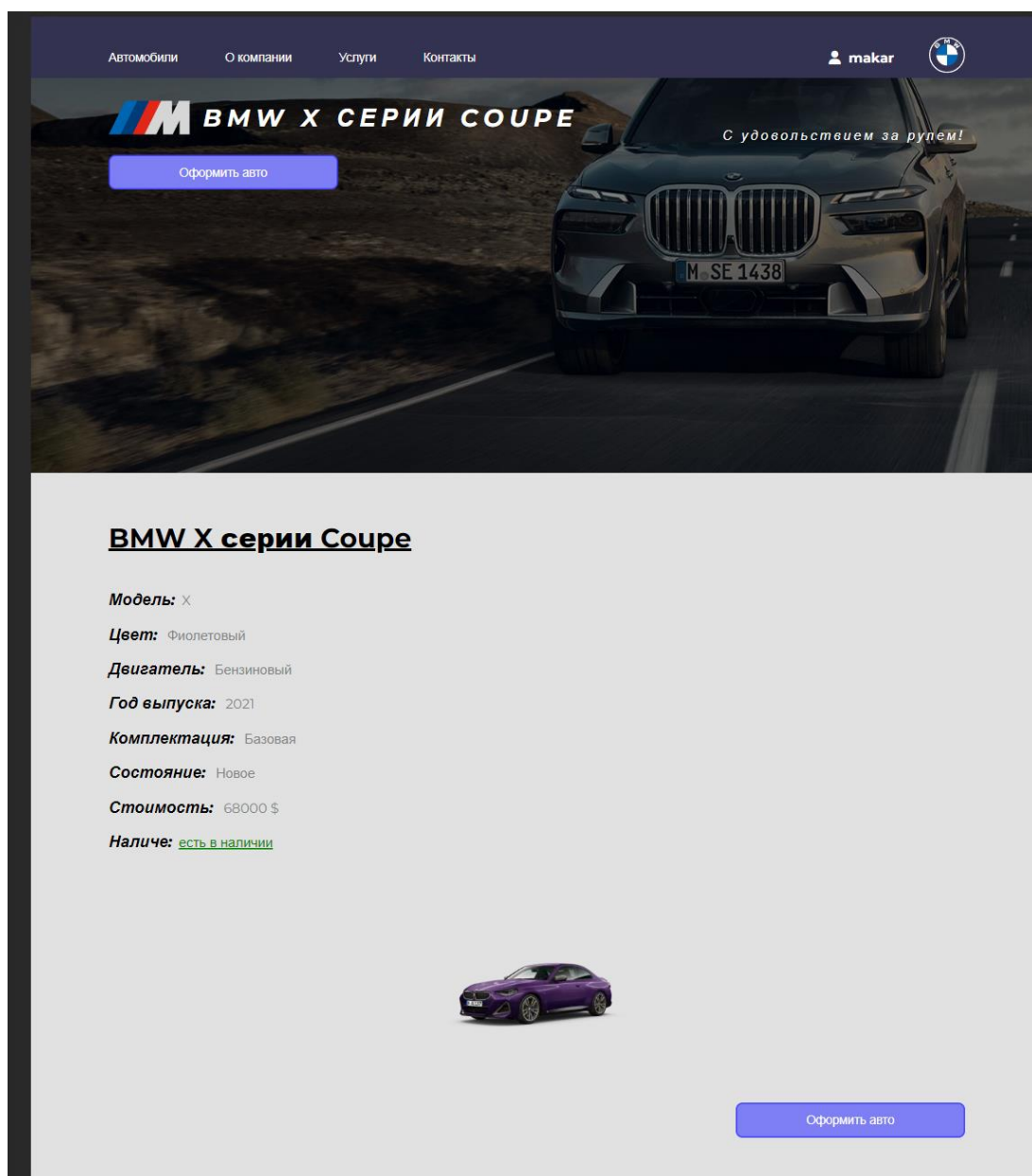


Рисунок 3.17 – Пример страницы с автомобилем

Оформление автомобиля возможно только клиентам, у которых заполнена вся личная информация, если же у клиента не заполнено хотя бы одно поле, клиент не может оформить авто. Пример оформления авто представлен на рисунке 3.18.

Оформление заказа на автомобиль **BMW X серии Coupe**

Место жительства: г.Брест, ул., д.18, кв.10
Серия и номер паспорта: МР9301212
Кем выдан: Брестким РОВД

Данные автомобиля:
Модель: BMW X серии Coupe
Коробка передач: Бензиновый
Цвет: Фиолетовый
Год выпуска: 2021
Состояние: Новое
Комплектация: Базовая

СТОИМОСТЬ: 68000

Оформление заказа

Ваша почта

Ваш логин

Введите пароль для оформления заказа

Повторите пароль для подтверждения заказа

Оформить заказ

Рисунок 3.18 – Пример оформления авто

Страница «Контактов» выводит все контактные данные автосалона, а также есть возможность добавления новых контактов, редактирования старых и удаление существующих. Страница контактов представлена на рисунке 3.19.

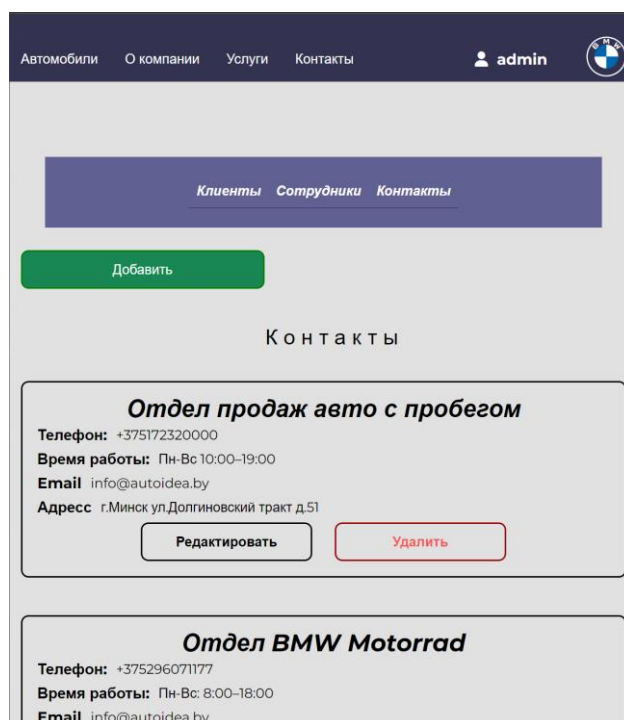


Рисунок 3.19 – Страница «Контактов» через админ панель

Страница добавления новых контактных данных представлена на рисунке 3.20.

The screenshot shows the 'Добавление контактных данных' (Add contact data) form. On the left is a sidebar with 'Клиенты', 'Сотрудники', and 'Контакты'. The form title is 'Добавление контактных данных'. A note states: '* - обязательное поле для заполнения' (required field for filling). The form contains the following fields: 'Название*' (Name), 'Телефон*' (Phone), 'Время работы*' (Working hours), 'Email*', 'Город*' (City), 'Улица*' (Street), and 'Дом*' (House). Each field has a placeholder text. A blue 'Добавить' (Add) button is at the bottom.

Рисунок 3.20 – Страница добавления контактных данных через админ панель

Страница редактирования контактных данных показана на рисунке 3.21.

Р е д а к т и р о в а н и е к о н т а к т н ы х
д а н н ы х

* - обязательное поле для заполнения

Название*
Отдел продаж авто с пробегом

Телефон*
+375172320000

Время работы*
Пн-Вс 10:00-19:00

Email*
info@autoidea.by

Город*
Минск

Улица*
Долгиновский тракт

Дом*
51

Сохранить

Рисунок 3.21 – Страница редактирования контактных данных

Страницы «Клиентов» и «Сотрудников» в админ панели однотипны. Имеют вывод всех клиентов/сотрудников, данные каждого можно редактировать, давать/убирать доступ для входа, удалять и добавлять новых, а также есть функция поиска по ФИО. Пример вывода всех клиентов/сотрудников представлена на рисунке 3.22.

Добавить

Поиск...

Клиенты
Сотрудники
Контакты

С о т р у д н и к и

Малиновский1 Юрий Вадимович

Дата рождения: 1979-06-02
Телефон: 80337593812
Место жительства: г.Могилев ул.Володарского д.45 кв.19

Серия и номер паспорта: МС18417212
Кем выдан: Могилевским РОВД

Логин: mal
Email: mal@mail.ru
Дата регистрации: 2023-11-24 19:22:35
Должность: Менеджер
Доступ: Нет доступа

Редактировать Доступ Удалить

Рисунок 3.22 – Вывод всех клиентов/сотрудников

Страница редактирования включает редактирование всех персональных данных, кроме логина, email и пароля. Страница редактирования клиента/сотрудника через админ панель представлена на рисунке 3.23.

Редактирование сотрудника

Малиновский1 Юрий Вадимович

* - обязательное поле для заполнения

Личные данные

Фамилия*

Имя*

Отчество

Дата рождения*

Номер телефона

Должность:

Данные по месту жительства

Город

Улица

Номер дома

Номер квартиры

Рисунок 3.23 – Страница редактирования клиента/сотрудника через админ панель

Страница добавления клиента/сотрудника через админ панель представлена на рисунке 3.24.

Добавление сотрудника

* - обязательное поле для заполнения

Личные данные

Фамилия*

Введите фамилию...

Имя*

Введите имя...

Отчество

Введите имя...

Дата рождения*

дд.мм.гггг

Номер телефона

Введите номер телефона...

Должность:

Выберите должность:

Данные по месту жительства

Город

Введите город...

Улица

Введите улицу...

Рисунок 3.24 – Добавление клиента/сотрудника через админ панель

Страница «Моделей» включает в себя вывод всех моделей авто в авто-салоне, удаление и редактирование существующих моделей и добавление новых. Пример страницы моделей через админ панель показан на рисунке 3.25.

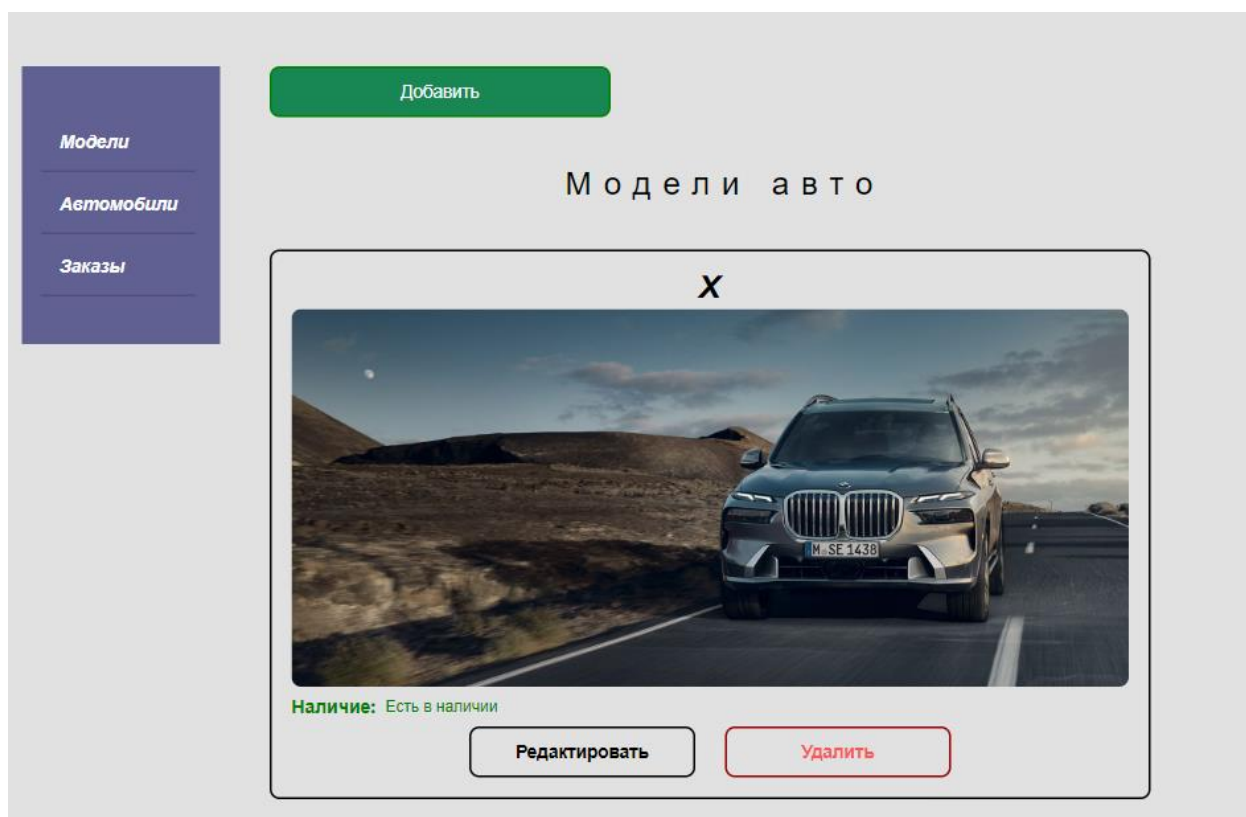


Рисунок 3.25 – Страница «Моделей» через админ панель

Страницы редактирования модели авто через админ панель показана на рисунке 3.26.

The screenshot shows a form titled 'Обновление модели авто' (Update car model). It includes a note: '* - обязательное поле для заполнения' (* - required field for filling). The first field is 'Модель*' (Model*), containing the text 'x'. The second section is 'Выберите главное фото модели (выбрано по умолчанию)' (Select the main photo of the model (selected by default)), with a file selection area showing 'Выберите файл' (Select file) and 'Файл не выбран' (File not selected). At the bottom is a blue 'Обновить' (Update) button.

Рисунок 3.26 – Редактирование модели автомобиля через админ панель

Страница добавления модели авто через админ панель представлена на рисунке 3.27.

The screenshot shows a web form titled "Добавление модели авто" (Add car model). Below the title is a note: "* - обязательное поле для заполнения" (Required field for filling). The form contains two main sections. The first section is labeled "Модель*" and features a text input field with the placeholder text "Модель". The second section is labeled "Выберите главное фото модели*" (Select the main photo of the model) and contains a file selection interface with a button labeled "Выберите файл" (Select file) and the text "Файл не выбран" (File not selected). At the bottom of the form is a blue button labeled "Добавить" (Add).

Рисунок 3.27 – Страница добавления модели авто через админ панель

Страница «Заказов» включает в себя вывод всех заказов, удаление существующих и добавление новых. Страница заказов через админ панель представлена на рисунке 3.28.

The screenshot displays the "Заказы" (Orders) section of the admin panel. At the top left is a green button labeled "Добавить" (Add). The main heading is "Заказы". Below this, there are two order cards. The first card is titled "Заказ №77" and lists the following details: "Клиент: Островская Анастасия", "Авто: X1", "Стоимость: 78000", "Дата оформления: 2023-11-24 20:33:39", "Номер телефона: 80447103212", and "Email: ost@mail.ru". A red button labeled "Удалить" (Delete) is positioned at the bottom right of this card. The second card is titled "Заказ №78" and shows: "Клиент: Артюшевская Ангелина" and "Авто: 8Gran Coupe".

Рисунок 3.28 – Страница «Заказов» через админ панель

Страница добавления новых заказов через менеджер панель представлена на рисунке 3.29.

The screenshot shows a form titled 'Добавление заказа' (Add Order). At the top, there is a note: '* - обязательное поле для заполнения' (* - required field for filling). The form contains two dropdown menus: 'Автомобиль*' (Car*) with the placeholder 'Выберите авто:' (Select car:) and 'Клиент*' (Client*) with the placeholder 'Выберите клиента:' (Select client:). Below these fields is a blue button labeled 'Добавить' (Add).

Рисунок 3.29 – Добавление заказа через менеджер панель

Страницы «Автомобилей» имеет функционал вывода всех автомобилей, данные каждого из которых можно редактировать, снимать с публикации/публиковать, удалять и добавлять новые авто. Пример вывода всех автомобилей через менеджер панель представлен на рисунке 3.30.

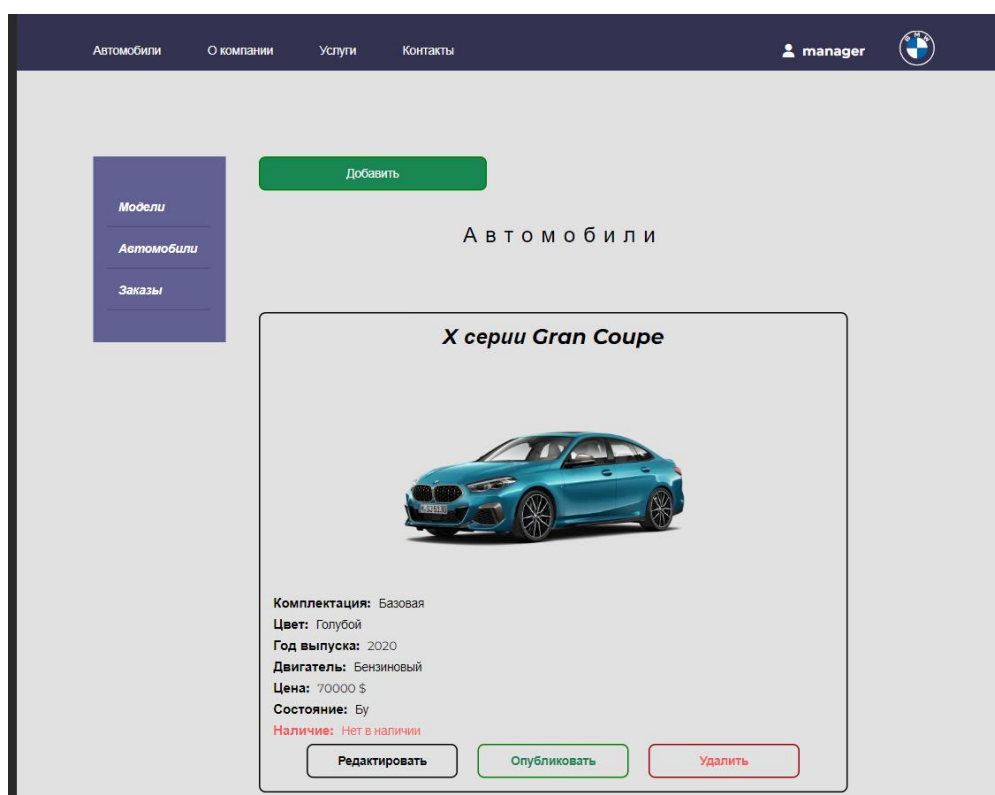


Рисунок 3.30 – Страница «Автомобилей» через менеджер панель

Добавление нового автомобиля через менеджер панель представлена на рисунке 3.31.

The screenshot shows a web application interface for adding a car. At the top, there is a dark blue header with navigation links: 'Автомобили', 'О компании', 'Услуги', and 'Контакты'. On the right side of the header, there is a user profile icon labeled 'manager' and a circular logo. Below the header, on the left, is a vertical sidebar with three buttons: 'Модели', 'Автомобили', and 'Заказы'. The main content area is titled 'Добавление авто'. Below the title, there is a note: '* - обязательное поле для заполнения'. The form consists of several fields: 'Название*' (text input), 'Год выпуска*' (text input), 'Комплектация*' (dropdown menu with 'Выберите комплектацию'), 'Цвет*' (text input), 'Двигатель*' (dropdown menu with 'Выберите двигатель'), 'Состояние*' (dropdown menu with 'Выберите состояние:'), 'Цена (\$)*' (text input with 'Цена \$' placeholder), 'Модель*' (dropdown menu with 'Выберите модель:'), and 'Выберите фото авто*' (file upload area with 'Выберите файл' and 'Файл не выбран' buttons). Below these fields is a checkbox labeled 'Наличие:'. At the bottom of the form is a blue button labeled 'Добавить'. The footer of the page is dark blue and contains a circular logo, navigation links: 'Автомобили', 'О компании', 'Услуги и сервис', and 'Контакты', social media icons for VK, YouTube, Telegram, and Instagram, and copyright information: 'Разработка сайта © 2023 dima.razumov.940@mail.ru'.

Автомобили О компании Услуги Контакты manager

Модели
Автомобили
Заказы

Добавление авто

* - обязательное поле для заполнения

Название*

Год выпуска*

Комплектация*

Выберите комплектацию

Цвет*

Двигатель*

Выберите двигатель

Состояние*

Выберите состояние:

Цена (\$)*

Цена \$

Модель*

Выберите модель:

Выберите фото авто*

Выберите файл Файл не выбран

Наличие: ☐

Добавить

Автомобили О компании Услуги и сервис Контакты

Разработка сайта © 2023 dima.razumov.940@mail.ru

Рисунок 3.31 – Добавление автомобиля через менеджер панель

ЗАКЛЮЧЕНИЕ

В результате выполнения работы была достигнута цель – разработана база данных для автосалона. Данный веб-сайт был написан на языке программирования PHP, поэтому для его запуска на компьютер необходимо установить подходящую среду разработки (рекомендуемая – Visual Code) или же перейти по ссылке находящейся в свободном доступе. Также веб-сайт доступен на любом устройстве с любой операционной системой.

Приложение просто в использовании: в нем имеется интуитивно понятный функционал, который не предоставит пользователю трудности в его эксплуатации.

Данное приложение может быть полезно для работы автосалона, работающих с крупными покупателями.

Клиент автосалона сможет просматривать всю информацию об автомобилях в автосалоне, контактную информацию. Также у клиента есть возможность быстрого оформления авто онлайн, а все заказы просмотреть в личном кабинете.

Администратор автосалона сможет со всеми удобствами и комфортом работать с данными о сотрудниках, просматривать результаты их труда, оценивать данную информацию, формируя их рейтинг в виде оценки по десятибалльной шкале. Также администратор сможет при необходимости получить данные всех клиентов и узнать количество заказов в автосалоне, и опираясь на данную информацию, делать скидки для постоянных клиентов.

Менеджер, работающий в автосалоне, сможет с помощью данного веб-сайта быстро находить нужные заказы и автомобили в базе данных, редактировать их при необходимости, а также добавлять новые в базу данных.

Таким образом, данный веб-сайт в перспективе сможет принести много пользы множеству автосалонов, поскольку оно создавалось специально для упрощения взаимодействия с данными в этой области.

В дальнейшем разработанную систему можно усовершенствовать, добавив дополнительный функционал.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] БГУИР ГОСТ СТП 01-2017 [Электронный ресурс]. – Режим доступа: https://www.bsuir.by/m/12_100229_1_122976.pdf - Дата доступа: 05.12.2023.
- [2] Wikipedia: автосалон [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Автосалон> – Дата доступа: 01.10.2023.
- [3] Wikipedia: PHP [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/PHP> – Дата доступа: 11.11.2023.
- [4] PHP.net [Электронный ресурс]. – Режим доступа: <https://www.php.net/> – Дата доступа: 22.11.2023.
- [5] Laravel [Электронный ресурс]. – Режим доступа: <https://laravel.com/> – Дата доступа: 19.11.2023.
- [6] MVC [Электронный ресурс]. – Режим доступа: <https://anton-pribora.ru/articles/php/mvc/> – Дата доступа: 19.11.2023.
- [7] Реализация MVC паттерна на примере создание сайте на PHP [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/150267/> – Дата доступа: 02.11.2023.
- [8] Основы HTML – Изучение веб-разработки [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/HTML> – Дата доступа: 27.10.2023.
- [9] Руководство по CSS: каскадные таблицы [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/CSS/Reference> – Дата доступа: 28.10.2023.
- [10] Основы SCSS [Электронный ресурс]. – Режим доступа: <https://sass-scss.ru/guide/> – Дата доступа: 29.10.2023.
- [11] Современный учебник JavaScript [Электронный ресурс]. – Режим доступа: <https://learn.javascript.ru/> – Дата доступа: 29.10.2023.
- [12] Интегрированная среда разработки [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Интегрированная_среда_разработки – Дата доступа: 11.10.2023.
- [13] VSCode [Электронный ресурс]. – Режим доступа: <https://code.visualstudio.com/> – Дата доступа: 29.10.2023.
- [14] OpenServer локальный сервер, установка [Электронный ресурс]. – Режим доступа: <https://www.youtube.com/watch?v=cVXWqyE6HoA&t=1553s> – Дата доступа: 10.10.2023.
- [15] MySQL [Электронный ресурс]. – Режим доступа: <https://www.mysql.com/> – Дата доступа: 29.11.2023.
- [16] PHPMyAdmin [Электронный ресурс]. – Режим доступа: <https://www.phpmyadmin.net/> – Дата доступа: 15.10.2023.
- [17] GIT [Электронный ресурс]. – Режим доступа: <https://git-scm.com/> – Дата доступа: 28.11.2023.
- [18] GitHub [Электронный ресурс]. – Режим доступа: <https://blog.hubspot.com/website/what-is-github-used-for> – Дата доступа: 05.12.2023.

- [19] Git и GitHub курс для новичков [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=zZBiln_2FhM – Дата доступа: 10.10.2023.
- [20] Gulp установка настройка плагина [Электронный ресурс]. – Режим доступа: <https://www.youtube.com/watch?v=jU88mLuLWlk&t=2055s> – Дата доступа: 11.10.2023.
- [21] Паттерн Dependency Injection [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/64061/> – Дата доступа: 29.10.2023.
- [22] Паттерн Singleton на PHP [Электронный ресурс]. – Режим доступа: <https://refactoring.guru/ru/design-patterns/singleton/php/example> – Дата доступа: 29.11.2023.
- [23] ORM [Электронный ресурс]. – Режим доступа: <https://php.zone/oop-v-php-prodvintyj-kurs/object-relational-mapping-orm-v-php> – Дата доступа: 24.11.2023.
- [24] Введение в REST API – RESTFUL веб-сервисы. – Режим доступа: <https://habr.com/ru/articles/483202/> – Дата доступа: 21.10.2023.
- [25] Принципы UI/UX. – Режим доступа: https://habr.com/ru/companies/SECL_GROUP/articles/182208/ – Дата доступа: 25.10.2023.
- [26] Что такое и зачем нужны алгоритмы [Электронный ресурс]. – Режим доступа: <https://htmlacademy.ru/blog/js/algorithm> – Дата доступа: 30.11.2023.
- [27] Виды алгоритмов и типы их схем [Электронный ресурс]. – Режим доступа: <https://otus.ru/nest/post/1784/> – Дата доступа: 01.12.2023.
- [28] Что такое хостинг [Электронный ресурс]. – Режим доступа: <https://hostiq.ua/info/what-is-hosting/> – Дата доступа: 01.12.2023.
- [29] Руководство по панели управления Beget [Электронный ресурс]. – Режим доступа: <https://beget.com/ru/kb/manual> – Дата доступа: 02.12.2023.
- [30] Выгрузка проекта на хостинг [Электронный ресурс]. – Режим доступа: <https://www.youtube.com/watch?v=UPnVcTZUwXE&t=854s> – Дата доступа: 04.12.2023.

ПРИЛОЖЕНИЕ А

(Обязательное)

Отчёт о проверке на заимствования в системе «Антиплагиат»



Отчет о проверке на заимствования №1



Автор: dima.razumov.940@mail.ru / ID: 11125416

Проверяющий:

Отчет предоставлен сервисом «Антиплагиат» - <http://users.antiplagiat.ru>

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 1
Начало загрузки: 05.12.2023 15:08:59
Длительность загрузки: 00:00:02
Имя исходного файла: Курсовая-записка-PDF.pdf
Название документа: Курсовая-записка-PDF
Размер текста: 65 кБ
Символов в тексте: 66081
Слов в тексте: 8236
Число предложений: 414

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 05.12.2023 15:09:02
Длительность проверки: 00:00:06
Комментарии: не указано
Модули поиска: Интернет Free



СОВПАДЕНИЯ
7,07%

САМОЦИТИРОВАНИЯ
0%

ЦИТИРОВАНИЯ
0%

ОРИГИНАЛЬНОСТЬ
92,93%

Совпадения — фрагменты проверяемого текста, полностью или частично сходные с найденными источниками, за исключением фрагментов, которые система отнесла к цитированию или самоцитированию. Показатель «Совпадения» — это доля фрагментов проверяемого текста, отнесенных к совпадениям, в общем объеме текста.

Самоцитирование — фрагменты проверяемого текста, совпадающие или почти совпадающие с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа. Показатель «Самоцитирование» — это доля фрагментов текста, отнесенных к самоцитированию, в общем объеме текста.

Цитирование — фрагменты проверяемого текста, которые не являются авторскими, но которые система отнесла к корректно оформленным. К цитированиям относятся также шаблонные фразы: библиография; фрагменты текста, найденные модулем поиска «СПС Гарант: нормативно-правовая документация». Показатель «Цитирование» — это доля фрагментов проверяемого текста, отнесенных к цитированию, в общем объеме текста.

Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.

Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.

Оригинальный текст — фрагменты проверяемого текста, не обнаруженные ни в одном источнике и не отмеченные ни одним из модулей поиска. Показатель «Оригинальность» — это доля фрагментов проверяемого текста, отнесенных к оригинальному тексту, в общем объеме текста.

Рисунок А.1 — Отчет о проверке на заимствования в система «Антиплагиат»

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинги программного кода

Файл AutoController.php

```
<?php

require(SITE_ROOT . '/app/services/AutoService.php');
$autoService = new AutoService();

class AutoController {

    public function addAuto(): void {
        global $autoService;

        if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['auto-create'])) {
            $autoService -> addAuto();
        }
    }

    public function updateAuto() {
        global $autoService;

        if($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['id'])) {

            $db = new DataB();

            $id = $_GET['id'];
            $auto = $db->selectOne('auto', ['id' => $id]);

            $id = $auto['id'];
            $name = $auto['name'];
            $complexion = $auto['complexion'];
            $color = $auto['color'];
            $year = $auto['year'];
            $engine = $auto['engine'];
            $price = $auto['price'];
            $status = $auto['status'];
            $img = $auto['img'];
            $model = $auto['id_model'];

            $arrRes = [$id, $name, $complexion, $color, $year, $engine, $price, $status, $img, $model];
            return $arrRes;
        }
    }
}
```

```

public function editAuto(): void {
    global $autoService;

    if($_SERVER['REQUEST_METHOD'] === 'POST' && is-
set($_POST['auto-edit'])) {
        $autoService -> updateAuto();
    }
}

public function editStatusAuto(): void {
    global $autoService;

    if($_SERVER['REQUEST_METHOD'] === 'GET' && is-
set($_GET['pub_id'])) {
        $id = $_GET['pub_id'];
        $autoService -> updateStatusAuto($id);
    }
}

public function deleteAuto(): void {
    global $autoService;

    if($_SERVER['REQUEST_METHOD'] === 'GET' && is-
set($_GET['del_id'])) {
        $id = $_GET['del_id'];
        $autoService -> deleteAuto($id);
    }
}

public function searchAuto() {
    $db = new DataB();

    if($_SERVER['REQUEST_METHOD'] === 'POST' && is-
set($_POST['search__auto'])) {

        $priceFrom = $_POST['price__from'];
        $priceTo = $_POST['price__to'];
        $yearFrom = $_POST['year__from'];
        $yearTo = $_POST['year__to'];
        $engine = $_POST['engine'];
        $status = $_POST['status'];
        $col = $_POST['color'];
        $name = $_POST['name'];
        $complex = $_POST['complexion'];
        $state = $_POST['state'];

        if($complex == 'Комплектация:') {
            $complex = '';
        }

        if($col == 'Цвет:') {
            $col = '';
        }
    }
}

```

```

    }

    $paramsPrice = [
        'price__from' => $priceFrom,
        'price__to' => $priceTo
    ];

    $paramsYear = [
        'year__from' => $yearFrom,
        'year__to' => $yearTo
    ];

    $params = [
        'name' => $name,
        'complexion' => $complex,
        'color' => $col,
        'status' => $status,
        'engine' => $engine,
        'state' => $state
    ];

    $arrRes =
    [
        $priceFrom, $priceTo, $yearFrom, $yearTo, $engine,
        $status, $col, $name, $complex, $state
    ];

    $autos = $db->searchAutos($params, $paramsPrice,
$paramsYear);

    return [$arrRes, $autos];
}
}
}

// ?>

```

Файл AutoService.php

```

<?php

class AutoService {

    private $AVAILABLE = 1;
    private $NO_AVAILABLE = 0;

    public function addAuto(): void {

        $db = new DataB();
    }
}

```

```

$status = trim($_POST['status']);
if(isset($_POST['status'])) {
    $status = $this -> AVAILABLE;
} else {
    $status = $this -> $NO_AVAILABLE;
}

$auto = [
    'name' => trim($_POST['name']),
    'engine' => trim($_POST['engine']),
    'year' => trim($_POST['year']),
    'price' => trim($_POST['price']),
    'color' => trim($_POST['color']),
    'complexion' => trim($_POST['complexion']),
    'img' => $_FILES['img']['name'],
    'status' => $status,
    'state' => trim($_POST['state']),
    'id_model' => trim($_POST['model'])
];

print_r($status);

$auto = $db->insert('auto', $auto);
$auto = $db->selectOne('auto', ['id' => $id]);
header('location:' . ADMIN_URL . '/auto');
}

public function updateAuto(): void {
    $db = new DataB();

    $status = trim($_POST['status']);
    if(isset($_POST['status'])) {
        $status = $this -> AVAILABLE;
    } else {
        $status = $this -> $NO_AVAILABLE;
    }

    if(empty($_FILES['img']['name'])) {
        $auto = [
            'name' => trim($_POST['name']),
            'engine' => trim($_POST['engine']),
            'year' => trim($_POST['year']),
            'price' => trim($_POST['price']),
            'color' => trim($_POST['color']),
            'complexion' => trim($_POST['complexion']),
            'status' => $status,
            'id_model' => trim($_POST['model'])
        ];
    } else {
        $auto = [
            'name' => trim($_POST['name']),
            'engine' => trim($_POST['engine']),

```

```

        'year' => trim($_POST['year']),
        'price' => trim($_POST['price']),
        'color' => trim($_POST['color']),
        'complexion' => trim($_POST['complexion']),
        'img' => $_FILES['img']['name'],
        'status' => $status,
        'id_model' => trim($_POST['model'])
    ];
}

$id = $_POST['id'];
$auto = $db->update('auto', $id, $auto);
header('location:' . ADMIN_URL . '/auto');
}

public function updateStatusAuto($id): void {
    $db = new DataB();
    $status = $_GET['status'];
    $autoId = $db->update('auto', $id, ['status' => $sta-
tus]);
    header('location:' . ADMIN_URL . '/auto');
}

public function deleteAuto($id): void {
    $db = new DataB();
    $db->delete('auto', $id);
    header('location:' . ADMIN_URL . '/auto');
}
}

?>

```

Файл OrderService.php

```

<?php

class OrderService {

    private $NEW_AUTO = 19;
    private $OLD_AUTO = 20;
    private $CLIENT = 0;
    private $NEW = 'Новое';
    public $errMsg = [];

    public function addOrderClient(): void {
        $db = new DataB();

        $email = $_POST['email'];
        $login = $_POST['login'];
        $passF = $_POST['password-first'];
        $passS = $_POST['password-second'];

        $idAuto = $_GET['auto'];
    }
}

```



```

        $auto = $db->selectOne('auto', ['id' => $idAuto]);
        $state = $auto['state'];

        $idSession = $_SESSION['id'];
        $roleSession = $_SESSION['role'];

        $user = $db->selectOne('authorization', ['id' => $idSession]);
        $idUser = $db->selectOne('clients', ['id_auth' => $idSession])['id'];

        $arrEmployees = $db->selectAll('employees', ['job' => 'Менеджер']);
        $randIndex = rand(0, count($arrEmployees) - 1);
        $idEmployee = $arrEmployees[$randIndex]['id'];

        if($login != $user['login'] || $email != $user['email']
        || (!password_verify($passS, $user['password'])) || $passF !=
        $passS) {
            array_push($this->errMsg, "Не верно введены данные!
            \n Заказ не был оформлен! \n Повторите попытку!");
        } else {

            if($state == $this->new) {
                $idContact = $this->NEW_AUTO;
            } else {
                $idContact = $this->OLD_AUTO;
            }

            $params = [
                'id_client' => $idUser,
                'id_auto' => $idAuto,
                'id_contact' => $idContact,
                'id_employee' => $idEmployee
            ];

            $db->insert('orders', $params);
            header('location:' . BASE_URL . '/profil');
        }
    }

    public function addOrderEmployee(): void {
        $db = new DataB();

        $idClient = $_POST['client'];
        $idAuto = $_POST['auto'];

        $idSession = $_SESSION['id'];
        $idEmployee = $db->selectOne('employees', ['id_auth' =>
        $idSession])['id'];

        $auto = $db->selectOne('auto', ['id' => $idAuto]);
        $state = $auto['state'];
    
```

```

        $CLIENTFullData = $db->selectAll('clientsview', ['id' =>
$idClient])[0];

        if($state == $this -> new) {
            $idContact = $this -> NEW_AUTO;
        } else {
            $idContact = $this -> OLD_AUTO;
        }

        $params = [
            'id_auto' => $idAuto,
            'id_client' => $idClient,
            'id_contact' => $idContact,
            'id_employee' => $idEmployee
        ];

        print_r($params);

        $db->insert('orders', $params);
        header('location:' . ADMIN_URL . '/order');
    }

    public function deleteOrder($id): void {
        $db = new DataB();

        $db->delete('orders', $id);
        if($_SESSION['role'] == $CLIENT) {
            header('location:' . BASE_URL . '/profil');
        } else {
            header('location:' . ADMIN_URL . '/order');
        }
    }
}

?>

```

Файл OrderController.php

```

<?php

require(SITE_ROOT . '/app/services/OrderService.php');
$orderService = new OrderService();

class OrderController {

    public function addOrderClient(): void {
        global $orderService;

        if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['button-order'])) {
            $orderService -> addOrderClient();
        }
    }
}

```

```

    }
}

public function addOrderEmployee(): void {
    global $orderService;

    if($_SERVER['REQUEST_METHOD'] === 'POST' && is-
set($_POST['order-create'])) {
        $orderService -> addOrderEmployee();
    }
}

public function deleteOrder(): void {
    global $orderService;

    if($_SERVER['REQUEST_METHOD'] === 'GET' && is-
set($_GET['del_order'])) {
        $id = $_GET['del_order'];
        $orderService -> deleteOrder($id);
    }
}

?>

```

ПРИЛОЖЕНИЕ В
(Обязательное)
Ведомость курсового проекта