

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ПРОЕКТНАЯ РАБОТА

ТЕМА: РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ
АВТОМОБИЛЬНОГО САЛОНА

Студент гр. 1308

Семенов М.Д.

Преподаватель

Водяхо А.И.

Санкт-Петербург
2024

ЗАДАНИЕ НА ПРОЕКТ

Студент Семенов М.Д.

Группа 1308

Тема работы: Разработка информационной системы
автомобильного салона

Исходные данные: Целью работы является разработка и обеспечение должной функциональности веб приложения информационной системы автомобильного салона, для автоматизации рабочих процессов администратора системы. В курсовой работе был создан проект приложения с выше перечисленными возможностями. Сформированы основные технические требования, архитектурное описание и тесты для проекта.

Содержание пояснительной записки: Введение, требования, архитектурное описание, архитектурное обоснование, модели, UML описание, use case, классы, активности, размещение, тесты, заключение, список используемых источников.

Предполагаемый объем пояснительной записки: Не менее 15 страниц.

Дата выдачи задания: 01.09.2024

Дата сдачи реферата: 06.10.2024

Студент гр. 1308

Семенов М.Д.

Преподаватель

Водяхо А.И.

АННОТАЦИЯ

Объектом разработки является веб-приложение информационной системы автомобильного салона. Цель работы – разработка и обеспечение должной функциональности веб приложения информационной системы автомобильного салона, для автоматизации рабочих процессов администратора системы. Для этих целей был проведен анализ предметной области и методов реализации проекта, были сформированы технические требования, было составлено архитектурное описание проекта, составлены тесты для информационной системы и ее последующая разработка.

ABSTRACT

The object of development is a web application of the information system of the automobile salon. The purpose of the work is to develop and ensure the proper functionality of the web application of the information system of the automobile salon, to automate the work processes of the system administrator. For these purposes, an analysis of the subject area and methods of project implementation was carried out, technical requirements were formed, an architectural description of the project was compiled, tests for the information system and its subsequent development were compiled.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	7
1. Анализ предметной области и методов реализации.....	8
1.1. Требования.....	8
1.1.1. Глоссарий.....	8
1.1.2. Бизнес – требования.....	8
1.1.3. Пользовательские требования.....	8
1.1.4. Системные требования.....	9
1.1.5. Функциональные требования.....	9
1.1.6. Нефункциональные требования.....	9
1.1.7. Требования к интерфейсу.....	9
1.1.8. Требования к продукту.....	9
1.1.9. Доменные требования	9
1.2. Рассмотрение предметной области	9
1.3. Обоснование выбора типа архитектуры разрабатываемого приложения ..	10
1.4. Обоснование выбора средств реализации проекта	12
2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ АВТОСАЛОНА.....	14
UML моделирование	16
Use case.....	16
Диаграмма классов.....	24
Диаграмма активности.....	27
Диаграмма развертывания	29
3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ АВТОСАЛОНА.....	30
Конфигурация среды разработки.....	30
Реализация веб-приложения	31
4. ТЕСТЫ.....	35
ЗАКЛЮЧЕНИЕ.....	36
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	37

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

БД – база данных

Десктопное приложение – настольное приложение

ИС – информационная система

ОС – операционная система

ПО – программное обеспечение

СУБД – система управления базами данных

UML – унифицированный язык моделирования (Unified Modeling Language)

Use case – это описание взаимодействия с системой в виде последовательности действий для достижения конкретного результата. По сути, это метамодель, которая иллюстрирует не саму систему, а набор функциональных требований к ней.

ВВЕДЕНИЕ

В настоящее время, автомобиль - самое распространённое средство передвижения. Сейчас автомобили продаются в автосалонах и покупка автомобиля коснулась огромного количества людей. В связи с возросшей активностью посещений автосалонов, становится сложнее обеспечивать комфортные условия не только для клиентов, но и для персонала. Решить эту проблему можно с помощью автоматизации рутинной части работы административного персонала автомобильного салона. Актуальность автоматизации процессов работы с информацией, при столь быстрорастущих ее объемах, сложно недооценить. Как в сфере предоставления услуг по продаже автомобилей, так и в любой другой, компании, автоматизирующие необходимую часть своей деятельности, оказываются в более выгодном положении на рынке среди конкурентов, что способствует благоприятной среде для их дальнейшего развития. Поэтому, разработка решения, эффективно взаимодействующего с потоками информации, является необходимым и современным для каждого автосалона. Основной целью данной работы, является разработка информационной системы, обеспечивающей автоматизацию рутинных задач работы администратора автомобильного салона.

Для успешного достижения поставленной цели, были поставлены следующие задачи:

- Анализ предметной области и методов реализации проекта
- Проектирование ИС автомобильного салона
- Разработка информационной системы автомобильного салона

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И МЕТОДОВ РЕАЛИЗАЦИИ

1.1. Требования

1.1.1. Глоссарий

Глоссарий приведен в таблице 1.

Таблица 1. Глоссарий

№	Термины	Определения
1	Use case	Описание поведения системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды
2	Интерфейс	Совокупность возможностей, способов и методов одновременного действия
3	База данных	Совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, причём такое собрание данных, которое поддерживает одну или более областей применения
4	Сайт	Одна или несколько логически связанных между собой веб-страниц; также место расположения контента сервера.
5	Фреймворк	Программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта
6	UML	Унифицированный язык моделирования (Unified Modeling Language)
7	Десктопное приложение	Это программа, которая требует ОС настольного компьютера для работы
8	СУБД	Система управления базой данных. Совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

1.1.2. Бизнес – требования

- 1) Возможность администрирования сайта;
- 2) Интуитивно понятный интерфейс;
- 3) Поддержка вёрстки всеми популярными браузерами.

1.1.3. Пользовательские требования

- 1) Возможность простой и быстрой регистрации;
- 2) Возможность использовать поиск информации по ключевым словам;
- 3) Обеспечение обратной связи с технической поддержкой сайта и с его администрацией;

1.1.4. Системные требования

- 1) Зарегистрированное доменное имя;
- 2) Для пользователя: любой браузер;
- 3) Для реализации проекта должен быть использован язык Python с фреймворком Django, HTML и CSS для создания внешнего вида сайта, а база данных - SQLite;
- 4) Доступ к глобальной сети Интернет

1.1.5. Функциональные требования

- 1) Возможность для пользователя регистрации и авторизации;
- 2) Возможность пользователя просматривать подробную информацию о работниках, клиентах, машинах и услугах;
- 3) Возможность пользователя редактировать данные;
- 4) Возможность поиска определенных определенных данных из разных разделов.

1.1.6. Нефункциональные требования

- 1) Гарантия безопасности сайта;
- 2) Защита сайта от вирусных или иного рода атак;
- 3) Данные работников и клиентов должны быть конфиденциальными.

1.1.7. Требования к интерфейсу

- 1) Интерфейс должен быть интуитивно понятным;
- 2) Размер и стиль шрифта надписей и слов, отображаемых на страницах сайта, должны быть комфортны глазу;
- 3) На начальной странице сайта должны отображаться все разделы сайта для предоставления дальнейшего выбора пользователю;
- 4) Минималистичный стиль

1.1.8. Требования к продукту

- 1) Стабильная и быстрая работа сайта;
- 2) Вёрстка сайта должна поддерживать все популярные браузеры.

1.1.9. Доменные требования

- 1) Доменное имя должно быть связано с терминами из индустрии «автотранспорт».
- 2) Доменное имя не должно быть очень большим (не более тридцати символов).

1.2. Рассмотрение предметной области

Автомобильный салон – это предприятие, которое занимается продажей автомобилей, обычно являющаяся частной, специализирующаяся на предоставлении комплекса услуг по продаже и обслуживанию автомобилей. В зависимости от конкретной организации, может включать в себя такие виды как продажа автомобилей, шиномонтаж, ремонт автомобиля, техническое обслуживание, химчистка(мойка). При этом услуги могут оказываться как по индивидуальным заказам, так и оптом(несколько машин). Информационная система (ИС) – это взаимосвязанное объединение средств, методов и персонала, используемое для сбора, хранения, обработки и выдачи информации, в целях успешной реализации управленческих решений. В современных реалиях основным техническим средством обработки информации является персональный компьютер. Большинство современных информационных систем преобразуют не информацию, а данные. Поэтому также их называют системами обработки данных.

Выделяют следующие основные принципы построения эффективных ИС:

Принцип интеграции, заключающийся в том, что обрабатываемая информация, однажды введенная в систему, позже многократно используется для решения типовых задач;

Принцип системности, состоящий в обработке данных тех или иных областей, для получения информации, требующейся для принятия решений на всех уровнях управления;

Принцип комплексности, заключающийся в механизации и автоматизации процедур преобразования данных на каждом из этапов функционирования информационной системы;

Деятельность любого автомобильного салона, подразумевает постоянное взаимодействие с данными и ведение их учета, например, сведений о нанятых специалистах или данных покупателя, информации о предоставляемых услугах и автомобилях. Именно для этих процессов, в автосалоны и внедряют информационные системы, позволяющие автоматизировать управление и учет данных. Такие ИС предназначены для оптимизации работы, в первую очередь, руководства и административного персонала, и являются залогом успешного повышения производительности их труда. Используя этот инструмент, сотрудникам становится значительно легче выполнять рутинные операции, связанные с получением и ведением данных о покупателях, работниках и иных объектах, вовлеченных в работу автосалона (по некоторым оценкам, использование ведения документации в электронном виде, позволяет сократить время обработки данных до 75%, в сравнении с бумажными носителями информации). Автоматизированная ИС, также дает возможность администратору избежать значительных временных затрат на поиск и фильтрацию необходимых данных, снизить трудоемкость обработки больших объемов информации. Рассмотрев предметную область, можно сделать вывод о том, что разработка информационной системы автомобильного салона является актуальной задачей и неотъемлемым условием конкурентоспособности в современных реалиях предоставления услуг автосалона.

1.3. Обоснование выбора типа архитектуры разрабатываемого приложения

Перед тем как определить методы и технологии для реализации информационной системы, необходимо выбрать тип архитектуры

разрабатываемого приложения. Информационную систему автомобильного салона можно представить как локальное (настольное) приложение или в виде веб-приложения.

Локальное приложение представляет собой клиентское программное обеспечение, реализующее стандартный интерфейс операционной системы и работающее независимо от других приложений. Для его работы необходима установка на каждую рабочую станцию пользователя, и оно обычно запускается локально. Подключение к интернету не требуется для его функционирования. Примерами локальных приложений являются продукты MS Office, такие как Word, Excel, Access.

Веб-приложение представляет собой программное обеспечение клиент-серверного типа, где клиент взаимодействует с веб-сервером через браузер. На стороне клиента реализуется пользовательский интерфейс, формируются запросы к серверу и обрабатываются его ответы. На стороне сервера осуществляется прием и обработка запросов, а затем формируется веб-страница, которая отправляется на клиентскую часть по сети с использованием технологий протокола HTTP. Обобщенная модель клиент-серверной архитектуры веб-приложения, показана на рисунке 1.1.

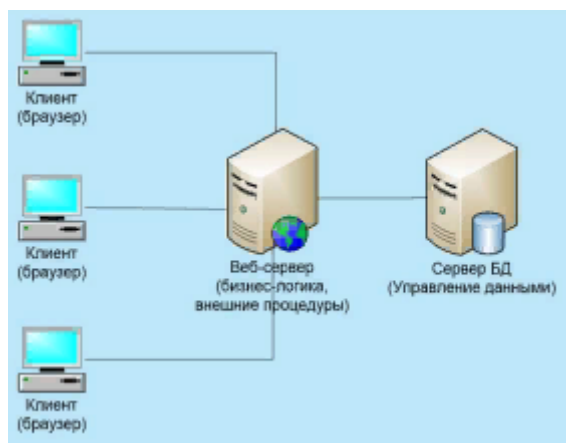


Рисунок 1.1 – Клиент-серверная архитектура веб-приложения

У каждого типа архитектуры приложений есть свои преимущества и недостатки. Однако для реализации информационной системы автосалона, вебприложение представляет собой оптимальное решение по следующим причинам:

- Веб-приложения не требуют развертывания на каждой рабочей станции;

- Отсутствие необходимости проверки текущей версии приложения на клиентском компьютере, что исключает возможные ошибки, связанные с несовместимостью версий;
- Обновления реализуются гораздо проще по сравнению с десктопными приложениями;
- Нет необходимости в проверке прав администратора системы, так как это осуществляется со стороны самого приложения;
- Возможность получения доступа из любой точки мира при наличии сетевого соединения, что позволяет выполнять работу вне автосалона, что особенно актуально в условиях распространенной практики удаленной работы;
- Независимость от платформы и операционной системы рабочей станции;
- Более низкие системные требования к аппаратной части клиентского компьютера, в сравнении с десктопным приложением;
- Возможность подключения сторонних API для упрощения расширения функционала веб-приложения;
- Адаптивность в мобильных приложениях;

1.4. Обоснование выбора средств реализации проекта

В создании любого веб-приложения, принцип разделения ответственностей между внутренней реализацией и внешним представлением выделяет два основных направления: Back-end и Front-end.

Область Front-end касается частей веб-приложения, с которыми взаимодействует пользователь через браузер, фокусируясь на клиентской стороне пользовательского интерфейса. Технологии Front-end включают HTML, CSS и JavaScript.

Back-end отвечает за программно-аппаратную часть и логику веб-приложения, обеспечивает функциональную составляющую серверной части. Для Back-end можно использовать различные инструменты и языки программирования, такие как PHP, Python, Java, Ruby, JavaScript/Node.js, а также системы управления базами данных.

Разработка Back-end части веб-приложений может осуществляться различными методами, такими как конструкторы сайтов, CMS (Content

Management System), собственные решения на языках программирования и использование фреймворков. Профессиональные разработчики предпочитают использовать фреймворки, поскольку они обеспечивают более широкие возможности для реализации функционала сложных систем.

Фреймворк представляет собой компонент программной платформы, который служит каркасом для программной системы. Фреймворки создаются для конкретных языков программирования и платформ, предоставляя готовые решения для типичных задач разработки веб-приложений, таких как шаблонизация, маршрутизация, интеграция с базами данных и многое другое.

Существует множество фреймворков, используемых в создании Back-end части веб-приложений, таких как Django, Flask, Express.js, Ruby on Rails, Laravel, Spring и другие. Перед выбором фреймворка важно определить предпочтительный язык программирования. Для создания веб-приложения для информационной системы автосалона, Python является оптимальным выбором из-за своего крупного сообщества программистов, чистого синтаксиса, обширной библиотеки, асинхронной поддержки и интеграции с другими языками.

Среди популярных фреймворков для Python выделяются Django и Flask, которые предоставляют обширные возможности для эффективной разработки Back-end части веб-приложений.

- При выборе языка программирования Python для разработки веб-приложения следует ознакомиться с существующими фреймворками. Среди наиболее популярных и подходящих для этой цели можно выделить Django и Flask.
- Django представляет собой широко используемый веб-фреймворк на основе Python, ориентированный на предоставление основных функций для создания веб-приложений. Он включает в себя свою систему именования компонентов, таких как "представления" для HTTP ответов, а также предоставляет встроенную панель администратора и ряд других характеристик, таких как простой синтаксис, встроенный веб-сервер, архитектура ядра MVC, собственная технология ORM (Объектно-Реляционное Отображение) и система миграции баз данных, библиотеки для работы с HTTP протоколами, поддержка промежуточного

ПО (middleware), возможность модульного тестирования Python и базовые инструменты для общих задач.

- Основным недостатком Django является его ограниченная гибкость, избыточная монолитность для некоторых задач и специфичная Django ORM технология, что может привести к проблемам при замене встроенной ORM на другую.
- Flask – еще один популярный фреймворк для веб-приложений на Python, относящийся к категории микрофреймворков. Он предоставляет минимальный набор базовых возможностей, сохраняя простоту конфигурации среды разработки. В отличие от Django, Flask не навязывает строгих правил структурирования программы, что делает его более гибким. Flask не привязывает к конкретной базе данных, а его шаблонизатор Jinja2 обеспечивает гибкость и удобство в разработке.
- Некоторые особенности и преимущества Flask включают его легковесность для создания простых веб-приложений, модульную структуру, интуитивно понятный синтаксис, и гибкий шаблонизатор Jinja2, который позволяет контролировать процесс разработки, не ограничивая функциональность. Проекты, такие как Red Hat, Airbnb, Netflix и Reddit, свидетельствуют о ведущих позициях Flask в мировой разработке.

Flask включает в себя встроенные инструменты для тестирования и отладки, предоставляя разработчику полноценный функционал, такой как встроенный сервер разработки, отладчик и обработчик запросов. Фреймворк также обеспечивает защиту от межсайтового скриптинга (XSS), представляя собой надежное решение для современных проблем безопасности веб-приложений.

В работе с базами данных в Flask используется библиотека SQLAlchemy, которая применяет технологию ORM. Эта библиотека позволяет описывать структуры баз данных и взаимодействовать с ними на языке Python, минуя использование SQL. SQLAlchemy действует как промежуточный уровень между приложением и базой данных, обеспечивая их соединение независимо друг от друга. Такой подход обеспечивает гибкость и эффективное использование базы данных, что делает его предпочтительным по сравнению с Django ORM.

Сравнив особенности фреймворков Django и Flask, для реализации веб-приложения информационной системы автосалона предпочтение отдается Flask. Это обосновывается несколькими факторами:

- Flask подходит для небольших и средних веб-приложений без острой необходимости в масштабировании, что соответствует потребностям автосалона без сетевой инфраструктуры
- Flask гибок и не перегружен невостребованным функционалом, что упрощает адаптацию под конкретные требования автосалона и предоставляет разработчику больше свободы для креативных решений.
- Flask обладает развитой системой защиты от уязвимостей веб-приложений, что важно для сохранения конфиденциальных данных клиентов и сотрудников автосалона.
- Flask требует меньше времени для изучения, что ускоряет процесс создания веб-приложения.

Относительно системы управления базами данных (СУБД), оптимальным выбором для проекта считается SQLite. Эта встраиваемая СУБД обладает компактной файловой структурой, не требует настройки сервера, обеспечивает высокую скорость операций выборки данных, совместима с SQLAlchemy и Flask, и не требует значительных ресурсов.

Таким образом, в обосновании выбора средств реализации проекта для Backend части используется следующий стек технологий: язык программирования Python, фреймворк Flask с шаблонизатором Jinja2, СУБД SQLite и библиотека SQLAlchemy. Front-end часть включает в себя верстку веб-страниц на HTML и CSS, с элементами использования CSS-фреймворка Bootstrap

2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ АВТОСАЛОНА

UML моделирование

Унифицированный язык моделирования (UML) используется в качестве средства обеспечения визуализации объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов и отображения организационных структур в программных системах. Построение моделей на этапе проектирования информационной системы автосалона,

позволит упростить и структурировать дальнейший процесс разработки веб-приложения.

Для проектирования данной системы, будут созданы диаграммы вариантов использования (use case diagram), служащие для отражения отношений между акторами и прецедентами, являющимися основными видами элементов. Актор – роль пользователя, взаимодействующая с некоторой сущностью. Графически реализуется в виде стилизованного человечка. Прецедент – выполняемые системой действия (включая доступные варианты), которые приводят к отслеживаемым актором результатам. Графически реализуется в виде эллипса с надписью. С помощью подобных диаграмм появляется возможность описывать поведение и функциональность проектируемой системы.

В качестве инструмента создания диаграмм вариантов использования, в данном проекте применялась веб-платформа для работы с диаграммами и схемами – Lucidchart.

В ходе выделения общих требований к разрабатываемой информационной системе автосалона, была выявлена единственная роль пользователя – администратор. Для данной роли доступны основные блоки взаимодействия с системой, такого типа: работа с информацией о клиентах, работниках, машинах и услугах. Все эти варианты активностей и отражаются на диаграммах вариантов использования. На общей диаграмме, демонстрируются указанные выше, основные подсистемы администратора, затем на этой основе создаются последующие диаграммы, детально рассматривающие содержимое подсистем.

Рассмотрим общую диаграмму (рис. 2.1.), состоящую из следующих подсистем:

- Ведение базы данных клиентов
- Ведение базы данных работников
- Ведение базы данных машин
- Ведение базы данных услуг

Use case

Use case diagram - диаграмма, на которой изображаются варианты использования проектируемой системы.

В данном проекте присутствуют один вид пользователей:

1. Администраторы приложения перенимают весь функционал с возможностями добавления новых и редактированием имеющихся клиентов, работников, машин и услуг, а также их поиском и удалением.

Описание для работы администратора представлено в таблице 2.1.

Таблица 2.1. Работа Администратора

Вариант использования	Внесение изменений в базах данных автосалона
1.1)Администратор авторизуется в приложении.После успешной авторизации приложение переводит администратора на главную страницу	
2.1)Выбор одного из разделов "Клиенты", "Работники", "Машины", "Услуги"	2.2)Открытие окна с подробной информацией о разделе (например "Клиенты"(остальные разделы аналогично))
3.1)Добавление нового клиента	3.2) Удаление клиента
3.3)Открытие окна с подробной информацией о клиенте для дальнейшего удаления или изменения данных клиента	3.4) Изменение данных клиента
4)Поиск клиента	

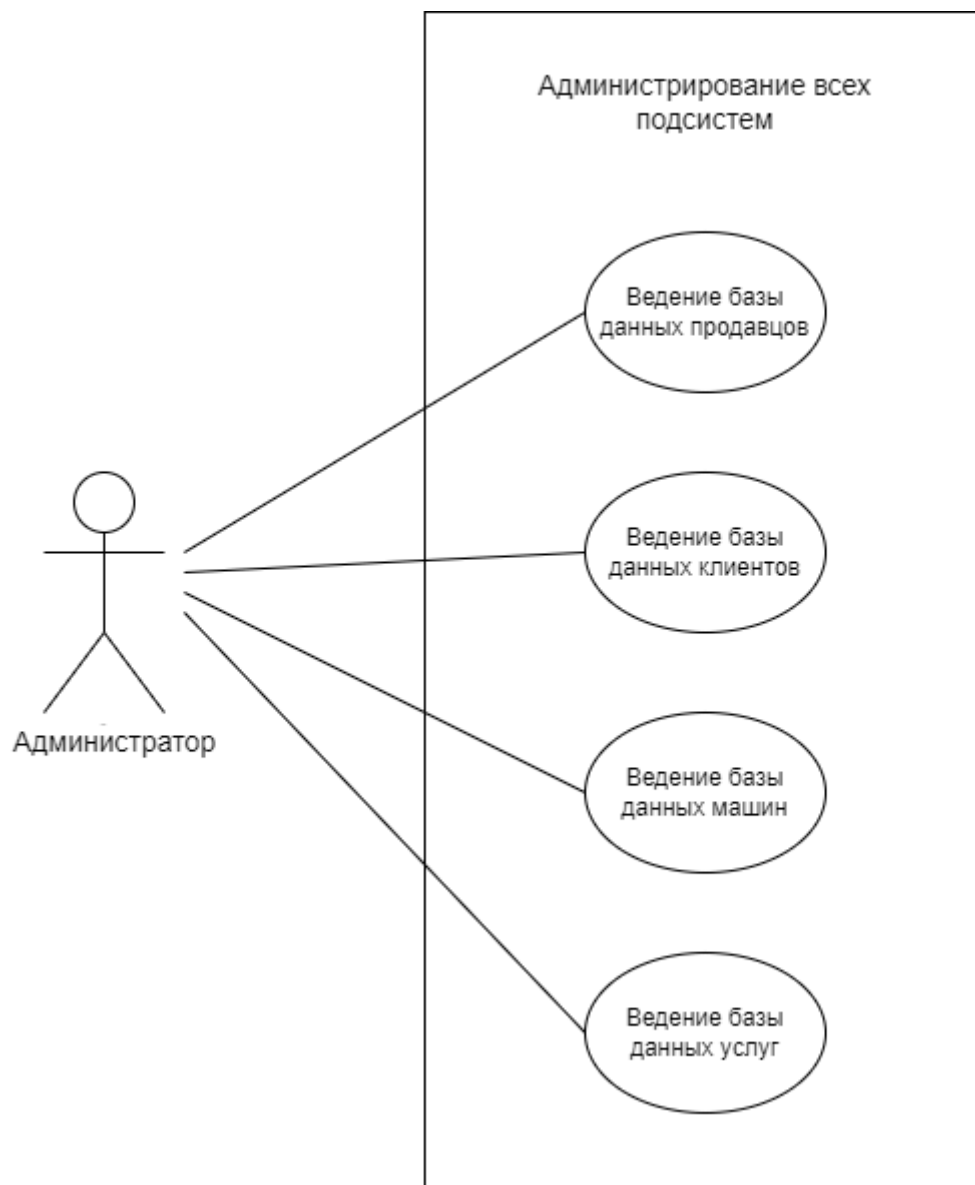


Рис. 2.1 - Общая диаграмма подсистем администратора

Детализация такого варианта использования, как «Ведение базы данных работников(продавцов)», изображена на рисунке 2.2. Основным вариантом использования в данной диаграмме является просмотр базы данных всех продавцов. Этот вариант предоставляет возможности поиска по данным клиента, добавление нового продавца и просмотр данных определенного продавца. Из просмотра данных определенного продавца можно осуществить редактирование или удаление его данных.

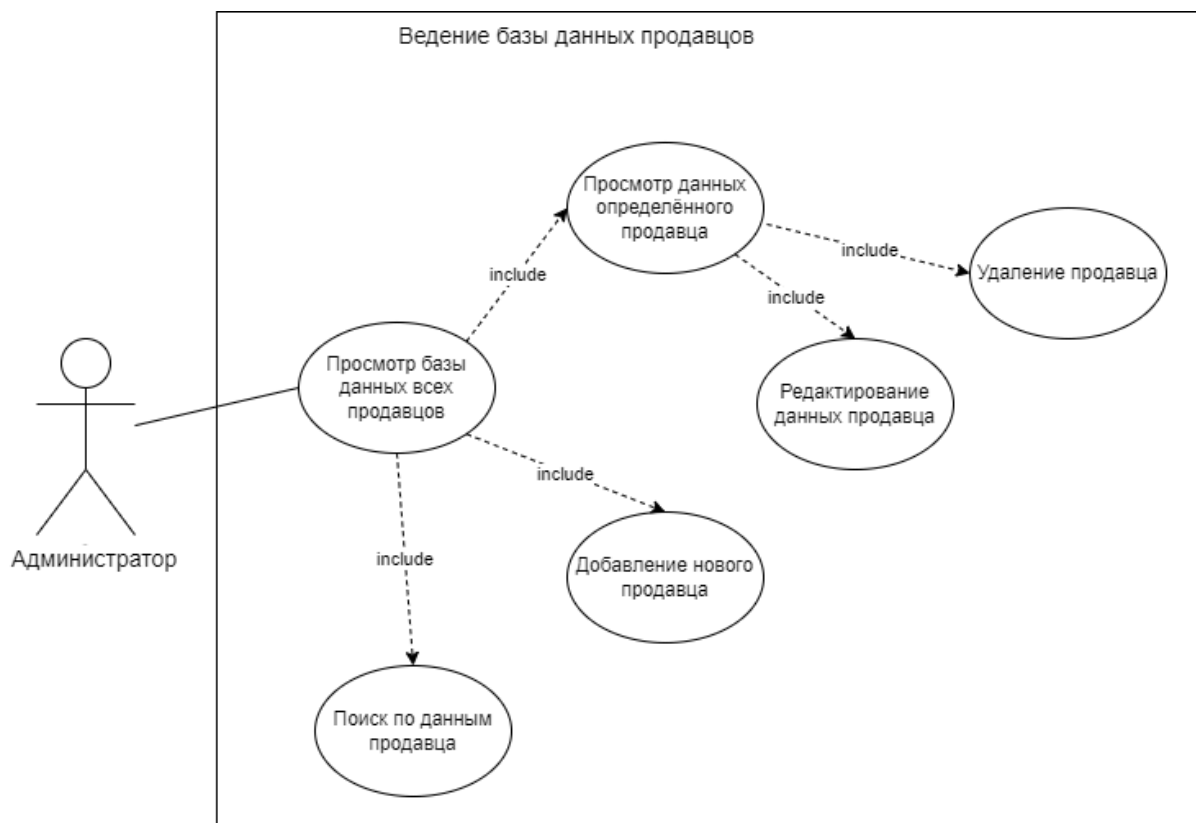


Рис. 2.2 - Диаграмма детализации подсистемы «Ведение базы данных продавца»

Детализация такого варианта использования, как «Ведение базы данных клиентов», изображена на рисунке 2.3. Основным вариантом использования в данной диаграмме является просмотр базы данных всех клиентов. Этот вариант предоставляет возможности поиска по данным клиента, добавление нового клиента и просмотр данных определенного клиента. Из просмотра данных определенного клиента можно осуществить редактирование или удаление его данных.

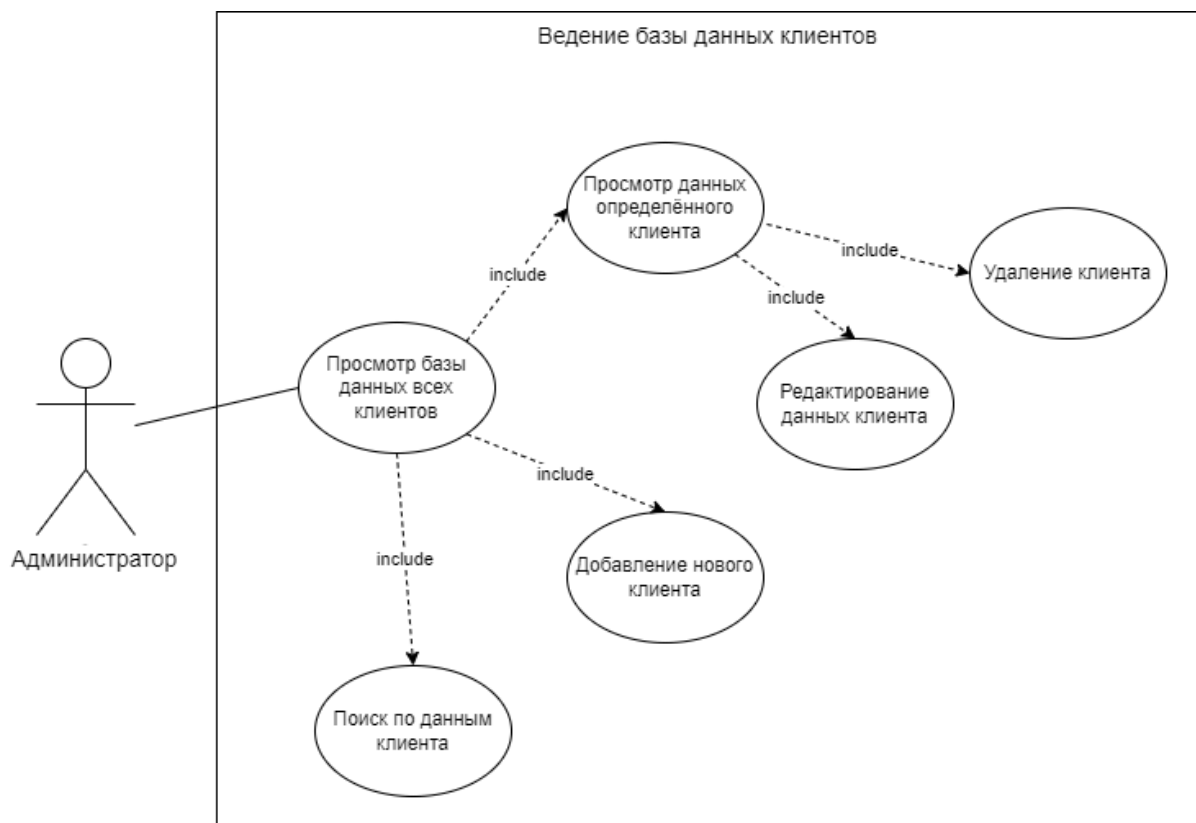


Рис. 2.3 - Диаграмма детализации подсистемы «Ведение базы данных клиента»

Детализация такого варианта использования, как «Ведение базы данных услуг». Ключевым вариантом использования в данной диаграмме является просмотр базы данных всех услуг. Этот вариант предоставляет возможности поиска по услугам, добавление новой услуги и просмотр данных определенной услуги. Из просмотра данных определенной услуги можно осуществить её редактирование или удаление.

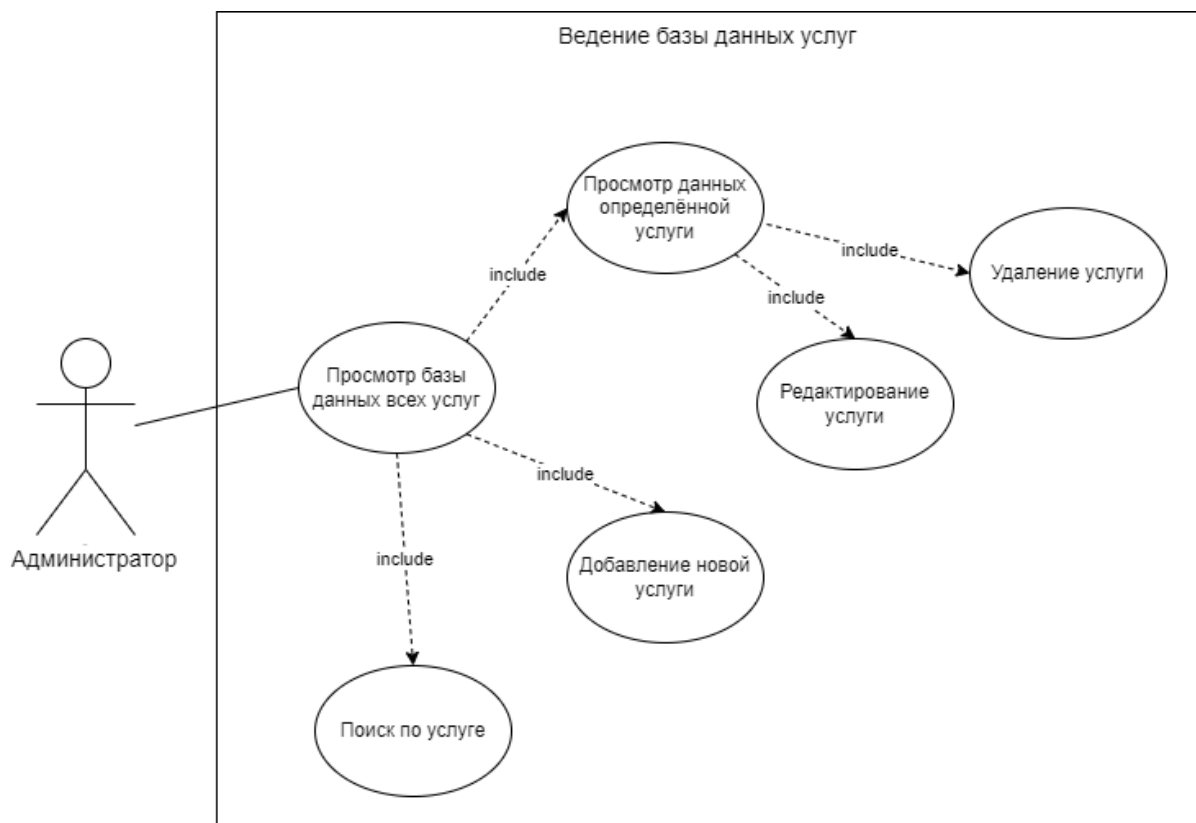


Рис. 2.4 – Диаграмма детализации подсистемы «Ведение базы данных услуг»

В заключении, рассмотрим детализацию варианта использования «Ведение базы данных машин», которая изображена на рисунке 2.5. Базовым вариантом использования в данной диаграмме является просмотр базы данных всех машин. Данный вариант дает возможность поиска по данным машин, добавления новой машины и просмотра данных определенной машин. Из просмотра данных определенной машины можно осуществить редактирование или удаление её данных.

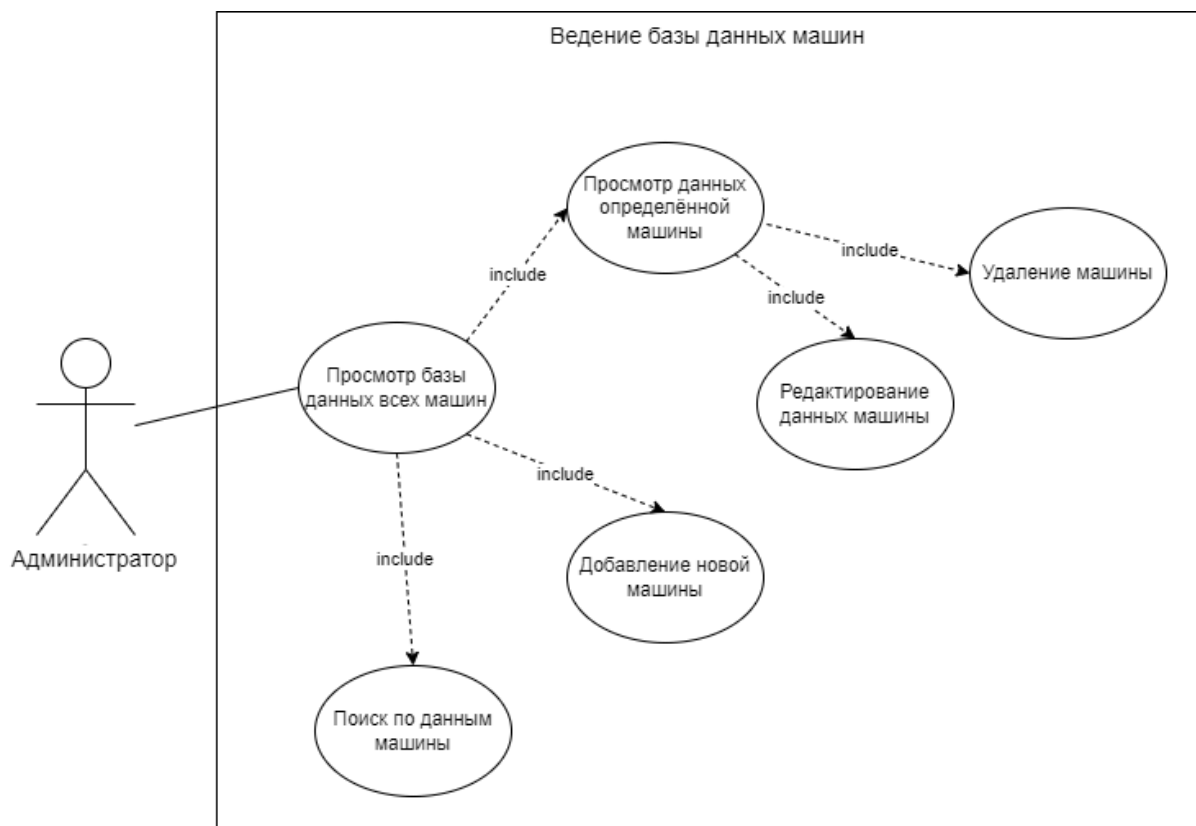


Рис. 2.5 – Диаграмма детализации подсистемы «Ведение базы данных машин»

Диаграмма классов

Диаграмма классов представлена на рисунке 2.6.

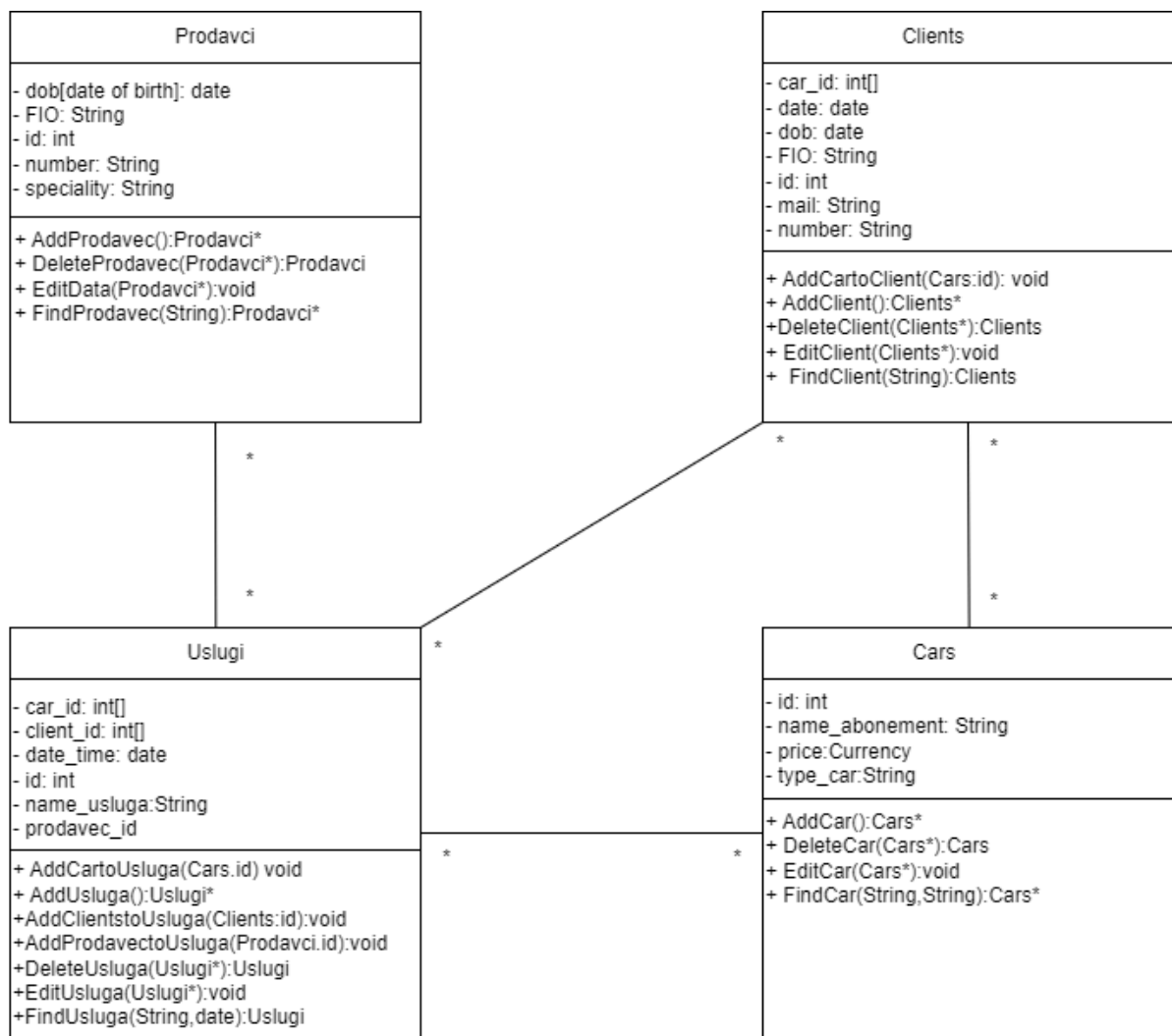


Рис. 2.6. Диаграмма классов

Описание диаграммы классов: в данном проекте основными классами являются классы Клиент, Работник, Машина, Услуга. Класс Клиенты описывает клиентов автосалона. За каждым клиентом закреплены его ФИО (FIO), номер телефона (number), электронная почта (mail), его уникальный идентификатор (id), его дата рождения (dob), дата занесения его данных в систему (date), и id машин, которые данный клиент приобрел (car_id). Для данного класса доступны методы добавления машины по идентификатору к клиенту (AddCartoClient (Cars.id)), добавление в базу данных нового клиента (AddClient()), удаление существующего клиента из базы данных (DeleteClient(Clients*)), метод для редактирования данных существующего в базе данных клиента (EditClient (Clients*)), а также метод поиска клиента из базы данных по ФИО (FindClient (String, date))

Класс Продавцы(Работники) необходим для хранения информации о работниках в автосалоне. Данный класс хранит в себе ФИО (FIO), номер телефона (number), его уникальный идентификатор (id), его дату рождения (dob), а также специальность (speciality). Для данного класса доступны методы: добавление в базу данных нового работника (AddProdavec()), удаление существующего работника из базы данных DeleteProdavec(Prodavci*), метод для редактирования данных существующего в базе данных клиента (EditData (Prodavci*)), а также метод поиска работника из базы данных по ФИО (FindProdavec (String)).

Третий класс – Услуги – описывает услуги, которые предоставляются автосалоном. Этот класс состоит из списка идентификаторов машин, которым оказывается данная услуга, (car_id), из списка идентификаторов клиентов, которые записаны на эту услугу (client_id), дата его проведения (date_time), собственно идентификатор самой услуги (id), название услуги (name_usluga), а также из идентификатора работника, который оказывает эту услугу (prodavec_id). Методы данного класса дают возможность добавить в базу данных новую услугу (AddUsluga()), удалить существующую услугу из базы данных DeleteUsluga(Uslugi*), редактирования данных существующей в базе данных услуги (EditUsluga (Uslugi*)), а также возможности редактирования определенных данных, таких как добавление машины к услуге (AddCartoUsluga(Cars.id)), добавление клиента к списку получающих эту услугу (AddClientstoUsluga(Uslugi.id)), добавление работника к работникам, оказывающим эту услугу (AddProdavectoUsluga(Prodavci.id)) а также поиск услуги из базы данных по названию и дате его проведения (FindUsluga (String, date)).

Класс Машины необходим для описания машин, которые может приобрести клиент. Данный класс содержит идентификатор машины (id), название машины (name_car), цену машины(price), тип машины (грузовая или легковая) (type_car). Методы класса Машины включают в себя добавление новой машины (AddCar()), удаление машины DeleteCar(Cars*), редактирование данных о машине EditCar(Cars*) и нахождение машины в базе данных по названию и типу (FindCar(String, String)).

Класс `Prodavci` связан с классом `Uslugi` отношением ассоциации «многие-ко многим», так как один работник может оказывать разные услуги, а одну услугу могут оказывать разные работники.

Между классами `Клиенты` и `Услуги` -- отношение агрегации «многие-ко многим», поскольку один клиент может получать разные услуги, а одну услугу могут получать множество клиентов.

Классы `Клиенты` и `Машины` связаны отношением ассоциации «один-ко многим», потому что у одного клиента может быть несколько машин.

Классы `Услуги` и `Машины` связаны также отношением «многие-ко многим», так как одной машина может оказываться несколько услуг, а одна услуга может оказываться нескольким машинам.

Диаграмма активности

Диаграмма активности представлена на рисунке 2.7.

При входе в информационную систему автосалона, пользователь перенаправляется на страницу авторизации, где ему предлагается ввести свой логин и пароль. В случае отсутствия регистрации, пользователю предоставляется возможность перейти по ссылке "Зарегистрироваться". После успешного входа в систему, пользователь может взаимодействовать с навигационной панелью, где ему доступны опции редактирования и удаления данных. Также присутствует элемент для добавления нового клиента, при активации которого пользователь переходит на соответствующую страницу с заполняемыми полями, использующими шаблон формата вводимых данных.

Кроме того, в рассматриваемом разделе реализована строка поиска, которая фильтрует данные клиентов по ФИО и дате их оформления в системе. Важно отметить, что функционал всех разделов навигационной панели идентичен, обеспечивая пользователю единый опыт взаимодействия с различными разделами системы.

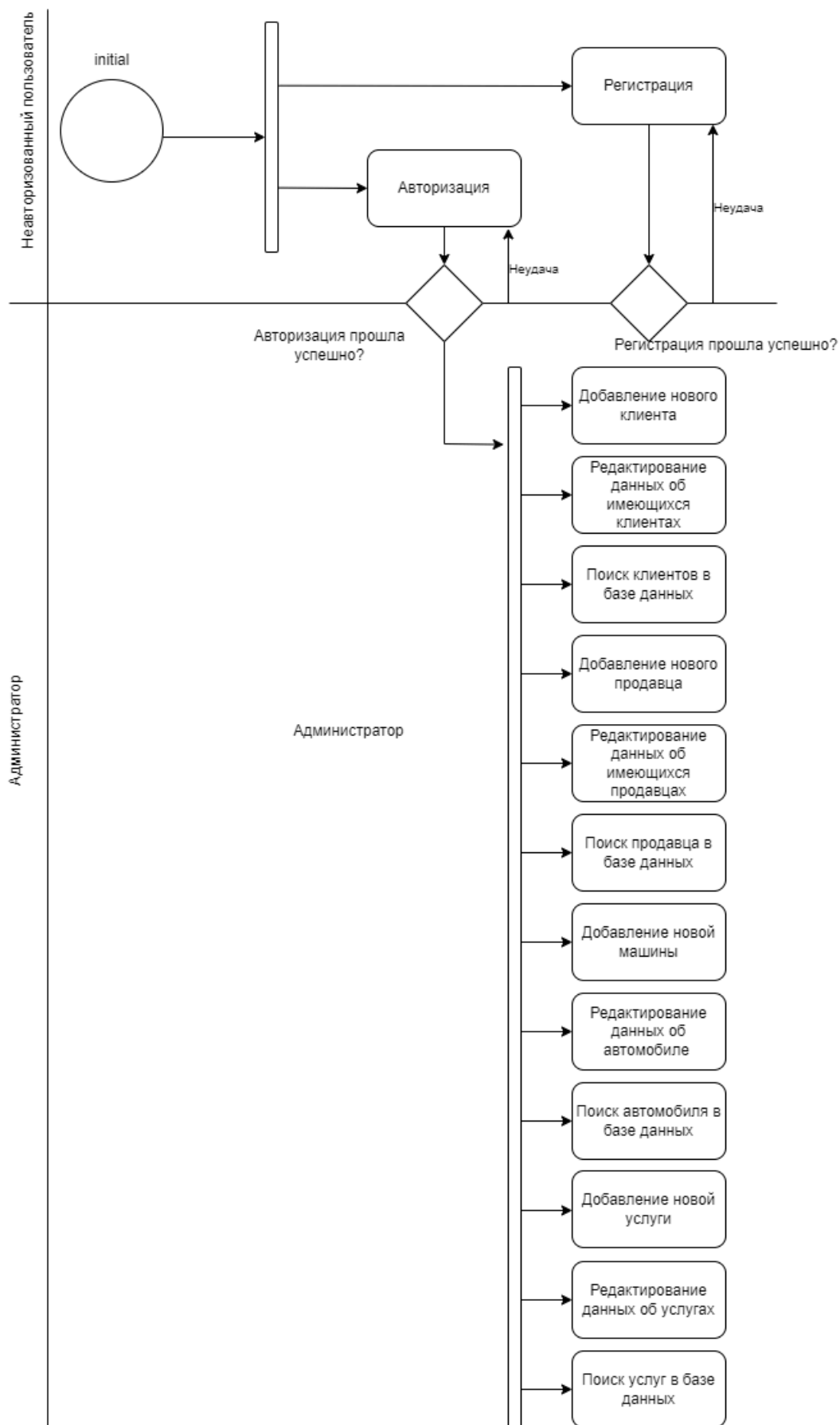


Рис.2.7 Диаграмма активности

Диаграмма развертывания

Диаграмма развертывания представлена на рисунке 2.8.

В обосновании выбора технологического стека для реализации проекта использовались следующие части:

Back-end:

Python

фреймворк Flask в паре с шаблонизатором Jinja2

БД: SQLite и библиотека SQLAlchemy

Front-end:

HTML

CSS, с использованием элементов CSS-фреймворка Bootstrap.

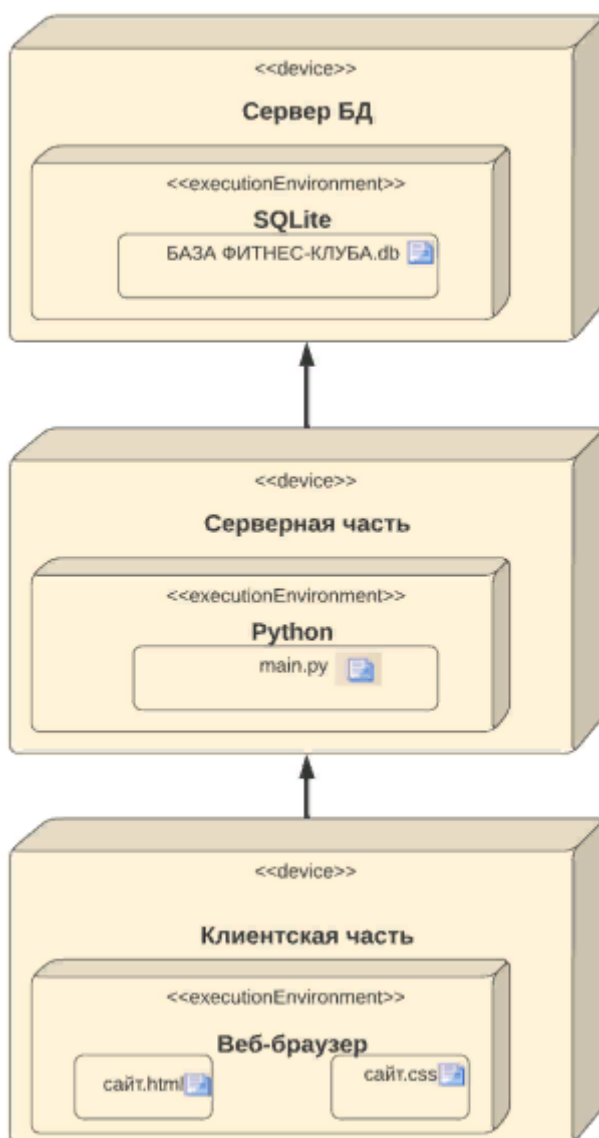


Рис.2.8 Диаграмма развертывания

3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ АВТОСАЛОНА

Конфигурация среды разработки

Прежде чем приступить к разработке веб-приложения, спроектированной ИС автосалона, необходимо настроить и сконфигурировать среду разработки. В качестве редактора кода, будет использован Visual Studio Code. Установив все используемое ПО, до начала работы с проектом, требуется создать виртуальную среду (Virtual Environment) Python. Виртуальная среда – это изолированная копия Python, куда устанавливаются пакеты и библиотеки, не затрагивающие глобальную версию Python, что позволяет избежать ряда проблем, способных возникнуть при использовании библиотек и пакетов различных версий. Далее, созданная среда должна быть активирована, после чего можно приступить к дальнейшей работе с проектом.

Следующим шагом является установка фреймворка Flask и импортирование библиотек, используемых в разработке. На рисунке 3.1. представлены все установленные для проекта пакеты, библиотеки и методы/

```
from flask import Flask, render_template, url_for, redirect, request, make_response, flash
from flask_login.utils import login_required
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, login_user, logout_user
from flask_login import UserMixin
from sqlalchemy.orm import backref
from werkzeug.security import check_password_hash, generate_password_hash
import json
from datetime import datetime
from collections import namedtuple
```

Рисунок 3.2 – Импортированные в проект библиотеки и методы.

После этого, среду разработки можно считать сконфигурированной

Реализация веб-приложения

Для создания веб-приложения ИС автосалона в редакторе VS Code, был создан проект со следующей структурой, представленной на рисунке 3.2.

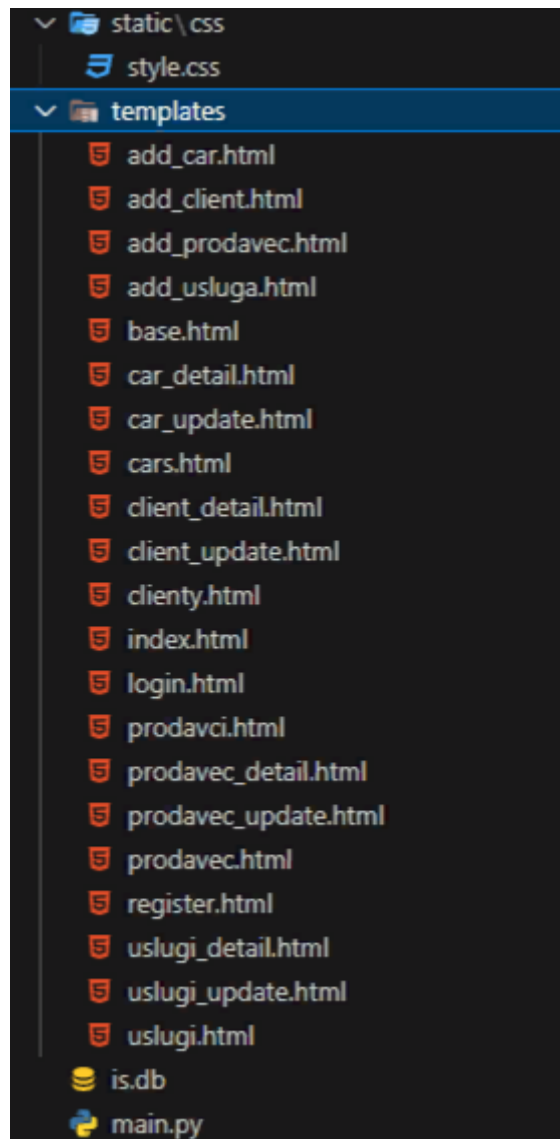


Рис.3.2 - Структура проекта веб-приложения

Папка `static` включает в себя файл `style.css`, в котором созданы все `css`-стили, используемые в верстке данного проекта. Папка `templates` включает в себя все шаблоны и `html` страницы, также используемые в верстке. Совокупность содержимого этих папок, отвечает за реализацию `Front-end` части разработки веб-приложения. В свою очередь, содержимое файла `main.py` отвечает за всю логику и обработку данных, одним словом, за реализацию `Back-end` части разработки. Файл `is.db` является реляционной базой данных, управляемой СУБД `SQLite`, используемой в проекте.

Рассмотрим некоторые функции предоставляемые `Flask`. Для обработки поступающих запросов и отправки ответов на клиентскую часть пользователю, используются декораторы, превращающие обычную функцию `Python` в

функцию просмотра фреймворка Flask. Декоратор конвертирует возвращаемое значение функции в ответ HTTP, который в свою очередь, отображается браузером. В Flask декоратор `route`, используется для связи URL адреса и функции. Именно такие декораторы повсеместно использовались в разработке веб-приложения автомобильного салона. Один из примеров маршрутизации URL адреса «/clienty/...» в проекте изображен на рисунке 3.3

```
@app.route('/clienty/<int:id>')
@login_required
def client_detail(id):
    client_det = Clients.query.get(id)
    return render_template("client_detail.html", client_det = client_det)
```

Рис.3.3 - Декоратор функции для URL маршрутизации

Для упрощения работы с HTML, в Flask реализована вспомогательная функция `render_template()`, позволяющая задействовать шаблонизатор Jinja2. Использование шаблонов позволяет задавать динамическое содержимое в стандартном HTML-коде и наследовать его в любых используемых страницах, сокращая объем написанного кода. Для генерации URL в Flask существует функция `url_for()`, позволяя не задавать его каждый раз вручную. В качестве инструмента для создания ответа от сервера используется функция `make_response()`. А для реализации всплывающих сообщений, обеспечивающих обратную связь с пользователем, фреймворк предоставляет функцию `flash()`. Все эти функции были использованы в ходе разработки веб-приложения ИС автомобильного салона.

Для создания таблиц базы данных, в проекте использовались модели. Модель является классом в Python, представляющим собой таблицу БД. Атрибуты модели соотносятся со столбцами таблицы, ее класс наследуется из `db.Model` определяя колонки в виде экземпляров класса `db.Column`. На рисунке 3.4, в качестве примера, демонстрируется объявления существующей в ИС модели таблицы `Cars`.

```
class Cars(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    type_car = db.Column(db.String(50), nullable=False)
    name_car = db.Column(db.String(50), nullable=False)
    price = db.Column(db.String(10), nullable=False)
```

Рис. 3.4 - Объявление модели таблицы `Cars`

В ходе разработки интерфейса клиентской части были использованы стандартные методы HTML и CSS верстки, с применением CSS-фреймворка Bootstrap, для импортирования иконок и стилей верстки веб-страниц.

Рассмотрев основные методы, которые использовались в ходе реализации веб-приложения, проведем обзор интерфейса пользователя реализованного проекта. Интерфейс, как и функционал, у каждого автосалона может отличаться исходя из требований конкретного заказчика. Данная реализация проекта, является одним из возможных вариантов. Запуск приложения осуществляется локально, используя встроенный функционал Flask, через порт 5000.

При входе в ИС автосалона, пользователь попадает на страницу авторизации и ему предлагается ввести свой логин и пароль. В системе осуществляется хеширование паролей, в целях безопасности данных. Также авторизованный пользователь не имеет возможности взаимодействовать с навигационной панелью веб-приложения, это реализовано декоратором функции `login_required`. Если пользователь еще не был зарегистрирован, необходимо перейти по ссылке «Зарегистрироваться». Страница авторизации, представлена на рисунке 3.6. Перейдя по ссылке «Зарегистрироваться», пользователь попадает на страницу регистрации. Так как ИС автосалона, реализована в виде веб-приложения, любой пользователь интернета может попытаться зарегистрироваться в системе. Во избежание подобных ситуаций, для регистрации новых администраторов системы, существуют зарезервированные логины, сообщаемые непосредственно конкретному администратору перед регистрацией. Страница регистрации, представлена на рисунке 3.6

Авторизация

Пожалуйста, заполните поля "Логин" и "Пароль"

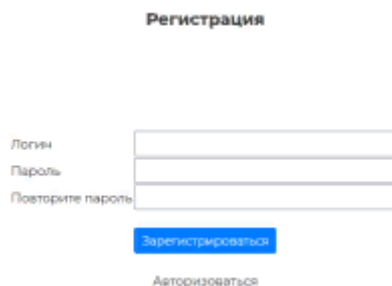
Логин

Пароль

[Авторизоваться](#)

[Зарегистрироваться](#)

Рис. 3.5 – Страница авторизации



Регистрация

Логин

Пароль

Повторите пароль

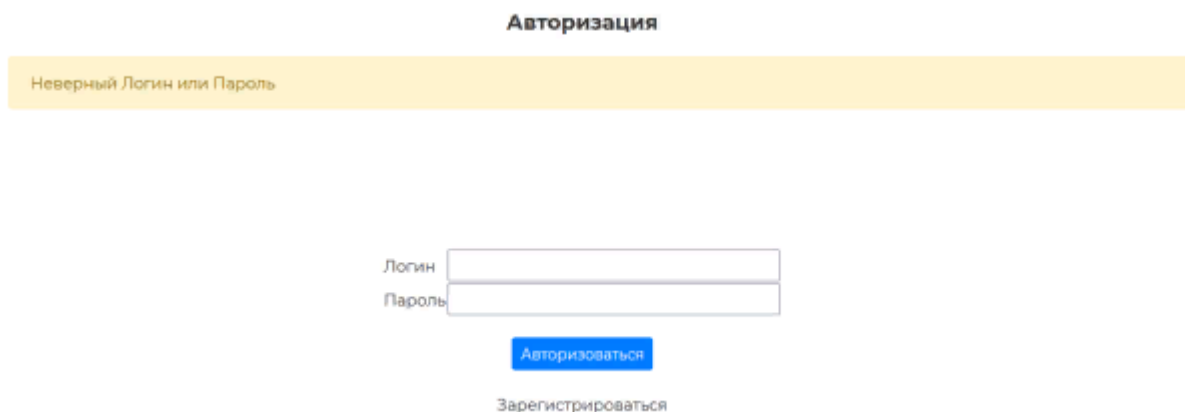
Зарегистрироваться

Авторизоваться

The registration form consists of three input fields for 'Логин' (Login), 'Пароль' (Password), and 'Повторите пароль' (Repeat password). Below the fields are two buttons: 'Зарегистрироваться' (Register) in blue and 'Авторизоваться' (Log in) in grey. The title 'Регистрация' is centered at the top.

Рис. 3.6 – Страница регистрации

Как для авторизации, так и для регистрации, предусмотрены появляющиеся уведомления, служащие для обратной связи с пользователем, в случаях, неверно введённых логина и пароля, или при попытках зарегистрироваться не по зарезервированному логину администратора. Варианты уведомлений представлены на рисунке 3.8.



Авторизация

Неверный Логин или Пароль

Логин

Пароль

Авторизоваться

Зарегистрироваться

The authorization form features a yellow error banner at the top with the text 'Неверный Логин или Пароль' (Incorrect Login or Password). Below the banner are two input fields for 'Логин' (Login) and 'Пароль' (Password). At the bottom are two buttons: 'Авторизоваться' (Log in) in blue and 'Зарегистрироваться' (Register) in grey. The title 'Авторизация' is centered above the input fields.

Рис. 3.7 – Уведомление при некорректном логине или пароле

После успешного входа в систему, пользователь может взаимодействовать с навигационной панелью, на которой представлены такие разделы как: «Клиенты», «Работники», «Машина», «Услуги» и «Выход» (возвращающий пользователя на страницу авторизации) и логотип автомобильного салона. Для удобства пользователя, при наведении курсором мыши на ссылки разделов и любых активных элементов системы, реализована анимация изменения отображения элемента, что представлено на рисунке 3.8.



Рис. 3.8 – Изменение отображения активного элемента при наведении курсора
мышы

Далее более подробно рассмотрим интерфейс пользователя и функционал для разделов навигационной панели ИС. Интерфейс раздела «Клиенты» представлен на рисунке 3.9.

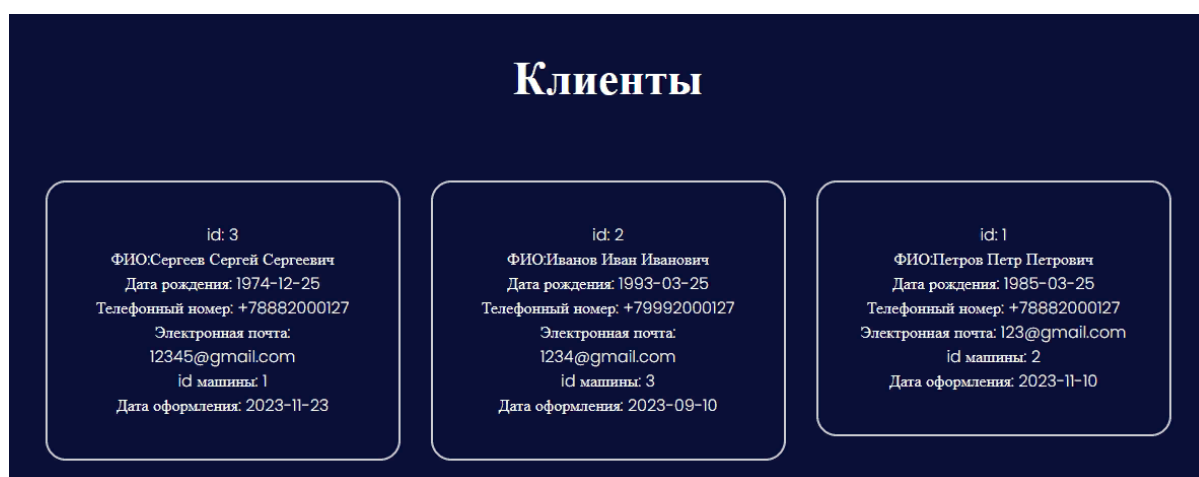


Рисунок 3.9 – Интерфейс раздела «Клиенты»

Данные клиентов представлены в виде карточек, которые демонстрируются в порядке последнего добавленного клиента. В каждой карточке клиента, есть рамка, нажав на которую пользователь ИС будет перемещен на страницу, в которой отображаются данные только выбранного клиента и появляется возможность редактирования или удаления данных (изображено на рисунке 3.10).



Рис. 3.10 – Интерфейс детализации определенного клиента

Добавление клиента

Введите ФИО клиента
Введите дату рождения клиента
Введите номер клиента[*7](_)_(_)-(_)-(_)
Введите адрес электронной почты клиента
Введите id абонента

Добавить

Рис. 3.11 – Интерфейс страницы добавления клиента

При добавлении, редактировании и удалении клиентов появляется уведомление об успешном выполненном действии. Подобные уведомления, появляются и в других разделах ИС. Приведем в качестве примера, вариант уведомления после редактирования данных клиента (изображено на рисунке 3.12)

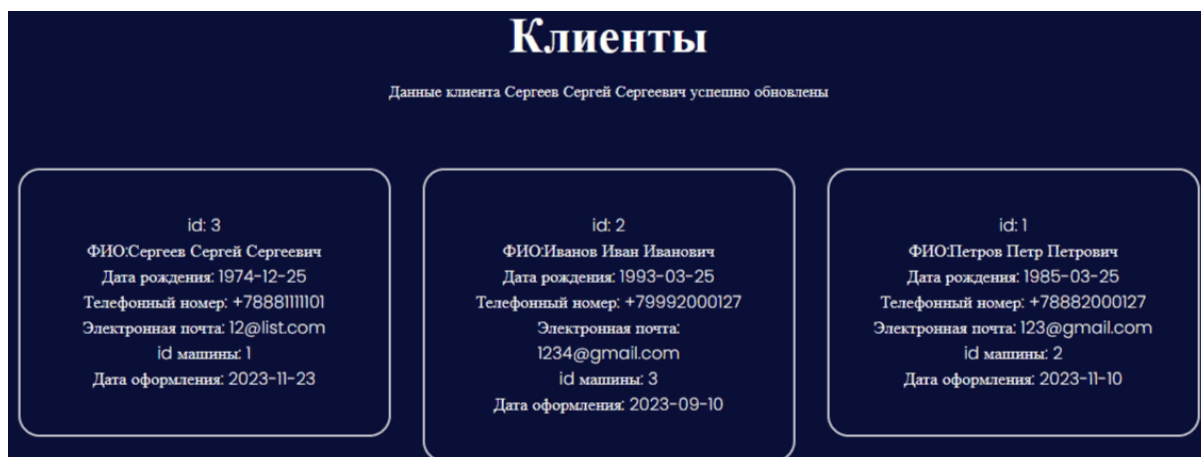


Рис. 3.12 – Уведомление о успешном редактировании данных клиента

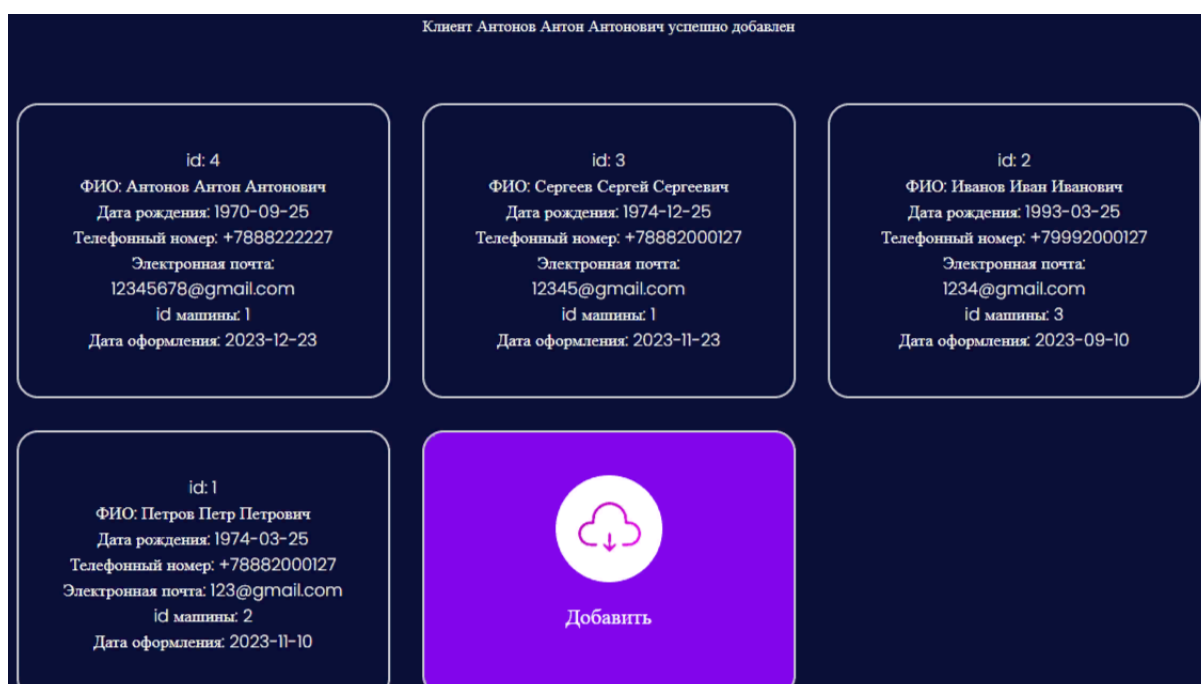


Рис. 3.13 - Добавление клиента

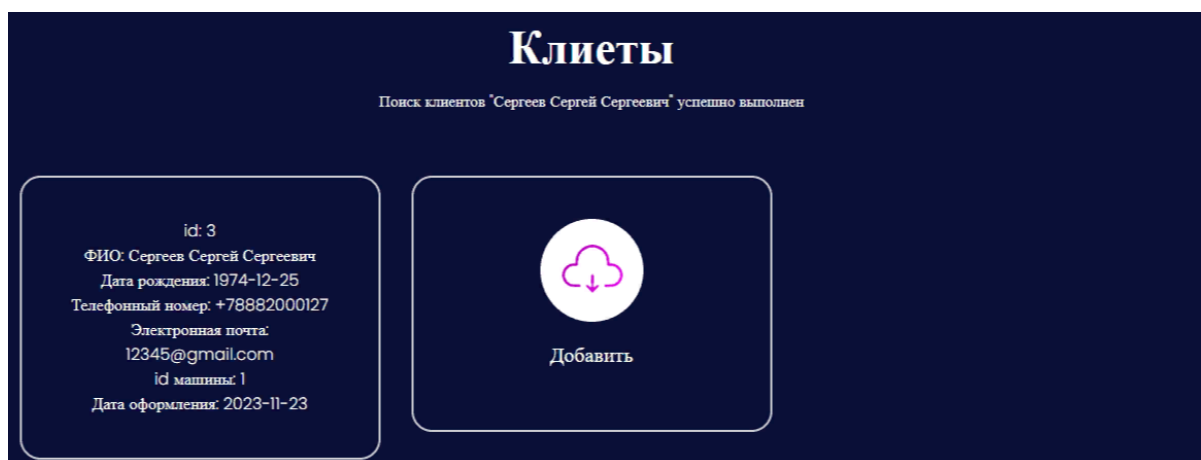


Рис. 3.14 - Поиск клиента

Подробно рассмотрев типичный для всех разделов ИС функционал на примере раздела «Клиенты», в заключении, продемонстрируем интерфейс оставшихся модулей: «Работники» (рисунок 3.13), «Машины» (рисунок 3.14) и «Услуги» (рисунок 3.15)

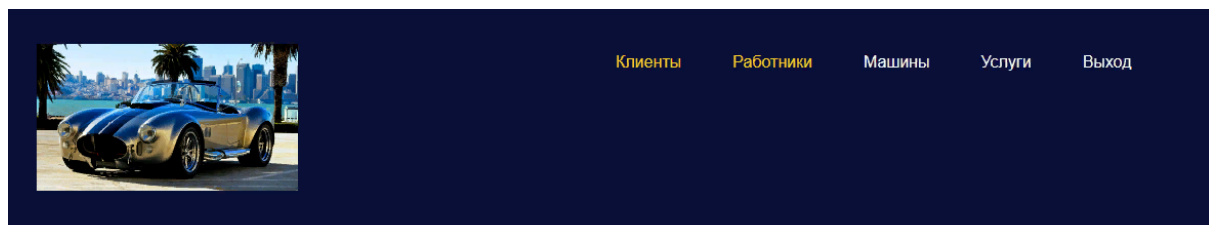


Рис.3.15 – Интерфейс раздела «Работники»



Рис.3.16 – Интерфейс раздела «Машины»

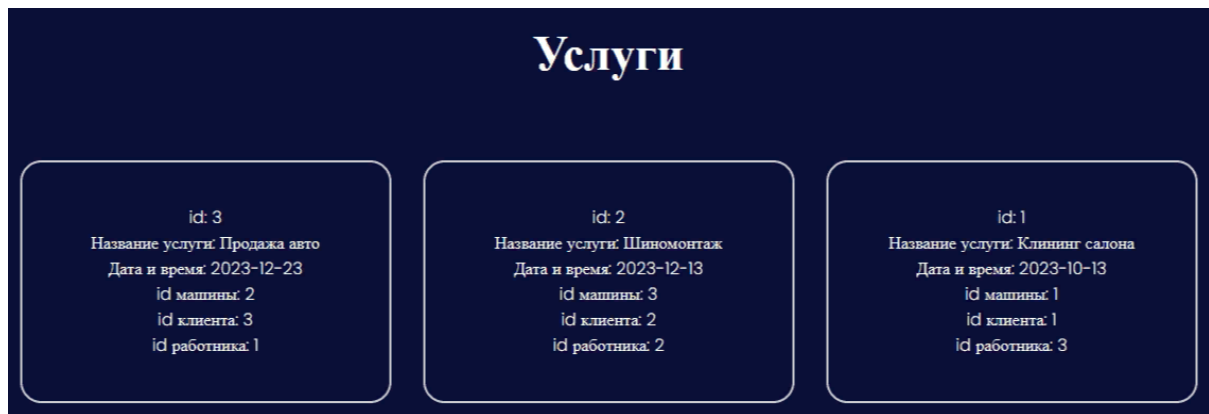


Рис.3.17 – Интерфейс раздела «Услуги»

Итогом реализации ИС автомобильного салона, является описанное в данном разделе курсовой работы веб-приложение, которое в полной мере соответствует выделенным на этапе проектирования системы требованиям.

4. ТЕСТЫ

Готовая система должна пройти тест основных функций, а именно:

- 1.1)Регистрация новых пользователей;
 - 1.2)Авторизация уже зарегистрированных пользователей;
 - 2.1)Добавление нового клиента
 - 2.2)Редактирование данных клиента
 - 2.3)Добавление нового работника
 - 2.4)Редактирование данных работника
 - 2.5)Добавление новой машины
 - 2.6)Редактирование данных о машине
 - 2.7)Добавление новой услуги
 - 2.8)Редактирования данных об услуге
 - 3.1))Поиск в базе данных уже имеющихся в системе клиентов
 - 3.2)Поиск в базе данных уже имеющихся в системе работников
 - 3.3)Поиск в базе данных уже имеющихся в системе машин
 - 3.4)Поиск в базе данных уже имеющихся в системе услуг
 - 4.1)Просмотр клиентов
 - 4.2)Просмотр работников
 - 4.3)Просмотр машин
 - 4.4)Просмотр услуг
- Выход из системы

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы на тему "Разработка информационной системы для автосалона" были выделены следующие этапы задач: анализ предметной области и методов реализации проекта, проектирование информационной системы, ее последующая разработка.

При проведении анализа предметной области и методов реализации проекта рассмотрена актуальность создания информационной системы в сфере автомобильного бизнеса. Вывод был сделан о том, что разработка информационной системы для автосалона является важной задачей и неотъемлемым условием конкурентоспособности в предоставлении услуг клиентам.

После рассмотрения возможных методов реализации ИС, был выбран вариант клиент-серверной архитектуры веб-приложения, реализованного на языке программирования Python с использованием фреймворка Flask, SQLAlchemy и СУБД SQLite.

В результате проектирования были определены требования к системе, построена функциональная модель информационной системы автосалона, а также определена схема хранения и взаимосвязи данных. С использованием настроенной среды разработки были реализованы Back-end и Front-end части приложения. В перспективе данная разработка, после анализа требований и корректировок функционала и интерфейса под конкретного заказчика, может быть успешно внедрена в любые автосалоны с целью улучшения эффективности работы административного персонала.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Flask [Электронный ресурс].
URL: <https://flask.palletsprojects.com/en/2.0.x/>
2. Документация SQLAlchemy [Электронный ресурс].
URL: <https://docs.sqlalchemy.org/en/14/>
3. Документация SQLite [Электронный ресурс].
URL: <https://www.sqlite.org/docs.html>

4. Документация Python [Электронный ресурс].

URL: <https://docs.python.org/3/>