

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01 «Информатика и технологии программирования»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту по дисциплине «ИГИ»**

на тему: «Автоматизированная система учета заявок СТО»

Исполнитель: студент гр. ИП-32
Пархоменко П.Л

Руководитель: преподаватель
Процкая М.А.

Дата проверки: _____
Дата допуска к защите: _____
Дата защиты: _____
Оценка работы: _____

Подписи членов комиссии
по защите курсовой работы: _____

Гомель 2022

СОДЕРЖАНИЕ

Введение	3
1 Аналитический обзор.....	4
1.1 Обзор аналогов	4
1.2 Обзор технологий	6
1.3 Постановка задачи	7
2 Проектирование приложения.....	9
2.1 Пользовательские роли и функции.....	9
2.2 Используемые средства	12
2.3 Описание модели данных	15
2.4 Архитектура приложения	16
3 Реализация приложения.....	17
3.1 Описание моделей базы данных	17
3.2 Описание интерфейса.....	19
4 Тестирование	28
4.1 Тестирование пользовательского интерфейса.....	28
4.2 Тестирование алгоритмов решения	28
Заключение	33
Список используемых источников	34
Приложение А.....	35

ВВЕДЕНИЕ

Для успешного функционирования станции технического обслуживания в современных условиях рынка ему требуется наличие развитой информационной системы, которая осуществляла бы автоматизированную работу автосервиса с возможностью хранения информации, ее постоянной обработки для проведения необходимых расчетов и выдачи требуемых результатов и сведений.

За последние десять лет количество автомобилей увеличилось в два раза и в настоящее время состоит более чем из трёх млн. машин. Ежегодно этот показатель повышается на 8-8,5%. По данным Департамента автомобильного транспорта РБ, в Беларуси насчитывается более 3 млн. машин. В свою очередь данная тенденция привела к значительному увеличению числа потенциальных потребителей услуг СТО.

Ручной учет и контроль всех сведений об автовладельцах, зарегистрированных как постоянные клиенты, представляется совершенно неэффективным, особенно в связи с все время увеличивающимся потоком новых. Длительное обслуживание каждого автовладельца заставляет клиентов подолгу засиживаться в очередях либо вовсе отказаться от услуг данного автосервиса.

Таким образом, существует насущная потребность в автоматизации работы автосервиса с разработкой программного обеспечения для ее эффективного функционирования, которое позволило бы усовершенствовать анализ сведений по каждому автомобилю.

Целью курсового проекта является разработка программного обеспечения для автоматизации и учета заказов на станции технического обслуживания.

Применение автосервисом данного программного обеспечения позволит уменьшить издержки на ручной труд, автоматизировать документооборот, оптимизировать бизнес-процессы, повысить качество работы с клиентами.

В представленной работе разрабатывается программное обеспечение, предназначенное для автоматизации учета и расчетов с клиентами, пользующимися услугами автосервиса.

1 АНАЛИТИЧЕСКИЙ ОБЗОР

1.1 Обзор аналогов

Сегодня в стране насчитывается большое количество различных станций технического обслуживания. Многие из них отличаются по степени обслуживания (марки авто, сложность работ и т.п.), но всех их объединяет общедоступное приложение, с помощью которого их клиенты могут с легкостью записаться на диагностику или полноценный осмотр своего автомобиля.

В данном подразделе будут рассмотрены существующие сайты станций технического обслуживания. Все сайты имеют свои особенности. Каждый пользователь выбирает приложение, которое наиболее удобно для него.

«Autoliga» на протяжении 14 лет уверенно занимает лидирующие позиции среди станций техобслуживания города Минска, что объясняется безусловной клиент ориентированностью, качеством и оперативностью производимых работ, а также сотрудничеством исключительно с квалифицированными мастерами. Они занимаемся ремонтом автомобилей следующих марок: «BMW», «Audi», «Nissan», «Лексус», «Fiat» и многих других.

На рисунке 1.1 представлена главная страница сайта кампания

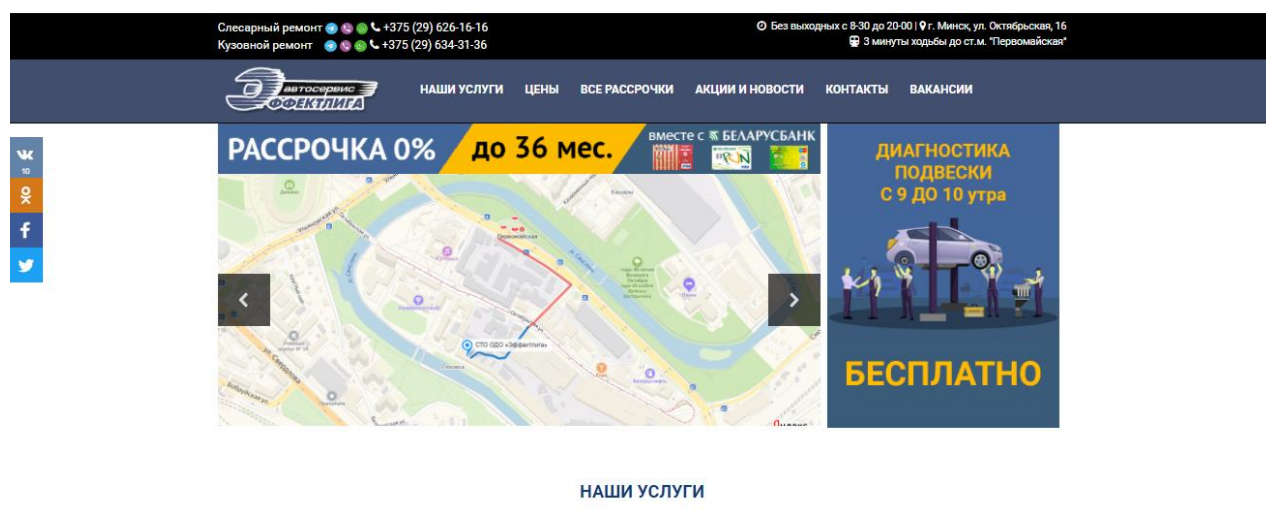


Рисунок 1.1 – Главная страница сайта *Autoliga*

На главной странице представлено удобное навигационное меню. В шапке сайте пользователи могут найти контакты для связи с менеджером СТО. После карты с расположением филиалов СТО и пути, пользователи могут ознакомиться с услугами, предоставляемыми данной СТО.

СТО «650услуг» - один из лучших сервисов в городе Гомель. Данный сервис представляет полный спектр услуг: в автосервисе также можно приобрести запчасти для иномарок и отечественных автомобилей. В СТО «650услуг» работают профессионалы с большим опытом работы, которые используют профессиональное оборудование и линейки материалов от ведущих мировых производителей. Отличным бонусом является то, что данный автосервис каждый месяц предоставляет акции и скидки на все виды услуг. Благодаря честным ценам и быстрой работы, данный сервис имеет исключительно положительные отзывы среди клиентов.

На рисунке 1.2 представлена главная страница сайта компании.

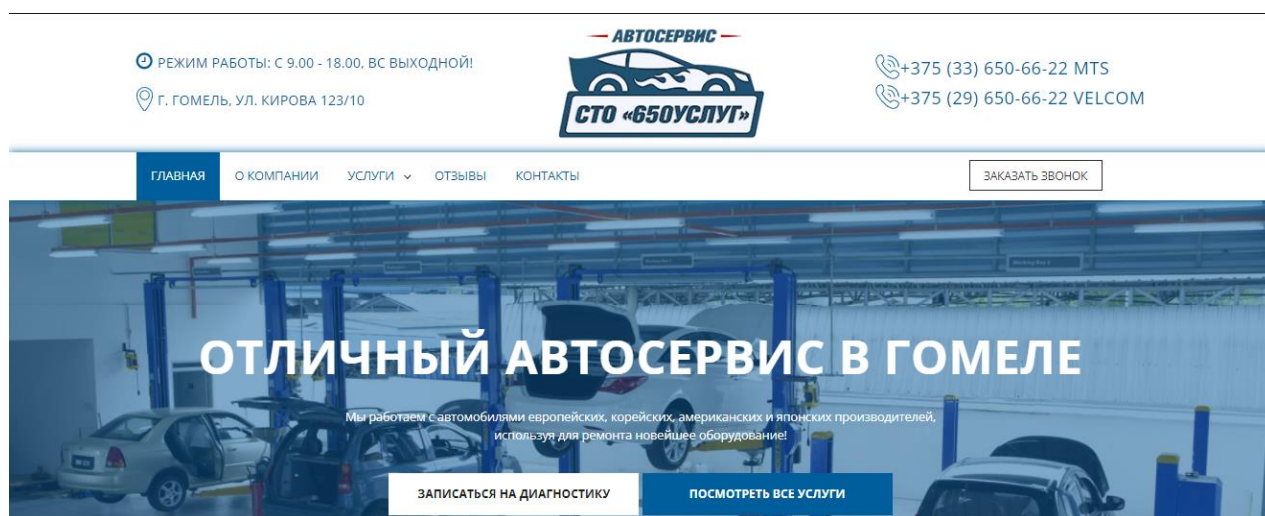


Рисунок 1.2 – Главная страница сайта «650услуг»

На главной странице по бокам от логотипа можно увидеть расписание и контакты компании. Также имеется навигационное меню. Из плюсов можно выделить большие кнопки «записаться на диагностику» и «посмотреть все услуги».

Автосервис «RemSto» является еще одним ведущим сервисом нашей страны. Этот автосервис за четырнадцать лет на рынке обслужил более двадцати пяти тысяч автомобилей. Данный автосервис направлен на обслуживание более ста двадцати марок автомобилей и имеет в штате сто тринадцать сотрудников.

На рисунке 1.3 представлена главная страница сайта компания

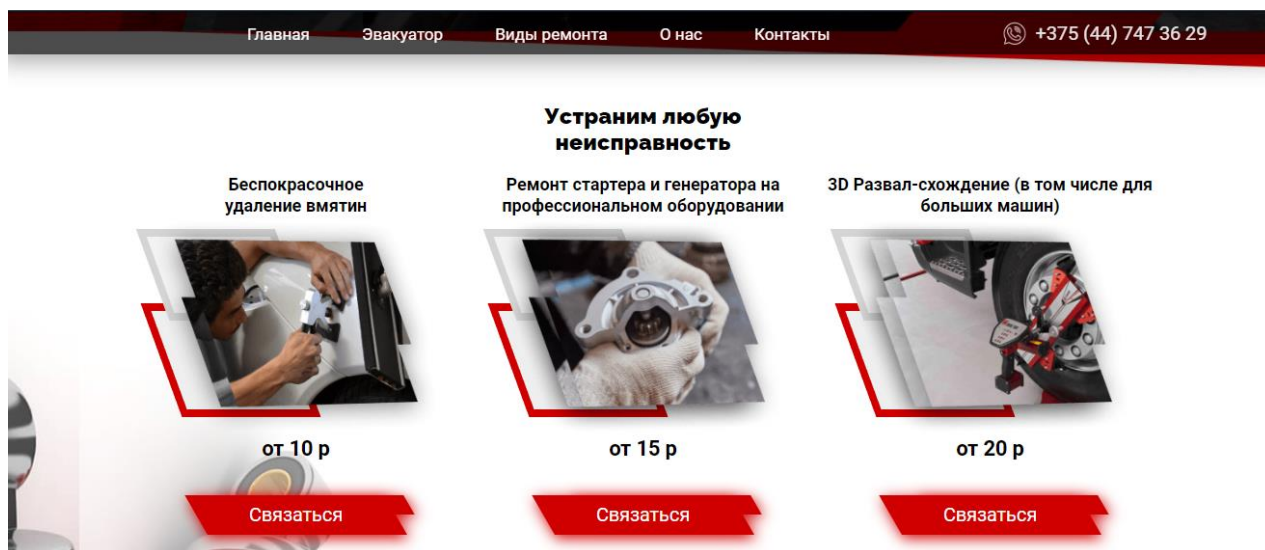


Рисунок 1.3 – Главная страница сайта *RemSto*

1.2 Обзор технологий

Прежде чем мы проведем обзор технологий, давайте разберемся, что такое веб-приложение и из чего оно состоит. Веб-приложение – это сайт с элементами интерактива. Они позволяют пользователям взаимодействовать с элементами на странице: нажимать кнопки, заполнять формы, запрашивать прайс, совершать покупки. Почтовые клиенты, соцсети, поисковики, интернет-магазины, программы для управления проектами – это всё примеры таких приложений.

Согласно [1], с точки зрения архитектуры веб-приложения состоят из двух частей: клиентской и серверной. Клиентская часть также называется фронтэнд. По сути это то, что пользователи видят на экране устройства. Основные технологии:

- *HTML* – это стандартный язык разметки, который применяют для создания веб-проектов. Его элементы позволяют отображать стандартные блоки страниц, а также представляют форматированный текст, изображения, таблицы и формы ввода данных;

- каскадные таблицы стилей (*CSS*) – это язык разметки, который определяет оформление и макет элементов *HTML*. Таким образом, *HTML* задаёт структуру, а *CSS* – стиль. С помощью *CSS* задаются шрифты, цвета, стили, расположение отдельных элементов, а также отображение страниц на разных устройствах;

– *JavaScript* – это язык программирования, который помогает реализовывать сложное поведение веб-страницы;

– *ASP.NET* – технология создания веб-приложений и веб-сервисов от компании Microsoft. Она является составной частью платформы *Microsoft.NET* и развитием более старой технологии *Microsoft ASP*.

Популярные Фреймворки и библиотеки *JavaScript*:

– *React* – это библиотека с открытым кодом для создания пользовательских интерфейсов. С помощью *React* разработчики создают веб-приложения, которые изменяют отображение без перезагрузки страницы;

– *Angular* – это фреймворк от компании *Google*. Прежде всего он нацелен на разработку *SPA*-решений;

– *Vue* – это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, *Vue* подходит для постепенного внедрения. Он легко интегрируется с другими библиотеками и существующими проектами.

В *web*-приложениях также важна серверная часть. Под серверной частью понимают набор аппаратно-программных средств, с помощью которых реализована логика работы приложения. Это то, что происходит вне браузера и компьютера пользователя. К бэкенду относится панель администрирования, управление данными, логика их передачи по запросам фронтенда.

Задача серверной разработки — сделать так, чтобы ответ от сервера доходил до клиента и спроектированные блоки функционировали нужным образом. А также создать для заказчика удобную и безопасную среду для наполнения и обновления контента на сайте. Основные технологии:

– *Node.js* – кроссплатформенная среда, которая выполняет код *JavaScript* вне браузера. *Node.js* позволяет разработчикам использовать *JavaScript*, чтобы получить инструменты командной строки. На стороне сервера с его помощью можно запускать сценарии для обработки динамического содержимого веб-страницы, перед тем как она будет доступна в веб-браузере пользователя.

– *Express* – фреймворк *Node.js*. *Express* сам использует модуль *http*, но вместе с тем предоставляет ряд готовых абстракций, которые упрощают создание сервера и серверной логики, в частности, обработка отправленных форм, работа с куками, *CORS* и т.д.

1.3 Постановка задачи

В курсовой работе необходимо разработать программное обеспечение (*web*-приложение) для автоматизации станции технического обслуживания.

Функционал программы состоит из четырех ролей: гостя, пользователя, мастера и администратора.

Функционал пользователя должен содержать:

- просмотр цен и услуг;
- возможность авторизоваться;
- просмотр своих заказов;
- просмотр отзывов;
- возможность добавить отзыв;
- возможность сделать заказ;
- оплата заказа.

Функционал гостя должен содержать:

- просмотр цен и услуг;
- просмотр общей информации;
- просмотр отзывов;
- возможность зарегистрироваться.

Функционал администратора должен содержать:

- добавление новых услуг;
- возможность авторизоваться;
- удаление услуги;
- удаление отзывов;
- принять заказ или отказать.

Функционал мастера должен содержать:

- просмотр своих заказов;
- возможность авторизоваться;
- закрытие заказа;
- просмотр отзывов.

2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

2.1 Пользовательские роли и функции

Для того, чтобы создать качественное *web*-приложение в данной предметной области, нужно четко понимать, какие роли у нас будут и какой функционал им будет доступен. А для этого нужно знать, какие объекты попадают в предметную область проектируемой системы и какие логические связи между ними существуют. Для формирования такого понимания используются логические модели предметной области.

Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области.

Для стабильной работы программного обеспечения необходимо чёткое распределение на роли, на основе которых будут формироваться функции взаимодействия со внутренней средой приложения.

Актер, согласно [2] – множество логически связанных ролей в *UML*, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Актером может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности.

Прецеденты представляют действия, выполняемые системой в интересах актеров. Проще говоря, прецедент – это описание последовательности действий (или нескольких последовательностей), которые выполняются системой и производят для отдельного актера видимый результат. Один актер может использовать несколько элементов прецедентов, и наоборот, один прецедент может иметь несколько актеров, использующих его. Каждый прецедент задает определенный путь использования системы. Набор всех прецедентов определяет полные функциональные возможности системы.

На рисунках 2.1 – 2.4 представлены диаграммы претендентов для всех реализованных ролей в приложении.

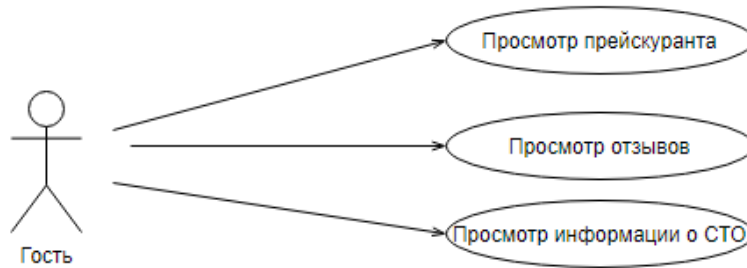


Рисунок 2.1 – Диаграмма актера «Гость» и его прецедентов

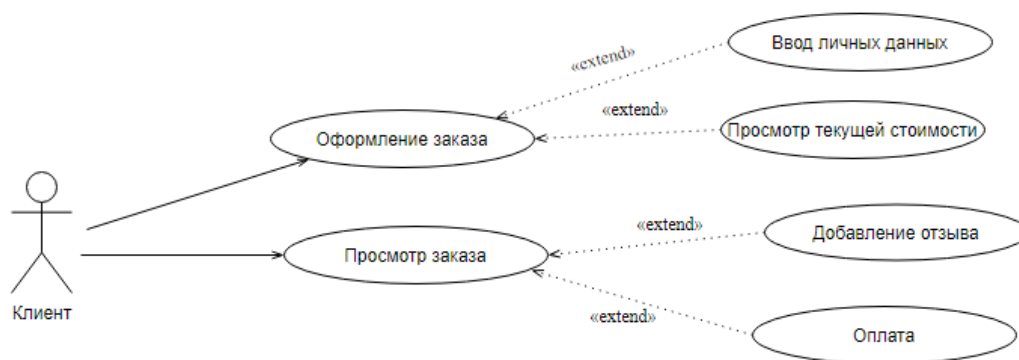


Рисунок 2.2 – Диаграмма актера «Клиент» и его прецедентов



Рисунок 2.3 – Диаграмма актера «Менеджер» и его прецедентов

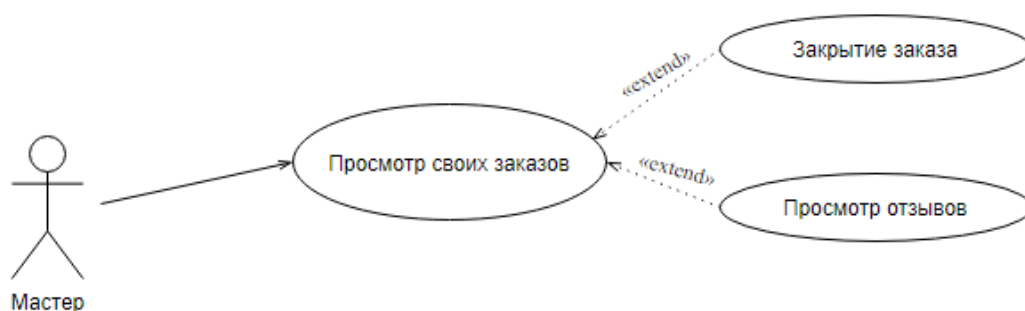


Рисунок 2.4 – Диаграмма актера «Мастер» и его прецедентов

Далее приведено описание актёров.

Актёр «Гость» – актёр, которому не доступны большинство функций. Единственное что он может – это просмотр справочника услуг и цен, также просмотр описания СТО и отзывов клиентов.

Актёр «Клиент» – это актёр, задачей которого является оформление заказа. Для удобства просмотра истории заказов данному уровню доступа доступен личный кабинет, где он может посмотреть подробности заказа, оставить отзыв и оплатить заказ.

Актёр «Менеджер» – актер, предназначенный для согласования заказа с клиентом, то есть основная задача которого является отказ, принятие и закрытие заказа. При принятии заказа он обязан указать фамилию актера «Мастер».

Актёр «Мастер» – это актер, предназначенный для выполнения того заказа, на который его назначил актер «Менеджер». В его полномочия входят: просмотр подробностей заказа, закрытие и просмотр отзывов.

Далее приведено описание прецедентов.

Прецедент «Просмотр прейскуранта» предназначен для ознакомления с услугами и ценами.

Прецедент «Просмотр отзывов» – это прецедент, предназначенный для просмотра отзывов, оставленных клиентами.

Прецедент «Просмотр информации о СТО» – это прецедент, предназначенный для просмотра общей информации о станции технического обслуживания, в который входят «3 причины выбрать наш автосервис» и «о нас».

Прецедент «Оформление заказа» служит для оформления заказа клиентом.

Прецедент «Ввод личных данных» предназначен для заполнения клиентом необходимых полей своими личными данными.

Прецедент «Просмотр текущей стоимости» – прецедент, необходимый для информирования клиента о стоимости услуги, которую он выбрал.

Прецедент «Просмотр заказа» предназначен для подробного просмотра выбранного заказа из общей таблицы.

Прецедент «Добавление отзыва» нужен для того, чтобы клиент смог оставить отзыв о проделанной работе.

Прецедент «Оплата» – это прецедент, предназначенный для оплаты заказа клиентом.

Прецедент «Назначение мастера» служит для установки соответствующего сотрудника на выбранный заказ.

Прецедент «Изменение статуса заказа» предназначен для предоставления актёру «Менеджер» возможности изменения состояния заказа: отказать в услуге, принять или закрыть.

Прецедент «Просмотр сотрудников» – это прецедент для просмотра данных о сотрудниках, зарегистрированных в системе.

Прецедент «Просмотр услуг» предназначен для просмотра оказываемых услуг.

Прецедент «Изменение услуги» вызывается актером «Менеджер» и служит для изменения всей или частичной информации об оказываемых услугах.

Прецедент «Статистика» – это прецедент, предназначенный для отображения статистики, генерируемой автоматически.

Прецедент «Объём услуг» практически полностью описывает прецедент «Статистика» за исключением того, что эти данные формируются на основании одного календарного месяца.

Прецедент «Просмотр своих заказов» служит для просмотра информации о заказах, которые выполняет сотрудник с данным *Id*.

Прецедент «Закрытие заказа» – это прецедент необходим для закрытия заказа непосредственно после его выполнения

2.2 Используемые средства

В качестве стека технологий для данной курсовой работы был выбран *MERN*. Согласно [3], *MERN* – это аббревиатура из следующих технологий:

- *MongoDb*;
- *Express.js*;
- *React.js*;
- *Node.js*.

MongoDB используется в качестве базы данных, *Node.js* и *Express.js* для серверной части, а *React.js* для создания клиентской части. Все эти технологии объединяет то, что в их основе используется язык *JavaScript*, что упрощает процесс разработки.

Согласно [4], *MongoDB* – это база данных *NoSQL*, которая ориентирована на использование документов и коллекций в отличие от использования строк и таблиц, как в стандартных базах данных. Базовой единицей данных в документах является пара ключ-значение.

Express – это фреймворк, созданный на базе платформы *Node.js*. Его преимущества значительно упростят и укоротят внутренний код. В стеке *Express* играет немаловажную роль, так как с его помощью настраивается маршрутизация и создается *API*-сервер для взаимодействия между клиентской и серверной частью приложения.

Node.js – среда выполнения, позволяющая запускать код *JavaScript* на сервере, благодаря чему фронтенд-разработчик может создавать легко интегрируемые приложения.

React – *JavaScript* библиотека. Согласно [5], она имеет следующие преимущества:

- эффективность. *React* хранит в памяти две версии виртуального *DOM* – обновленный виртуальный *DOM* и его резервную копию, созданную до обновления. После обновления *React* сравнивает обе версии между собой, чтобы найти измененные элементы, а затем – обновляет исключительно изменившуюся часть реального *DOM*. Подобный процесс на первый взгляд кажется пересложненным и трудоемким, однако он занимает гораздо меньше времени, чем обновление реальной объектной модели документа целиком, следовательно, он оптимизирует работу с *DOM*;

- высокая производительность. Одна из жизненно важных целей любого стартапа – написать веб-приложение быстрым и отзывчивым, обеспечить наилучшее обслуживание клиентов. Виртуальная *DOM* (рисунок 2.5), в отличие от реального *DOM*, занимает мало места и быстро обновляется, тем самым повышая производительность приложения. Виртуальная *DOM* позволяет странице немедленно получать ответы от сервера и отображать обновления. Например, *Facebook* применяет технологию виртуального *DOM* для обновления чатов и лент пользователей без перезагрузки страницы;

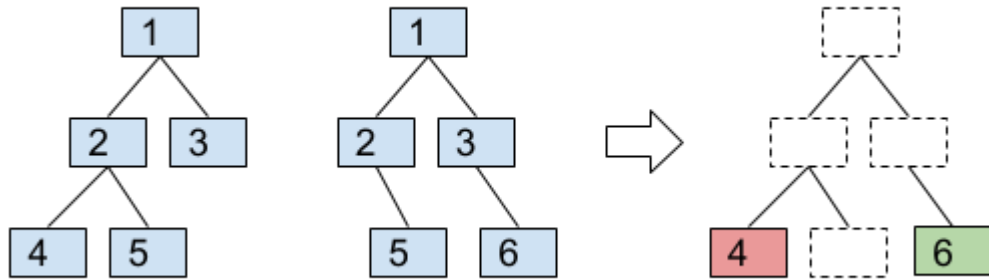


Рисунок 2.5 – Диаграмма виртуальной DOM

– повторное применение компонентов. При работе с *ReactJS* создаются многоразовые компоненты: чаще всего, компонент пользовательского интерфейса можно использовать в других частях кода или даже в разных проектах практически без изменений. Более того, разработчикам *React*-приложений доступны библиотеки готовых компонентов с открытым исходным кодом. Программирование на *React* сводится к тому, что мы разрабатываем различные компоненты, а затем встраиваем их в *DOM* (рисунок 2.6).

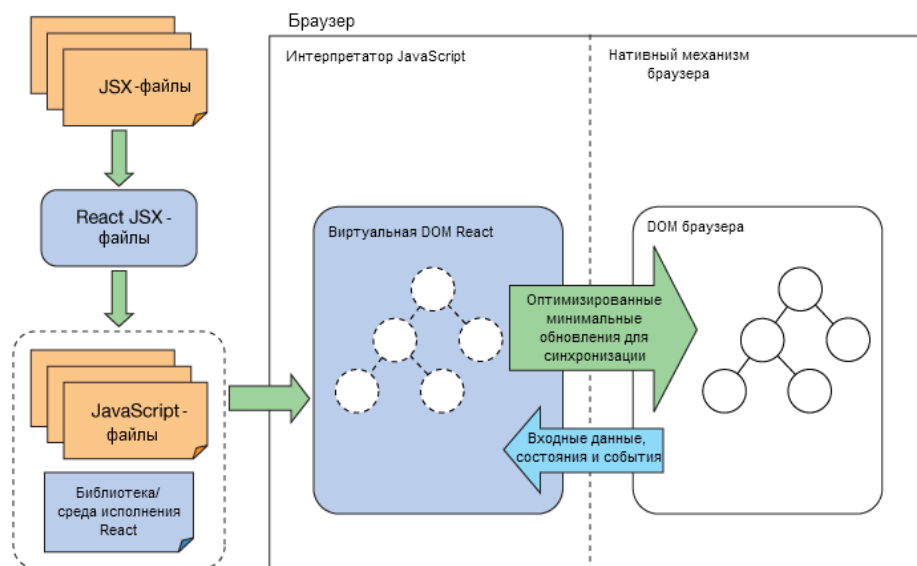


Рисунок 2.6 – Диаграмма приложения на *React*

MERN – это эффективный стек для разработки веб-приложений. Он удобный в использовании и лёгок в изучении.

2.3 Описание модели данных

В курсовой работе была применена документная модель данных. Документная модель – это СУБД, специально предназначенная для хранения иерархических структур данных (документов) и обычно реализуемая с помощью подхода *NoSQL*.

На рисунке 2.7 представлена схемы используемой базы данных.

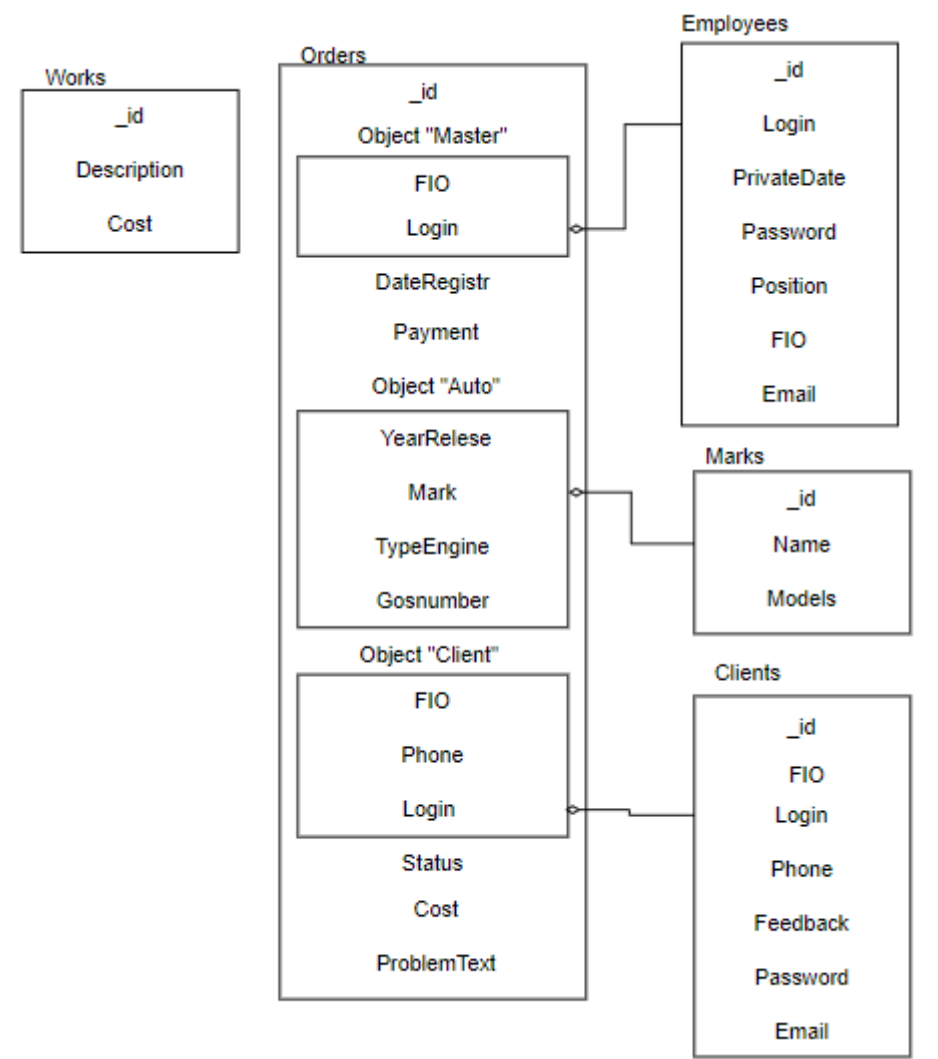


Рисунок 2.7 – Схема базы данных

Данная схема отражает модель базы данных, на которой мы можем заметить, что у нас имеется пять коллекций: *Clients*, *Employees*, *Marks*, *Orders*, *Works*. В коллекциях содержатся документы, которые в свою очередь содержат вложенные документы. Коллекция *Works* выступает в качестве справочника.

Документная база данных – это тип нереляционных баз данных, предназначенный для хранения и запроса данных в виде документов в формате, подобном *JSON*. Документные базы данных позволяют разработчикам хранить и запрашивать данные в БД с помощью той же документной модели, которую они используют в коде приложения. Гибкий, полуструктурированный, иерархический характер документов и документных баз данных позволяет им развиваться в соответствии с потребностями приложений.

2.4 Архитектура приложения

Как уже упоминалось в главе 2.2 в качестве стека технологий был выбран *MERN*. Данный стек предлагает следующую архитектуру (рисунок 2.8):

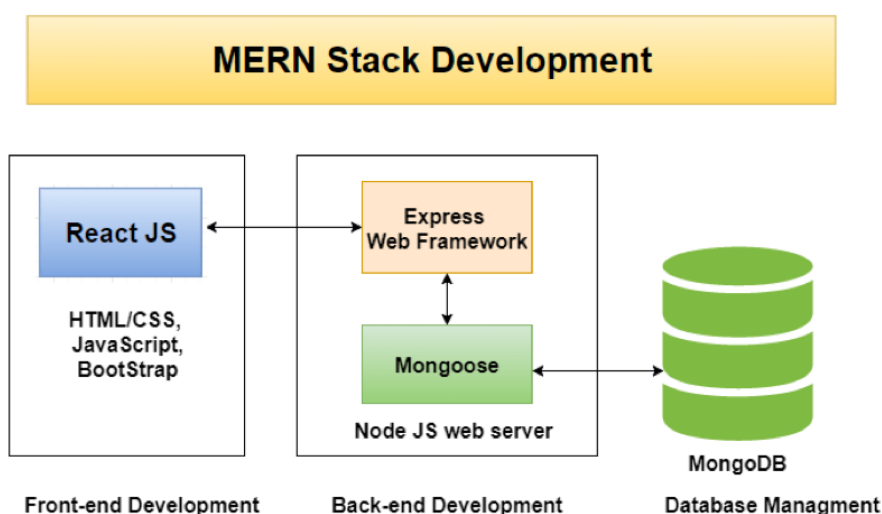


Рисунок 2.8 – Архитектура приложения с использованием стека *MERN*

На рисунке 2.8 можно выделить следующие уровни:

- уровень данных. Данный уровень хранит все данные приложения. Он представлен сервером;
- бизнес-уровень работает как посредник между уровнем данных и уровнем представления. Все данные проходят через бизнес-уровень перед тем, как их будет использовать уровень представления;
- уровень представления. Согласно [6], это уровень, на котором пользователи взаимодействуют с приложением. В данной курсовой работе он реализован с помощью библиотеки *React*. Уровень представления содержит отдельный код (код, который отвечает только за разметку, но не поведение).

3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1 Описание моделей базы данных

В приложение было создано 5 моделей, которые отражают документы из коллекций (схема представлена на рисунке 2.7).

В таблице 3.1 представлена модель *Client*.

Таблица 3.1 – Описание модели *Client*

Название	Тип	Обязательное (<i>required</i>)	Уникальное (<i>unique</i>)
<i>Feedback</i>	<i>String</i>	Нет	Нет
<i>Fio</i>	<i>String</i>	Да	Нет
<i>Login</i>	<i>String</i>	Да	Да
<i>Phone</i>	<i>String</i>	Да	Да
<i>Password</i>	<i>String</i>	Да	Нет
<i>Email</i>	<i>String</i>	Да	Да

В таблице 3.2 представлено описание модели *Employee*.

Таблица 3.2 – Описание модели *Employee*

Название	Тип	Обязательное (<i>required</i>)	Уникальное (<i>unique</i>)
<i>Fio</i>	<i>String</i>	Да	Нет
<i>Login</i>	<i>String</i>	Да	Да
<i>Privatedata</i>	<i>String</i>	Да	Нет
<i>Password</i>	<i>String</i>	Да	Нет
<i>Email</i>	<i>String</i>	Да	Да
<i>Position</i>	<i>String</i>	Да	Нет

В таблице 3.3 представлено описание модели *Mark*.

Таблица 3.3 – Описание модели *Mark*

Название	Тип	Обязательное (required)	Уникальное (unique)
<i>Name</i>	<i>String</i>	Да	Да
<i>Models</i>	<i>Array</i>	Да	—

В таблице 3.4 представлено описание модели *Order*.

Таблица 3.4 – Описание модели *Order*

Название	Тип	Обязательное (required)	Уникальное (unique)
<i>_id</i>	<i>objectid</i>	да	да
<i>auto</i>	<i>object</i>	да	нет
<i>auto.yearRelease</i>	<i>number</i>	да	нет
<i>auto.mark</i>	<i>string</i>	да	нет
<i>auto.typeengine</i>	<i>string</i>	да	нет
<i>auto.gosnumber</i>	<i>string</i>	да	да
<i>client</i>	<i>object</i>	да	нет
<i>client.fio</i>	<i>string</i>	да	нет
<i>client.phone</i>	<i>string</i>	да	нет
<i>client.login</i>	<i>string</i>	да	да
<i>dateRegistr</i>	<i>date</i>	да	нет
<i>master</i>	<i>object</i>	нет	нет
<i>master.fio</i>	<i>string</i>	нет	нет
<i>master.login</i>	<i>string</i>	нет	да
<i>master.email</i>	<i>string</i>	нет	да
<i>payment</i>	<i>boolean</i>	нет	нет
<i>status</i>	<i>string</i>	да	нет
<i>problemText</i>	<i>string</i>	да	нет
<i>cost</i>	<i>number</i>	нет	нет

В таблице 3.5 представлено описание модели *Work*.

Таблица 3.5 – Описание модели *Work*

Название	Тип	Обязательное (<i>required</i>)	Уникальное (<i>unique</i>)
<i>_id</i>	<i>objectid</i>	да	да
<i>cost</i>	<i>number</i>	да	нет
<i>description</i>	<i>string</i>	да	нет
<i>name</i>	<i>string</i>	да	да

3.2 Описание интерфейса

Интерфейс пользователя, он же пользовательский интерфейс (*UI*) — интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы. *UI* должен быть интуитивно понятным и простым, чтобы на дольше удерживать пользователя на странице.

3.2.1 При запуске программы потенциальный клиент видит главную страницу, часть которой приведен на рисунке 3.1.


+375 (25) 6768356
Сделать заказ
Услуги Отзывы Прейскурант Регистрация

СТО Гомель




Для автолюбителей в Гомеле 1AK.by предоставляет услуги СТО. Удобное расположение станции технического обслуживания на пр. Речицкий 135 позволяет клиентам осуществлять выгодные покупки в магазине «Первой аккумуляторной компании» и получать услуги по обслуживанию автомобиля.

СТО на пр. Речицкий 135 является официальной станцией G-EnergyService, сертифицированной ООО «Газпромнефть-СМ». Клиентам предоставляется широкий ассортимент оригинальных смазочных материалов G-Energy и Gazpromneft. Имеются также два поста для экспресс-замены масла.

На станции технического обслуживания в г. Гомель на пр. Речицкий 135 также оказывают услуги по ремонту подвески, шиномонтажу, развалу-схождению, компьютерной диагностике различных систем автомобиля.

Рисунок 3.1 – Главная страница


На данной странице пользователю доступны ссылки Услуги, Отзывы, Прейскурант и Регистрация. Ссылка «Прейскурант» приведет пользователя к таблице с названием, описанием и стоимостью услуг (рисунок 3.2).

 +375 (25) 6768356 Сделать заказ Услуги Отзывы Прейскурант Регистрация		
Прейскурант отпускных цен на услуги Service Station в г. Гомель		
Наименование	Описание	Стоимость
Двигатель	Диагностика двигателя, смена комплекта ГРМ, масляного фильтра, моторного масла, ремня генератора, прокладок ДВС масляного поддона и клапанной крышки, установка защиты двигателя, компьютерная диагностика ДВС.	1000
Рулевое управление	Замена жидкости гидроусилителя, ремонт рулевых тяг, рулевой рейки.	1200
Салон	Замена салонного фильтра, установка дополнительного оборудования.	1200
Система выхлопа	Замена глушителей	1200








***Примечание:** прейскурант ориентировочный, стоимость услуг уточняйте на СТО.

Рисунок 3.2 – Таблица прейскуранта


При нажатии на ссылку «Услуги» пользователь увидит перечень услуг. Ссылка «Отзывы» приведет пользователя к карточкам с отзывами и фамилиями клиентов (рисунок 3.3).


+375 (25) 6768356
Сделать заказ
Услуги Отзывы Прейскурант Регистрация


Гомеле:

-  продажа аккумуляторов,
-  моторных масел,
-  охлаждающих жидкостей,
-  тормозных жидкостей,
-  трансмиссионных масел,
-  автохимии,
-  автозапчастей.

Отзывы наших клиентов



Хоменков С.С.
Все отлично



Суглобов В.В.
Сервис очень хороший, рекомендую

Прейскурант отпускных цен на услуги Service Station в г. Гомель

Рисунок 3.3 – Карточки отзывов клиентов

Кнопка сделать заказ приведет незарегистрированного пользователя на страницу регистрации. Это же пользователь может сделать, нажав на ссылку «Регистрация». После откроется окно регистрации (рисунок 3.4).

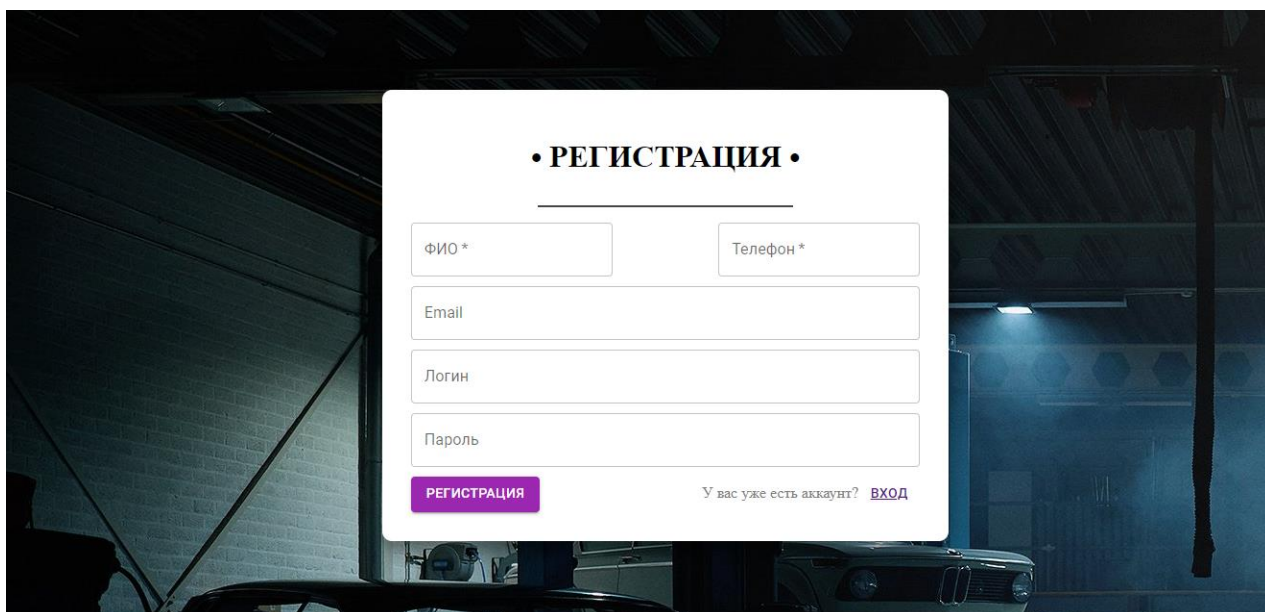


Рисунок 3.4 – Окно регистрации

В случае, если пользователь уже зарегистрирован, он может нажать на кнопку «Вход» и войти в систему под своей учетной записью (рисунок 3.5).

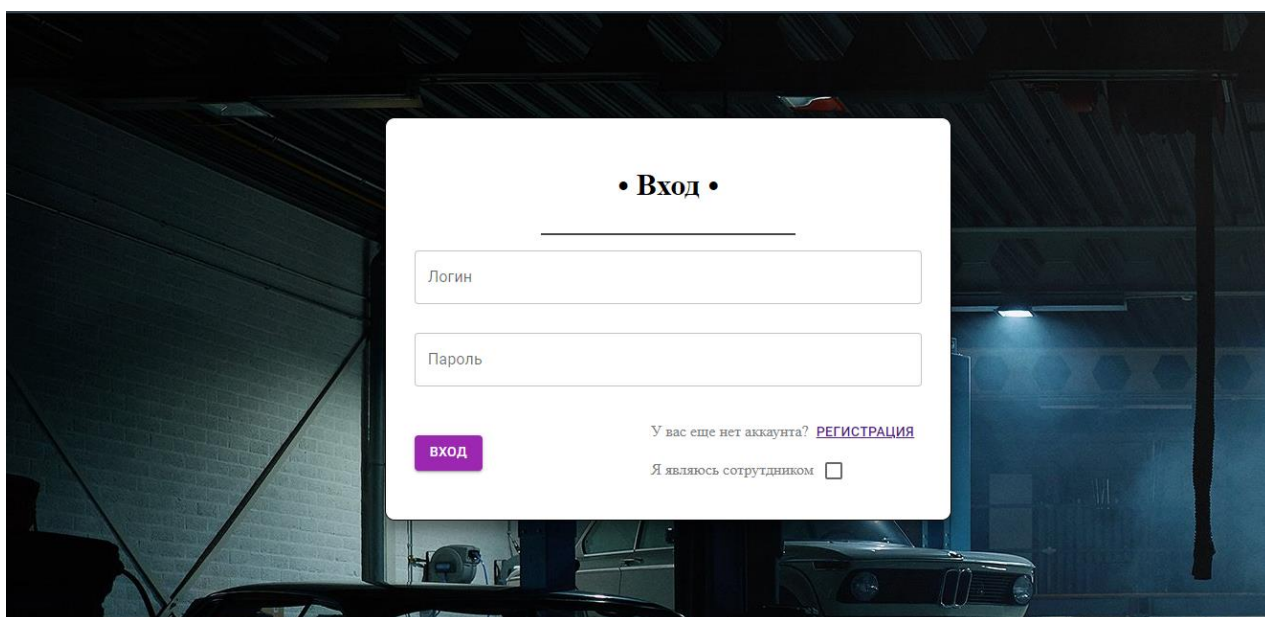


Рисунок 3.5 – Окно входа

После входа в систему пользователю предстает следующее окно, интерфейс которого приведен на рисунке 3.6.



Рисунок 3.6 – Главная страница в личном кабинете

На главной странице пользователь может просмотреть список своих заказов, отзывы и услуги, перейдя по соответствующим ссылкам.

Меню добавления и редактирования отзыва представлено на рисунке 3.7.

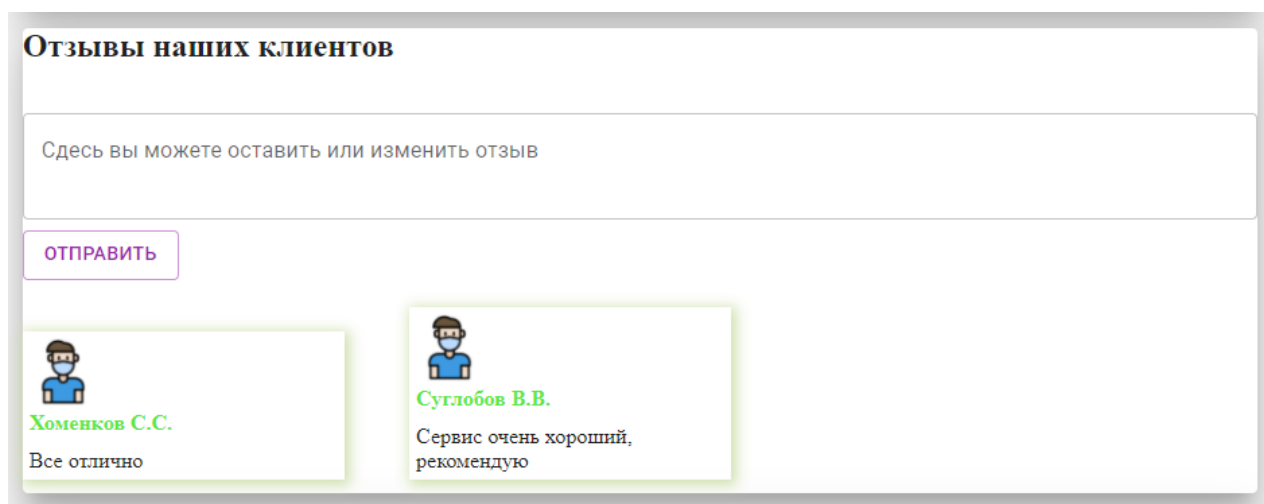


Рисунок 3.7 – Меню изменения отзыва клиента

Также в личном кабинете пользователю доступен прейскурант в виде таблице, аналогичный, как на рисунке 3.2.

В случае, если заказ новый и еще не был оплачен (пример такого заказа представлен на рисунок 3.8), пользователь может оплатить его, нажав на кнопку «Оплатить». Перед ним откроется соответствующее окно (рисунок 3.9)

Заказ от 12.04.2022, 01:30:36 №6254ac0c6df493425a892141		
Статус заказа: Принято		
Автомобиль	Клиент	Мастер
<ul style="list-style-type: none"> • Выпуск: 2020 • Марка: test • Двигатель: Diesel • Номер: 1234TR 	<ul style="list-style-type: none"> • Хоменков С.С. • +375257639813 	<ul style="list-style-type: none"> • Кирпиченко Д.Д • mr.dima.kirpichenko@gmail.com
sdasd xvх sdf dsgrfg		
Заказ не оплачен (к оплате)1000 руб. ОПЛАТИТЬ		

Рисунок 3.8 – Пример неоплаченного заказа

Оплата заказа № 6254ac0c6df493425a892141

Все данные защищены службой по защите данных

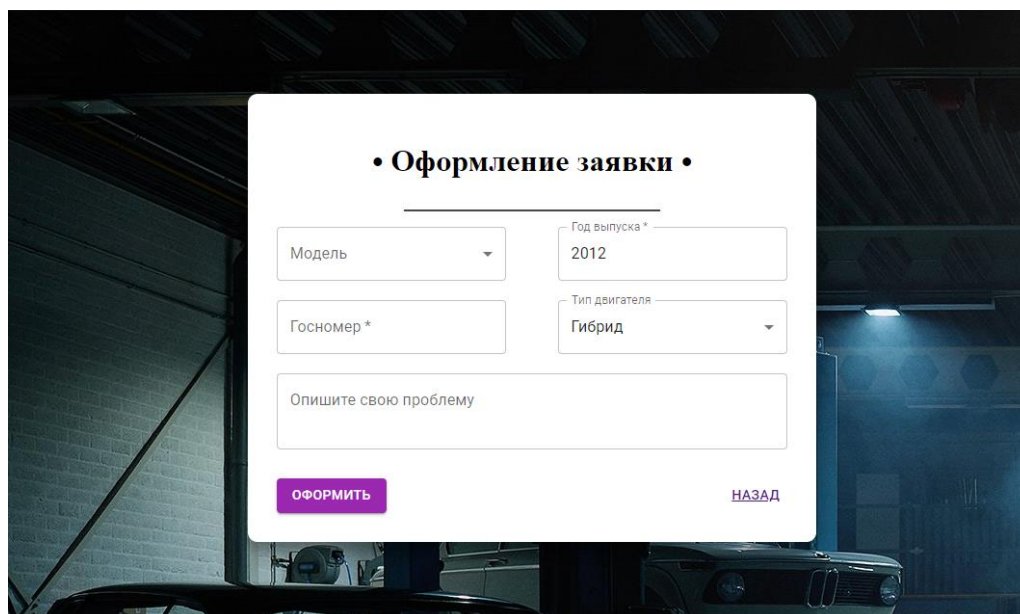
Номер карты

Год
Месяц
День

[ОТМЕНА](#) [ОПЛАТИТЬ](#)

Рисунок 3.9 – Окно оплаты заказа

В личном кабинете пользователь может сделать новый заказ. Для этого ему необходимо в шапке сайта или в навигационном меню нажать кнопку «Сделать заказ». Интерфейс окна представлен на рисунке 3.10.



• Оформление заявки •

Модель ▼

Госномер *

Год выпуска *
2012

Тип двигателя
Гибрид ▼

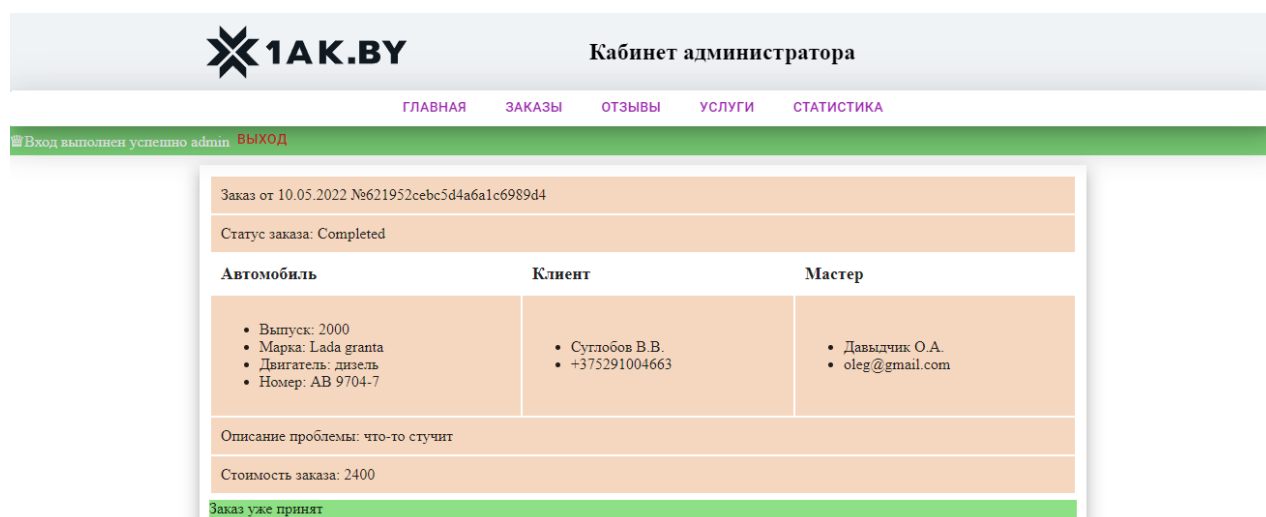
Опишите свою проблему

ОФОРМИТЬ
НАЗАД

Рисунок 3.10 – Окно заказа

После нажатия на кнопку «Оформить», новый заказ попадает в базу данных и будет доступен в личном кабинете.

3.2.2 При входе в аккаунт администратор попадает в свой личный кабинет, интерфейс которого представлен на рисунке 3.11.



1AK.BY Кабинет администратора

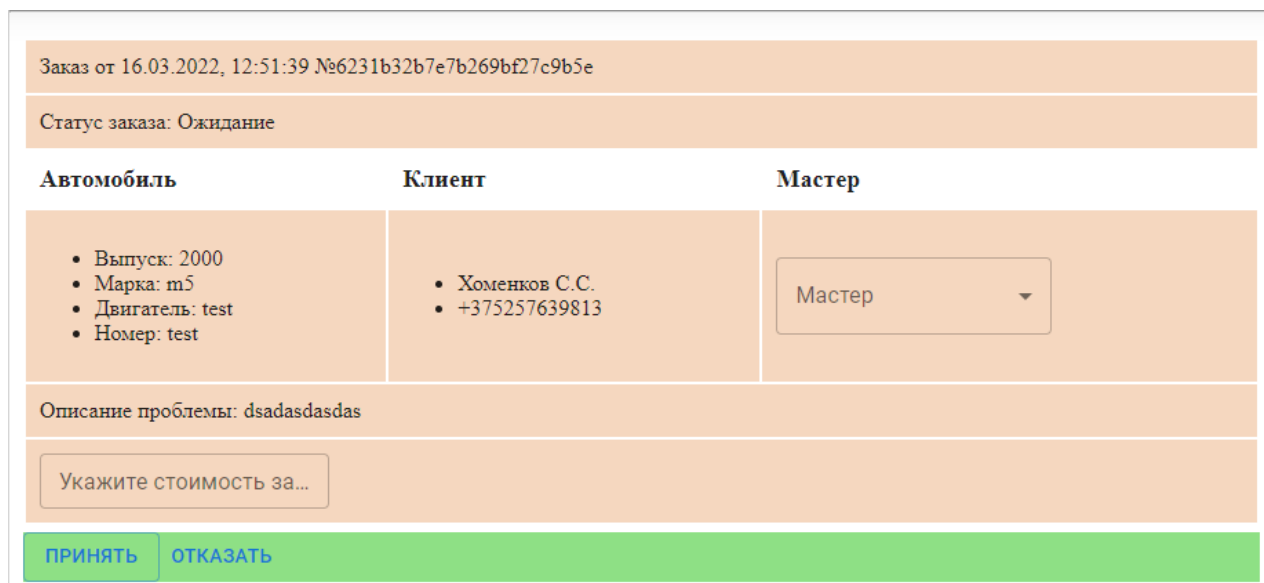
ГЛАВНАЯ ЗАКАЗЫ ОТЗЫВЫ УСЛУГИ СТАТИСТИКА

Вход выполнен успешно admin [Выход](#)

Заказ от 10.05.2022 №621952cebc5d4a6a1c6989d4		
Статус заказа: Completed		
Автомобиль	Клиент	Мастер
<ul style="list-style-type: none"> • Выпуск: 2000 • Марка: Lada granta • Двигатель: дизель • Номер: АВ 9704-7 	<ul style="list-style-type: none"> • Суглобов В.В. • +375291004663 	<ul style="list-style-type: none"> • Давыдчик О.А. • oleg@gmail.com
Описание проблемы: что-то стучит		
Стоимость заказа: 2400		
Заказ уже принят		

Рисунок 3.11 – Личный кабинет администратора

В личном кабинете администратору доступно навигационное меню. При нажатии на ссылку «Заказы», администратор перейдет к заказам, которые необходимо принять. Пример такого заказа представлен на рисунке 3.12.



Заказ от 16.03.2022, 12:51:39 №6231b32b7e7b269bf27c9b5e

Статус заказа: Ожидание

Автомобиль	Клиент	Мастер
<ul style="list-style-type: none"> • Выпуск: 2000 • Марка: m5 • Двигатель: test • Номер: test 	<ul style="list-style-type: none"> • Хоменков С.С. • +375257639813 	<div>Мастер ▼</div>

Описание проблемы: dsadasdasdas

Укажите стоимость за...

ПРИНЯТЬ **ОТКАЗАТЬ**

Рисунок 3.12 – Пример непринятого заказа

В данном окне администратор может назначить мастера и указать стоимость. Далее администратор должен принять заказ, нажав на кнопку «Принять», либо отказать.

Администратору также доступно несколько действий с услугами: добавить новую, удалить или отредактировать.

Для удаления услуги администратору необходимо нажать на иконку «X» рядом с потенциально удаляемой записью. Далее ему необходимо подтвердить свое действие (рисунок 3.13).

Для редактирования данных об услуге, администратору необходимо нажать на иконку «✎». Далее перед администратором появятся 3 поля для ввода, и кнопка «Сохранить изменения» (рисунок 3.14)

Добавить новую услугу

Прейскурант отпускных цен на услуги Service Station в г. Гомель

Наименование			Описание	Стоимость
		Двигатель	Диагностика двигателя, смена комплекта ГРМ, масляного фильтра, моторного масла, ремня генератора, прокладок ДВС масляного поддона и клапанной крышки, установка защиты двигателя, компьютерная диагностика ДВС.	1000
		Рулевое управление	Замена жидкости гидроусилителя, ремонт рулевых тяг, рулевой рейки.	1200
		Салон	Замена салонного фильтра, установка дополнительного оборудования.	1200
		Система выхлопа	Замена глушителей	1200

Рисунок 3.13 – Просмотр и добавление новой услуги

Вы действительно хотите удалить? **НЕТ** **УДАЛИТЬ**

Изменить данные

Рисунок 3.14 – Меню удаления и редактирования услуги

Так же у администратора есть возможность просмотреть все отзывы клиентов и удалить некорректный (рисунок 3.15)

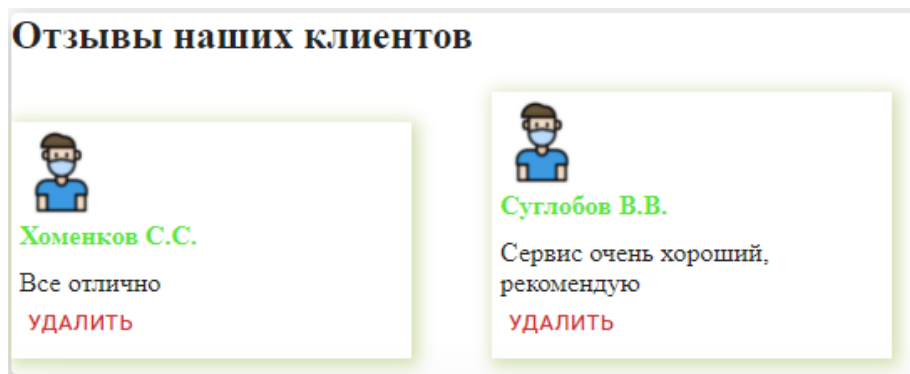


Рисунок 3.15 – Панель отзывов в личном кабинете администратора

Администратору доступна статистика. Она выражена в двух диаграммах: круговой (рисунок 3.16) и столбиковой (рисунок 3.17)

Диаграмма марок автомобилей на 100% заказов



Рисунок 3.16 – Круговая диаграмма

Диаграмма ожидаемых заказов и реальных

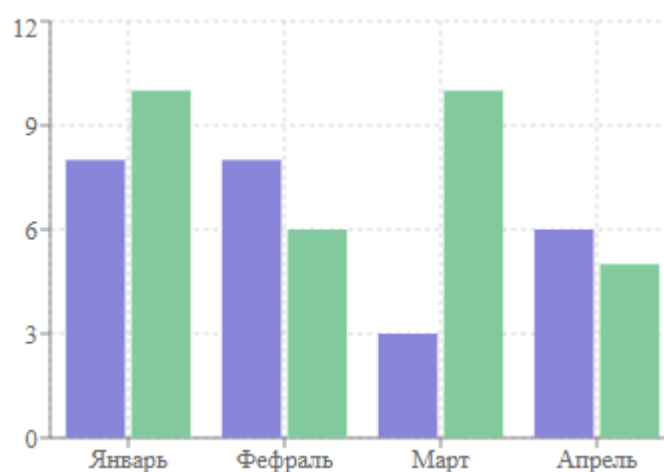


Рисунок 3.17 – Столбиковая диаграмма

4 ТЕСТИРОВАНИЕ

4.1 Тестирование пользовательского интерфейса

В таблице 4.1 приведена информация о результатах проведения тестирования пользовательского интерфейса.

Таблица 4.1 – Тестирование пользовательского интерфейса

№	Вид проверки	Результат
1	Реализуется ли размещение всех сообщений об ошибках, уведомлений	Да
2	Адаптация под разные размеры экрана	Частично
3	Отсутствуют ли орфографически, пунктуационные ошибки	Да
4	Читабелен ли используемый шрифт	Да
5	Имеется ли смена темы	Нет
6	Интуитивно понятное меню	Да

4.2 Тестирование алгоритмов решения

В таблице 4.2 проведены пошаговые тесты, которые в конечном итоге должны привести к ожидаемому результату.

Таблица 4.2 – Примеры проводимых тестов

Краткое описание	Шаги по воспроизведению	Ожидаемый результат	Актуальный результат
1	2	3	4
Проверка введенных данных в форме для авторизации	1. Перейти на страницу для авторизации. 2. Оставить поля пустыми. 3. Нажать кнопку «Вход».	Приложение должно вывести следующее уведомление «Некорректные данные при входе»	Приложение выводит уведомление «Некорректный данные при входе»

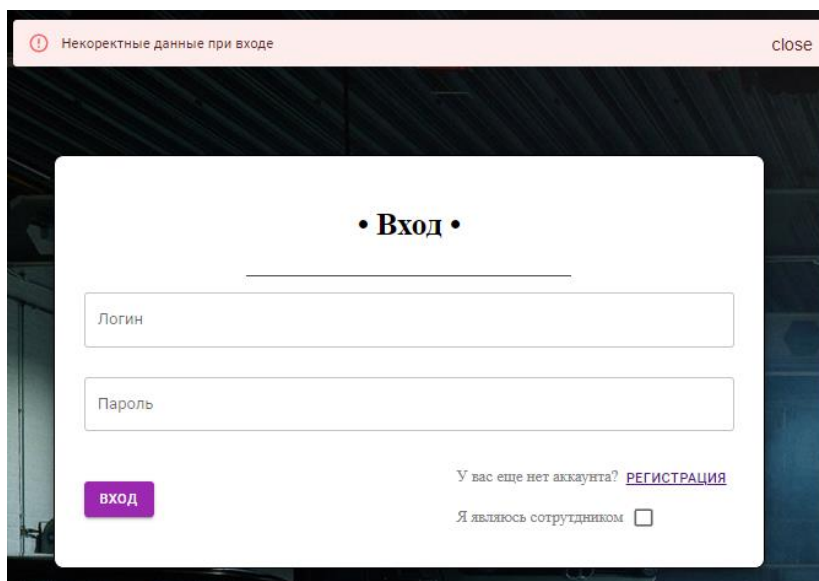
Продолжение таблицы 4.2

1	2	3	4
Проверка на неверный ввод пароля	<p>1. Перейти на страницу для авторизации.</p> <p>2. Ввести следующие данные: логин – <i>login1</i>, пароль – <i>Incorrect</i>.</p> <p>3. Нажать кнопку «Войти».</p>	Приложение должно вывести следующее уведомление «Неверный пароль»	Приложение выводит уведомление «Неверный пароль»
Регистрация существующего пользователя	<p>1. Нажать на ссылку «Регистрация».</p> <p>2. Ввести следующие данные: <i>login1, password1</i>.</p> <p>3. Подтвердить введенные данные.</p>	Приложение должно вывести следующее уведомление «Такой пользователь уже существует»	Приложение выводит уведомление «Такой пользователь уже существует»
Проверка правильности ввода логина при регистрации	<p>1. Перейти на страницу для авторизации.</p> <p>2. Ввести следующие данные: логин – <i>log</i>, пароль – <i>password</i>.</p>	Приложение должно вывести следующее уведомление: «Неверный логин»	Приложение выводит уведомление «Неверный логин»
Проверка на число	<p>1. Авторизоваться как администратор.</p> <p>2. В поле «<i>Cost</i>» ввести строку.</p>	Приложение должно вывести следующее уведомление: «Введены неверные данные»	Приложение выводит уведомление «Введены неверные данные»

Окончание таблицы 4.2

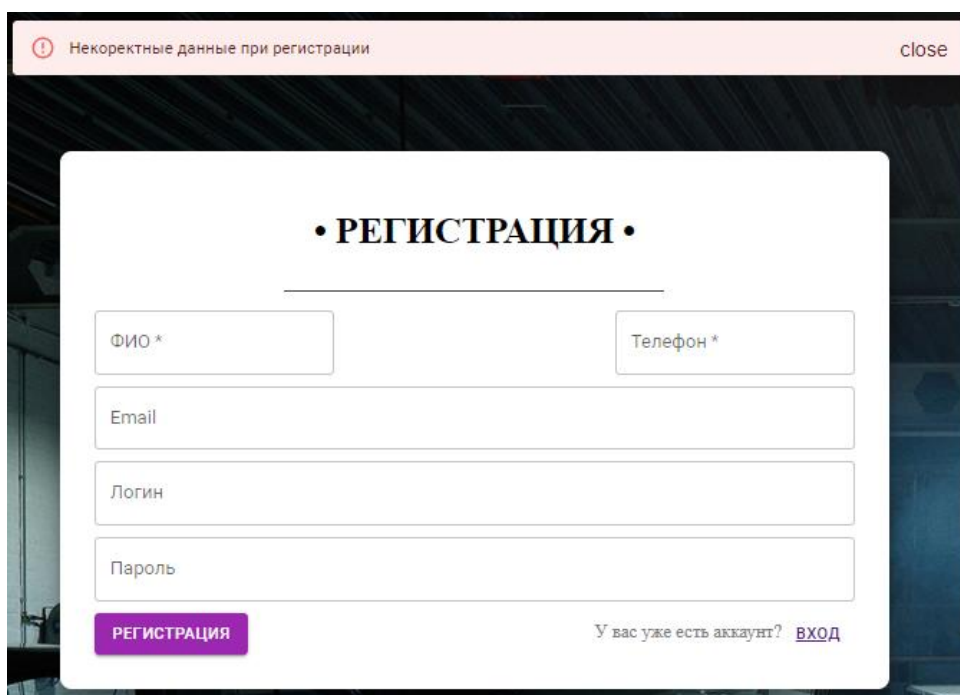
1	2	3	4
Проверка на сохранение нового заказа в базе данных	<p>1. Авторизоваться, введя следующие данные: логин <i>login1</i> и пароль <i>password1</i>.</p> <p>2. Перейти в личный кабинет.</p> <p>3. Нажать на кнопку «Сделать заказ».</p> <p>4. Вернуться в личный кабинет.</p>	Данные о новом заказе добавятся к остальным заказам, которые берутся из базы данных	При открытии личного кабинета, новый заказ добавился в конец к остальным. Также заказ добавился в коллекцию « <i>orders</i> »
Проверка на некорректный <i>email</i>	1. В форме регистрации в поле <i>email</i> ввести следующие данные: <i>ravel#gmail.com</i> .	Приложение должно вывести следующее уведомление «Некорректные данные при регистрации. Неверный <i>email</i> »	Приложение выводит уведомление «Некорректные данные при регистрации. Неверный <i>email</i> »
Проверка на пустые поля при регистрации нового заказа	<p>1. Авторизоваться, введя следующие данные: <i>login1</i>, <i>password1</i>.</p> <p>2. Перейти в личный кабинет.</p> <p>3. Нажать на кнопку «Сделать заказ».</p> <p>4. Оставить поля пустыми и нажать на кнопку «Сделать заказ».</p>	Приложение должно вывести предупреждения о не введенных полях	Приложение выводит предупреждения о не введенных полях

На рисунках 4.1 – 4.2 представлена проверка на пустые поля при попытке входа и регистрации.



The screenshot shows a web application interface for login. At the top, a pink error banner displays the message "Некорректные данные при входе" (Incorrect data during login) with a close button. The main form is titled "• Вход •" (Login). It contains two input fields: "Логин" (Login) and "Пароль" (Password). Below the password field is a purple button labeled "ВХОД" (Login). To the right of the button, there is a link "У вас еще нет аккаунта? РЕГИСТРАЦИЯ" (Don't you have an account yet? REGISTRATION) and a checkbox labeled "Я являюсь сотрудником" (I am an employee).

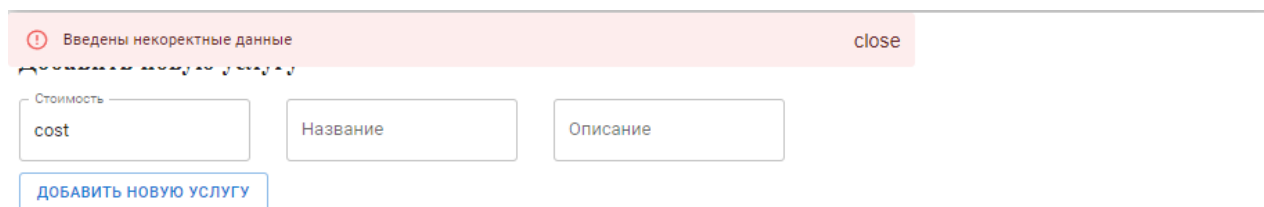
Рисунок 4.1 – Проверка на пустые поля при входе



The screenshot shows a web application interface for registration. At the top, a pink error banner displays the message "Некорректные данные при регистрации" (Incorrect data during registration) with a close button. The main form is titled "• РЕГИСТРАЦИЯ •" (Registration). It contains five input fields: "ФИО *" (Full Name), "Телефон *" (Phone), "Email", "Логин" (Login), and "Пароль" (Password). Below the password field is a purple button labeled "РЕГИСТРАЦИЯ" (Registration). To the right of the button, there is a link "У вас уже есть аккаунт? ВХОД" (Do you already have an account? LOGIN).

Рисунок 4.2 – Проверка на пустые поля при регистрации

Так же некорректный ввод обрабатывается на всех полях ввода в приложении. На рисунке 4.3 представлена обработка ввода в поле «Стоимость» некорректного значения (значение должно быть числом, пользователь пытается ввести строку).



Введен некорректный текст

Стоимость

cost

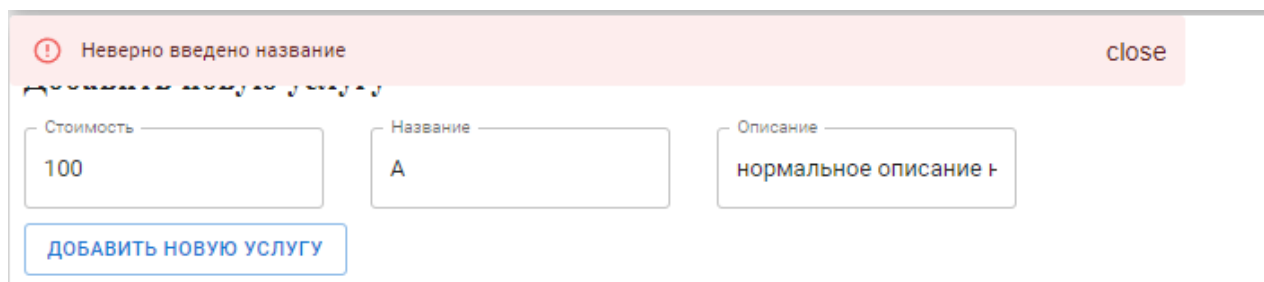
Название

Описание

ДОБАВИТЬ НОВУЮ УСЛУГУ

Рисунок 4.3 – Проверка на некорректные данные

Так же в приложении строки и числа проверяются на диапазон. На рисунке 4.4 представлена обработка некорректного значения в поле «Название». Данная строка не может состоять из одного символа, поэтому пользователь увидит следующее предупреждение:



Неверно введено название

Стоимость

100

Название

А

Описание

нормальное описание

ДОБАВИТЬ НОВУЮ УСЛУГУ

Рисунок 4.4 – Проверка на некорректный ввод

ЗАКЛЮЧЕНИЕ

В результате курсового проекта было разработано программное обеспечение для автоматизации работы станции технического обслуживания.

В данном проекте удалось решить следующие задачи:

- автоматизация оформления заказа клиентом;
- предоставление клиенту всей истории его заказов в удобном виде;
- сбор и анализ статистики;
- предоставление сотрудникам полной информации о клиенте в отдельных личных кабинетах;
- предоставление возможности клиенту оставлять отзыв о проделанной работе.

В процессе создания приложения были пройдены такие этапы разработки, как постановка задачи, составление логической модели предметной области, создание и реализация алгоритма работы программы с последующим тестированием.

Преимуществом разработанной нами системы является ее относительно низкие требования к установке и использования, а также простота и удобство в эксплуатации. Использование одного из самого популярного стека технологий позволило нам достичь большой гибкости программирования. В случае внесения каких-либо изменений в комплексной работе автосервиса это позволит нам в дальнейшем дорабатывать и улучшать приложение.

Список использованных источников

1. Стек технологий для разработки веб-приложений [Электронный ресурс] – Режим доступа: <https://www.azoft.ru/blog/web-development-stack/#:~:text=Веб-технологии%20разработки%20приложений%3A%20Java%20C%20PHP> – Дата доступа : 13.03.2022.
2. Актеры и прецеденты [Электронный ресурс] – Режим доступа: https://studopedia.su/6_42762_akteri-i-pretseidenti.html – Дата доступа: 16.03.2022.
3. Разработка на стеке MERN [Электронный ресурс] – Режим доступа: <https://worksolutions.ru/useful/autsorsing-ili-zakaznaya-razrabotka-na-steke-MERN/> – Дата доступа : 13.03.2022.
4. Введение в MongoDB [Электронный ресурс] – Режим доступа: <https://metanit.com/nosql/mongodb/1.1.php> – Дата доступа: 16.03.2022.
5. Плюсы и минусы React: виртуальная DOM, синтаксис JSX и другие аргументы для спора [Электронный ресурс] – Режим доступа: <https://nuancesprog.ru/p/14500/> – Дата доступа : 17.03.2022.
6. Райтман М.А., Изучаем React. 2-е изд. – СПб.:Москва, 2019 – 368 с.