**Министерство образования Иркутской области**

Государственное бюджетное профессиональное образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
| **ДП.09.02.07-3.25.211.04. ПЗ** |  |

**ВЕБ-ПРИЛОЖЕНИЕ «АВТОСАЛОН»**

|  |  |  |
| --- | --- | --- |
| Нормоконтролер: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Е.С. Кудрявцева) |
| Консультат по экономической части: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.П. Юргина) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (П.А. Стош) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (К.С. Мясников) |

Иркутск 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc199197120)

[1 Предпроектное исследование 5](#_Toc199197121)

[1.1 Описание предметной области 5](#_Toc199197122)

[1.2 Обзор инструментов, используемых в проектировании и разработке программного обеспечения 7](#_Toc199197123)

[2 Техническое задание 10](#_Toc199197124)

[3 Проектирование программного продукта 11](#_Toc199197125)

[3.1 Архитектура программного продукта 11](#_Toc199197126)

[3.2 Функциональное и структурное проектирование 12](#_Toc199197127)

[3.3 Проектирование базы данных 15](#_Toc199197128)

[3.4 Проектирование пользовательского интерфейса 23](#_Toc199197129)

[3.4.1 Разработка прототипов интерфейса 23](#_Toc199197130)

[3.4.2 Выбор цветовой гаммы и шрифтов 28](#_Toc199197131)

[3.4.3 Разработка элементов интерфейса 28](#_Toc199197132)

[3.4.4 Разработка дизайн макетов 29](#_Toc199197133)

[4 Реализация программного продукта 33](#_Toc199197134)

[4.1 Разработка клиентской части 33](#_Toc199197135)

[4.2 Разработка базы данных 35](#_Toc199197136)

[4.3 Разработка серверной части 37](#_Toc199197137)

[5 Отладка и тестирование программного продукта 39](#_Toc199197138)

[6 Документация программного продукта 45](#_Toc199197139)

[6.1 Руководство пользователя 45](#_Toc199197140)

[6.2 Руководство администратора 52](#_Toc199197141)

[6.3 Руководство менеджера 57](#_Toc199197142)

[7 Стоимость разработки 59](#_Toc199197143)

[7.1 Расчет трудоемкости разработки программного обеспечения 59](#_Toc199197144)

[7.2 Затраты на оплату 60](#_Toc199197145)

[7.3 Затраты на амортизацию оборудования 61](#_Toc199197146)

[7.4 Расчет затрат на электроэнергию 63](#_Toc199197147)

[7.5 Затраты на материалы, израсходованные при проведении разработки и прочие 64](#_Toc199197148)

[7.7 Расчет цены 65](#_Toc199197149)

[ЗАКЛЮЧЕНИЕ 67](#_Toc199197150)

[Список используемых источников 68](#_Toc199197151)

[Приложение А Техническое задание 70](#_Toc199197152)

[Приложение Б код клиентской части 76](#_Toc199197153)

[Приложение В код серверной части 108](#_Toc199197154)

ВВЕДЕНИЕ

Автосалон – это коммерческая организация, занимающаяся продажей новых и поддержанных автомобилей. В современном мире, особенно в условиях роста городской инфраструктуры и увеличения числа владельцев транспортных средств, спрос на услуги автосалонов продолжает расти. Это связано с изменением потребительских привычек, развитием цифровых технологий и стремлением к удобству при совершении крупных покупок. Многие потенциальные покупатели всё чаще предпочитают заранее ознакомиться с ассортиментом через интернет, чтобы сэкономить время и сделать более осознанный выбор.

Для бизнеса продажа автомобилей — это не только вопрос получения прибыли, но и важный элемент в формировании имиджа компании, уровня доверия клиентов и конкурентоспособности на рынке. Удобство выбора авто, наличие подробной информации и возможность оформления заявки онлайн положительно влияют на лояльность покупателей и повышают эффективность работы автосалона.

Спектр возможностей, предоставляемых автосалонами, постоянно расширяется и адаптируется под запросы клиентов. Среди наиболее востребованных функций можно выделить: просмотр каталога автомобилей с детальными характеристиками, фильтрация по параметрам (цена, марка, модель, пробег), возможность оставить заявку на интересующую машину. Такие решения делают процесс покупки автомобиля максимально простым и доступным как для частных лиц, так и для организаций.

В условиях высокой конкуренции среди автосалонов автоматизация процессов становится важным инструментом для повышения эффективности работы и улучшения клиентского сервиса. Использование веб-приложений позволяет оперативно обновлять информацию о наличии автомобилей, упрощает взаимодействие с клиентами, сокращает время на обработку заявок и повышает уровень удовлетворённости пользователей. Это способствует не только привлечению новых клиентов, но и укреплению доверия со стороны постоянных покупателей, что является залогом успешного развития бизнеса.

Таким образом, автосалоны играют важную роль в автомобильной индустрии, предлагая комплексное решение для подбора и покупки автомобиля. Они экономят время клиентов, предоставляют полную информацию о транспортных средствах и обеспечивают высокое качество обслуживания, что делает их незаменимыми партнерами при приобретении личного авто.

Целью дипломного проектирования является разработка веб-приложения автосалона, которое позволит автоматизировать процессы отображения и выбора автомобилей, а также приема заявок от клиентов.

Основными задачами являются:

− произвести исследование деятельности автосалона;

− определить инструментальные средства проектирования и реализации;

− разработать техническое задание для веб-приложения;

− спроектировать базу данных;

− спроектировать интерфейс пользователя веб-приложения;

− реализовать веб-приложение;

− провести тестирование веб-приложения;

− составить программную документацию.

1. Предпроектное исследование
   1. Описание предметной области

В качестве предметной области была выбрана автоматизация процессов в автосалоне, специализирующемся на продаже новых и поддержанных автомобилей. Основной задачей разработки данного веб-приложения является оптимизация процессов взаимодействия между клиентами, менеджерами и администратором, упрощение процедуры выбора и бронирования автомобиля, а также обеспечение удобного и понятного интерфейса для всех участников процесса.

Веб-приложение должно автоматизировать ключевые этапы работы автосалона: от просмотра каталога автомобилей до оформления заявки на покупку. Это позволит сократить время на обработку запросов, повысить прозрачность информации и улучшить качество обслуживания клиентов.

Работа администратора заключается в управлении данными о автомобилях (добавление, редактирование, удаление), а также в управлении учетными записями менеджеров и клиентов.

Менеджер может просматривать список автомобилей, информацию по каждому из них, а также список оформленных клиентами заявок и бронирований. Он отвечает за редактирование статуса авто и статуса заявки клиента.

Клиент имеет возможность зарегистрироваться, просмотреть каталог автомобилей, выбрать интересующий вариант, оставить заявку на покупку автомобиля, а также просмотреть историю своих заявок и бронирований.

Для эффективной работы веб-приложения будет разработан интуитивно понятный интерфейс как для клиентов, так и для сотрудников автосалона.

В соответствии с предметной областью можно выделить следующие объекты:

− Автомобиль. Атрибуты: код автомобиля, vin номер, марка, модель, поколение, комплектация, год выпуска, пробег, стоимость, статус (в наличии/продан), описание, фото, цвет, тип кузова, тип двигателя, название двигателя, объем двигателя, мощность, тип трансмиссии, название трансмиссии, привод, страна производства, вес, грузоподъемность, число мест, расход топлива, объем бака, емкость батареи, запас хода, максимальная скорость, клиренс, трехмерная модель, автосалон.

− Клиент. Атрибуты: код клиента, имя, контактные данные, история бронирований.

− Менеджер. Атрибуты: код менеджера, имя, контактные данные, список бронирований клиентов.

− Бронирование. Атрибуты: код бронирования, код клиента, код автомобиля, дата визита в автосалон, дата бронирования, комментарий менеджера, статус (ожидание/подтвержденное/отменено/выполненное).

Веб-приложение создается для обслуживания следующих категорий пользователей:

− Администратор — управляет автомобилями, менеджерами и филиалами.

− Менеджер — работает с информацией о бронированиях автомобилей, обрабатывает бронирования.

− Клиент — регистрируется, просматривает каталог автомобилей, оставляет заявки, отслеживает статусы.

Таким образом, веб-приложение для автосалона должно стать надежным инструментом для повышения эффективности работы, улучшения качества обслуживания клиентов и увеличения скорости продаж автомобилей.

1.2 Обзор инструментов, используемых в проектировании и разработке программного обеспечения

Для проектирования и разработки веб-приложения были выбраны современные инструменты и технологии, которые обеспечивают эффективную реализацию всех этапов проекта.

Draw.io — это удобный онлайн-инструмент для построения диаграмм. Он предоставляет широкий набор возможностей для визуального представления различных процессов и структур, что особенно важно на этапе проектирования архитектуры приложения.

Figma — бесплатная платформа для создания дизайна и прототипирования пользовательских интерфейсов. С её помощью можно разработать наглядный и функциональный макет веб-приложения ещё до начала программирования.

MySQL Workbench — мощный инструмент для работы с базами данных. Он позволяет не только управлять данными, но и создавать ER-диаграммы, отражающие структуру таблиц и связи между ними. Этот инструмент будет использован в рамках дипломного проекта для проектирования базы данных автосалона.

Для реализации клиентской части веб-приложения выбраны следующие технологии:

HTML — язык разметки, используемый для построения структуры веб-страниц. Он определяет расположение элементов на странице, но не влияет на их внешний вид.

Tailwind CSS — популярный фреймворк для стилизации веб-страниц. Он позволяет быстро и гибко создавать современный дизайн, применяя классы непосредственно в HTML.

JavaScript — язык программирования, применяемый для добавления интерактивности к веб-страницам. Он обеспечивает динамическое обновление контента, обработку событий и взаимодействие с сервером.

Создание надёжной серверной части является важной частью разработки веб-приложения, поскольку она отвечает за обработку запросов, работу с базой данных и безопасность системы.

PHP — один из самых популярных языков программирования для создания серверной логики. Он хорошо подходит для работы с базами данных и позволяет строить масштабируемые веб-приложения.

Для ускорения и упрощения разработки используется Laravel — современный PHP-фреймворк, предоставляющий готовые решения для маршрутизации, аутентификации, управления сессиями и других задач. Использование Laravel повышает качество кода и снижает риск возникновения ошибок.

Для хранения и управления данными выбрана система MySQL — реляционная СУБД, в которой данные организованы в виде таблиц. Она отличается высокой производительностью, хорошей документацией и активным сообществом разработчиков.

В качестве среды разработки используется Visual Studio Code (VS Code) — мощный и гибкий редактор кода, поддерживающий множество расширений. Он позволяет удобно работать с файлами, настраивать автоматизацию и писать чистый, структурированный код.

Для контроля версий и совместной работы над проектом используются:

Git — распределённая система контроля версий, которая помогает отслеживать изменения в коде, создавать отдельные ветки для разных задач и восстанавливать предыдущие версии проекта при необходимости. Git обеспечивает стабильное управление кодовой базой при командной разработке.

GitLab — веб-платформа, позволяющая размещать и управлять репозиториями на основе Git. Платформа предоставляет дополнительные возможности для автоматизации тестирования и развертывания приложений, а также систему управления задачами. Это делает рабочий процесс более прозрачным и организованным.

Выбранные инструменты и технологии обеспечивают надежную реализацию, простоту сопровождения и защиту кодовой базы веб-приложения на всех этапах его жизненного цикла.

1. Техническое задание

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

Для создания технического задания использовался стандарт ГОСТ 34.602-2020.

Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

1. введение;
2. основания для разработки;
3. назначение системы;
4. требования к системе;
5. требования к техническому обеспечению;
6. требования к программному обеспечению;
7. организационно-технические требования.

Техническое задание на разработку веб-приложения «Автосалон» представлено в приложении А.

1. Проектирование программного продукта
   1. Архитектура программного продукта

В веб-приложении используется клиент-серверная архитектура. Данная архитектура делится на две основные части: клиентскую и серверную.

Клиентская часть представляет собой интерфейс, с помощью которого пользователь взаимодействует с системой, пользователь отправляет HTTP-запросы к серверу и получает ответ.

Серверная часть обрабатывает запросы, взаимодействует с базой данных и возвращает ответ.

Основные принципы:

* Разделение обязанностей: клиент отвечает за интерфейс и отправку запросов, сервер – за обработку и хранение данных.
* Сетевое взаимодействие: обмен данными происходит по стандартным протоколам (HTTP, TCP/IP, WebSocket).
* Централизованное управление: сервер часто выступает единой точкой контроля данных и бизнес-логики.

Так же используется архитектурный паттерн MVC (Model-View-Controller), который разделяет веб-приложение на три основных компонента:

* Model (Модель) отвечает за работу с базой данных.
* View (Представление) отображает данные пользователю.
* Controller (Контроллер) является посредником между моделью и представлением. Обрабатывает входящие запросы, взаимодействует с моделью для получения данных и передает их в представление.

Особенности клиент-серверной архитектуры:

* Сервер может быть легко масштабирован для обработки большого количества запросов.
* Клиентская часть может быть оптимизирована для различных устройств.
  1. Функциональное и структурное проектирование

Диаграмма прецедентов – диаграмма, показывающая связи между действиями и лицами, выполняющими их.

В веб-приложении выделяются шесть ролей: администратор, пользователь, менеджер.

На основе ролей, была создана диаграмма прецедентов, показывающая все необходимые функции:

1. Пользователь: регистрация и авторизация в системе, возможность осуществлять поиск, оформление бронирования на покупку.
2. Администратор: добавление новых менеджеров, управление автомобилями.
3. Менеджер: рассмотрение и редактирование бронирования клиента.

Как итог, данная диаграмма прецедентов (рисунок 1) должна показывать основные действия пользователей в информационной системе и их взаимодействие между собой.

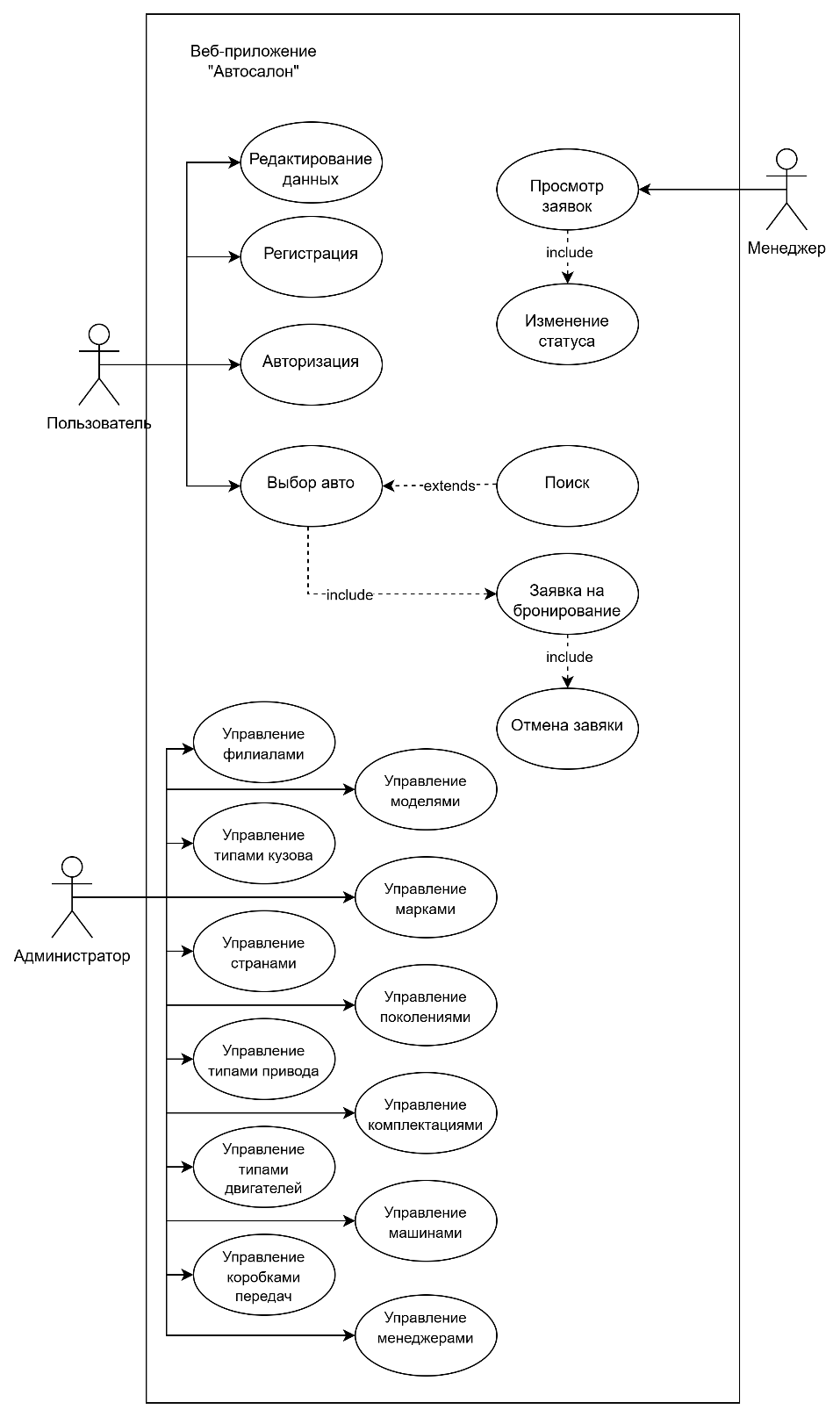


Рисунок 1 – Диаграмма прецедентов

Диаграмма деятельности представляет собой блок-схему, которая показывает, как поток управления переходит от одной деятельности к другой. На рисунке 2 изображено какой путь проходит пользователь, оформляя заявку на бронирование.

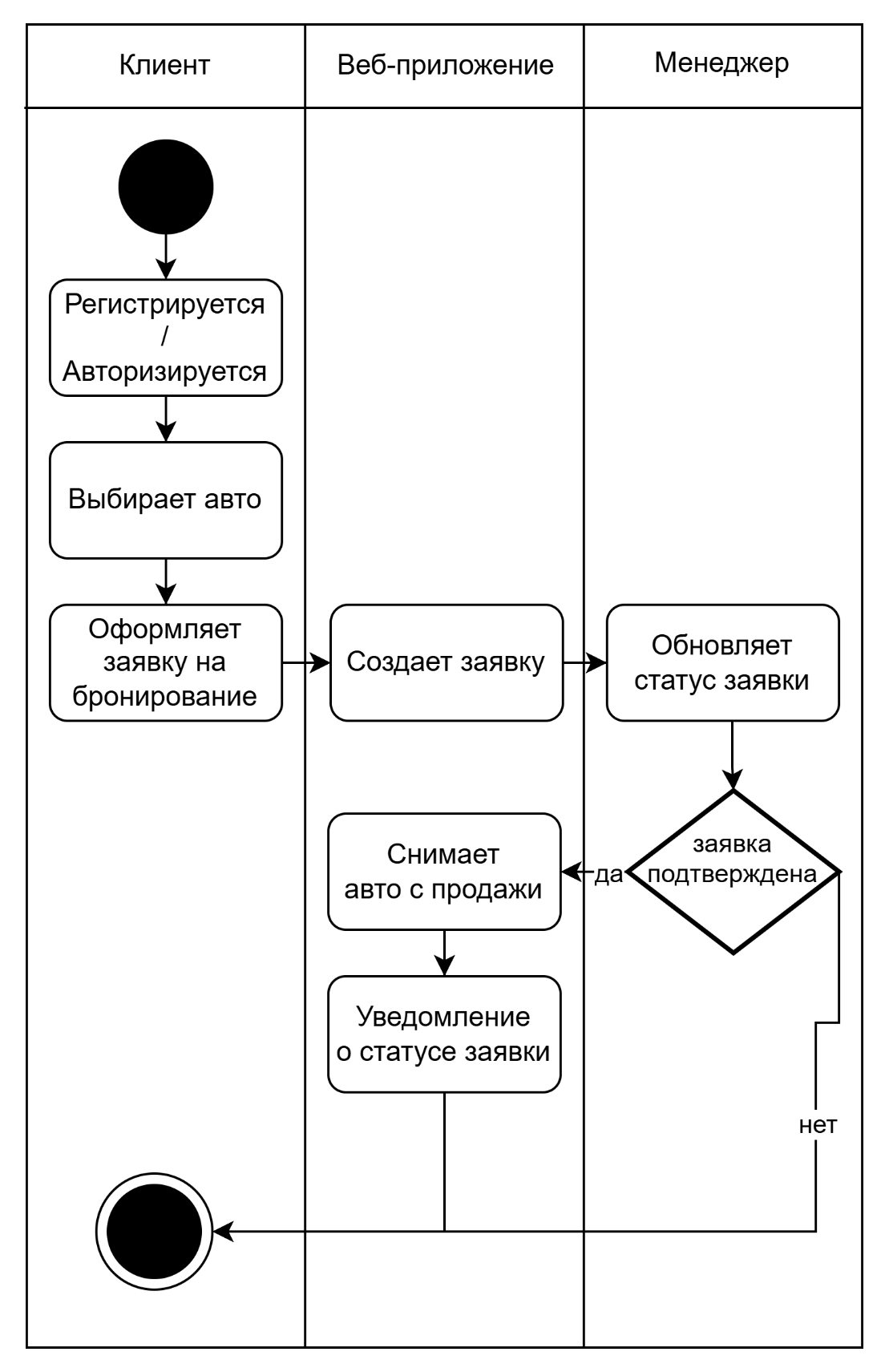


Рисунок 2 – Диаграмма деятельности

Диаграмма представляет блок-схему деятельности, демонстрируя последовательность действий от регистрации клиента до выполнения заказа и оставления отзыва.

* 1. Проектирование базы данных

База данных – это структурированный набор упорядоченной информации, которая храниться в связанных электронных таблицах.

ER-модель – это концептуальная модель взаимосвязей объектов с набором атрибутов, позволяющая визуализировать то, как объекты связаны. ER-модель веб-приложения представлена на рисунке 3.

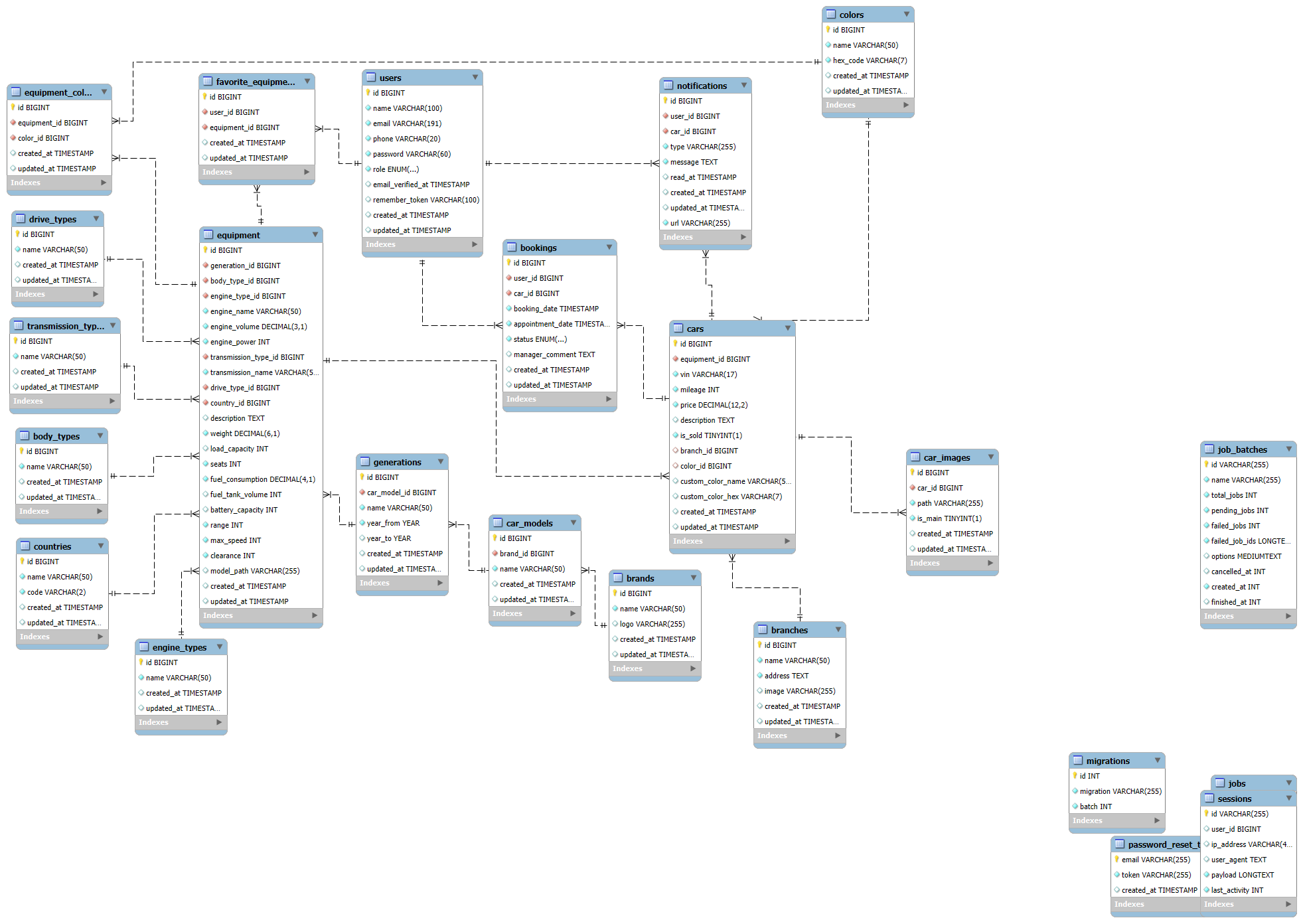


Рисунок 3 – ER-модель веб-приложения

Нормальные формы – это набор правил, которые помогают улучшить структуру базы данных

Все таблицы представлены в третьей нормальной форме (3НФ), что означает, что каждая таблица удовлетворяет условиям 1НФ и 2НФ, а также не имеет транзитивных зависимостей. Каждый столбец содержит атомарное значение, не ключевые атрибуты зависят от каждого потенциального ключа, и нет транзитивных зависимостей от первичного ключа.

Далее в таблицах 1-18 представлены таблицы ER модели, их поля, типы данных и описание.

Таблица 1 – Users (пользователи)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор пользователя |
| name | VARCHAR(100) | Имя пользователя |
| email | VARCHAR(191) UNIQUE | Адрес электронной почты |
| phone | VARCHAR(20) UNIQUE | Номер телефона |
| password | VARCHAR(60) | Хэшированный пароль |
| role | ENUM('admin', 'manager', 'user') DEFAULT 'user' | Роль пользователя |
| email\_verified\_at | TIMESTAMP NULLABLE | Дата и время подтверждения email |
| remember\_token | VARCHAR(100) NULLABLE | Токен для "Remember me" |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 2 – Brands (бренды)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| name | VARCHAR(50) | Название бренда |
| logo | VARCHAR(255) NULLABLE | Путь к изображению логотипа |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 3 – Countries (страны)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| name | VARCHAR(50) | Название страны |
| code | VARCHAR(2) | Код страны |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 4 – Body\_types (типы кузова)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| name | VARCHAR(50) | Название типа кузова |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 5 – Car\_models (модели авто)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| brand\_id (FK) | FOREIGN KEY → brands.id | Внешний ключ бренда |
| name | VARCHAR(50) | Название модели автомобиля |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 6 – Generations (поколения)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| car\_model\_id (FK) | FOREIGN KEY → car\_models.id | Внешний ключ модели автомобиля |

Продолжение таблицы 6

|  |  |  |
| --- | --- | --- |
| name | VARCHAR(50) | Название поколения |
| year\_from | YEAR | Год начала выпуска |
| year\_to | YEAR NULLABLE | Год окончания выпуска |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 7 – Drive\_types (типы привода)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| name | VARCHAR(50) | Название привода |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 8 – Engine\_types (типы двигателей)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| name | VARCHAR(50) | Название типа двигателя |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 9 – Transmission\_types (типы трансмиссий)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| name | VARCHAR(50) | Название трансмиссии |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 10 – Colors (цвета)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| name | VARCHAR(50) | Название цвета |
| hex\_code | VARCHAR(7) | HEX-код цвета |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 11 – Equipment (комплектации)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| generation\_id (FK) | FOREIGN KEY → generations.id | Внешний ключ поколения |
| body\_type\_id (FK) | FOREIGN KEY → body\_types.id | Внешний ключ типа кузова |
| engine\_type\_id (FK) | FOREIGN KEY → engine\_types.id | Внешний ключ типа двигателя |
| engine\_name | VARCHAR(50) | Название двигателя |
| engine\_volume | DECIMAL(3,1) | Объем двигателя (литры) |
| engine\_power | INTEGER | Мощность двигателя (л.с.) |
| transmission\_type\_id (FK) | FOREIGN KEY → transmission\_types.id | Внешний ключ типа трансмиссии |
| transmission\_name | VARCHAR(50) | Название трансмиссии |
| drive\_type\_id (FK) | FOREIGN KEY → drive\_types.id | Внешний ключ типа привода |
| country\_id (FK) | FOREIGN KEY → countries.id | Внешний ключ страны производства |
| description | TEXT NULLABLE | Описание комплектации |
| weight | DECIMAL(6,1) | Вес автомобиля (кг) |

Продолжение таблицы 11

|  |  |  |
| --- | --- | --- |
| load\_capacity | INTEGER NULLABLE | Грузоподъемность (кг) |
| seats | INTEGER | Количество сидений |
| fuel\_consumption | DECIMAL(4,1) | Расход топлива (л/100 км) |
| fuel\_tank\_volume | INTEGER NULLABLE | Объем топливного бака (л) |
| battery\_capacity | INTEGER NULLABLE | Ёмкость аккумулятора (А·ч или кВт·ч) |
| range | INTEGER | Запас хода (км) |
| max\_speed | INTEGER | Максимальная скорость (км/ч) |
| clearance | INTEGER | Дорожный просвет (мм) |
| model\_path | VARCHAR(255) NULLABLE | Путь для 3d модели |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 12 – Equipments\_colors (цвета комплектаций)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| equipment\_id (FK) | FOREIGN KEY → equipment.id | Внешний ключ комплектации |
| color\_id (FK) | FOREIGN KEY → colors.id | Внешний ключ цвета |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 13 – Branches (Филиалы)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор филиала |

Продолжение таблицы 13

|  |  |  |
| --- | --- | --- |
| name | VARCHAR(50) | Название филиала |
| address | TEXT | Адрес филиала |
| image | VARCHAR(255) NULLABLE | Путь к изображению филиала |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 14 – Cars (Авто)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор автомобиля |
| equipment\_id (FK) | FOREIGN KEY → equipment.id | Внешний ключ комплектации |
| vin | VARCHAR(17) UNIQUE | VIN-номер автомобиля |
| mileage | INTEGER | Пробег (в км) |
| price | DECIMAL(12, 2) | Цена автомобиля |
| description | TEXT NULLABLE | Описание автомобиля |
| is\_sold | BOOLEAN DEFAULT false | Статус продажи авто |
| branch\_id (FK) | FOREIGN KEY → branches.id NULLABLE | Внешний ключ филиала (может быть NULL) |
| color\_id (FK) | FOREIGN KEY → colors.id NULLABLE | Внешний ключ цвета автомобиля (может быть NULL) |
| custom\_color\_name | VARCHAR(50) NULLABLE | Название пользовательского цвета |
| custom\_color\_hex | VARCHAR(7) NULLABLE | HEX-код пользовательского цвета |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 15 – Car\_images (изображения авто)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор изображения |
| car\_id (FK) | FOREIGN KEY → cars.id | Внешний ключ автомобиля |
| path | VARCHAR(255) | Путь к файлу изображения |
| is\_main | BOOLEAN DEFAULT false | Статус основного изображения |
| created\_at | TIMESTAMP | Дата добавления изображения |
| updated\_at | TIMESTAMP | Дата последнего обновления |

Таблица 16 – Bookings (бронирования)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор брони |
| user\_id (FK) | FOREIGN KEY → users.id | Внешний ключ пользователя |
| car\_id (FK) | FOREIGN KEY → cars.id | Внешний ключ авто |
| booking\_date | TIMESTAMP DEFAULT CURRENT\_TIMESTAMP | Дата бронирования |
| appointment\_date | TIMESTAMP | Дата визита в автосалон |
| status | ENUM('pending', 'confirmed', 'rejected', 'completed') | Статус бронирования |
| manager\_comment | TEXT NULLABLE | Комментарий менеджера |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |
| INDEX | INDEX(car\_id, status) | Для оптимизации поиска по статусам и авто |

Таблица 17 – Favorite\_equipments (избранные комплектации)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор |
| user\_id (FK) | FOREIGN KEY → users.id | Внешний ключ пользователя |
| equipment\_id (FK) | FOREIGN KEY → equipment.id | Внешний ключ комплектации |
| created\_at | TIMESTAMP | Дата добавления |
| updated\_at | TIMESTAMP | Дата последнего обновления |
| UNIQUE | UNIQUE(user\_id, equipment\_id) | Одна пара пользователь/комплектация может быть только одна |

Таблица 18 – Notifications (уведомления)

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id (PK) | BIGINT UNSIGNED AUTO\_INCREMENT | Уникальный идентификатор уведомления |
| user\_id (FK) | FOREIGN KEY → users.id | Кому отправлено уведомление |
| car\_id (FK) | FOREIGN KEY → cars.id | С каким автомобилем связано уведомление |
| type | VARCHAR | Тип уведомления |
| message | TEXT | Текст уведомления |
| read\_at | TIMESTAMP NULLABLE | Время прочтения уведомления |
| created\_at | TIMESTAMP | Дата создания записи |
| updated\_at | TIMESTAMP | Дата последнего обновления |
| url | VARCHAR | Ссылка для перехода |

* 1. Проектирование пользовательского интерфейса
     1. Разработка прототипов интерфейса

Для того что бы наглядно показать, как страницы связанны между собой на рисунке 4 представлена схема навигации.

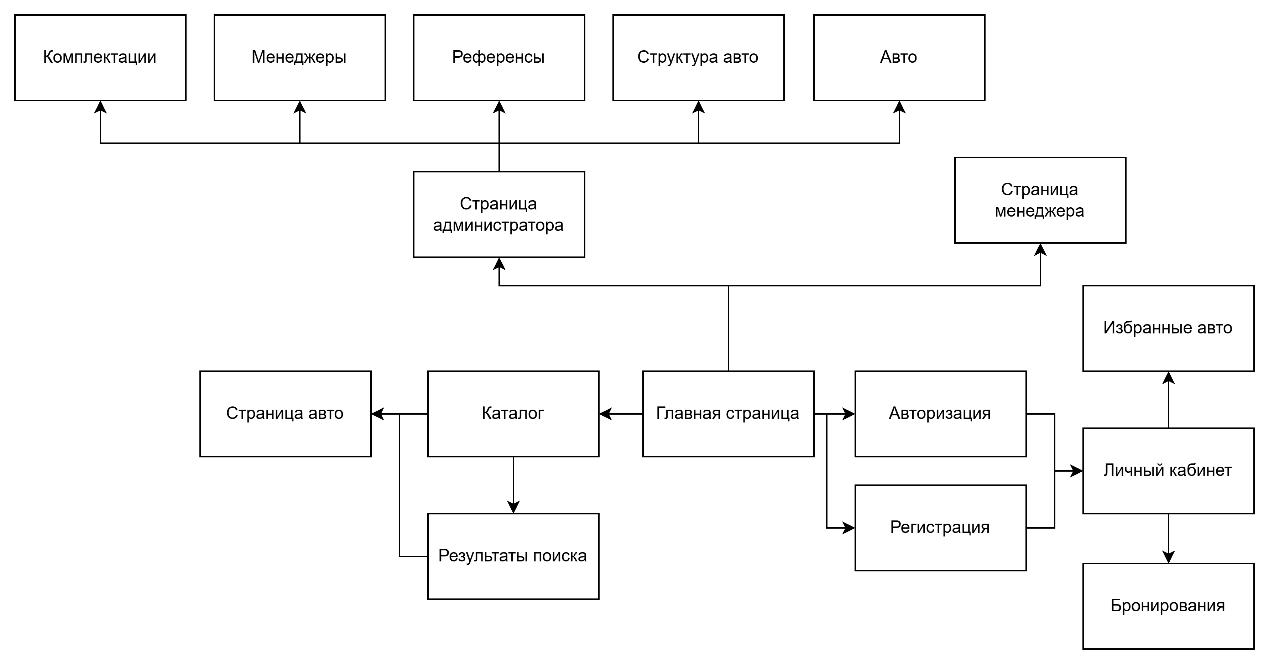


Рисунок 4 – Схема навигации

Для разработки пользовательского интерфейса используется онлайн-сервис «Figma». На рисунках 5 – 9 представлены прототипы страниц веб-приложения автосалона.

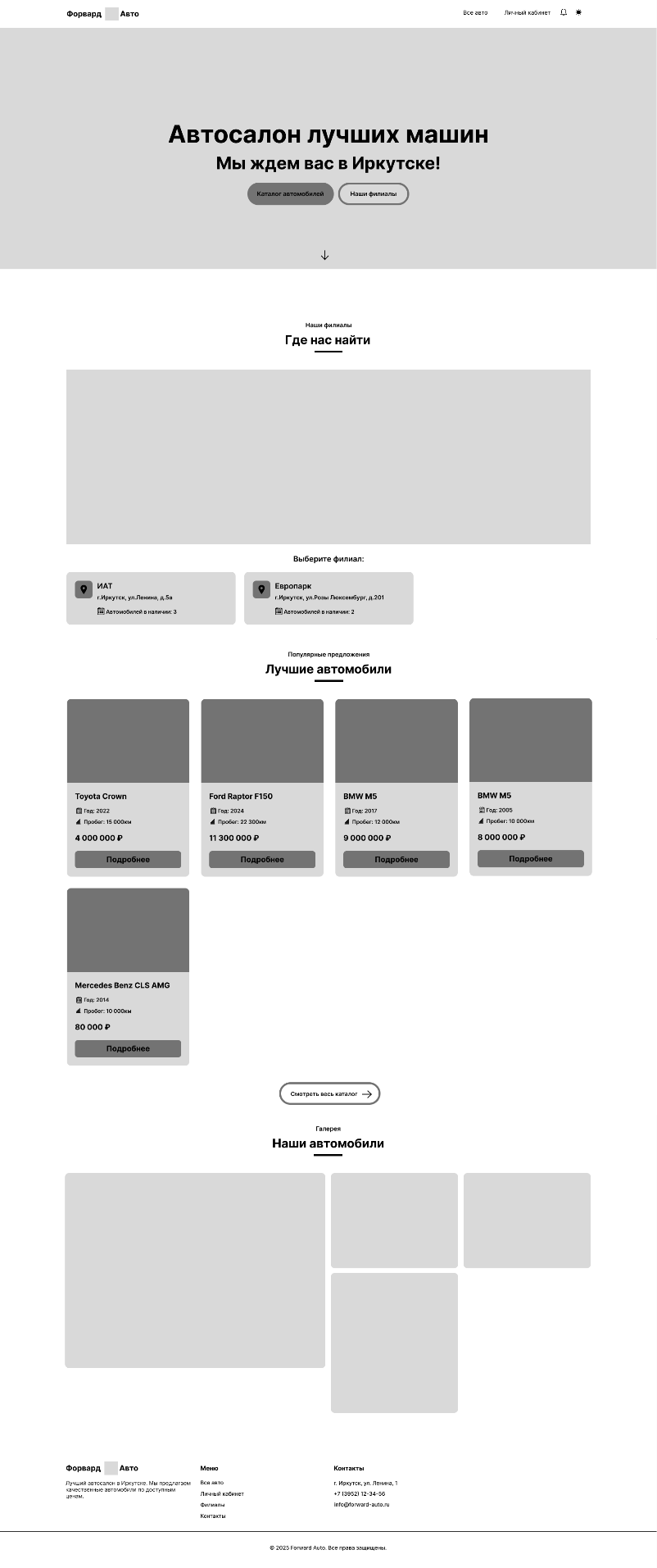


Рисунок 5 – Прототип главной страницы

На главной странице пользователь может, найти необходимую услугу, перейти на страницу каталога со всеми авто, посмотреть адреса филиалов на карте, и зайти в личный кабинет.

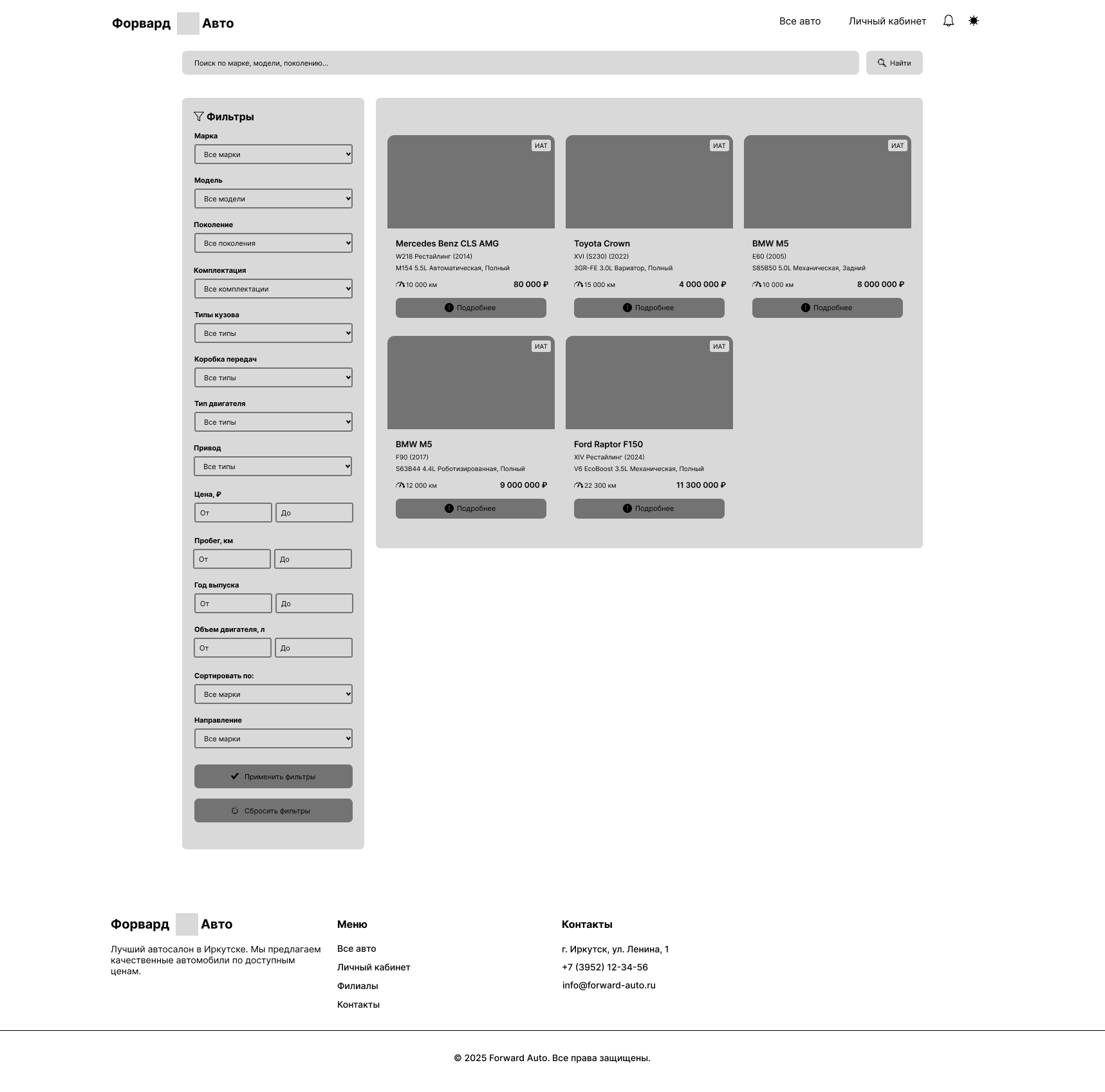


Рисунок 6 – Прототип страницы каталога

На данной странице пользователь может посмотреть какие авто сейчас находятся в продаже, найти интересующий авто и отсортировать по параметрам.

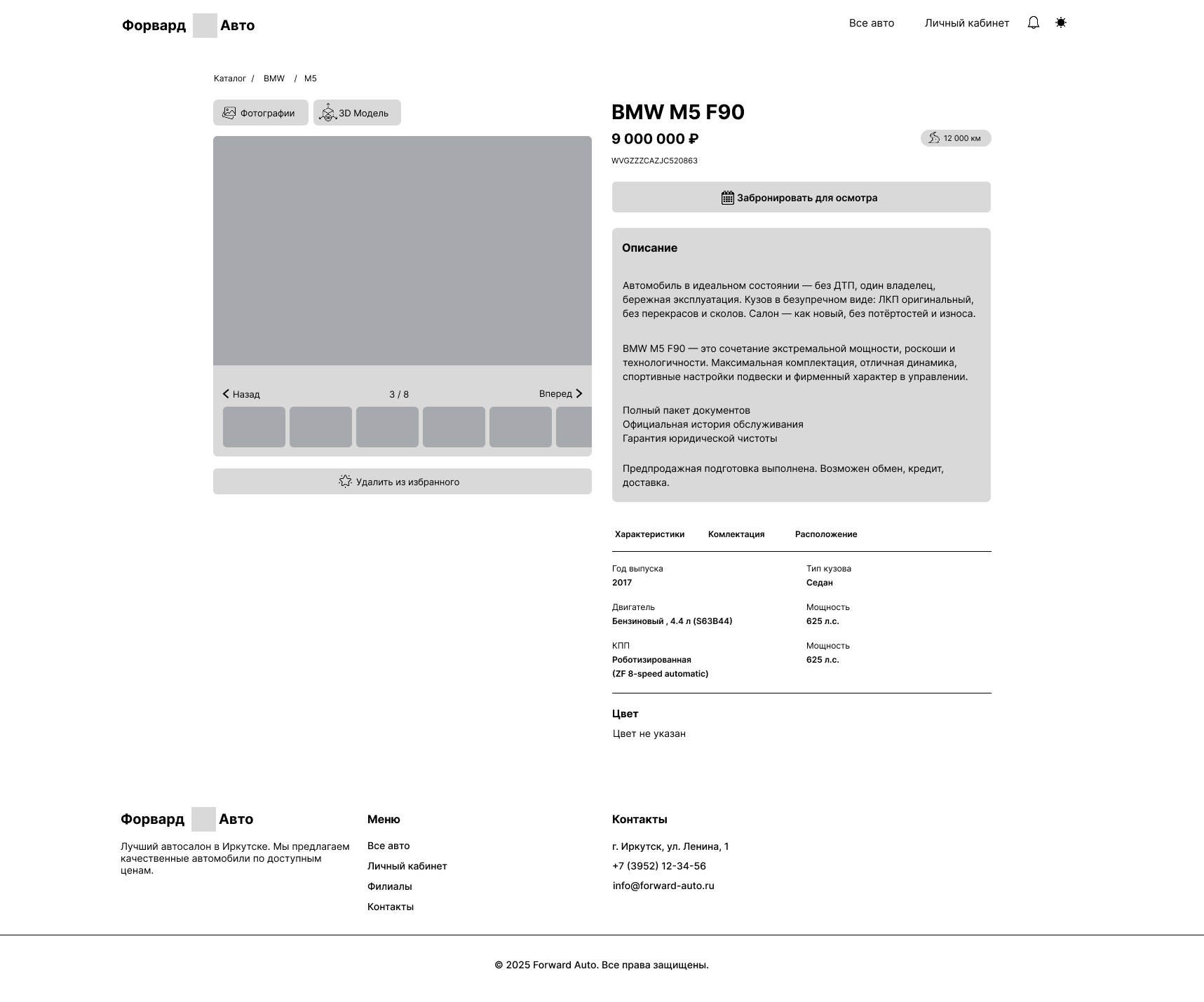


Рисунок 7 – Прототип страницы автомобиля

На представленной странице пользователь может ознакомиться с продаваемым автомобилем, посмотреть характеристики и описание, а также оформить заявку на покупку автомобиля и добавить авто в избранное.

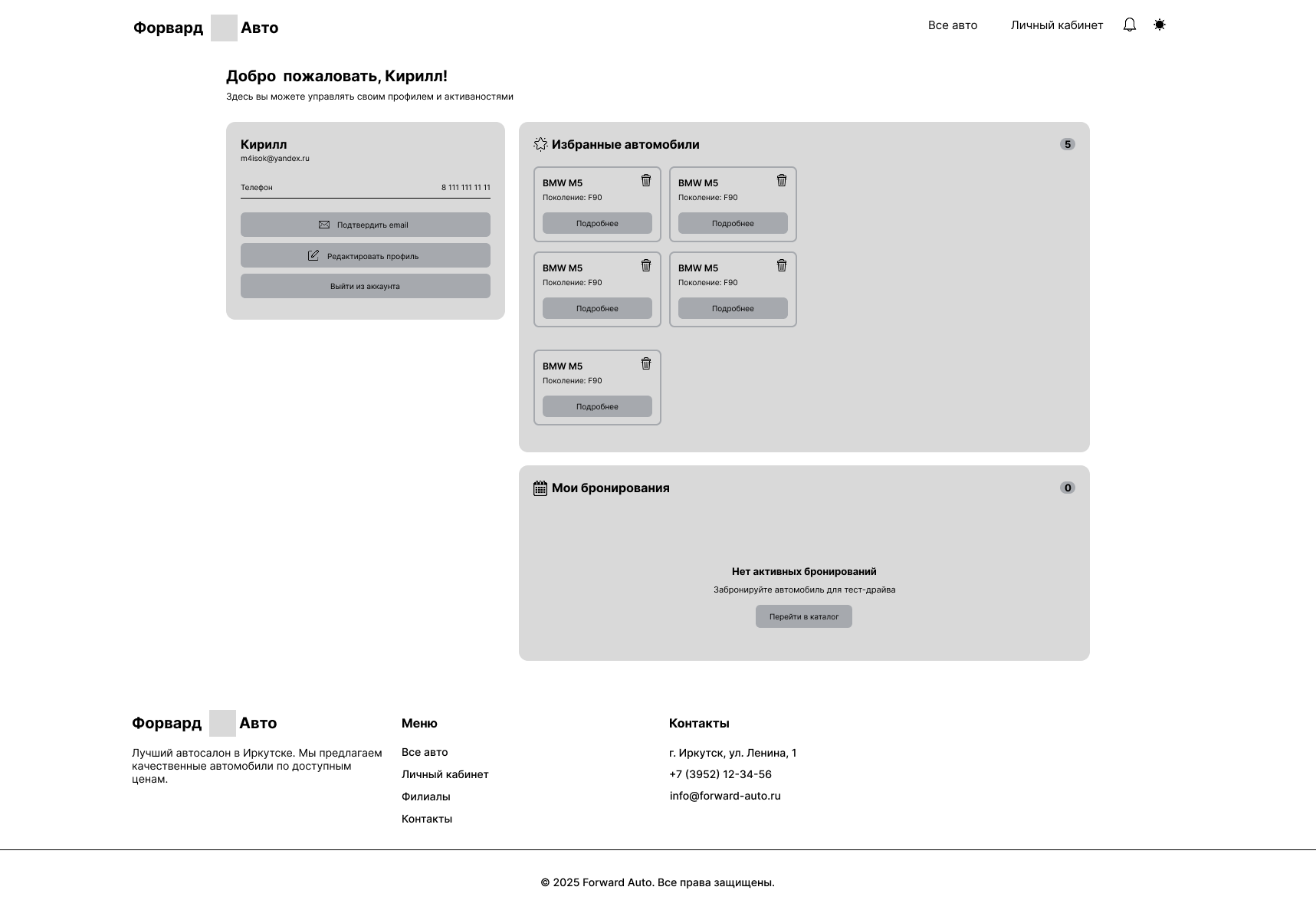


Рисунок 8 – Прототип страницы личного кабинета

На данной странице пользователь может поменять личные данные, ознакомиться со списком бронирований и списком «избранных» автомобилей.

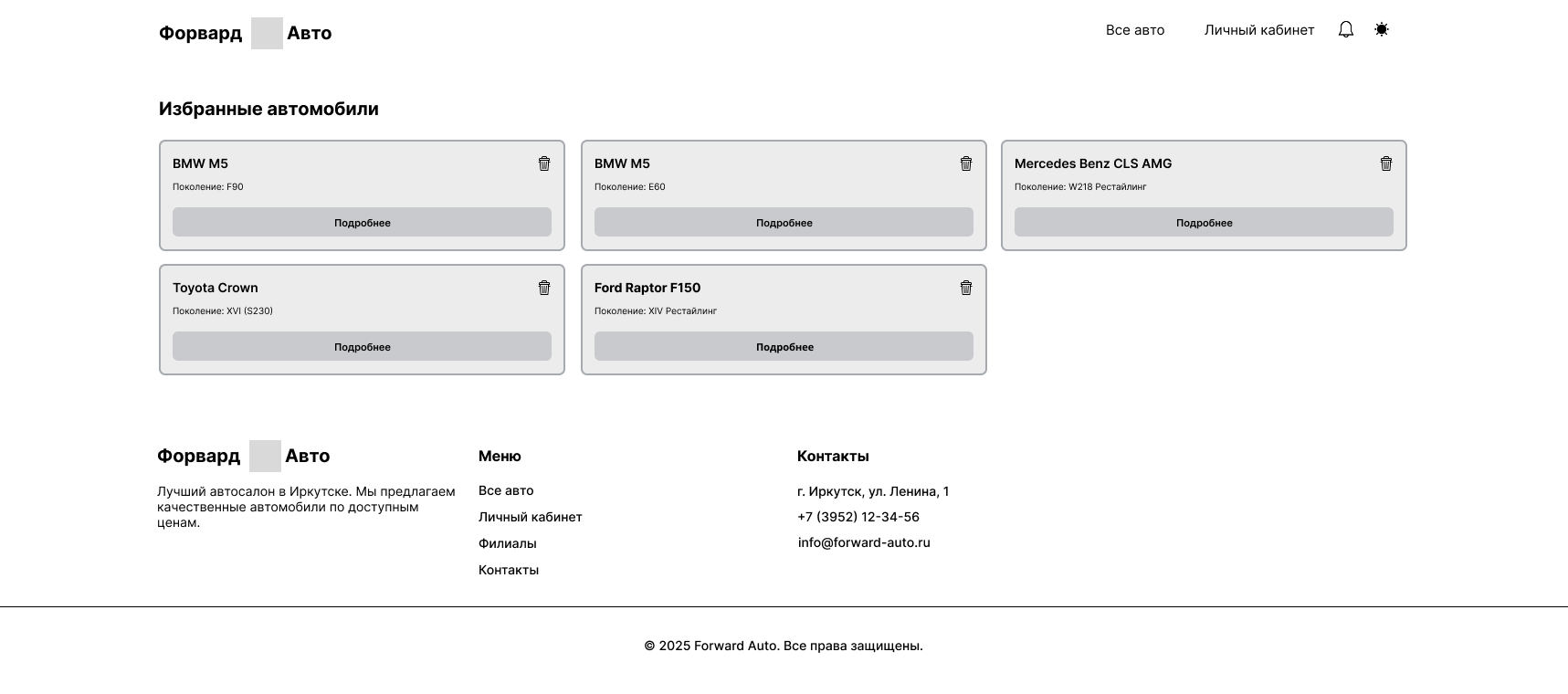


Рисунок 9 – Оформление заявки

На представленной странице пользователь может полностью посмотреть список «избранных» автомобилей, также может удалить авто из «избранного» или перейти на страницу продажи самого автомобиля.

* + 1. Выбор цветовой гаммы и шрифтов

Для создания единого стиля интерфейса веб-приложения были подобраны цвета и шрифты, обеспечивающие комфортный опыт взаимодействия.

Веб-приложение выполнено в сочетании фиолетово-белого цвета, для достижения единой цветовой гармонии (Рисунок 10).

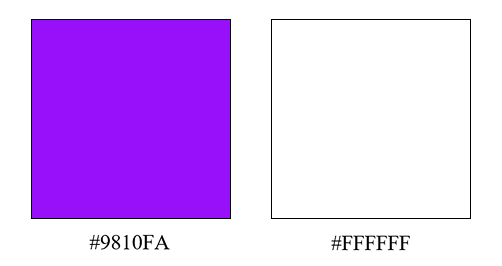


Рисунок 10 – Цветовая гамма

Основным шрифтом для информационной системы был выбран «Instrument Sans», который обеспечивает хорошую читаемость заголовков и основного текста.

* + 1. Разработка элементов интерфейса

Для наиболее удобного использования веб приложения клининговой компании были разработаны такие элементы как: поле ввода, кнопка, карточка услуги.

Поле ввода – это графический элемент, предназначенный для ввода текста. В веб-приложении оно имеет серый фон, с черной обводкой (рисунок 11).



Рисунок 11 – Поле ввода информационной системы

Кнопка – важный элемент, с помощью которого совершаются какое-либо действие. В информационной системе кнопка выполнена в фиолетовом цвете с белым текстом внутри (рисунок 12).



Рисунок 12 – Кнопка информационной системы

Карточка автомобиля – элемент интерфейса, благодаря которому, пользователь может кратко ознакомиться с основной информацией об автомобиле на странице каталога (рисунок 13).



Рисунок 13 – Карточка автомобиля

* + 1. Разработка дизайн макетов

После проектирования ранее упомянутых элементов была начата разработка дизайн-макета веб-приложения.

Далее на рисунках 13 – 17 представлены дизайн-макеты страниц

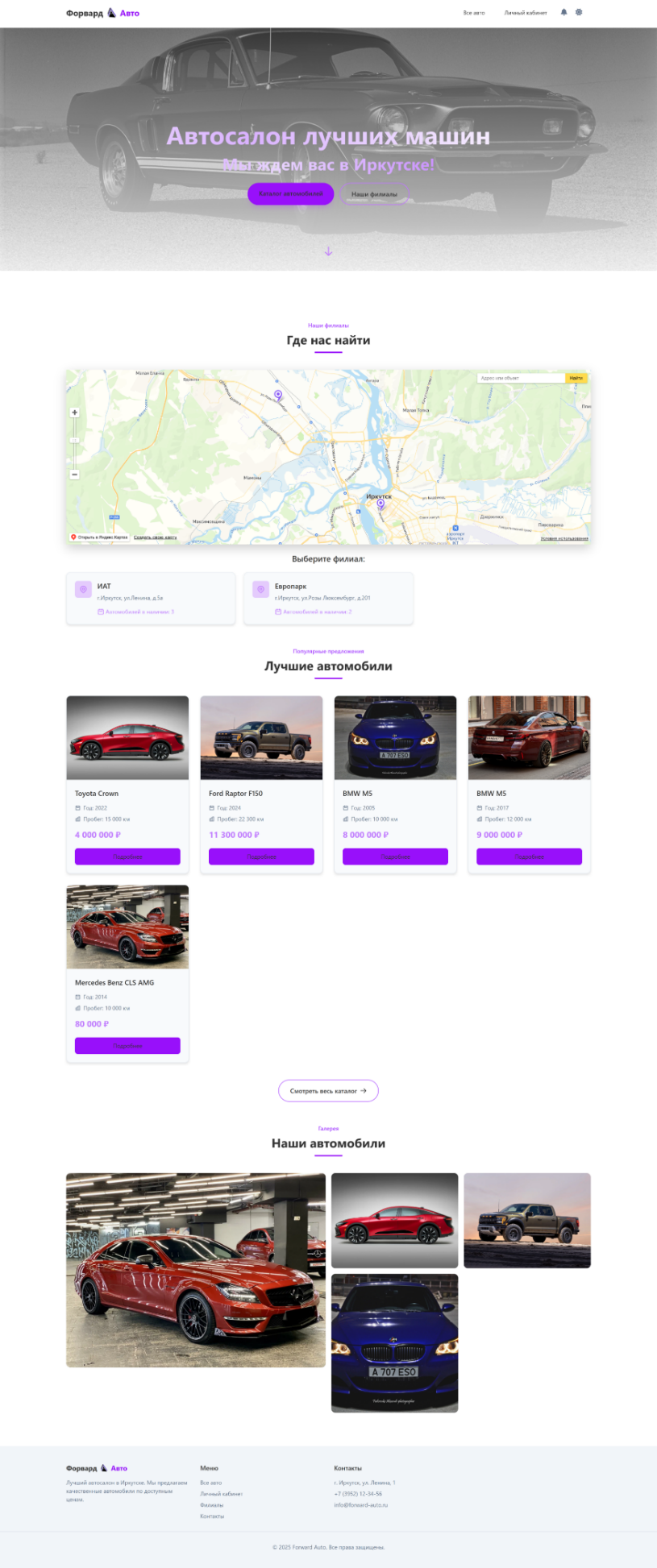


Рисунок 13 – Дизайн-макет главной страницы

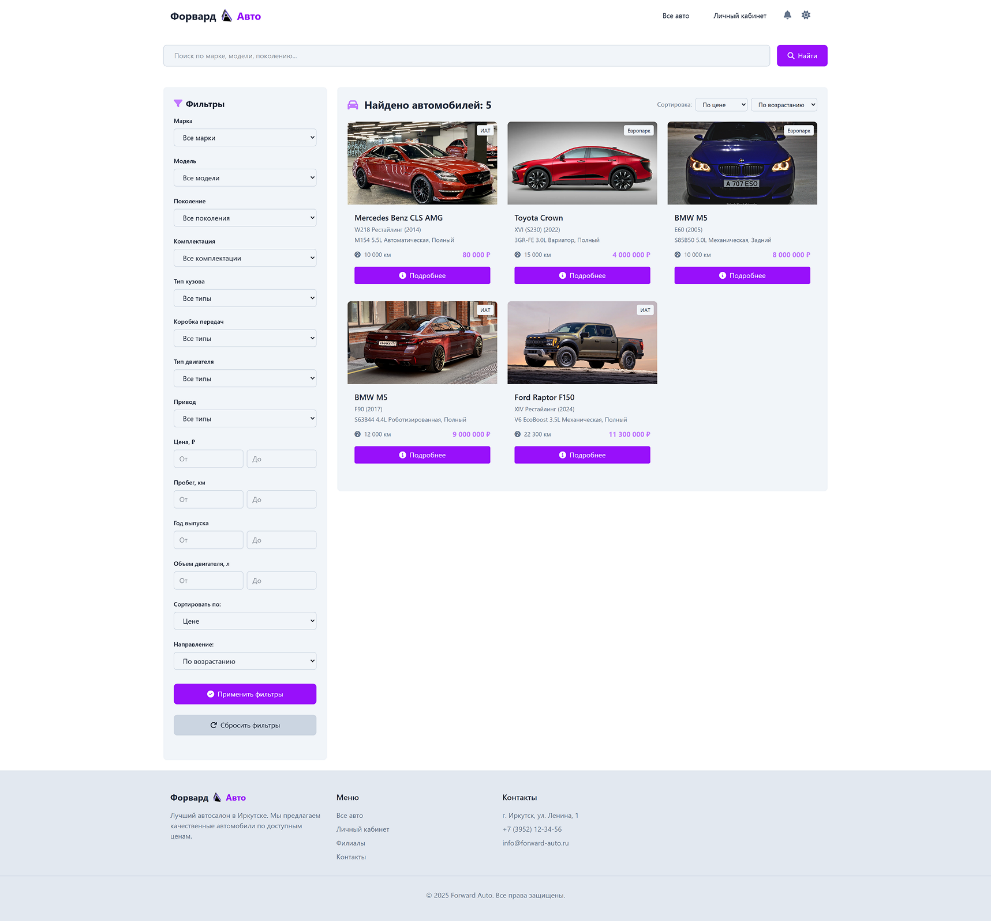


Рисунок 14 – Каталог автомобилей

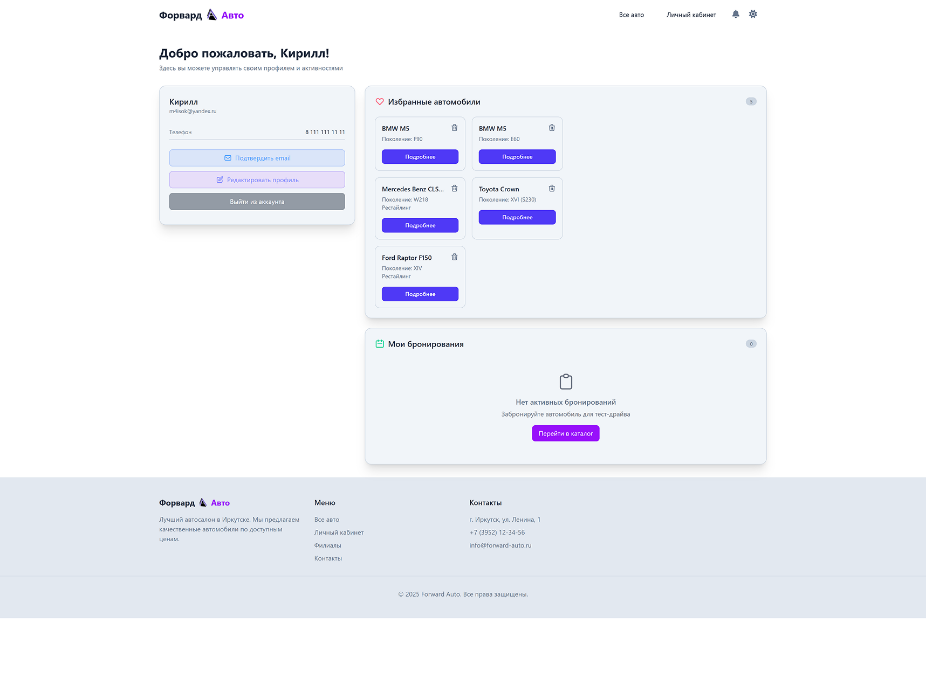


Рисунок 15 – Личный кабинет

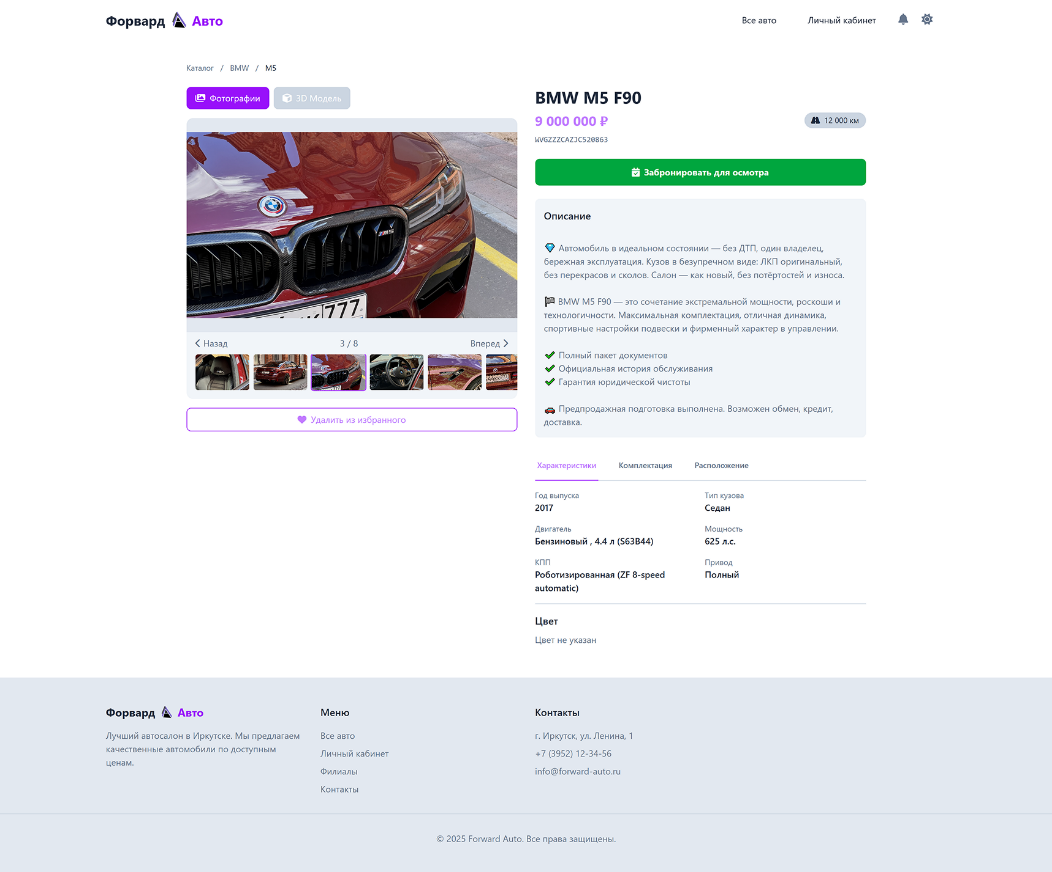


Рисунок 16 – Страница авто

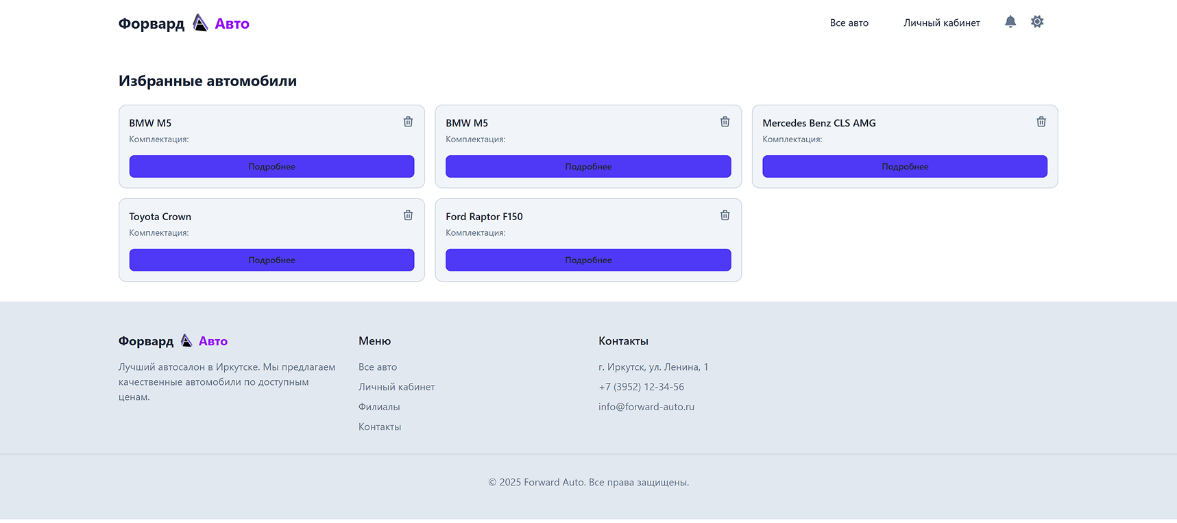


Рисунок 17 – Страница избранного

1. Реализация программного продукта

4.1 Разработка клиентской части

Разработка клиентской части веб-приложения является важным этапом, ведь это определяет пользовательский опыт и так же может влиять на производительность и безопасность. В дипломном проекте клиентская часть реализована с помощью фреймворка Tailwind. На рисунке 18 представлен код макет всех клиентских страниц (user.blade.php).



Рисунок 18 – Макет страниц

После файл макета страниц подключается на всех клиентских страницах, определяя базовую структуру всех HTML-страниц веб-приложения. На рисунке 19 представлен код страницы каталога с использованием макета.



Рисунок 19 – Главный шаблон

Далее шаблон страницы администратора использует структуру другого макета (admin.blade.php) с помощью директивы «@extends('layouts.admin)». Это позволяет разделять макетами страницы админа и клиента. Основной контент страницы админа определяется в секции «@section('content')», которая затем вставляется в соответствующую область макета (@yield('content')). Часть кода шаблона администратора представлено на рисунке 20. С полным кодом пользовательского макета можно ознакомиться в приложении Б.



Рисунок 20 – Часть страницы администратора

4.2 Разработка базы данных

База данных веб-приложения состоит из 18 таблиц. Таблицы были созданы с использованием миграций Laravel. Это упростило процесс управления базой данных в ходе разработки. Все файлы миграций представлены на рисунке 21.



Рисунок 21 – Все файлы миграций

На рисунке 22 в качестве примела приведен код одного из файлов – создание таблицы комплектаций.

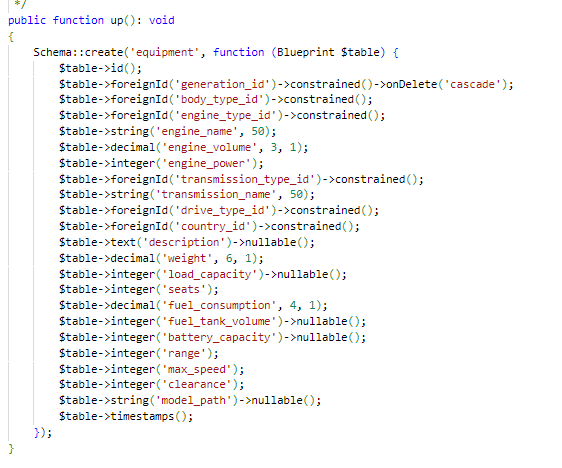


Рисунок 22 – Миграция таблицы услуг

4.3 Разработка серверной части

Первым и самым важным этапом в разработке серверной части веб-приложения, является создание моделей, которые позволяют упростить работу с запросами к базе данных. На рисунке 23 представлена часть кода модели комплектаций (Equipment).



Рисунок 23 – Модель жалоб

После создания необходимых моделей, необходимо создать контроллеры, которые будут обрабатывать все HTTP-запросы. Часть кода контроллера комплектации представлено на рисунке 24. Полный код контроллера EquipmentController представлен в приложении В.

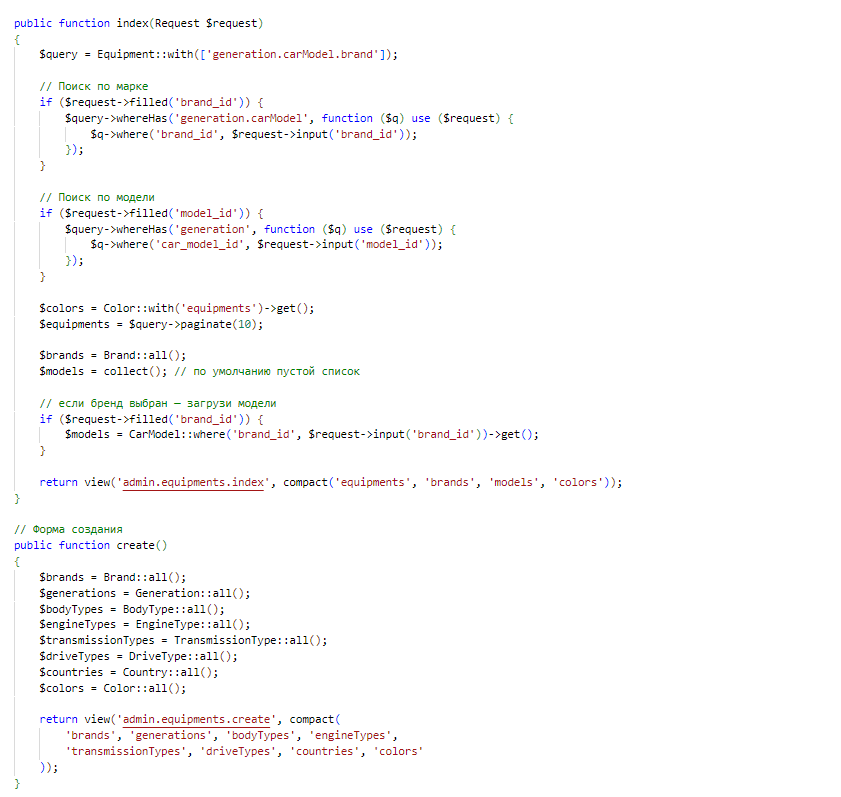


Рисунок 24 – Часть контроллера комплектаций

5 Отладка и тестирование программного продукта

Одним из этапов разработки любого программного продукта является тестирование и отладка.

Тестирование – это процесс выполнения программного продукта с целью выявления ошибок, несоответствий требованиям и оценки его качества. Главная цель тестирования веб-приложения – обеспечить надежность, безопасность и эффективность функционирования.

Отладка – это процесс поиска и устранения ошибок в программном коде. Отладка проводилась в процессе написания программного кода, методом пошагового выполнения кода.

Веб-приложение тестировалось с помощью PHPUnit, включая следующие методы:

* Интеграционное тестирование – проверка взаимодействия контроллеров, базы данных и HTTP-запросов.
* Функциональное тестирование – валидация (создание, обновление, удаление сотрудников).
* Тестирование безопасности – проверка роли пользователя.

Для тестирования веб-приложения был разработан сценарий тестирования для добавления комплектации. В таблице 12 представлен данный сценарий.

Таблица 19 – Сценарий тестирования добавления комплектации.

|  |  |
| --- | --- |
| Поле | Описание |
| Дата тестирования | 04.05.2025 |
| Приоритет тестирования | Высокий |
| Этапы теста | 1. Авторизация в системе с правами администратора 2. Создание комплектации с корректными данными 3. Попытка создания комплектации без прав |

Продолжение таблицы 19

|  |  |
| --- | --- |
|  | администратора 4. Проверка валидации обязательных полей 5. Проверка валидации числовых полей |
| Тестовые данные | Данные для создания комплектации: generation\_id - ID поколения (XV70), body\_type\_id - ID типа кузова (Sedan), engine\_type\_id - ID типа двигателя (Gasoline) engine\_name - V6, engine\_volume - 3.5, engine\_power – 301, transmission\_type\_id - ID типа трансмиссии (Automatic), transmission\_name - 8-speed, drive\_type\_id - ID типа привода (Front-wheel drive), country\_id - ID страны (Japan) , description - Top of the line weight – 1600, load\_capacity – 500, seats – 5 fuel\_consumption - 10.5, fuel\_tank\_volume – 65, range – 500 max\_speed – 220, clearance - 150 Некорректные данные: generation\_id - "not-a-number" engine\_volume - "not-a-number" engine\_power - "invalid" |
| Ожидаемый результат | 1. Система создает новую комплектацию для администратора 2. Система запрещает создание для не-администратора (403 или редирект) 3. Система возвращает ошибки валидации для обязательных полей 4. Система возвращает ошибки валидации для числовых полей |
| Фактический результат | Все ожидаемые результаты соответствуют фактическим: |

На рисунке 25, 26 представлены одна из функций тестирования и результаты всего теста.

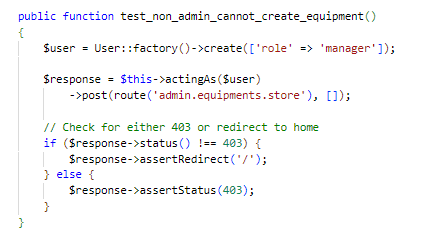


Рисунок 25 – Функция тестирования

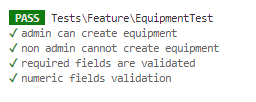


Рисунок 26 – Результаты теста добавления комплектации

Далее был разработан второй сценарий тестирования, редактирование профиля администратором, менеджером и обычным пользователем.

Таблица 20 – Сценарий тестирования операций с инвентарем

|  |  |
| --- | --- |
| Поле | Описание |
| Дата тестирования | 04.05.2025 |
| Приоритет тестирования | Высокий |

Продолжение таблицы 20

|  |  |
| --- | --- |
| Этапы теста | 1. Редактирование профиля обычным пользователем 2. Проверка сброса верификации email при изменении 3. Попытка редактирования профиля администратором 4. Попытка редактирования профиля менеджером 5. Проверка обязательности поля имени 6. Проверка обязательности и уникальности email 7. Проверка обязательности, уникальности и формата телефона |
| Тестовые данные | **Данные пользователя:** Имя - Иван Иванов Email - user@example.com Телефон - 8 123 456 78 90 Роль - user **Данные для обновления (корректные):** Имя - Новое имя Email - new-user@example.com Телефон - 8 999 999 99 99 **Некорректные данные:** 1. Пустое имя 2. Пустой email 3. Занятый email (other@example.com) 4. Пустой телефон 5. Неверный формат телефона (1234567890) 6. Занятый телефон (8 999 999 99 99) |
| Ожидаемый результат | 1. Пользователь может обновить свои данные (имя, email, телефон) |

Продолжение таблицы 20

|  |  |
| --- | --- |
|  | 2. При изменении email сбрасывается верификация 3. Администратор не может редактировать профиль (ошибка) 4. Менеджер не может редактировать профиль (ошибка) 5. Система требует указать имя 6. Система требует уникальный и валидный email 7. Система требует уникальный телефон в правильном формате |
| Фактический результат | Все ожидаемые результаты соответствуют фактическим: |

На рисунке 27, 28 представлены одна из функций тестирования, а именно возможность редактировать профиль пользователем.



Рисунок 27 – Одна из функций тестирования

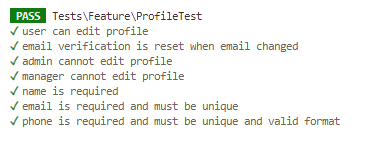


Рисунок 28 – Результаты теста

6 Документация программного продукта

6.1 Руководство пользователя

Для запуска веб-приложения необходимо запустить приложение OpenServer и включить его, чтобы иконка флажка поменяла цвет на зеленый. Далее в терминале необходимо зайти в директорию веб приложения, если это еще не было сделано, введя команду «cd ./» и нажав клавишу Tab выбрать нужную директорию, затем ввести команду «php artisan serve». Открыть еще один терминал, также зайти в директорию информационной системы и ввести команду «npm run dev».

Далее необходимо зайти в браузер и перейти по адресу «http://localhost:8000». После загрузки пользователь увидит главную страницу (рисунок 29) информационной системы.

На главной странице пользователь видит приветственное сообщение с названием автосалона и местом его расположения. Вверху находятся две кнопки одна из которых ведет к каталогу автомобилей а другая позволяет быстро перейти к разделу филиалов. Ниже расположена секция с картой и адресами представительств автосалона. Пользователь может выбрать интересующий его филиал и посмотреть сколько автомобилей доступно в наличии. При нажатии на конкретный филиал откроется список автомобилей, которые в нем находятся что позволяет заранее определиться с посещением нужного места. Также есть возможность перейти ко всем филиалам для более детального ознакомления. Ниже можно найти секцию с популярными автомобилями, представленными в продаже. Каждый автомобиль сопровождается фотографией основными характеристиками и ценой. Нажав на кнопку подробнее, пользователь может ознакомиться с полной информацией об автомобиле. Если же хочется посмотреть больше вариантов, то предусмотрена кнопка которая ведет к полному каталогу машин с возможностью фильтрации и подбора по параметрам. Еще ниже расположена галерея с фотографиями автомобилей что позволяет пользователю визуально ознакомиться с тем какие машины представлены в салоне. Общая структура страницы удобна для навигации и дает понимание, где находится нужная информация без лишних кликов.

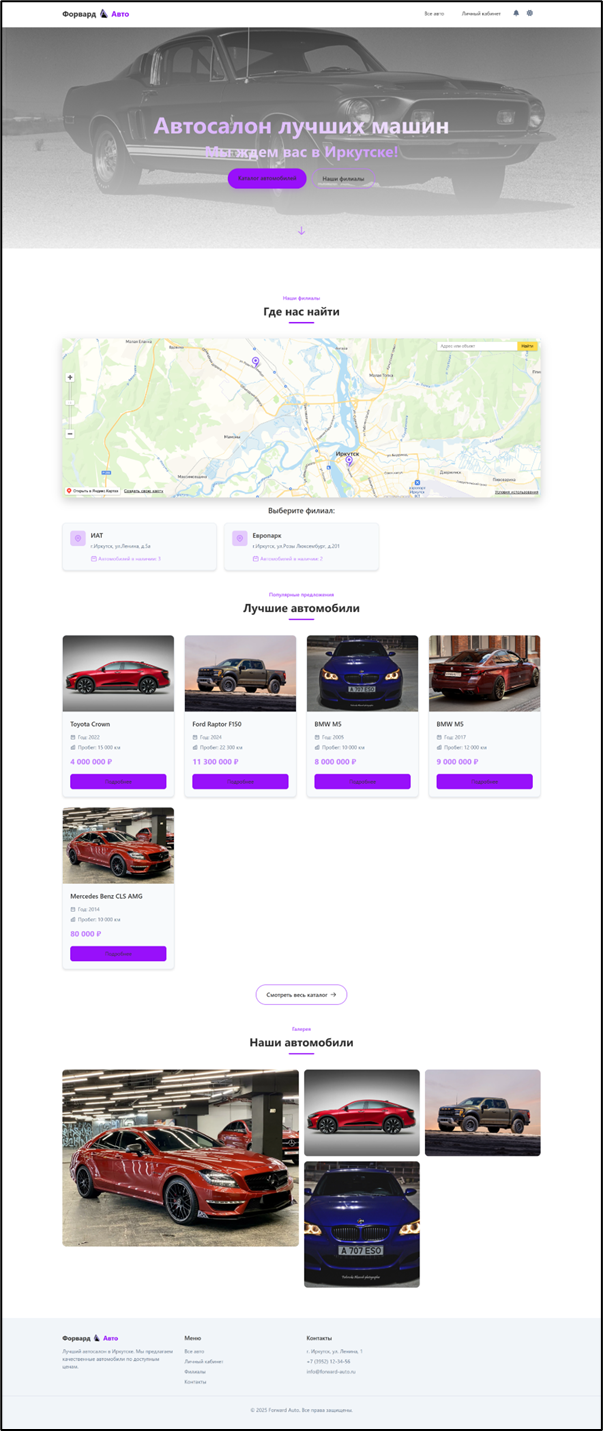


Рисунок 29 – Главная страница

На странице каталога (рисунок 30) пользователь видит поисковую строку, с помощью которой можно найти нужные автомобили по марке, модели или поколению. Ниже расположены фильтры, позволяющие уточнить выборку по различным параметрам: марка, модель, поколение, комплектация, тип кузова, коробка передач, тип двигателя, привод, цена, пробег, год выпуска и объем двигателя. Можно выбрать минимальные и максимальные значения для числовых параметров, а также задать направление сортировки. В правой части экрана отображается список найденных автомобилей, каждый из которых имеет фото, основные характеристики и цену. Нажав на кнопку «Подробнее», пользователь перейдет на страницу с полной информацией об автомобиле. Если автомобилей не нашлось, будет предложено изменить параметры поиска или сбросить фильтры. Также в верхней части есть возможность быстро вернуться к начальным настройкам каталога.

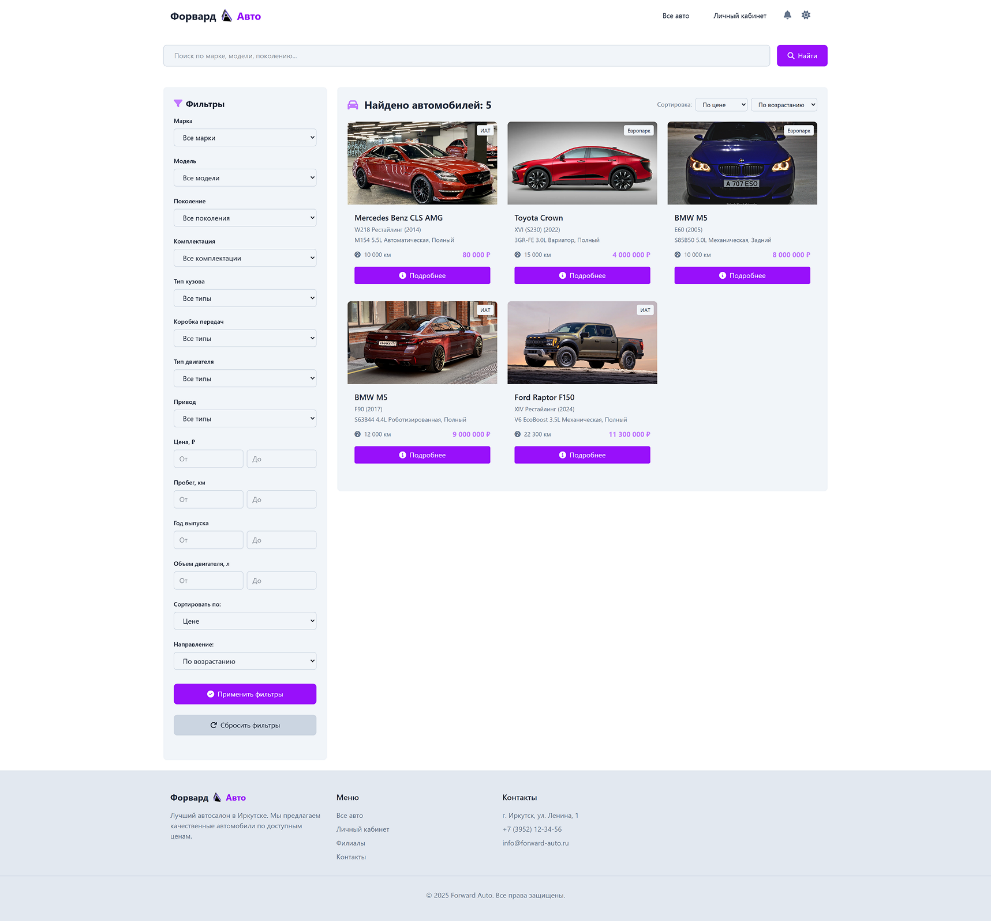


Рисунок 30 – Страница каталога

На странице автомобиля (рисунок 31) пользователь видит хлебные крошки, которые помогают понять где он находится на сайте и быстро вернуться обратно в каталог или к выбранной марке. Ниже расположены уведомления о том что автомобиль успешно забронирован или произошла ошибка это позволяет сразу получить обратную связь после действия. В левой части экрана находится основное изображение автомобиля с возможностью просмотра в модальном окне а при наличии нескольких фото можно переключаться между ними с помощью кнопок или миниатюр. Также есть кнопка для просмотра 3D модели если она доступна. Для удобства пользователя реализована возможность добавить автомобиль в избранное которая сохраняет понравившиеся варианты для дальнейшего просмотра. В правой части страницы отображается подробная информация об автомобиле: марка модель и год выпуска пробег цена и VIN-номер если он указан. Ниже расположена кнопка для бронирования автомобиля на осмотр которая становится недоступной если пользователь уже авторизован но не подтвердил email или если авто уже забронировано. Подробное описание автомобиля если оно заполнено поможет лучше ознакомиться с его состоянием и историей. Далее представлены вкладки с техническими характеристиками комплектацией и местоположением. На вкладке характеристик можно найти данные о двигателе трансмиссии приводе цвете и других параметрах. В разделе комплектации указаны дополнительные параметры такие как вес вместимость расход топлива и другие. На вкладке расположения показан адрес филиала где находится автомобиль и карта для ориентира. Такая структура страницы позволяет пользователю получить всю необходимую информацию об автомобиле и принять решение о посещении салона.).

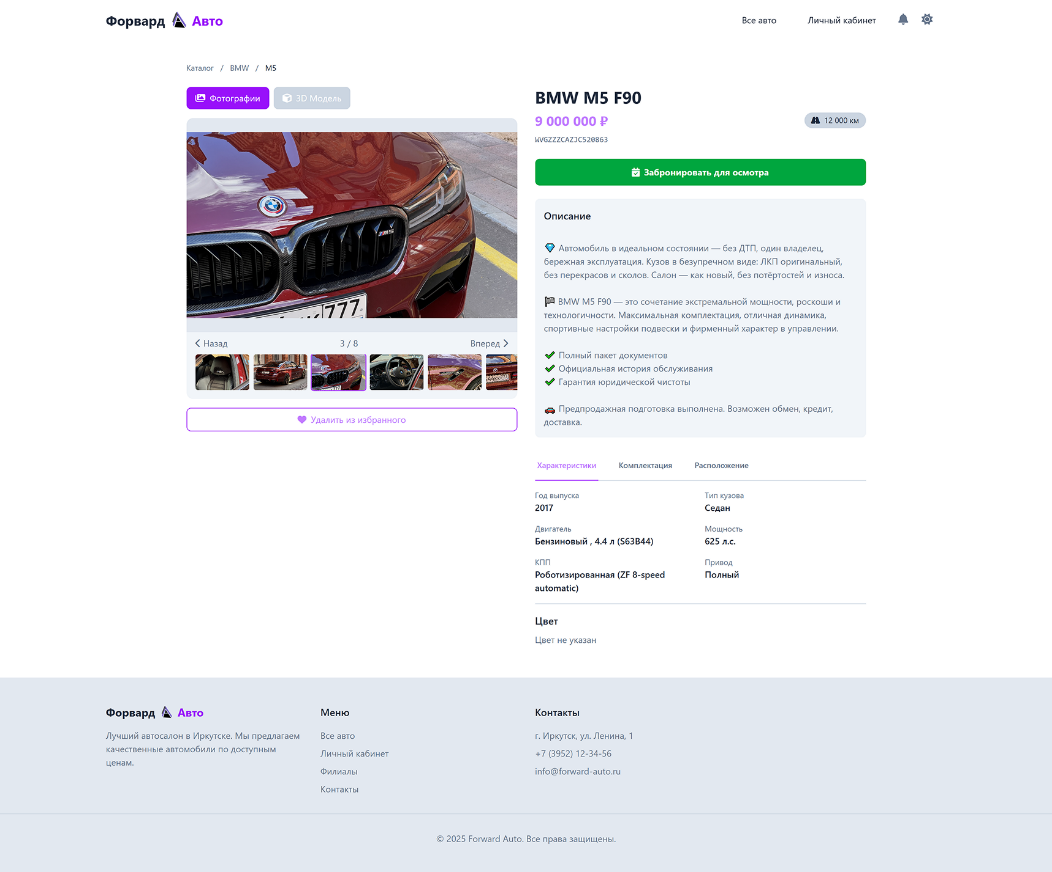


Рисунок 31- Страница авто

На странице регистрации (рисунок 32) пользователь видит форму с полями для ввода данных. Вверху отображается заголовок "Регистрация" и подсказка о том, что если у пользователя уже есть аккаунт, он может войти, нажав на соответствующую ссылку. Форма содержит следующие поля: имя, email, телефон, пароль и подтверждение пароля. Для каждого поля предусмотрены сообщения об ошибках, которые появляются в случае некорректного ввода данных. Также указаны требования к паролю — минимум 8 символов, одна заглавная буква, одна строчная буква и цифра.

После заполнения всех полей пользователь может нажать кнопку "Зарегистрироваться", чтобы создать учетную запись. Если данные введены правильно, происходит переход на страницу авторизации или отображается сообщение об успешной регистрации. Если же возникают ошибки, они подсвечиваются рядом с соответствующими полями, и пользователь должен исправить их перед повторной отправкой формы.

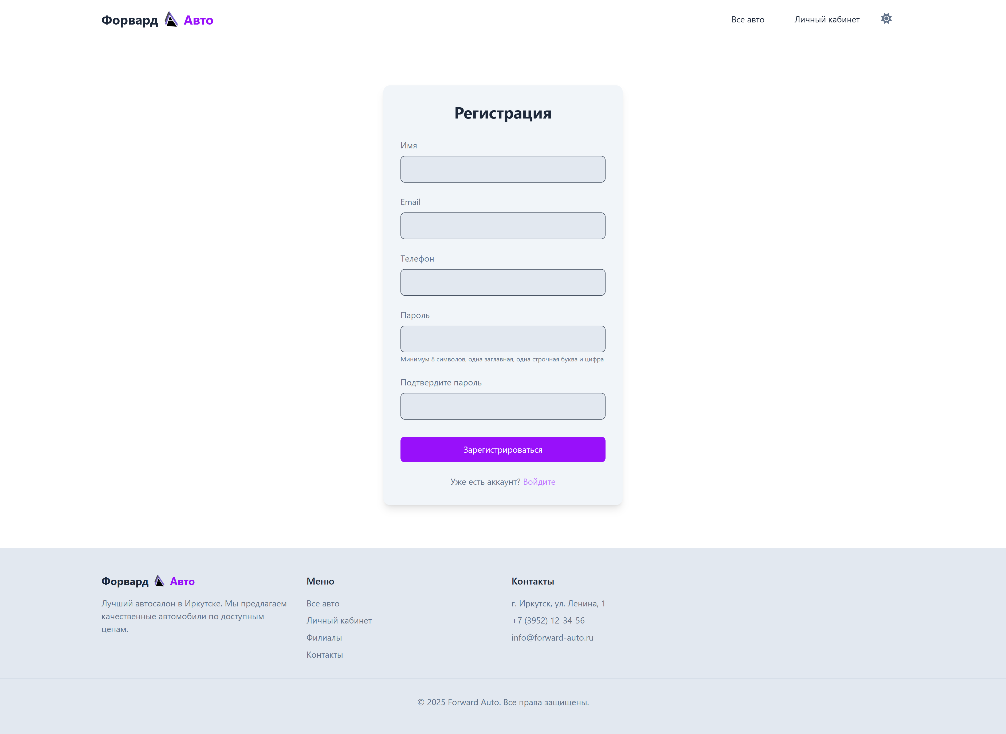


Рисунок 32 – Страница регистрации

Если пользователь уже зарегистрирован, то может перейти к странице входа, которая представлена на рисунке 33.

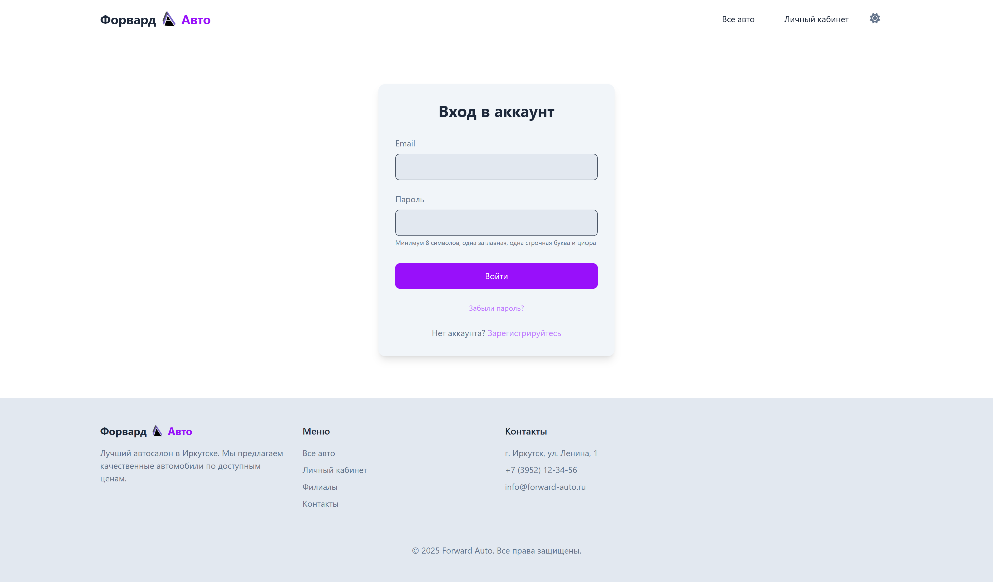


Рисунок 33 – Страница входа

На странице личного кабинета (рисунок 34) пользователь видит приветствие с указанием имени и основной информации о профиле. В левой части экрана находится блок с данными аккаунта: имя, email, телефон, статус (пользователь, менеджер или администратор). Также здесь доступны действия: подтверждение email, редактирование профиля и выход из аккаунта.

В правой части расположены два основных раздела. Первый — "Избранные автомобили", где отображаются до 10 машин, добавленных пользователем в избранное. Каждый автомобиль сопровождается названием и поколением. При необходимости можно перейти на страницу автомобиля или удалить его из избранного. Если автомобилей больше 10, есть ссылка для просмотра полного списка.

Второй раздел — "Мои бронирования". Здесь показаны все забронированные автомобили с датами визита и текущим статусом бронирования (ожидание, подтверждено, отклонено, отменено). Для каждого бронирования доступна кнопка "Подробнее", а также возможность отмены при наличии соответствующего статуса.

Пользователь может редактировать данные профиля или изменить пароль через модальное окно, которое открывается по кнопке "Редактировать профиль".).

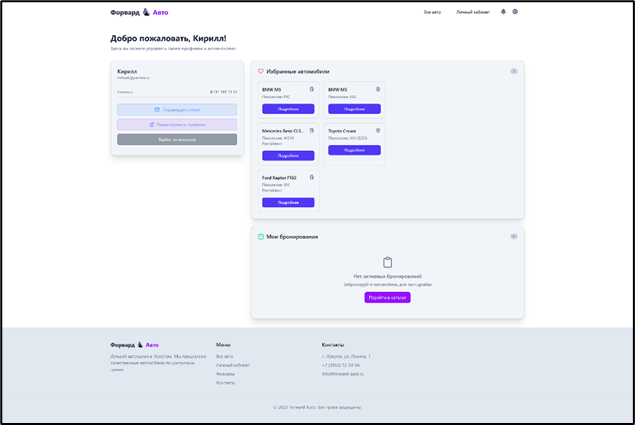


Рисунок 34 – Личный кабинет

6.2 Руководство администратора

На странице администратора (рисунок 35) пользователь видит панель управления, разделённую на блоки по типам данных: страны, филиалы, марки, модели, типы кузова, двигателя, привода и другие. Каждый блок содержит список существующих записей и форму добавления новой. Для каждой записи доступны действия: редактирование и удаление.

В верхней части есть строка поиска для фильтрации справочников. При вводе текста система автоматически подгружает подходящие результаты. Также можно сбросить фильтр, нажав на соответствующую кнопку.

Для добавления нового элемента, например, новой марки или типа двигателя, нужно заполнить поле ввода и нажать «Добавить». Чтобы изменить существующую запись, достаточно отредактировать текст в поле и нажать кнопку сохранения. Удаление выполняется через подтверждающее окно.

Все изменения сохраняются сразу, а при ошибках ввода система выводит уведомления рядом с полями.

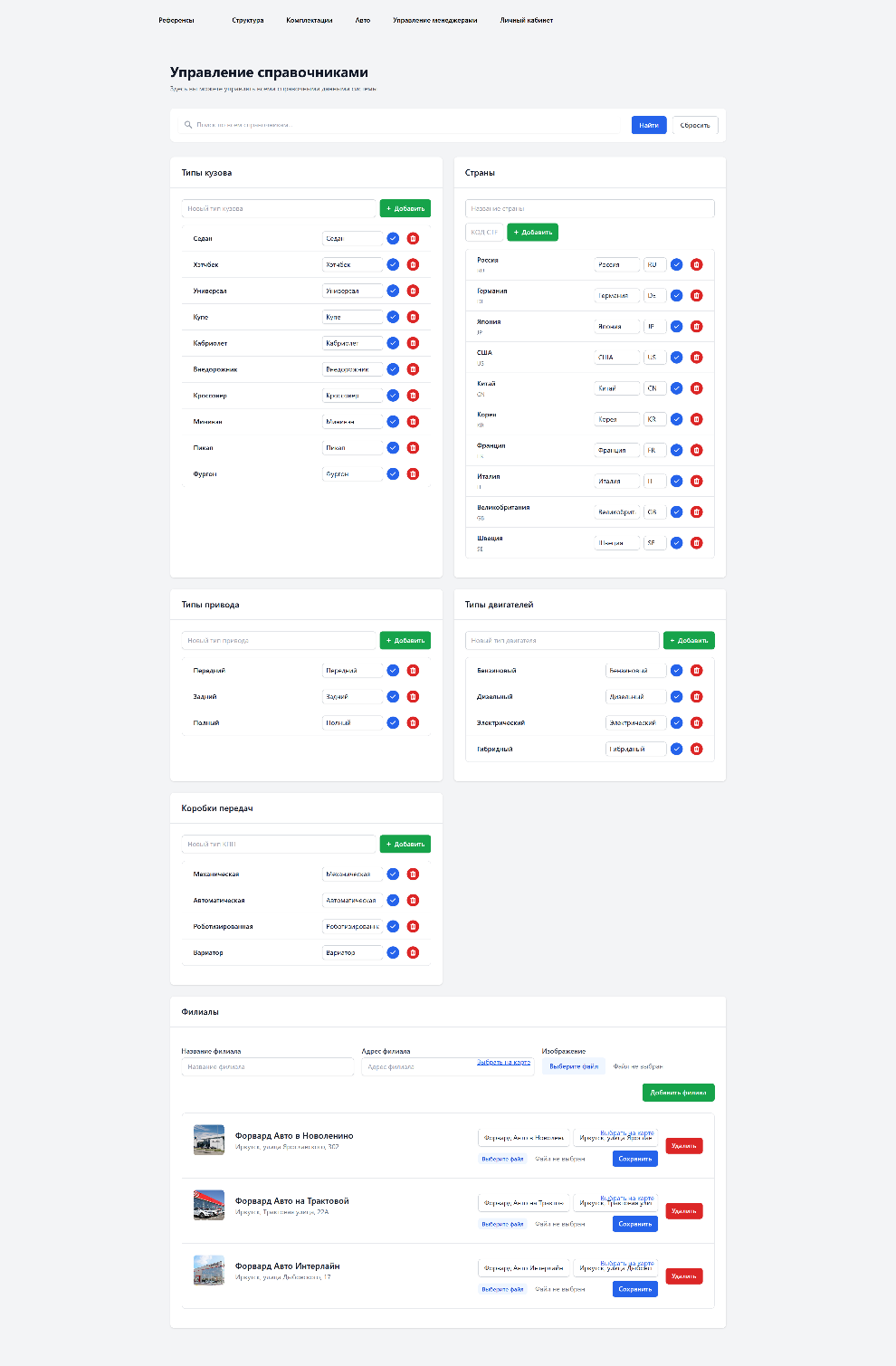


Рисунок 35 – Управление референсами

На странице управления справочниками (рисунок 36) администратор видит список разделов: марки, модели, поколения и другие. Каждый раздел содержит таблицу с текущими записями и форму для добавления новых. В верхней части страницы есть строка поиска, которая фильтрует данные по названию.

Для добавления новой записи нужно заполнить поле ввода и нажать «Добавить». Чтобы отредактировать существующую запись, следует нажать кнопку редактирования, изменить данные и подтвердить сохранение. Удаление выполняется через модальное окно с подтверждением действия.

Все изменения применяются сразу, а при ошибках система выводит уведомления рядом с полями. Также доступна пагинация для просмотра других страниц справочника. Такая структура позволяет быстро управлять данными, не покидая интерфейс.

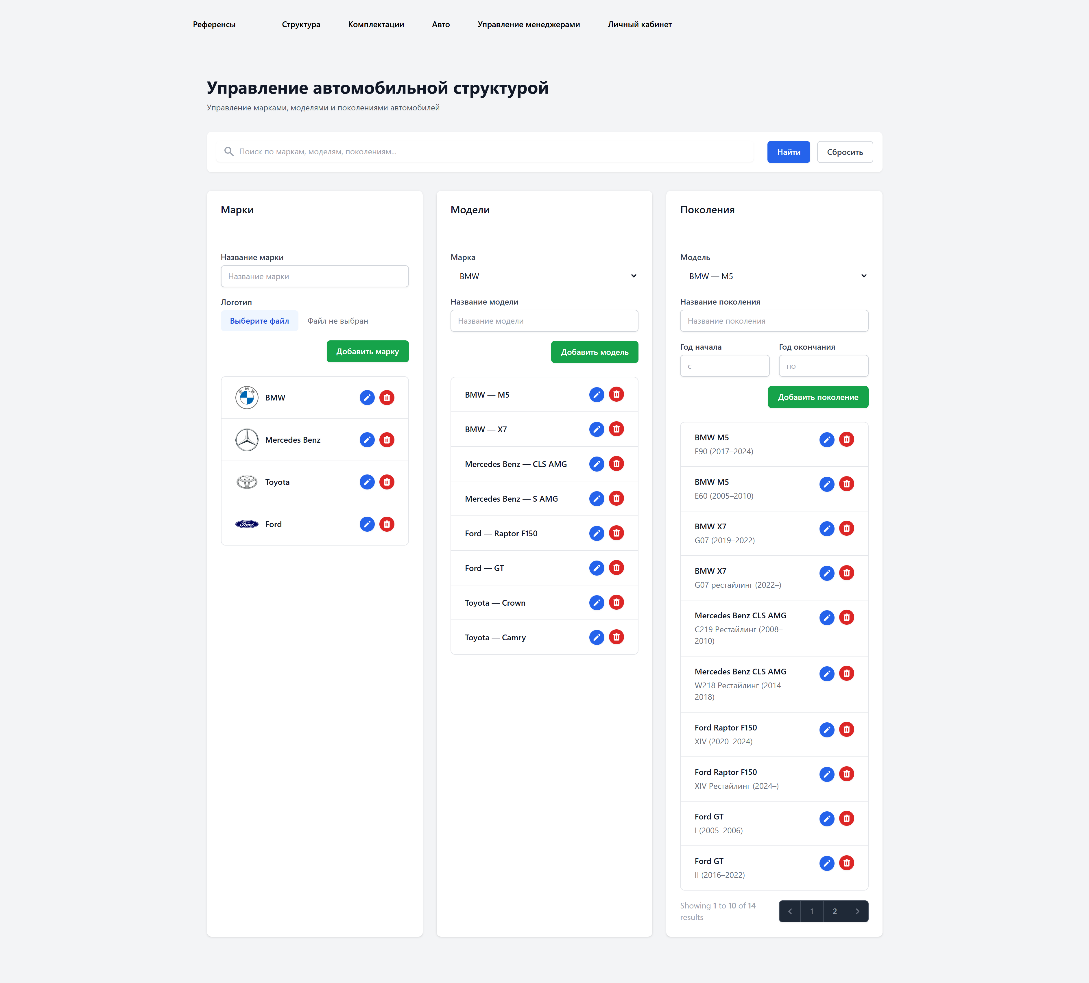


Рисунок 36 – Управление структурой

На странице управления комплектациями (рисунок 38) администратор видит фильтры по марке и модели, форму поиска и кнопку добавления новой комплектации. Ниже отображается таблица с текущими записями. Каждая строка содержит информацию о марке, модели, поколении, типе двигателя, приводе, доступных цветах и действиях: редактирование и удаление.

В нижней части страницы находится раздел цветов. Здесь отображаются существующие цвета с их названием, HEX-кодом, визуальным обозначением и списком комплектаций, в которых они используются. Для каждого цвета доступны редактирование и удаление. Изменения сохраняются при нажатии на кнопку «Сохранить».

Пагинация позволяет просматривать другие страницы списка. Сообщения об ошибках выводятся вверху, если данные введены некорректно или произошла ошибка при выполнении действия. (рисунок 37).

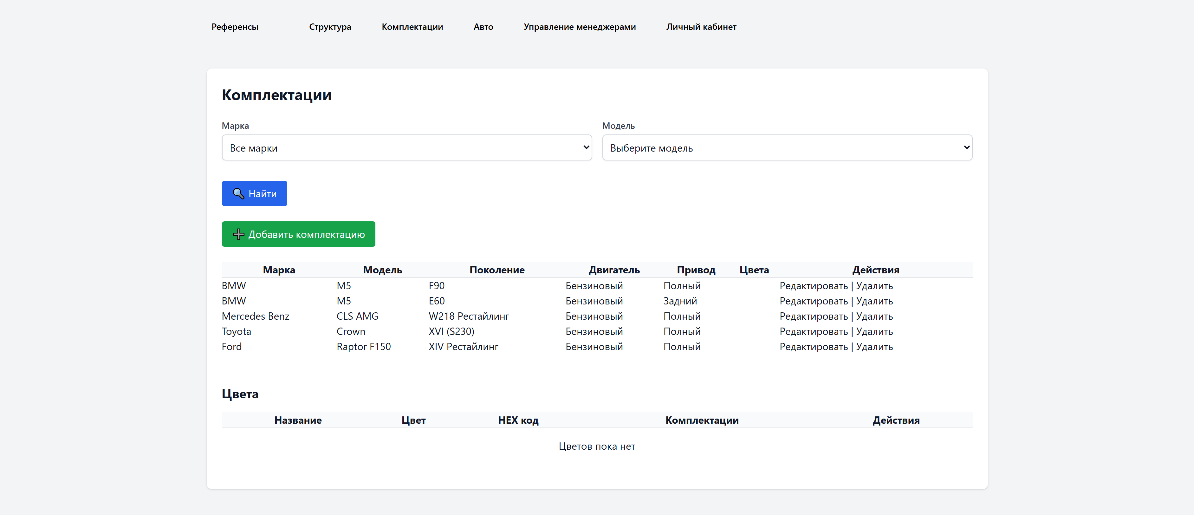


Рисунок 37 – Управление комплектациями

На странице управления автомобилями (рисунок 38) администратор видит строку поиска, с помощью которой можно находить авто по VIN, марке, модели или поколению. Рядом расположены кнопки поиска и сброса фильтра. Ниже находится таблица со списком автомобилей, включающая информацию о VIN, марке/модели/поколении, цвете, цене, пробеге, филиале и статусе (на продаже или продан).

Каждая запись содержит две кнопки: «Редактировать» и «Удалить». Нажав на «Редактировать», администратор переходит к форме изменения данных автомобиля. Кнопка «Удалить» удаляет запись после подтверждения.

В правом верхнем углу есть кнопка «Добавить автомобиль», которая открывает форму для добавления нового автомобиля. Также доступна пагинация для навигации по страницам списка. Сообщения об ошибках или успешных действиях отображаются вверху, если такие имеются.

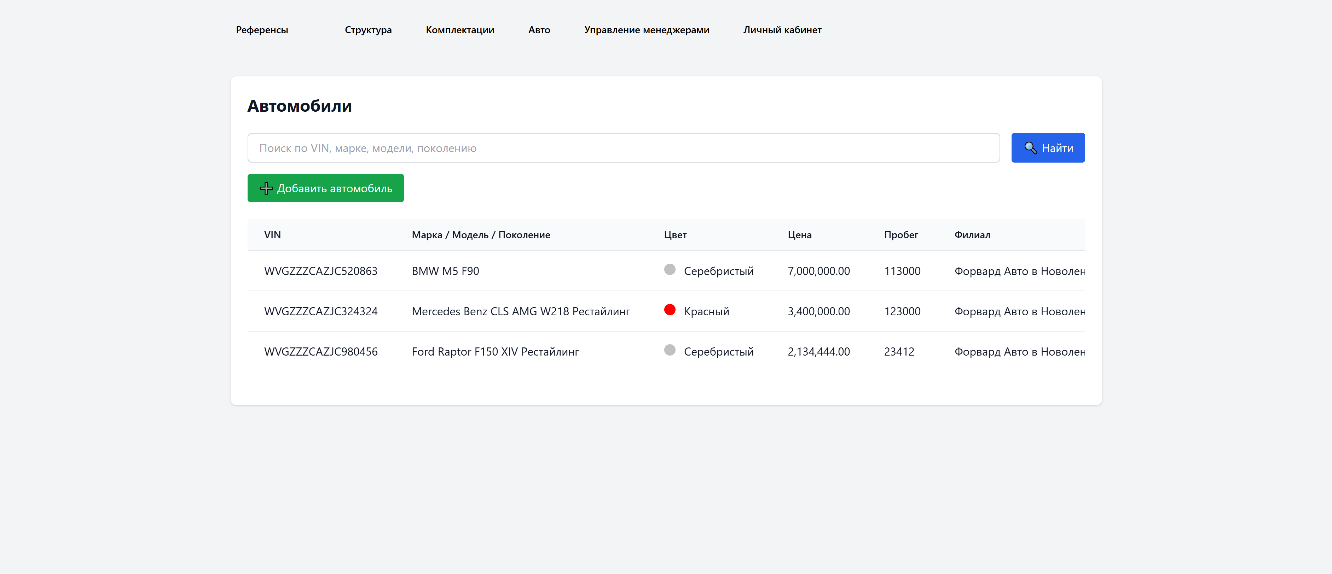


Рисунок 38 – Управление авто

На странице управления менеджерами (рисунок 39) администратор видит форму добавления нового менеджера с полями: имя, email, телефон, пароль и подтверждение пароля. После заполнения всех полей можно нажать кнопку «Добавить менеджера», и новый пользователь появится в системе. Ниже отображается список уже добавленных менеджеров в виде таблицы с колонками: ID, имя, email, телефон и действия. Для каждого менеджера доступны две кнопки: Редактировать — открывает возможность изменить данные (поля становятся редактируемыми), а также ввести новый пароль при необходимости. Кнопка "Сохранить" применяет изменения, удалить — удаляет запись после подтверждения. После успешного добавления, редактирования или удаления выводится сообщение об успехе. Эта страница позволяет удобно управлять учетными записями менеджеров без лишних кликов.

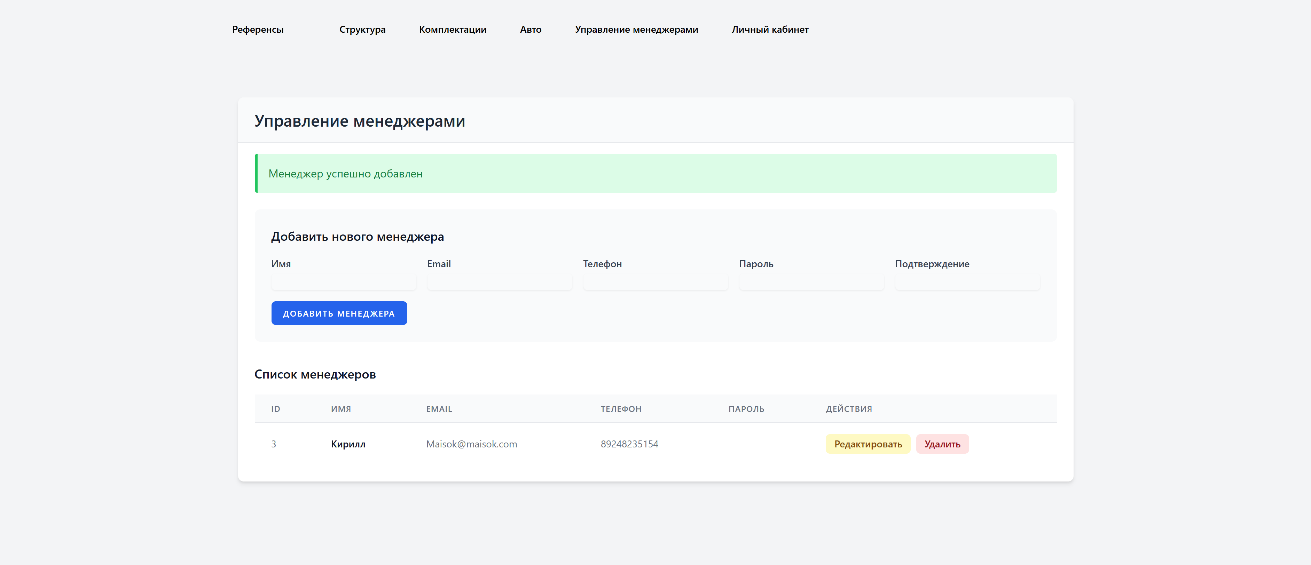


Рисунок 39 – Управление менеджерами

6.3 Руководство менеджера

На странице управления бронированиями (рисунок 40) менеджер видит фильтры для поиска записей по статусу и дате визита. После выбора нужных параметров можно нажать «Применить», чтобы отобразить подходящие бронирования, или «Сбросить» для отображения всех записей.

Ниже расположен список бронирований в виде таблицы. Каждая запись содержит информацию: ID, автомобиль (марка, модель, VIN), клиента (имя, email, телефон), дату бронирования, дату визита и текущий статус. Статусы окрашены в разные цвета для удобного восприятия: ожидание, подтверждено, отклонено, завершено.

Для каждой записи доступны две кнопки:

Редактировать — открывает форму изменения статуса и других данных бронирования.

Удалить — удаляет запись после подтверждения.

Если бронирований нет или фильтры не дают результатов, отображается соответствующее сообщение. Также доступна пагинация для просмотра других страниц списка. Сообщения об успехе или ошибках выводятся вверху страницы при наличии.

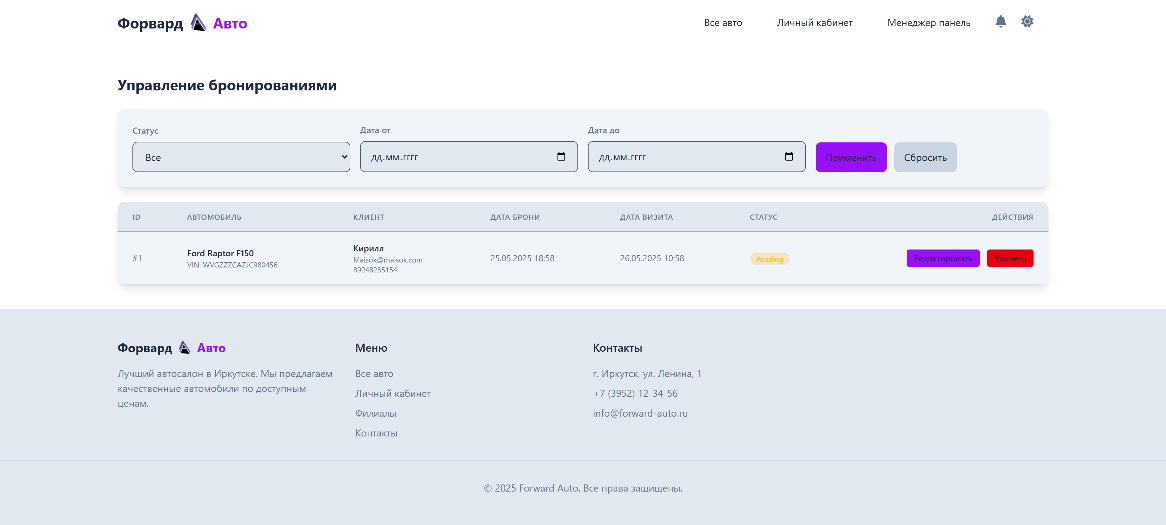


Рисунок 40 – Управление бронированиями

7 Стоимость разработки

Разработка веб-приложения для автосалона — это комплексный и многоуровневый процесс, требующий не только технической проработки, но и тщательного экономического обоснования. В условиях растущего спроса на цифровые сервисы и автоматизацию бизнес-процессов внедрение онлайн-платформы позволяет автосалонам выйти на новый уровень: повысить качество клиентского сервиса, упростить взаимодействие с покупателями и оптимизировать управление продажами автомобилей.

7.1 Расчет трудоемкости разработки программного обеспечения

Трудоемкость считается ключевым показателем экономической эффективности (характеристика расхода труда). Позволяет оценить использование рабочего времени в результате производственной деятельности или при оказании услуг. Формула трудоемкости показывает количество труда, затрачиваемое на единицу продукции.

Суммарные затраты труда рассчитываются как сумма составных затрат труда по формуле (1):

(1)

Таблица 21 – Суммарные затраты времени

|  |  |  |
| --- | --- | --- |
| Вид работ | Трудоемкость в часах | |
| Всего | В том числе машинное время |
| Анализ предметной области проекта и | 25 | 25 |

Продолжение таблицы 21

|  |  |  |
| --- | --- | --- |
| Анализ предметной области проекта и описание структуры | 25 | 25 |
| Анализ инструментов | 20 | 20 |
| Проектирование проекта и описание этапов проектировании | 35 | 35 |
| Создание технического задания на разработку проекта | 15 | 10 |
| Разработка программного продукта | 100 | 100 |
| Тестирование программного продукта | 30 | 30 |
| Подготовка документации | 25 | 15 |
| Итого: | 250 | 235 |

7.2 Затраты на оплату

Эффективная разработка программного продукта требует значительных трудозатрат.

Затраты на оплату (ЗОТ) труда разработчика ПО включают затраты на оплату труда и отчисления от фонда заработной платы.

Затраты на заработную плату определяются произведением часовой тарифной ставки разработчика и трудоемкости разработки программного продукта по следующей формуле:

Основная заработная плата рассчитывается в рублях по формуле (2)

(2)

Где *t* – суммарные затраты труда, вычисляемые по формуле (1), час.;

*ТС*МЕС– часовая тарифная ставка, руб.

Часовая тарифная ставка: 800 руб.

Основная зарплата:

7.3 Затраты на амортизацию оборудования

Использование оборудования в процессе разработки приводит к его постепенному износу. Далее рассчитана сумма амортизационных отчислений для ноутбука и принтера, которые применялись при создании веб-приложения, что отражает часть затрат проекта.

Срок полезного использования согласно классификатору основных средств равен 2-3 года.

Расчет амортизации производится при помощи данных формул (3-6):

(3)

где

Ан – годовая норма амортизации;

Кк – срок полезного использования в соответствии с классификатором.

(4)

где

Аг – ежегодная сумма амортизации;

Со – начальная стоимость оборудования.

(5)

где

Ам – ежемесячная сумма амортизации.

Далее необходимо сумму ежемесячной амортизации умножить на количество месяцев эксплуатации оборудования.

(6)

Стоимость компьютера с комплектующими и периферией: 106674 руб.

Комплектующие:

1. Процессор AMD Ryzen 5 5500 OEM (6399 руб)
2. SATA накопитель Kingston A400 [SA400S37/960G] (5999 руб)
3. Видеокарта Sapphire AMD Radeon RX 6700 PULSE (32999 руб)
4. Блок питания DEEPCOOL PF600 (3599 руб)
5. Кулер для процессора ID-COOLING SE-214-XT ARGB (1599 руб)
6. Материнская плата ASRock B450M Pro4 R2.0 (6999 руб)
7. Корпус DEEPCOOL CC560 WH V2 (4799 руб)

Периферия:

1. Клавиатура Аnne pro 2 (7483 руб)
2. Мышь DragonFly R1 PRO MAX (3299 руб)
3. Монитор Prime Box 24.5 (13500 руб)
4. Монитор ARDOR GAMING INFINITY AF27H1 (19999 руб)

Срок полезного использования 4 лет.

Годовая норма амортизации получается равна:

= 25%

Ежегодная сумма амортизации равна:

Ежемесячная сумма амортизации равна:

.

Время разработки: 235 часов ≈ 1,5 месяца

Сумма амортизации:

Стоимость принтера Epson L200: 5996 руб.

Срок полезного использования 4 года.

Годовая норма амортизации получается равна:

= 25%

Ежегодная сумма амортизации равна:

Ежемесячная сумма амортизации равна:

Время использования: 15 часов ≈ 0,02 месяца

Сумма амортизации:

Итого амортизация оборудования:

3333 (компьютер) + 25 (принтер) = 3358 руб.

7.4 Расчет затрат на электроэнергию

Эксплуатация технических средств сопровождается потреблением электроэнергии. Здесь приведен расчет затрат на электроэнергию, необходимую для работы ноутбука в течение всего периода разработки, что является важной составляющей общих расходов.

Для расчета затрат на электроэнергию первоочередное необходимо рассчитать расход электроэнергии оборудования по следующей формуле (7):

(7)

где

P – электрическая мощность в киловаттах;

t – время в часах.

Мощность компьютера: 0,5 кВт⋅ч.

Согласно таблице 1, машинное время работы ноутбука равно 235 часов. Стоимость одного киловатт-часа равна 1,58 рубля для городского населения Иркутской области. Общее потребление равно 117.5 кВт⋅ч.

Расчет стоимости электроэнергии по следующей формуле (8):

(8)

где Ц – стоимость киловатт-час.

Затраты на электроэнергию:

Мощность принтера: 0,2 кВт

Принтер использовался только во время подготовки документации и согласно таблице 1 это равно 15 часов.

Затраты на электроэнергию:

Общие затраты на электроэнергию:

7.5 Затраты на материалы, израсходованные при проведении разработки и прочие

Для успешной реализации проекта требуются различные вспомогательные материалы. В данном разделе перечислены и рассчитаны затраты на канцелярские товары и другие расходные материалы, которые были использованы в ходе работы.

Таблица 15 – Затраты на материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу (руб.) | Кол-во (шт.) | Всего (руб.) |
| Бумага | 360 | 1 | 360 |
| Чернила | 950 | 2 | 1900 |
| Ручка | 75 | 2 | 150 |
| Папка для документов | 150 | 1 | 150 |
| Итого | 2560 | | |

Таблица 16 – Использование платного ПО

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за использование | Срок использования | Всего (руб.) |
| Хостинг (1 год) | 10000 | 4 мес. | 3333 |
| Итого | | | 3333 |

После расчета показателей можно определить стоимость программы по формуле (9):

Зразр = Ззп + Заморт + Зэкспл + Змат+ Зпр, (9)

Таблица 17 – общие расходы

|  |  |  |  |
| --- | --- | --- | --- |
| Статьи затрат | Индекс | Сумма, руб. | Удельный вес затрат, % |
| Заработная плата | Зп. | 200000 | 95,49 |
| Амортизация оборудования | Заморт. | 3358 | 1,6 |
| Затраты на электроэнергию | Зэкспл. | 190,39 | 0,09 |
| Затраты на материалы | Змат. | 2560 | 1,22 |
| Прочие затраты | Зпр | 3333 | 1,59 |
| Итого: | Зразр. | 209441,39 | 100 |

Зразр = 200000 + 3358 + 190,39 + 2560 + 3333= 133184,82

7.7 Расчет цены

Самозанятые могут сотрудничать как с физическими, то есть с обычными гражданами, так и с юридическими лицами и ИП. При работе с физическим лицом ставка, по которой рассчитывается налог, – 4% (Нфз) от суммы дохода, а при работе с компанией или ИП – 6% (Нюл).

Следовательно, данный вид налога рассчитывается от суммы разработки полностью по формуле 10.

или (10)

Налог для самозанятых:  или

Определение итоговой стоимости продукта включает не только покрытие всех затрат, но и учет прибыли и налогов. Далее представлен итоговый расчет цены веб-приложения, который учитывает себестоимость разработки, планируемую прибыль и НДС.

Состав расчетной цены на разработку определяется по формуле (11):

(11)

Где Зразр – затраты на разработку;

Пр – прибыль от реализации.

Пр = 209441,39×12%= 25133 руб.

НДС = (209441,39+ 25133) \* 0,2 = 46914.88

Итоговая цена:

Ц = 209441,39 + 25133 + 46914,88= 281489,27

ЗАКЛЮЧЕНИЕ

В рамках выполнения дипломного проекта была сформулирована основная задача – создание веб-приложения клининговой компании. Для достижения этой задачи был сформирован перечень необходимых действий.

Для реализации информационной системы были выбраны следующие технологии: MySQL и Laravel.

Задачи были успешно выполнены, и веб-приложение было разработано в соответствии с техническим заданием. В ходе работы над дипломным проектом были получены и использованы новые знания о технологиях и средствах разработки. Было создано веб-приложение, которое позволяет оформлять заказы, обладает удобным интерфейсом и простой навигацией по содержимому.

В результате дипломного проектирования была достигнута основная цель – создание веб-приложения клининговой компании. Были изучены средства и выполнены поставленные задачи для достижения цели:

* + произвести исследование деятельности клининговой компании;
  + определить инструментальные средства проектирования и реализации;
  + разработать техническое задание для веб-приложения;
  + спроектировать базу данных;
  + спроектировать интерфейс пользователя веб-приложения;
  + реализовать веб-приложение;
  + провести тестирование веб-приложения;
  + составить программную документацию.

Список используемых источников

1. sqlzoo.net – Interactive SQL Tutorial – URL: https://sqlzoo.net (дата обращения: 25.04.2025). – Текст: электронный.
2. laravel.com – Official Documentation – URL: https://laravel.com/docs (дата обращения: 22.04.2025). – Текст: электронный.
3. laravel.ru – Laravel Documentation in Russian – URL: https://laravel.ru/docs (дата обращения: 23.04.2025). – Текст: электронный.
4. laracasts.com – Video Lessons on Laravel and PHP – URL: https://laracasts.com (дата обращения: 24.04.2025). – Текст: электронный.
5. w3schools.com/sql – SQL Tutorial – URL: https://www.w3schools.com/sql (дата обращения: 26.04.2025). – Текст: электронный.
6. tailwindcss.com – Official Tailwind CSS Documentation – URL: https://tailwindcss.com/docs (дата обращения: 27.04.2025). – Текст: электронный.
7. tailwindcss.ru – Tailwind CSS Russian Language Documentation – URL: https://tailwindcss.ru/docs (дата обращения: 28.04.2025). – Текст: электронный.
8. plugins.jetbrains.com – Laravel Plugin for PHPStorm Guide – URL: https://plugins.jetbrains.com/plugin/9869-laravel (дата обращения: 29.04.2025). – Текст: электронный.
9. dev.mysql.com – MySQL Documentation – URL: https://dev.mysql.com/doc (дата обращения: 30.04.2025). – Текст: электронный.
10. nova.laravel.com – Laravel Nova Admin Panel – URL: https://nova.laravel.com (дата обращения: 01.05.2025). – Текст: электронный.
11. github.com/laravel/laravel – Laravel GitHub Repository – URL: https://github.com/laravel/laravel (дата обращения: 02.05.2025). – Текст: электронный.
12. css-tricks.com – Tailwind CSS Examples and Articles – URL: https://css-tricks.com (дата обращения: 03.05.2025). – Текст: электронный.
13. laravel-news.com – Laravel News and Updates – URL: https://laravel-news.com (дата обращения: 04.05.2025). – Текст: электронный.
14. stackoverflow.com/questions/tagged/laravel – Laravel Questions and Answers – URL: https://stackoverflow.com/questions/tagged/laravel (дата обращения: 05.05.2025). – Текст: электронный.
15. dev.to/t/laravel – Laravel Community Discussions – URL: https://dev.to/t/laravel (дата обращения: 06.05.2025). – Текст: электронный.

Приложение А Техническое задание

**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

**Техническое задание**

**ВЕБ-ПРИЛОЖЕНИЕ «АВТОСАЛОН»**

|  |  |  |
| --- | --- | --- |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (П.А. Стош) |
|  | (подпись, дата) |  |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (К.С. Мясников) |
|  | (подпись, дата) |  |

Иркутск, 2025

**1. Введение**

**1.1.** **Общие сведения**

Документ представляет собой техническое задание на создание веб-приложения автосалона, предназначенной для организации работы веб-приложения автосалона.

**1.2.** **Цели и задачи**

Целью дипломного проекта является создание веб-приложения, которое позволит автоматизировать процессы, происходящие в автосалоне.

Задачи веб-приложения включают:

* Управление бронированиями.
* Управление базой автомобилей.
* Управление менеджерами.
* Управление продаваемыми автомобилями.

**2. Основания для разработки**

**2.1.** **Нормативные документы**

Документ основывается на следующих нормативных документах:

* ГОСТ 34.602-2020 "Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы".
* ГОСТ Р 56477-2015 "Проектирование и внедрение информационных систем. Общие требования".

**2.2.** **Проектные документы**

Проектные документы включают:

* Пояснительную записку.
* Руководство пользователя.
* Задание на дипломный проект.

**3. Назначение системы**

**3.1. Общее описание**

Веб-приложение автосалона – это веб-приложение, предназначенное для автоматизации и оптимизации процессов, связанных с бронированием автомобилей, обслуживанием клиентов и управление данными автосалона.

**3.2.** **Преимущества и новизна**

Веб-приложение будет предоставлять:

* Интуитивно понятный интерфейс для пользователей.
* Автоматизацию процессов, связанных с управлением автомобилями, бронированиями, менеджерами и взаимодействием их с клиентами.

**4. Требования к системе**

**4.1.** **Функциональные требования**

1. Общие функции системы
   * Авторизация пользователей.
   * Операции с отображением данных с возможностью сортировки по год, пробег, цена.
   * Функции фильтрации.
   * Функции поиска данных по марке, модели, поколению.
   * Адаптация под различные виды устройства.
   * Регистрация пользователей.
   * Разграничение прав доступа.
   * Интеграция с Яндекс картами.
2. Пользователь:
   * Оформление и отмена бронирований
   * Редактирование данных аккаунта
3. Менеджер:
   * Редактирование статуса бронирований с указанием комментария и даты
   * Формирование отчетов с функцией экспорта данных бронирований
4. Администратор:
   * Удаление, редактирование, удаление данных об автомобилях с сохранением целостности данных.
   * Удаление, редактирование, удаление данных.
   * Формирование отчетов с функцией экспорта данных авто.

**4.2.** **Технические требования**

1. Производительность:
   1. Обработка до 500 задач одновременно.
   2. Время отклика системы не более 7 секунд при загрузке данных.
2. Надежность:
   1. Доступность системы не менее 99,5% в год.
   2. Регулярное резервное копирование данных.
3. Безопасность:
   1. Регистрация и мониторинг всех действий пользователей.
   2. Шифрование данных на уровне передачи хранения.

**4.3. Эксплуатационные требования**

* Удобство использования.
* Дружественный пользовательский интерфейс.

**5. Требования к техническому обеспечению**

**5.1. Оборудование**

* Сервер: Серверная платформа с процессором не менее 8 ядер, 16 ГБ ОЗУ, SSD объемом 512 ГБ.
* Клиентские рабочие станции: ПК с ОС Windows 10 или Linux, 4 ГБ ОЗУ, 2 ГБ свободного места на диске.

**5.2. Сетевые требования**

* Сеть: Доступ в Интернет со скоростью не менее 10 Мбит/с.
* Сетевые протоколы: Поддержка TCP/IP, HTTPS.

**6. Требования к программному обеспечению**

**6.1. Программные компоненты**

* Операционная система: Windows Server.
* Базы данных: MySQL версии не ниже 8.0.
* Программное обеспечение: Веб-сервер Nginx, интерпретатор PHP 8.2 или выше.

**6.2. Интерфейсы**

* Интерфейс пользователя: Веб-интерфейс с поддержкой браузеров Chrome, Firefox, Edge.

**7. Организационно-технические требования**

**7.1. Этапы разработки**

В таблице 1 представлены сроки и этапы разработки системы.

Таблица 1 – Сроки и этапы разработки

|  |  |
| --- | --- |
| Наименование этапов дипломного проекта | Срок |
| Провести предпроектное исследование. | 22.02.2025 |
| Составить техническое задание на разработку программного продукта в соответствии с ГОСТ. | 28.02.2025 |
| Провести проектирование программного продукта. | 15.03.2025 |
| Разработать программный продукт. | 30.04.2025 |
| Выполнить тестирование и отладку программного продукта. | 06.05.2025 |
| Разработать документы для программного продукта. | 16.05.2025 |
| Составить пояснитиельную записку. | 18.05.2025 |

Приложение Б код клиентской части

<!DOCTYPE html>

<html lang="en" class="dark">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<link rel="shortcut icon" href="{{ asset('images/logo.png') }}" type="image/x-icon">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

@vite('resources/css/app.css')

@vite('resources/js/app.js')

<meta name="user-id" content="{{ auth()->id() }}">

<meta name="csrf-token" content="{{ csrf\_token() }}">

<script src="https://api-maps.yandex.ru/2.1/?apikey=fa8f0187-f7ba-4f94-aa91-c6a61473cec3&lang=ru\_RU" type="text/javascript"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>

<script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>

<script defer src="https://cdn.jsdelivr.net/npm/alpinejs@3.14.8/dist/cdn.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/three@0.128.0/examples/js/controls/OrbitControls.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/three@0.128.0/examples/js/loaders/GLTFLoader.min.js"></script>

<title>Forward Auto</title>

<script>

tailwind.config = {

darkMode: 'class',

theme: {

extend: {

animation: {

'float': 'float 6s ease-in-out infinite',

'pulse-slow': 'pulse 4s cubic-bezier(0.4, 0, 0.6, 1) infinite',

},

keyframes: {

float: {

'0%, 100%': { transform: 'translateY(0)' },

'50%': { transform: 'translateY(-10px)' },

}

},

colors: {

dark: {

100: '#f5f5f7',

200: '#e5e5e7',

300: '#d4d4d8',

400: '#a1a1aa',

500: '#71717a',

600: '#52525b',

700: '#3f3f46',

800: '#27272a',

900: '#18181b',

},

light: {

900: '#f8fafc',

800: '#f1f5f9',

700: '#e2e8f0',

600: '#cbd5e1',

500: '#94a3b8',

400: '#64748b',

300: '#475569',

200: '#334155',

100: '#1e293b',

}

}

}

}

}

</script>

<style>

/\* Custom scrollbar \*/

::-webkit-scrollbar {

width: 8px;

}

::-webkit-scrollbar-track {

background: #2D2D2D;

}

::-webkit-scrollbar-thumb {

background: #8B5CF6;

border-radius: 4px;

}

.light ::-webkit-scrollbar-track {

background: #e2e8f0;

}

.light ::-webkit-scrollbar-thumb {

background: #7C3AED;

}

/\* Theme switcher \*/

.theme-switch {

position: relative;

display: inline-block;

width: 60px;

height: 30px;

}

.theme-switch input {

opacity: 0;

width: 0;

height: 0;

}

.slider {

position: absolute;

cursor: pointer;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: #4a4a4a;

transition: .4s;

border-radius: 34px;

}

.slider:before {

position: absolute;

content: "";

height: 22px;

width: 22px;

left: 4px;

bottom: 4px;

background-color: white;

transition: .4s;

border-radius: 50%;

}

input:checked + .slider {

background-color: #8B5CF6;

}

input:checked + .slider:before {

transform: translateX(30px);

}

/\* Light theme styles \*/

.light {

color: #1e293b;

}

.light body {

background-color: #f8fafc;

}

.light .bg-\[\#1C1B21\] {

background-color: #ffffff;

}

.light .bg-\[\#2A2A2A\],

.light .bg-\[\#3C3C3C\] {

background-color: #f1f5f9;

}

.light .bg-\[\#2D2D2D\] {

background-color: #e2e8f0;

}

.light .text-white {

color: #1e293b;

}

.light .text-gray-300,

.light .text-gray-400 {

color: #64748b;

}

.light .border-gray-700 {

border-color: #cbd5e1;

}

.light .bg-gray-800 {

background-color: #e2e8f0;

}

.light .bg-gray-700 {

background-color: #cbd5e1;

}

.light .bg-gray-600 {

background-color: #94a3b8;

}

.light .text-black {

color: #1e293b;

}

.light .notification-dropdown,

.light .notification-item.unread {

background-color: #ffffff;

}

.light .notification-item {

border-color: #e2e8f0;

}

.light .notification-item:hover {

background-color: #f1f5f9;

}

/\* Smooth theme transition \*/

body {

transition: background-color 0.5s ease, color 0.5s ease;

}

/\* Fixed header \*/

.main-header {

position: fixed;

top: 0;

left: 0;

right: 0;

z-index: 50;

background-color: rgba(28, 27, 33, 0.9);

backdrop-filter: blur(10px);

transition: background-color 0.5s ease;

}

.light .main-header {

background-color: rgba(255, 255, 255, 0.9);

}

/\* Content offset for fixed header \*/

main {

padding-top: 80px;

}

/\* Notification styles \*/

.notification-dropdown {

max-height: 70vh;

overflow-y: auto;

width: 350px;

right: 0;

left: auto;

}

.notification-item.unread {

background-color: rgba(139, 92, 246, 0.1);

}

.notification-badge {

font-size: 0.7rem;

min-width: 18px;

height: 18px;

line-height: 1;

}

/\* Filter sidebar \*/

.filter-sidebar {

position: sticky;

top: 100px;

height: calc(100vh - 120px);

overflow-y: auto;

}

/\* Car item animation \*/

.car-item {

transition: all 0.3s ease;

}

.car-item:hover {

transform: translateY(-5px);

box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);

}

.light .bg-white {

background-color: var(--tw-bg-opacity);

background-color: rgba(249, 250, 251, var(--tw-bg-opacity));

}

.light .bg-gradient-to-r.from-blue-50.to-indigo-50 {

background-image: linear-gradient(to right, #f0f9ff, #eef2ff);

}

.light .text-gray-800 {

color: #1f2937;

}

.light .text-gray-600 {

color: #4b5563;

}

.light .border-gray-200 {

border-color: #e5e7eb;

}

.light .bg-gray-100 {

background-color: #f3f4f6;

}

.light .bg-blue-100 {

background-color: #dbeafe;

}

.light .bg-green-100 {

background-color: #d1fae5;

}

.light .bg-yellow-100 {

background-color: #fef3c7;

}

.light .bg-red-100 {

background-color: #fee2e2;

}

.light .bg-purple-100 {

background-color: #ede9fe;

}

.light .divide-gray-200 {

border-color: #e5e7eb;

}

.light .hover\:bg-gray-50:hover {

background-color: #f9fafb;

}

</style>

</head>

<body class="bg-[#1C1B21] text-white transition-colors duration-300">

<div class="min-h-screen">

<header class="main-header">

<div class="container mx-auto px-4 py-4 flex justify-between items-center">

<a href="{{route('welcome')}}" class="flex items-center space-x-2">

<span class="text-2xl font-bold">

Форвард

</span>

<img src="{{asset("images/logo.png")}}" alt="" class="w-8 h-8 animate-pulse-slow">

<span class="text-2xl font-bold text-purple-600">

Авто

</span>

</a>

<div class="flex items-center space-x-6">

<nav id="menu" class="hidden md:flex md:space-x-6 items-center">

<a class="px-4 py-2 rounded-lg hover:bg-purple-600/20 transition-all" href="{{route('catalog')}}">Все авто</a>

<a class="px-4 py-2 rounded-lg hover:bg-purple-600/20 transition-all" href="{{route('dashboard')}}">Личный кабинет</a>

@auth

@if(auth()->user()->isManager())

<a class="px-4 py-2 rounded-lg hover:bg-purple-600/20 transition-all" href="{{route('manager.bookings.index')}}">Менеджер панель</a>

@endif

@if(auth()->user()->isAdmin())

<a class="px-4 py-2 rounded-lg hover:bg-purple-600/20 transition-all" href="{{route('admin.car-structure.index')}}">Админ панель</a>

@endif

@if(auth()->user()->isSuperAdmin())

<a class="px-4 py-2 rounded-lg hover:bg-purple-600/20 transition-all" href="{{route('admin.admins.index')}}">Админ панель</a>

@endif

@endauth

</nav>

@auth

<div class="notification-icon relative">

<a href="#" id="notification-bell" class="text-gray-300 hover:text-white relative">

<i class="fas fa-bell text-xl"></i>

@if(auth()->user()->unreadNotifications()->count() > 0)

<span class="notification-badge absolute -top-1 -right-1 bg-red-500 text-white rounded-full flex items-center justify-center">

{{ auth()->user()->unreadNotifications()->count() }}

</span>

@endif

</a>

<div class="notification-dropdown hidden absolute mt-2 bg-[#2A2A2A] rounded-md shadow-lg z-50 border border-gray-700">

<div class="p-3 bg-[#2A2A2A] ">

<div class="flex justify-between items-center">

<h4 class="font-semibold">Уведомления</h4>

<a href="#" id="mark-all-read" class="text-xs text-purple-400 hover:text-purple-300">Отметить все как прочитанные</a>

</div>

</div>

<div class="notification-list max-h-96 overflow-y-auto">

@foreach(auth()->user()->notifications()->latest()->take(10)->get() as $notification)

<div class="notification-item px-3 py-3 border-b bg-[#2A2A2A] hover:bg-gray-700 cursor-pointer flex justify-between items-start {{ $notification->read\_at ? '' : 'unread' }}" data-id="{{ $notification->id }}">

<div>

<p class="text-sm">

{{ \Illuminate\Support\Str::limit($notification->message, 60, '...') }}

</p>

<small class="text-xs text-gray-400 block mt-1">

{{ $notification->created\_at->diffForHumans() }}

</small>

</div>

@if($notification->url)

<a href="{{ $notification->url }}" class="text-xs text-purple-400 hover:text-purple-300 ml-2">

<i class="fas fa-external-link-alt"></i>

</a>

@endif

</div>

@endforeach

@if(auth()->user()->notifications()->count() > 10)

<div class="p-3 text-center text-sm text-gray-400">

Показаны последние 10 уведомлений

</div>

@endif

@if(auth()->user()->notifications()->count() === 0)

<div class="p-6 text-center text-gray-400">

<i class="far fa-bell-slash text-2xl mb-2"></i>

<p>Нет уведомлений</p>

</div>

@endif

</div>

<div class="p-3 border-t border-gray-700 text-center">

<a href="{{ route('notifications.index') }}" class="text-sm text-purple-400 hover:text-purple-300">Все уведомления</a>

</div>

</div>

</div>

@endauth

<div class="hidden md:block">

<button id="theme-toggle" class="text-gray-300 hover:text-white transition-colors">

<!-- Здесь будет только одна иконка - динамически меняется -->

<i id="desktop-theme-icon" class="fas fa-moon text-xl"></i>

</button>

</div>

<div class="md:hidden">

<button id="menu-toggle" class="text-white focus:outline-none">

<svg class="w-8 h-8" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16"></path>

</svg>

</button>

</div>

</div>

</div>

</header>

<!-- Mobile Menu -->

<div id="mobile-menu" class="hidden fixed inset-0 z-40 bg-black bg-opacity-75 md:hidden">

<div class="fixed inset-y-0 right-0 w-3/4 bg-[#2D2D2D] p-4 overflow-y-auto">

<div class="flex justify-end mb-6">

<button id="menu-close" class="text-white">

<svg class="w-8 h-8" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6 6l12 12"></path>

</svg>

</button>

</div>

<nav class="flex flex-col space-y-4">

<a class="px-4 py-3 rounded-lg hover:bg-purple-600/20 transition-all text-lg" href="{{route('catalog')}}">Все авто</a>

<a class="px-4 py-3 rounded-lg hover:bg-purple-600/20 transition-all text-lg" href="{{route('dashboard')}}">Личный кабинет</a>

<a class="px-4 py-3 rounded-lg hover:bg-purple-600/20 transition-all text-lg" href="#contact">Контакты</a>

@auth

<div class="mt-6 pt-6 border-t border-gray-700">

<a href="{{ route('notifications.index') }}" class="flex items-center px-4 py-3 rounded-lg hover:bg-purple-600/20 transition-all text-lg">

<span class="relative mr-3">

<i class="fas fa-bell"></i>

@if(auth()->user()->unreadNotifications()->count() > 0)

<span class="notification-badge absolute -top-1 -right-1 bg-red-500 text-white rounded-full flex items-center justify-center">

{{ auth()->user()->unreadNotifications()->count() }}

</span>

@endif

</span>

Уведомления

</a>

</div>

@endauth

</nav>

<div class="mt-8 pt-6 border-t border-gray-700">

<button id="mobile-theme-toggle" class="flex items-center justify-between w-full px-4 py-3 text-lg">

<span>Тема</span>

<span>

<!-- Здесь будет только одна иконка - динамически меняется -->

<i id="mobile-theme-icon" class="fas fa-moon "></i>

</span>

</button>

</div>

</div>

</div>

<!-- Main Content -->

<main class="text-current">

@yield('content')

</main>

<!-- Footer -->

<footer class="bg-[#2D2D2D] py-12">

<div class="container mx-auto px-4">

<div class="grid grid-cols-1 md:grid-cols-4 gap-8">

<div>

<div class="flex items-center space-x-2 mb-4">

<span class="text-xl font-bold">Форвард</span>

<img src="{{asset("images/logo.png")}}" alt="" class="w-6 h-6">

<span class="text-xl font-bold text-purple-600">Авто</span>

</div>

<p class="text-gray-400">Лучший автосалон в Иркутске. Мы предлагаем качественные автомобили по доступным ценам.</p>

</div>

<div>

<h4 class="text-lg font-semibold mb-4">Меню</h4>

<ul class="space-y-2">

<li><a href="{{route('catalog')}}" class="text-gray-400 hover:text-purple-400 transition-colors">Все авто</a></li>

<li><a href="{{route('dashboard')}}" class="text-gray-400 hover:text-purple-400 transition-colors">Личный кабинет</a></li>

<li><a href="#branches" class="text-gray-400 hover:text-purple-400 transition-colors">Филиалы</a></li>

<li><a href="#contact" class="text-gray-400 hover:text-purple-400 transition-colors">Контакты</a></li>

</ul>

</div>

<div>

<h4 class="text-lg font-semibold mb-4">Контакты</h4>

<ul class="space-y-2">

<li class="text-gray-400">г. Иркутск, ул. Ленина, 1</li>

<li class="text-gray-400">+7 (3952) 12-34-56</li>

<li class="text-gray-400">info@forward-auto.ru</li>

</ul>

</div>

<div>

</div>

</div>

</div>

<div class="border-t border-gray-700 mt-8 pt-8 text-center text-gray-400">

<p>© {{ date('Y') }} Forward Auto. Все права защищены.</p>

</div>

</div>

</footer>

</div>

<script>

document.addEventListener('DOMContentLoaded', function() {

// Theme switcher

const themeToggle = document.getElementById('theme-toggle');

const mobileThemeToggle = document.getElementById('mobile-theme-toggle');

const html = document.documentElement;

// Иконки темы (теперь у нас по одной иконке для каждого переключателя)

const desktopThemeIcon = document.getElementById('desktop-theme-icon');

const mobileThemeIcon = document.getElementById('mobile-theme-icon');

// Функция для обновления иконок темы

function updateThemeIcons() {

const isLight = html.classList.contains('light');

const iconClass = isLight ? 'fa-sun' : 'fa-moon';

// Обновляем иконку в десктопном переключателе

if (desktopThemeIcon) {

desktopThemeIcon.className = `fas ${iconClass} text-xl`;

}

// Обновляем иконку в мобильном переключателе

if (mobileThemeIcon) {

mobileThemeIcon.className = `fas ${iconClass} text-xl`;

}

}

// Проверяем сохраненную тему при загрузке

if (localStorage.getItem('theme') === 'light') {

html.classList.remove('dark');

html.classList.add('light');

} else {

// Убедимся, что темная тема установлена

html.classList.remove('light');

html.classList.add('dark');

}

// Инициализируем иконки

updateThemeIcons();

// Функция переключения темы

function toggleTheme() {

const isLight = html.classList.contains('light');

if (isLight) {

// Переключаем на темную тему

html.classList.remove('light');

html.classList.add('dark');

localStorage.setItem('theme', 'dark');

} else {

// Переключаем на светлую тему

html.classList.remove('dark');

html.classList.add('light');

localStorage.setItem('theme', 'light');

}

// Обновляем иконки после смены темы

updateThemeIcons();

}

// Назначаем обработчики

if (themeToggle) themeToggle.addEventListener('click', toggleTheme);

if (mobileThemeToggle) mobileThemeToggle.addEventListener('click', toggleTheme);

// Mobile menu toggle

const menuToggle = document.getElementById('menu-toggle');

const menuClose = document.getElementById('menu-close');

const mobileMenu = document.getElementById('mobile-menu');

if (menuToggle && menuClose && mobileMenu) {

// Переключение меню по кнопке меню

menuToggle.addEventListener('click', function () {

mobileMenu.classList.toggle('hidden');

document.body.style.overflow = mobileMenu.classList.contains('hidden') ? '' : 'hidden';

});

// Закрытие по кнопке "крестик"

menuClose.addEventListener('click', function () {

mobileMenu.classList.add('hidden');

document.body.style.overflow = '';

});

// Закрытие по клику вне меню

mobileMenu.addEventListener('click', function (e) {

if (e.target === mobileMenu) {

mobileMenu.classList.add('hidden');

document.body.style.overflow = '';

}

});

}

// Notification system

const bell = document.getElementById('notification-bell');

const dropdown = document.querySelector('.notification-dropdown');

if (bell && dropdown) {

// Функция для обновления уведомлений

async function updateNotifications() {

try {

const response = await fetch('/notifications');

const data = await response.json();

// Обновляем бейдж

updateBadge(data.unread\_count);

// Обновляем список уведомлений

renderNotifications(data.notifications);

} catch (error) {

console.error('Ошибка при обновлении уведомлений:', error);

}

}

// Функция для обновления бейджа

function updateBadge(count) {

let badge = document.querySelector('.notification-badge');

if (count > 0) {

if (!badge) {

badge = document.createElement('span');

badge.className = 'notification-badge absolute -top-1 -right-1 bg-red-500 text-white rounded-full flex items-center justify-center';

bell.appendChild(badge);

}

badge.textContent = count;

} else if (badge) {

badge.remove();

}

}

// Функция для отметки как прочитанного

async function markAsRead(id) {

try {

const response = await fetch(`/notifications/${id}/read`, {

method: 'POST',

headers: {

'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content,

'Content-Type': 'application/json'

}

});

const data = await response.json();

if (data.success) {

updateBadge(data.unread\_count);

document.querySelector(`.notification-item[data-id="${id}"]`).classList.remove('unread');

}

} catch (error) {

console.error('Ошибка при отметке уведомления:', error);

}

}

// Открытие/закрытие выпадающего списка

bell.addEventListener('click', function(e) {

e.preventDefault();

e.stopPropagation();

dropdown.classList.toggle('hidden');

});

// Закрытие при клике вне области

document.addEventListener('click', function(e) {

if (!bell.contains(e.target) && !dropdown.contains(e.target)) {

dropdown.classList.add('hidden');

}

});

// Обработка кликов на уведомления

dropdown.addEventListener('click', function(e) {

const item = e.target.closest('.notification-item');

if (item && item.classList.contains('unread')) {

markAsRead(item.dataset.id);

item.classList.remove('unread');

}

});

// Отметить все как прочитанные

document.getElementById('mark-all-read')?.addEventListener('click', async function(e) {

e.preventDefault();

try {

const response = await fetch('/notifications/read-all', {

method: 'POST',

headers: {

'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content,

'Content-Type': 'application/json'

}

});

const data = await response.json();

if (data.success) {

updateBadge(0);

document.querySelectorAll('.notification-item').forEach(item => {

item.classList.remove('unread');

});

}

} catch (error) {

console.error('Ошибка при отметке всех уведомлений:', error);

}

});

// Обновляем каждые 30 секунд

setInterval(updateNotifications, 1000);

// Первоначальная загрузка

updateNotifications();

}

// Smooth scrolling for anchor links

document.querySelectorAll('a[href^="#"]').forEach(anchor => {

anchor.addEventListener('click', function (e) {

e.preventDefault();

const targetId = this.getAttribute('href');

if (targetId === '#') return;

const targetElement = document.querySelector(targetId);

if (targetElement) {

targetElement.scrollIntoView({

behavior: 'smooth',

block: 'start'

});

// Close mobile menu if open

if (mobileMenu && !mobileMenu.classList.contains('hidden')) {

mobileMenu.classList.add('hidden');

document.body.style.overflow = '';

}

}

});

});

});

function renderNotifications(notifications) {

const list = document.querySelector('.notification-list');

if (!list) return;

list.innerHTML = '';

if (notifications.length === 0) {

list.innerHTML = `

<div class="p-6 text-center text-gray-400">

<i class="far fa-bell-slash text-2xl mb-2"></i>

<p>Нет уведомлений</p>

</div>

`;

return;

}

notifications.forEach(notification => {

const limitedMessage = notification.message.substring(0, 60) + (notification.message.length > 60 ? '...' : '');

const item = document.createElement('div');

item.className = `notification-item px-3 py-3 border-b bg-[#2A2A2A] hover:bg-gray-700 cursor-pointer flex justify-between items-start ${notification.read\_at ? '' : 'unread'}`;

item.setAttribute('data-id', notification.id);

item.innerHTML = `

<div>

<p class="text-sm">${limitedMessage}</p>

<small class="text-xs text-gray-400 block mt-1">${timeAgo(notification.created\_at)}</small>

</div>

${notification.url ? `<a href="${notification.url}" class="text-xs text-purple-400 hover:text-purple-300 ml-2"><i class="fas fa-external-link-alt"></i></a>` : ''}

`;

list.appendChild(item);

});

// Показываем "показаны последние 10"

if (notifications.length >= 10) {

const info = document.createElement('div');

info.className = 'p-3 text-center text-sm text-gray-400';

info.textContent = 'Показаны последние 10 уведомлений';

list.appendChild(info);

}

}

// Функция форматирования даты (простой пример)

function timeAgo(dateString) {

const date = new Date(dateString);

const seconds = Math.floor((new Date() - date) / 1000);

const intervals = [

{ limit: 3600, divisor: 60, suffix: 'минуту назад' },

{ limit: 86400, divisor: 3600, suffix: 'час назад' },

{ limit: 604800, divisor: 86400, suffix: 'день назад' },

{ limit: 2629743, divisor: 604800, suffix: 'неделю назад' },

];

const interval = intervals.find(i => seconds < i.limit);

if (interval) {

const count = Math.floor(seconds / interval.divisor);

return `${count} ${interval.suffix}`;

}

return 'более месяца назад';

}

</script>

@yield('scripts')

</body>

</html>

Приложение В код серверной части

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use App\Models\Brand;

use App\Models\CarModel;

use App\Models\Equipment;

use Illuminate\Http\Request;

use App\Models\Color;

use Illuminate\Support\Facades\Log;

use App\Models\Generation;

use App\Models\BodyType;

use App\Models\EngineType;

use App\Models\TransmissionType;

use App\Models\DriveType;

use App\Models\Country;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Validator;

use Zipper;

use Illuminate\Support\Facades\Storage;

use Illuminate\Support\Str;

use Illuminate\Support\Facades\File;

use Illuminate\Validation\ValidationException;

class EquipmentController extends Controller

{

public function index(Request $request)

{

$query = Equipment::with(['generation.carModel.brand']);

if ($request->filled('brand\_id')) {

$query->whereHas('generation.carModel', function ($q) use ($request) {

$q->where('brand\_id', $request->input('brand\_id'));

});

}

if ($request->filled('model\_id')) {

$query->whereHas('generation', function ($q) use ($request) {

$q->where('car\_model\_id', $request->input('model\_id'));

});

}

$colors = Color::with('equipments')->get();

$equipments = $query->paginate(10);

$brands = Brand::all();

$models = collect(); // по умолчанию пустой список

if ($request->filled('brand\_id')) {

$models = CarModel::where('brand\_id', $request->input('brand\_id'))->get();

}

return view('admin.equipments.index', compact('equipments', 'brands', 'models', 'colors'));

}

public function create()

{

$brands = Brand::all();

$generations = Generation::all();

$bodyTypes = BodyType::all();

$engineTypes = EngineType::all();

$transmissionTypes = TransmissionType::all();

$driveTypes = DriveType::all();

$countries = Country::all();

$colors = Color::all();

return view('admin.equipments.create', compact(

'brands', 'generations', 'bodyTypes', 'engineTypes',

'transmissionTypes', 'driveTypes', 'countries', 'colors'

));

}

public function store(Request $request)

{

if (!auth()->user()->isAdmin()) {

return back()->with('error', 'Доступ запрещён');

}

$data = $request->validate([

'generation\_id' => 'required|exists:generations,id',

'body\_type\_id' => 'required|exists:body\_types,id',

'engine\_type\_id' => 'required|exists:engine\_types,id',

'engine\_name' => 'nullable|string|max:50',

'engine\_volume' => 'nullable|numeric|min:0.8|max:8.0',

'engine\_power' => 'nullable|integer|min:40|max:2000',

'transmission\_type\_id' => 'required|exists:transmission\_types,id',

'transmission\_name' => 'nullable|string|max:50',

'drive\_type\_id' => 'required|exists:drive\_types,id',

'country\_id' => 'required|exists:countries,id',

'description' => 'nullable|string|max:1000',

'weight' => 'nullable|numeric|min:500|max:10000',

'load\_capacity' => 'nullable|integer|min:0|max:3000',

'seats' => 'nullable|integer|min:2|max:9',

'fuel\_consumption' => 'nullable|numeric|min:0.1|max:50',

'fuel\_tank\_volume' => 'nullable|integer|min:30|max:200',

'battery\_capacity' => 'nullable|integer|min:10|max:200',

'range' => 'nullable|integer|min:50|max:1000',

'max\_speed' => 'nullable|numeric|min:50|max:450',

'clearance' => 'nullable|numeric|min:100|max:400',

'model\_folder' => 'nullable|file|mimes:zip|max:51200',

// Цвета

'new\_colors' => 'array|nullable',

'new\_colors.\*.name' => 'required\_with:new\_colors.\*.hex|string|max:255',

'new\_colors.\*.hex' => [

'required\_with:new\_colors.\*.name',

'regex:/^#([A-Fa-f0-9]{6})$/i'

],

]);

DB::beginTransaction();

try {

$equipment = Equipment::create($data);

if ($request->hasFile('model\_folder')) {

$this->process3dModel($request->file('model\_folder'), $equipment);

}

$this->processColors($request, $equipment);

DB::commit();

return redirect()->route('admin.equipments.index')

->with('success', 'Комплектация добавлена');

} catch (\Exception $e) {

DB::rollBack();

\Log::error('Ошибка при создании комплектации: ' . $e->getMessage());

return back()->withInput()

->with('error', 'Ошибка: ' . $e->getMessage());

}

}

protected function process3dModel($file, $equipment)

{

// Явные пути с обратными слешами

$tempDir = str\_replace('/', '\\', storage\_path('app\\temp'));

$publicDir = str\_replace('/', '\\', storage\_path('app\\public'));

if (!File::exists($tempDir)) {

File::makeDirectory($tempDir, 0755, true);

}

$tempName = 'model\_'.$equipment->id.'\_'.time().'.zip';

$tempPath = $tempDir.'\\'.$tempName;

file\_put\_contents($tempPath, file\_get\_contents($file->getRealPath()));

$extractPath = $publicDir.'\\3d\_models\\equipment\_'.$equipment->id;

if (File::exists($extractPath)) {

File::deleteDirectory($extractPath);

}

File::makeDirectory($extractPath, 0755, true);

$zip = new \ZipArchive;

if ($zip->open($tempPath) !== true) {

unlink($tempPath);

throw new \Exception("Ошибка открытия ZIP");

}

$zip->extractTo($extractPath);

$zip->close();

unlink($tempPath);

if (!File::exists($extractPath.'\\scene.gltf')) {

File::deleteDirectory($extractPath);

throw new \Exception("Не найден scene.gltf");

}

$equipment->update([

'model\_path' => '3d\_models/equipment\_'.$equipment->id

]);

}

protected function getZipErrorMessage($code)

{

$errors = [

\ZipArchive::ER\_EXISTS => 'Файл уже существует',

\ZipArchive::ER\_INCONS => 'Архив противоречив',

\ZipArchive::ER\_INVAL => 'Некорректный аргумент',

\ZipArchive::ER\_MEMORY => 'Ошибка выделения памяти',

\ZipArchive::ER\_NOENT => 'Нет такого файла',

\ZipArchive::ER\_NOZIP => 'Не ZIP архив',

\ZipArchive::ER\_OPEN => 'Не удалось открыть файл',

\ZipArchive::ER\_READ => 'Ошибка чтения',

\ZipArchive::ER\_SEEK => 'Ошибка поиска',

];

return $errors[$code] ?? 'Неизвестная ошибка';

}

protected function processColors($request, $equipment)

{

if ($request->has('colors')) {

$equipment->colors()->sync($request->input('colors'));

}

if ($request->has('new\_colors')) {

foreach ($request->input('new\_colors', []) as $colorData) {

if (!empty($colorData['name']) && !empty($colorData['hex'])) {

$color = Color::firstOrCreate(

['name' => $colorData['name']],

['hex\_code' => $colorData['hex']]

);

$equipment->colors()->attach($color);

}

}

}

}

public function edit(Equipment $equipment)

{

$brands = Brand::all();

$models = collect(); // по умолчанию пустой список

if ($equipment->generation && $equipment->generation->carModel && $equipment->generation->carModel->brand) {

$models = $equipment->generation->carModel->brand->carModels;

}

$generations = collect(); // по умолчанию пустой список

if ($equipment->generation && $equipment->generation->carModel) {

$generations = $equipment->generation->carModel->generations;

}

$colors = Color::all();

// Вернуть представление с данными

return view('admin.equipments.edit', compact(

'equipment',

'brands',

'models',

'generations',

'colors'

));

}

public function update(Request $request, Equipment $equipment)

{

$data = $request->validate([

'generation\_id' => 'required|exists:generations,id',

'body\_type\_id' => 'required|exists:body\_types,id',

'engine\_type\_id' => 'required|exists:engine\_types,id',

'engine\_name' => 'nullable|string|max:50',

'engine\_volume' => 'nullable|numeric|min:0.8|max:8.0',

'engine\_power' => 'nullable|integer|min:40|max:2000',

'transmission\_type\_id' => 'required|exists:transmission\_types,id',

'transmission\_name' => 'nullable|string|max:50',

'drive\_type\_id' => 'required|exists:drive\_types,id',

'country\_id' => 'required|exists:countries,id',

'description' => 'nullable|string',

'weight' => 'nullable|numeric|min:500|max:10000',

'load\_capacity' => 'nullable|integer|min:0|max:3000',

'seats' => 'nullable|integer|min:2|max:9',

'fuel\_consumption' => 'nullable|numeric|min:0.1|max:50',

'fuel\_tank\_volume' => 'nullable|integer|min:30|max:200',

'battery\_capacity' => 'nullable|integer|min:10|max:200',

'range' => 'nullable|integer|min:50|max:1000',

'max\_speed' => 'nullable|numeric|min:50|max:450',

'clearance' => 'nullable|numeric|min:100|max:400',

'model\_folder' => 'nullable|file|mimes:zip|max:51200', // 50MB

'remove\_model' => 'nullable|boolean',

// Валидация новых цветов

'new\_colors' => 'array|nullable',

'new\_colors.\*.name' => 'required\_with:new\_colors.\*.hex|string|max:255',

'new\_colors.\*.hex' => [

'required\_with:new\_colors.\*.name',

'regex:/^#([A-Fa-f0-9]{6})$/i'

],

]);

DB::beginTransaction();

try {

if ($request->has('remove\_model')) {

$this->delete3dModel($equipment);

}

if ($request->hasFile('model\_folder')) {

$this->process3dModel($request->file('model\_folder'), $equipment);

}

$equipment->update($request->except(['colors', 'new\_colors', 'model\_folder', 'remove\_model']));

if ($request->has('new\_colors')) {

foreach ($request->input('new\_colors', []) as $colorData) {

if (!empty($colorData['name']) && !empty($colorData['hex'])) {

$color = Color::firstOrCreate(

['name' => $colorData['name']],

['hex\_code' => $colorData['hex']]

);

$equipment->colors()->attach($color);

}

}

}

DB::commit();

return redirect()

->route('admin.equipments.index')

->with('success', 'Комплектация обновлена');

} catch (\Exception $e) {

DB::rollBack();

\Log::error('Equipment update failed', [

'error' => $e->getMessage(),

'trace' => $e->getTraceAsString()

]);

return back()->withInput()

->with('error', 'Ошибка: '.$e->getMessage());

}

}

protected function delete3dModel(Equipment $equipment)

{

if ($equipment->model\_path) {

$path = storage\_path('app/public/' . $equipment->model\_path);

if (File::exists($path)) {

File::deleteDirectory($path);

}

$equipment->update(['model\_path' => null]);

}

}

public function detachColor(Equipment $equipment, Color $color)

{

$equipment->colors()->detach($color);

return back()->with('success', 'Цвет удален');

}

public function destroy(Equipment $equipment)

{

$equipment->delete();

return back()->with('success', 'Комплектация удалена');

}

public function updateColor(Request $request)

{

$validator = Validator::make($request->only('name', 'hex\_code', 'color\_id'), [

'name' => 'required|string|max:50',

'hex\_code' => ['required', 'regex:/^#([A-Fa-f0-9]{6})$/'],

'color\_id' => 'required|exists:colors,id',

]);

if ($validator->fails()) {

return back()

->withErrors(['colorErrors' => $validator->errors()->all()])

->withInput();

}

$color = Color::findOrFail($request->input('color\_id'));

$color->update([

'name' => $request->input('name'),

'hex\_code' => $request->input('hex\_code'),

]);

return back()->with('success', 'Цвет успешно обновлён');

}

public function deleteColor(Request $request)

{

$request->validate([

'color\_id' => 'required|exists:colors,id',

]);

$color = Color::findOrFail($request->input('color\_id'));

$color->delete();

return back()->with('success', 'Цвет успешно удалён');

}

}