Содержание

[Введение 5](#_Toc185070221)

[1. Анализ технического задания 6](#_Toc185070222)

[1.1. Описание предметной области 9](#_Toc185070223)

[1.2. Аналоги 10](#_Toc185070224)

[2. Модели данных 13](#_Toc185070225)

[2.1. Концептуальная модель 13](#_Toc185070226)

[2.2. Логическая модель 18](#_Toc185070227)

[2.3. Приведение логической модели к физической 21](#_Toc185070228)

[3. Разработка и реализация АИС 30](#_Toc185070229)

[3.1. Создание SQL-запросов 30](#_Toc185070230)

[3.2. Руководство пользователя 33](#_Toc185070231)

[3.3. Руководство программиста 43](#_Toc185070232)

[4. Тестирование АИС 51](#_Toc185070233)

[Заключение 61](#_Toc185070234)

[Список литературы 62](#_Toc185070235)

[Приложение А: «Модели данных» 63](#_Toc185070236)

[Приложение Б: Текст кода 66](#_Toc185070237)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | МИВУ 10.03.01-14ПЗ | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум | Подпись | Дата |
| Разраб | | Фадеев Н. Д. |  |  | АРМ администратора салона по аренде автомобилей | Литера | | | Лист | Листов |
| Пров | | Колпаков А. А. |  |  |  | y |  | 4 | 66 |
|  | |  |  |  | МИВлГУ  ИБ-122 | | | | |
| Н. Контр. | |  |  |  |
| Утв | | Орлов А. А. |  |  |

# Введение

Большой выбор автомобилей, предлагаемых к прокату, необходимость хранения данных о большом количестве производителей, поставщиков и клиентов усложняют задачу учета поставок, продаж и остатков автомобилей в автосалоне. Ручная обработка этих данных не позволяет получить оперативную информацию об уменьшении количества автомобилей, пользующихся спросом и, следовательно, необходимости пополнения этого числа автомобилей. Расчет прибыли, затрат, рентабельности, поиск решений, приводящих к повышению эффективности работы автосалона, становятся трудоемкими. В любой организации, как большой, так и маленькой, возникает проблема такой организации управления данными, которая бы обеспечила наиболее эффективную работу. Некоторые организации используют шкафы для хранения папок с документами и ручной поиск для обработки необходимой информации. Но большинство предпочитают компьютеризированные базы данных и программы, позволяющие эффективно хранить, быстро извлекать нужную информацию и управлять большими объёмами данных. Большое количество информации, высокие требования к точности, многочисленные вычисления, потребность в постоянном обновлении данных делают необходимым применение баз данных для учета автомобилей в автосалоне.

Целью данной курсовой работы является создание информационной системы (далее ИС) для работы с базой данных (далее БД) автосалона, которая будет обеспечивать удобное взаимодействие пользователей и персонала.

При работе с данной базой данных каждый клиент сможет получить информацию об интересующем его автомобиле.

Для достижения этой цели я поставил перед собой ряд задач:

1. Сравнить производительность систем управления базами данных (далее СУБД) и средств программирования.
2. Разработать модели данных.
3. Разработать базу данных.
4. Создать многопользовательское приложение.

# Анализ технического задания

Современная разработка информационных систем требует выбора инструментов, которые наилучшим образом соответствуют требованиям проекта. В контексте создания системы управления базой данных для автосалона я рассмотрю и сравню несколько технологий, чтобы понять, какие из них оптимально использовать.

Этот анализ охватывает языки программирования C# и C++, среды разработки Visual Studio и Eclipse, а также системы управления базами данных SQLite и MySQL. Введение этих сравнений поможет выявить наиболее подходящие инструменты, учитывая их производительность, простоту использования, гибкость, безопасность и интеграцию с другими системами. Это обеспечит надежность и эффективность системы управления базой данных, поддерживая высокие стандарты разработки и эксплуатации.

1. Характеристики C++:

Производительность: очень высока, поскольку это язык низкого уровня, позволяет манипулировать памятью и системными ресурсами. [1]

Гибкость: предоставляет детальный контроль над аппаратным обеспечением. [1]

Использование: широко используется для разработки системного ПО, драйверов и высокопроизводительных приложений. [1]

1. Характеристики C#:

Простота: более высокого уровня, чем C++, более легкий для изучения и использования. [3]

Платформа: тесно интегрирован с .NET, что облегчает разработку веб-приложений и бизнес-логики. [3]

Безопасность: предоставляет автоматическое управление памятью (сборка мусора), что снижает вероятность ошибок, связанных с памятью. [3]

1. Характеристики Visual Studio:

Функциональность: обширный набор инструментов для отладки, тестирования и разработки. [6]

Интеграция: отлично интегрируется с экосистемой Microsoft, поддерживает множество языков программирования. [6]

Использование: очень удобная и интуитивно понятная среда. [6]

4. Характеристики Eclipse:

Открытый исходный код: бесплатная и доступная, поддерживаемая сообществом. [2]

Кроссплатформенность: работает на различных операционных системах (Windows, macOS, Linux). [2]

Плагины: огромное количество доступных плагинов для расширения функциональности. [2]

5. Характеристики SQLite

Легковесность: не требует установки и настройки сервера. [5]

Простота: легка в использовании и имеет компактную архитектуру. [5]

Встроенная СУБД: отлично подходит для приложений, где требуется локальное хранилище данных. [5]

6. Характеристики MySQL

Производительность: высокая производительность и масштабируемость для крупных проектов. [4]

Функциональность: поддерживает сложные запросы, транзакции и интеграцию с другими СУБД. [4]

Популярность: широко используется в веб-разработке и поддерживается большим сообществом. [4]

Результат анализа характеристик можно представить в виде таблиц: 1 – таблица сравнения языков программирования, 2 – таблица сравнения сред, 3 – таблица сравнения СУБД.

Таблица 1 – Сравнение языков программирования

|  |  |  |
| --- | --- | --- |
| Критерий | C++ | C# |
| Производительность | Высокая | Средняя |
| Простота | Сложная | Простая |
| Гибкость | Высокая | Средняя |
| Интеграция | Ограниченная | Отличная с .NET |
| Безопасность | Низкая (ручное управление памятью) | Высокая (автоматическая сборка мусора) |

Таблица 2 – Сравнение сред программирования

|  |  |  |
| --- | --- | --- |
| Критерий | Visual Studio | Eclipse |
| Производительность | Высокая | Высокая |
| Простота | Удобная | Средняя |
| Гибкость | Высокая | Высокая |
| Интеграция | Отличная с экосистемой MS | Кроссплатформенная |
| Безопасность | Высокая | Высокая |

Таблица 3 – Сравнение СУБД

|  |  |  |
| --- | --- | --- |
| Критерий | SQLite | MySQL |
| Производительность | Высокая | Высокая |
| Простота | Простая | Средняя |
| Гибкость | Низкая | Высокая |
| Интеграция | Локальное хранилище данных | Веб-разработка |
| Безопасность | Высокая | Высокая |

В данном конкретном случае C#, Visual Studio и SQLite являются оптимальными выбором. Они предлагают следующие преимущества:

• C#: Простота использования, высокая производительность, отличная совместимость с .NET и высокая безопасность. [3]

• Visual Studio: Мощная и интуитивно понятная интегрированная среда разработки, поддержка множества языков и богатый набор инструментов. [6]

• SQLite: Легковесная и простая в использовании СУБД, идеально подходящая для встроенных систем и приложений с локальным хранением данных. [5]

Эти инструменты обеспечат стабильную, безопасную и эффективную работу системы управления базой данных для автосалона.

## Описание предметной области

Предметная область — часть реального мира, данные о которой мы хотим отразить в базе данных. В данной курсовой работе рассматривается предметная область «Салон по аренде автомобилей».

Основной задачей является проектирование и разработка приложения для автоматизации оформления аренды. Предполагаемая база данных должна обеспечивать работу автосалона по учету аренды автомобилей различных видов.

Процесс аренды автомобиля осуществляется следующим образом:

1) Поиск клиентов. Встреча с потенциальными покупателями для выяснения их бюджета и желаний. Эту задачу выполняют продавцы-консультанты.

На данном этапе клиент знакомится с ассортиментом автосалона в зависимости от своих возможностей и желаний.

2) Аренда автомобиля. При аренде автомобиля менеджер регистрируют клиента в базе.

Для аренды клиенту необходимо предоставить номер телефона, e-mail, а также свои паспортные данные.

3) Возможное событие «Автомобиль недоступен». В данной ситуации клиент не может арендовать автомобиль т. к. на данный момент он отсутствует в автосалоне.

4) Заключение сделки может оформлять менеджер.

В автосалоне есть свой спектр сотрудников: менеджеры по продажам и администратор. Менеджеры оформляют договоры по аренде, администратор контролирует работу системы.

Клиент, приходя в автосалон взаимодействует с менеджером, и исходя из его запроса тот оформляет сделку. При аренде машины клиент получает договор купле-продажи с датой, маркой авто, условиями оплаты, номером продавца, заключившим сделку и сумму сделки.

Исходя из данных требований к приложению, база данных должна содержать несколько таблиц и данные таблицы должны быть связаны для целостности системы.

Для того чтобы создать программу, необходимо учесть то, что она создается, прежде всего, для пользователя, и поэтому немаловажным требованием к программе должен стать удобный и интуитивно понятный интерфейс. Необходимо предусмотреть все возможности управления приложением, чтобы упростить работу пользователя и максимально обеспечить эффективность работы.

Программа должна правильно работать с данными, т.е. всегда должен выводится нужный результат, требуемый пользователю. Приложение должно мгновенно реагировать на действия пользователя и в зависимости от запроса с его стороны формировать выходной результат.

## Аналоги

В настоящее время реализовано огромное количество АС для автосалонов. Примером такой системы может являться Kayak.com - онлайн-агрегатор для поиска и бронирования автомобилей:

Достоинства:

* Удобный интерфейс для поиска и сравнения предложений от различных автодилеров.
* Возможность онлайн-бронирования и оплаты выбранного автомобиля.
* Интеграция с другими сервисами для бронирования отелей, авиабилетов и т.д.
* Сбор и анализ больших объемов данных для улучшения предложений пользователям.

Недостатки:

* Комиссии за использование платформы, которые платят автодилеры.
* Необходимость поддержания актуальной информации об автомобилях и ценах.
* Конкуренция с другими агрегаторами за привлечение клиентов.

В качестве другого примера можно привести 1С:Автодилер – ПО, предназначенное для автоматизации учета, планирования и анализа работы любых автопредприятий: крупных и мелких автомастерских, автосалонов, магазинов автозапчастей, автомоек, шиномонтажных мастерских и станций замены масла, автостраховщиков.

Достоинства:

* Полная автоматизация ключевых бизнес-процессов автосалона.
* Единая база данных автомобилей, клиентов, сотрудников, финансов.
* Гибкие инструменты управления продажами, сервисом и маркетингом.
* Развитая аналитика, отчетность и панели мониторинга.
* Возможность интеграции с внешними системами.

Недостатки:

* Значительные первоначальные затраты на внедрение и обучение.
* Сложность интеграции с существующими IT-системами автосалона.
* Необходимость перестройки бизнес-процессов под возможности системы.
* Зависимость от стабильной работы системы и риски простоев.

Особенности:

* Модули:
  + Управление складом, закупками и поставками автомобилей.
  + Управление продажами, маркетингом и взаимоотношениями с клиентами.
  + Управление сервисным обслуживанием и запчастями.
  + Управление финансами, бухгалтерией и кадрами.
* Веб-интерфейс для клиентов и мобильные приложения для сотрудников.
* Аналитическая подсистема с гибкими отчетами и панелями мониторинга.

# Модели данных

Данный этап является самым важным при создании АИС. Здесь выделяются сущности, атрибуты сущностей и связи между сущностями. На основе полученной диаграммы “Сущность – связь” или логической модели строятся функциональные модели системы и диаграмма потоков данных. Для создания базы данных, нужно логическую модель представить в виде физической.

Разработка моделей данных включает 3 этапа: [7]

1) Концептуальная модель:

- определение сущностей (объектов, понятий) предметной области и их взаимосвязей;

- выявление атрибутов сущностей и описание их характеристик;

- построение диаграммы сущность-связь (ER-диаграммы) для визуального представления модели.

2) Логическая модель:

- трансформация концептуальной модели в структуру, подходящую для конкретной СУБД;

- определение типов данных, ключей, индексов, ограничений целостности;

- нормализация данных для устранения избыточности и аномалий;

- спецификация отношений между сущностями (таблицами).

3) Физическая модель:

- оптимизация логической модели для конкретной СУБД и аппаратной платформы;

- определение физического хранения данных: файлов, томов, буферов и т.д.;

- результат - физическая схема базы данных, готовая для реализации.

## 2.1. Концептуальная модель

Концептуальная модель данных является первым и наиболее абстрактным уровнем проектирования базы данных. На этом этапе определяются ключевые сущности предметной области, их атрибуты и взаимосвязи. [7]

В ходе анализа процесса аренды автомобиля, были выделены следующие базовые сущности организации и их атрибуты:

1) Персонал организации.

Атрибуты персонала: ID, Фамилия, Имя, Отчество, Должность, Телефон, Почта. Они представлены на рисунке 1.

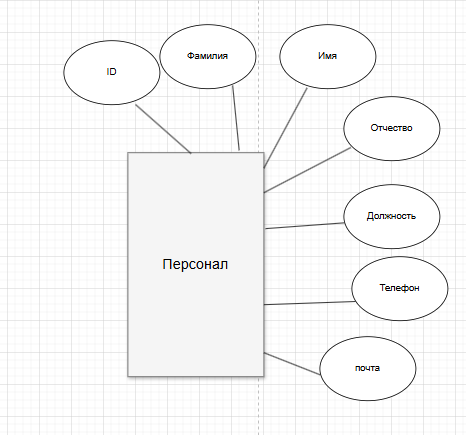


Рис. 1 – Атрибуты персонала

2) Автомобили.

Атрибуты автомобилей: ID, Страна производитель, Марка, Название, Цвет, Цена, Доступность автомобиля представлены на рисунке 2.

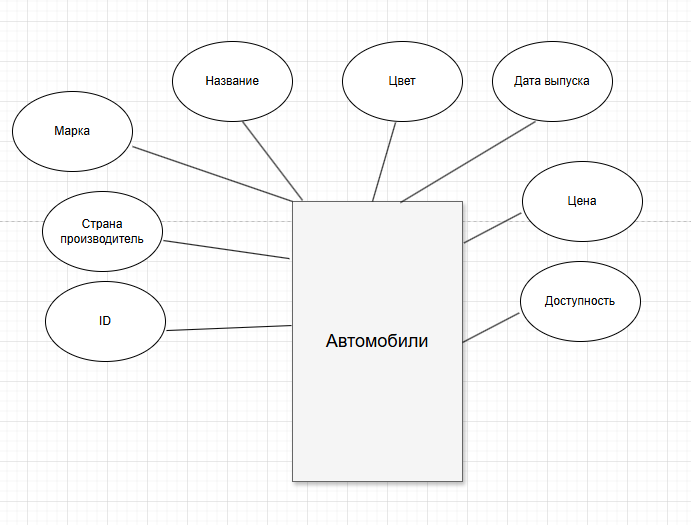


Рис. 2 – Атрибуты автомобилей

3) Договор.

Атрибуты договора: ID, Серия паспорта клиента, Номер паспорта клиента, Дата, Стоимость, ID клиента, ID автомобиля, ID сотрудника, Время начала аренды,   
Время окончания аренды можно увидеть на рисунке 3.

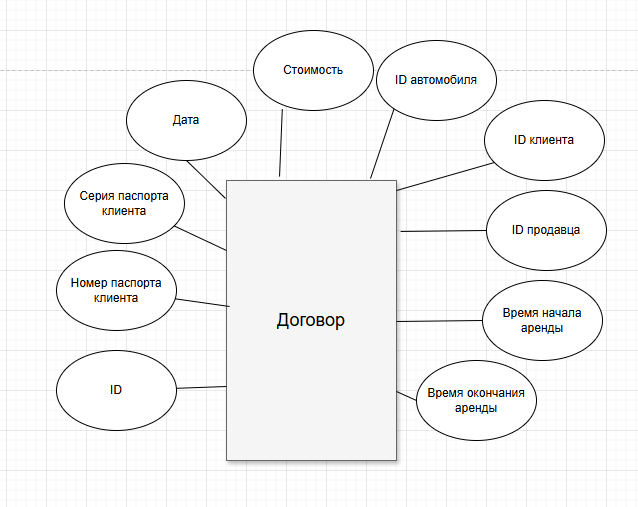


Рис. 3 – Атрибуты договора

4) Клиенты.

Атрибуты клиента: ID, № паспорта, Серия паспорта, Фамилия, Имя, Отчество, Телефон изображены на рисунке 4.

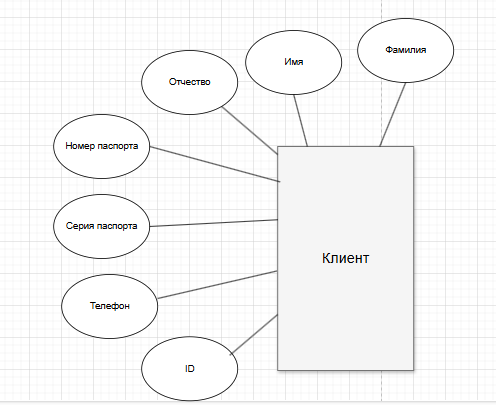


Рис. 4 – Атрибуты клиента

Все выделенные сущности были объединены в общую концептуальную модель данных, которая представлена на рисунке 5.

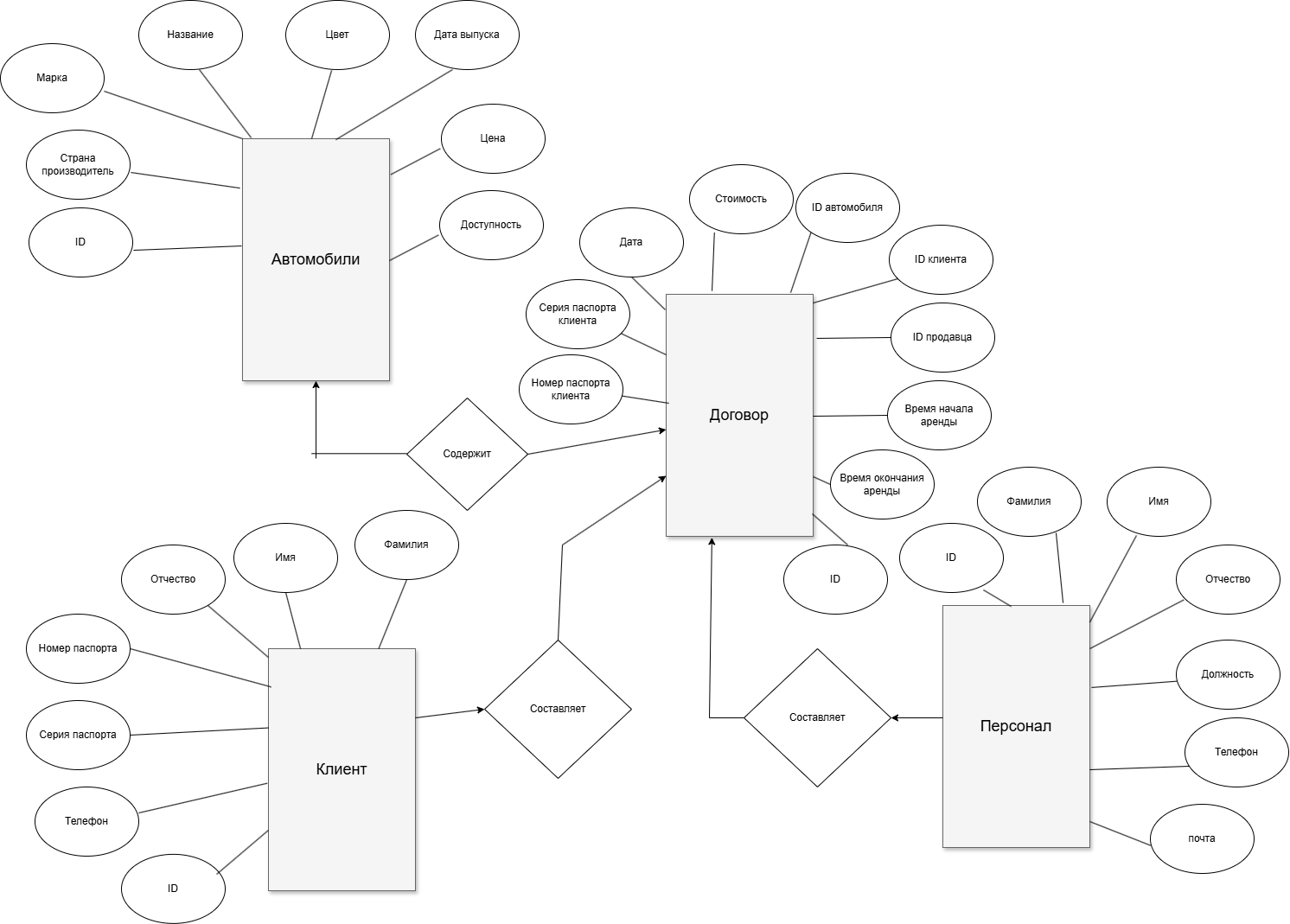


Рис. 5 – Концептуальная модель данных.

Между всеми таблицами существуют связи один ко многим.

## 2.2. Логическая модель

Логическая модель данных — это абстрактное представление структуры данных, которое используется для планирования и проектирования баз данных. [7]

Ее создание начинается с анализа предметной области и выделения сущностей.

Данная логическая модель данных отражает основные сущности, связанные с

автосалоном: клиенты, договор, автомобили и персонал. Она позволяет хранить и управлять данными, необходимыми для эффективного функционирования автосалона.

Основные сущности и их атрибуты:

Клиенты:

* + ID – Уникальный идентификатор клиента, первичный ключ
  + Фамилия
  + Имя
  + Отчество
  + Серия\_паспорта
  + Номер\_паспорта
  + Телефон

Договор:

* + ID\_Клиента – внешний ключ
  + ID\_Персонала – внешний ключ
  + ID\_Автомобиля – внешний ключ
  + ID
  + Серия\_паспорта\_клиента
  + Номер\_паспорта\_клиента
  + Стоимость
  + Дата\_оформления\_договора
  + Время\_начала\_аренды
  + Время\_окончания\_аренды

Автомобили:

* + ID – уникальный идентификатор, первичный ключ
  + ID\_Страны – уникальный идентификатор страны, внешний ключ
  + ID\_Авто – уникальный идентификатор авто, внешний ключ
  + Цена
  + Доступность

Страна\_производитель

* ID – уникальный идентификатор, первичный ключ
* Страна

Характеристики

* ID – уникальный идентификатор, первичный ключ
* Марка
* Название
* Цвет

Персонал:

* + ID – уникальный идентификатор, первичный ключ
  + Фамилия
  + Имя
  + Отчество
  + Должность
  + Почта
  + Телефон

Архив – аналогична таблице «Договор», только вместо ID сотрудников, клиентов и автомобилей, в ней хранятся удаленные договоры с уже готовыми данными.

На основе этих данных была составлена логическая модель данных, представленная на рисунке 6.

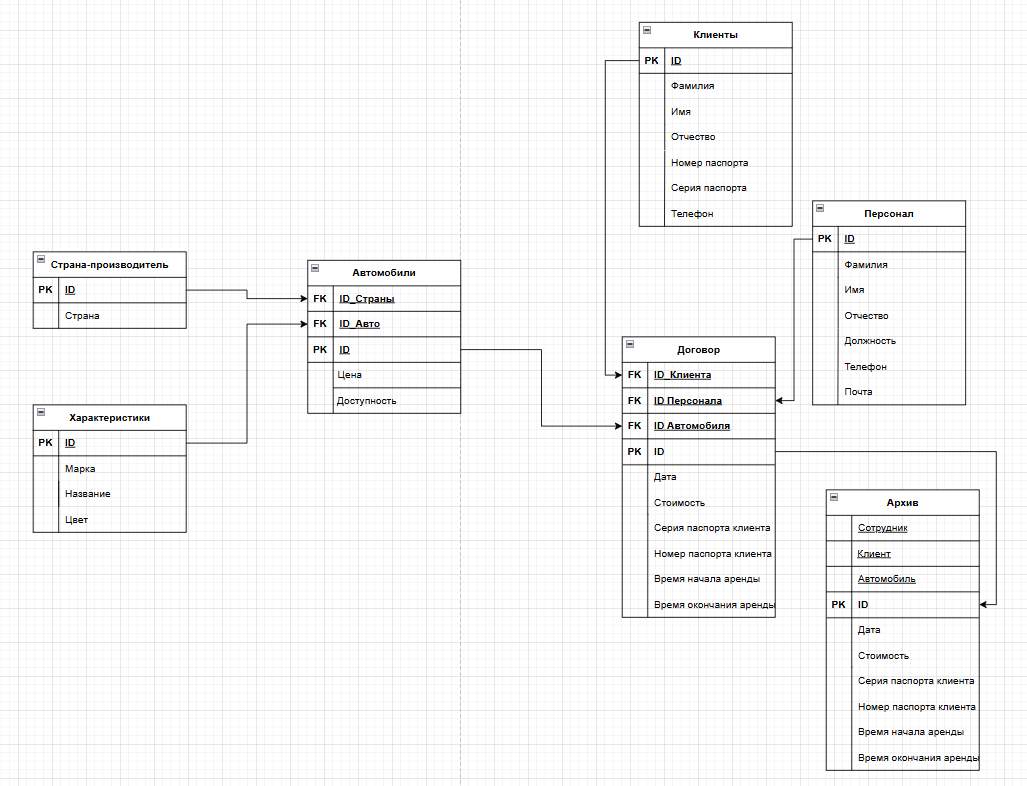


Рис. 6 – Логическая модель данных

## 2.3. Приведение логической модели к физической

Физическая модель данных — это заключительный шаг в процессе моделирования данных. [7] Она представляет фактические детали реализации в конкретной системе управления базами данных (СУБД).

К основные задачи физической модели данных относится:

1) Определение физических характеристик таблиц и полей:

- Типы данных для каждого атрибута (integer, varchar, date и т.д.)

- Размеры полей (длина строк)

- Ограничения целостности данных (первичные/внешние ключи, уникальность, проверка значений)

2) Проектирование физической структуры базы данных:

- Определение физического хранения данных (файлы, тома, блоки)

- Настройка индексов для ускорения доступа к данным

- Решения по кластеризации таблиц

- Механизмы резервного копирования и восстановления данных

3) Оптимизация производительности:

- Анализ запросов и создание оптимальных индексов

- Денормализация данных для повышения скорости выборки

- Использование материализованных представлений

Для создания этой модели были созданы домены: FIO(varchar(30)), Number(integer(4)), Seria(integer(6)), Telephonenumber(integer(11)). Домен FIO будет применен к полям: «Name», «Surname», «Last\_name» в таблицах «Klients» и «Personal». Домен «Number» к полям «Number» и «Number\_klients» в таблицах «Klients» и «Contract» соответственно. Домен «Seria» к полям «Seria» и «Seria\_klients» в таблицах «Klients» и «Contract» соответственно. Домен «Telephonenumber» к полям «Telephone» в таблицах «Klients» и «Personal».

Физическую модель можно преобразовывать по частям. Поле «ID» для таблицы «Klients» связано с полем «ID\_klients» в таблице «Conract». Данные поля также являются и первичным ключом для таблицы «Klients» и внешним ключом для таблицы «Conract». Связь между двумя таблицами представлена на рисунке 7.

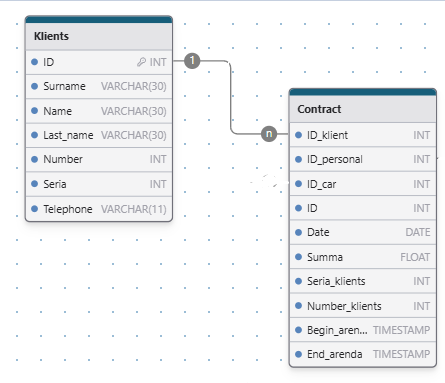


Рис. 7 – Связь между таблицами «Klients» и «Conract»

В таблице «Personal» ключевым также будет поле «ID», оно будет иметь связь с полем «ID\_personal» в таблице «Conract». По аналогии с таблицей «Klients» эти поля будут первичным и внешним ключами. Связь показана на рисунке 8.

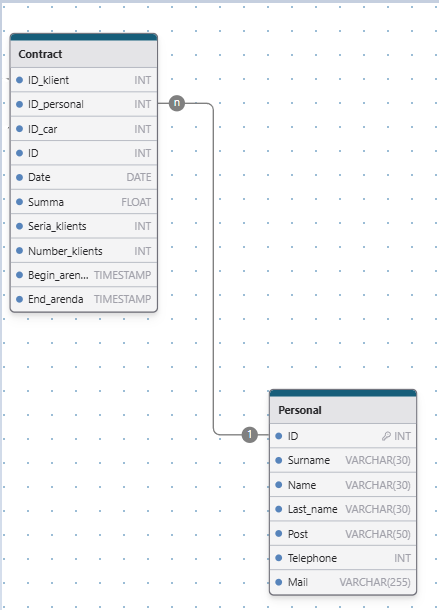


Рис. 8 – Связь между таблицами «Personal» и «Contract»

Связь между таблицами «Cars» и «Contract» организована через поля «ID» и «ID\_car» соответственно. Это можно увидеть на рисунке 9.

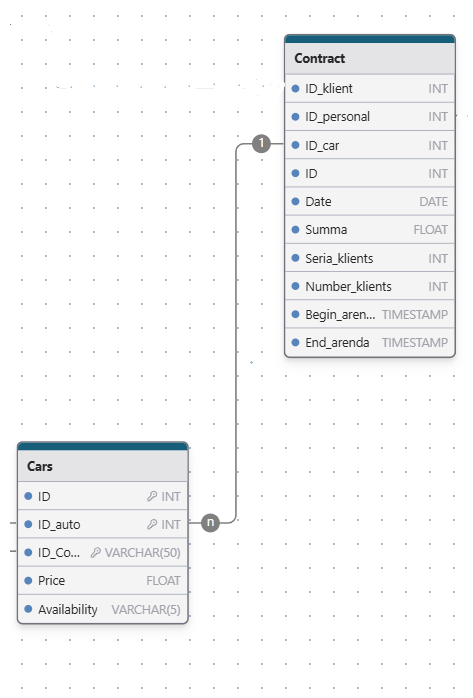


Рис. 9 – Связь между таблицами «Cars» и «Contract»

Также таблица «Cars» связана с еще 2 таблицами: «Specifications» и «Country», связаны они через поля ID в таблицах «Specifications» и «Country», и ID\_auto и ID\_country в таблице «Cars». Это можно увидеть на рисунке 10.

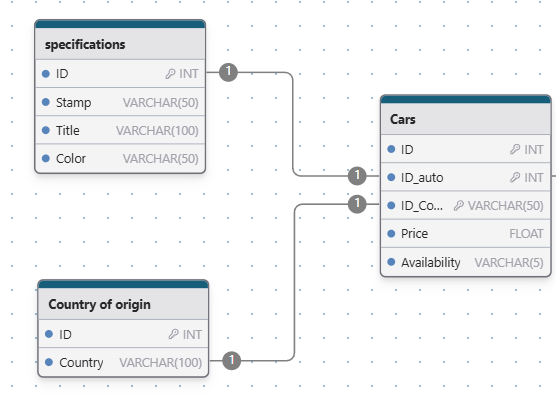


Рис. 10 – Связь между таблицами «Cars», «Specifications» и «Country»

Полная схема физической модели данных представлена на рисунке 11.

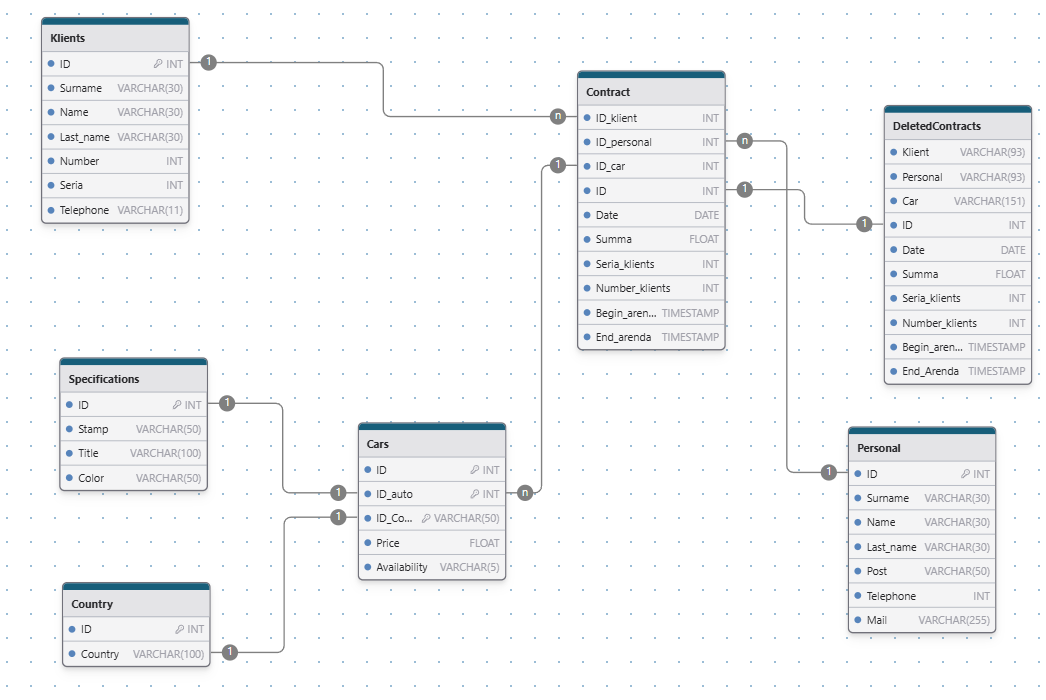
****

Рис. 11 – Физическая модель данных

Таблица "Klients":

ID\_PK (integer) – Уникальный идентификатор, первичный ключ

Surname (FIO(varchar(30)))

Name (FIO(varchar(30))

Last\_name (FIO(varchar(30)))

Seria (Seria(integer(6)))

Number (Number(integer(4)))

Telephone (Telephonenumber(integer(11))

Для данной таблицы могут быть определены:

-первичный ключ ID для уникальной идентификации записей;

-индекс по полю Telephone для ускорения поиска клиентов по номеру;

-ограничение уникальности для Seria + Number и Telephone

Таблица "Conract":

ID\_klients\_FK (integer), внешний ключ на таблицу Klients)

ID\_personal\_FK (integer), внешний ключ на таблицу Personal)

ID\_car\_FK (integer), внешний ключ на таблицу Cars)

ID (integer) - уникальный идентификатор, первичный ключ

Seria\_klient (Seria(integer(6)))

Number\_klient (Number(integer(4)))

Summa (float)

Date (date)

Begin\_arenda (timestamp)

End\_arenda (timestamp)

Ключевые особенности:

-поле ID для уникальной идентификации записей

-связь с таблицами Klients, Personal и Cars через внешние ключи;

-индекс по ID\_klients\_FK для ускорения поиска бронирований клиента;

Таблица "Cars":

ID\_PK (integer) – уникальный идентификатор, первичный ключ

ID\_country (integer, внешний ключ на таблицу Country)

ID\_auto (integer, внешний ключ на таблицу Cars)

Price (float)

Availability (varchar(5))

Ключевые особенности:

-первичный ключ ID для уникальной идентификации записей;

Таблица «Specificatoins»

ID\_FK – Уникальный идентификатор, внешний ключ

Stamp(varchar(50))

Title(varchar(100))

Color(carchar(50))

Ключевые особенности:

-индекс по Stamp и Title для быстрого поиска

Таблица «Country»

ID\_PK (integer) – Уникальный идентификатор, первичный ключ

Country(varchar(50))

Ключевые особенности:

-ограничение Country на уникальность

Таблица "Personal":

ID\_PK (integer) - Уникальный идентификатор, первичный ключ

Surname (FIO(varchar(30))

Name (FIO(varchar(30))

Last\_name (FIO(varchar(30))

Post (varchar(50))

Telephone (Telephonenumber(integer(11))

Mail (varchar(255))

Ключевые особенности:

-первичный ключ ID для уникальной идентификации;

-индекс по Post для быстрого поиска сотрудников определенной должности;

-ограничение Telephone и Mail на уникальность.

Таблица «DeletedContracts»

ID (integer) – Уникальный идентификатор, первичный ключ

Klient (varchar(93))

Personal (varchar(93))

Car (varchar(151))

Date (date)

Summa (integer)

BeginArenda (timestamp)

EndArenda (timestamp)

# Разработка и реализация АИС

## Создание SQL-запросов

SQL-запросы являются основным механизмом взаимодействия с реляционными базами данных. Они позволяют эффективно управлять данными, обеспечивая их хранение, извлечение, модификацию и удаление. [7]

Результатом запроса является представление — виртуальная таблица, представляющая собой поименованный запрос, который будет подставлен как подзапрос при использовании представления. [7] В отличие от обычных таблиц реляционных баз данных, представление не является самостоятельной частью набора данных, хранящегося в базе.

* + 1. Запрос создания таблицы (CREATE TABLE):

Запрос для создания новой таблицы Contract.

CREATE TABLE Contract (ID\_klient INTEGER NOT NULL, ID\_personal INTEGER, ID\_car INTEGER NOT NULL, ID INTEGER NOT NULL PRIMARY KEY, Date TEXT (10), Summa REAL NOT NULL, SeriaKlient INTEGER (4) NOT NULL, NumberKlient INTEGER (6) NOT NULL, BeginArenda TEXT (16) NOT NULL, EndArenda TEXT (16) NOT NULL, FOREIGN KEY (ID\_klient) REFERENCES Klients (ID) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (ID\_car) REFERENCES Cars (ID) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (ID\_personal) REFERENCES Personal (ID) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE Contract - Эта команда создаёт новую таблицу с названием Contract.

ID INTEGER NOT NULL UNIQUE PRIMARY KEY – создается столбец ID с типом данных INTEGER со значением NOT NULL, являющийся первичным ключом PRIMARY KEY.

BeginArenda TEXT (16) NOT NULL – команды данного типа создают столбцы с указанными типами данных.

FOREIGN KEY (ID\_klient) REFERENCES Klients (ID) ON DELETE CASCADE ON UPDATE CASCADE – данный тип команд связывает столбцы через внешние ключи, в данном случае столбец ID\_klient является внешним ключом к таблице Klients, где первичным ключом является столбец ID.

* + 1. Запросы для выборки данных из таблиц (SELECT)
    2. Запрос для вывода данных из таблицы Personal

SELECT ID, Surname AS 'Фамилия', Name AS 'Имя', LastName AS 'Отчество', Post AS 'Должность', Telephone AS 'Телефон', Mail AS 'Электронная почта' FROM Personal

Данный запрос выбирает столбцы в таблице Personal с помощью команды FROM. При помощи ключевого слова AS присваивает им название и выводит их на экран.

Результат выполнения показан на рисунке 12.

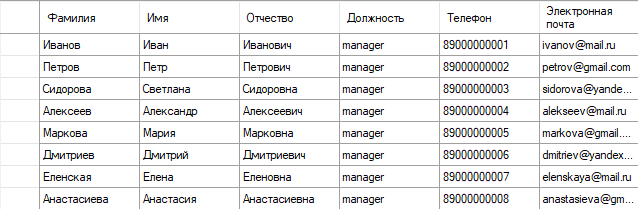


Рис. 12 – Результат выполнения запроса на таблице Personal

* + 1. Запрос для извлечения данных из таблицы Contract и связанных с ней таблиц

SELECT Contract.ID, klients.Surname || ' ' || klients.Name || ' ' || klients.LastName AS 'Клиент', personal.Surname || ' ' || personal.Name || ' ' || personal.LastName AS 'Персонал', specifications.Stamp || ' ' || specifications.Title AS 'Автомобиль', contract.Date AS 'Дата', contract.Summa AS 'Сумма', contract.BeginArenda AS 'Начало аренды', contract.EndArenda AS 'Конец аренды' FROM contract LEFT JOIN klients ON contract.ID\_klient = klients.ID LEFT JOIN personal ON contract.ID\_personal = personal.ID LEFT JOIN cars ON contract.ID\_car = cars.ID LEFT JOIN specifications ON cars.ID\_auto = specifications.ID

В данном запросе помимо методов, описанных в прошлом примере, функция || ' ' || разделяет данные пробелами. LEFT JOIN – объединяет все данные в одну таблицу через внешние ключи.

Результат выполнения показан на рисунке 13.

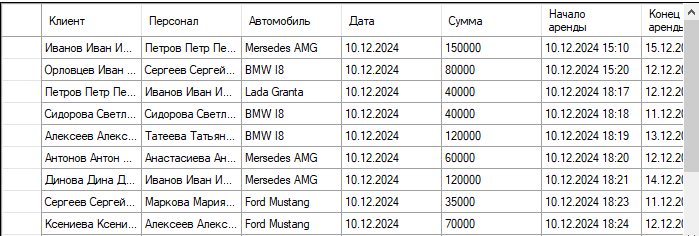


Рис. 13 – Результат выполнения запроса на таблице Contract

* + 1. Запрос для вставки в таблицу (INSERT):

Вставка в таблицу Contract, при аренде автомобиля:

INSERT INTO Contract (ID\_Klient, ID\_Personal, ID\_Car, Summa, Date, BeginArenda, EndArenda, SeriaKlient, NumberKlient) VALUES (@ID\_Klient, @ID\_Personal, @ID\_Car, @Summa, @Date, @BeginArenda, @EndArenda, @Seria\_Client, @Number\_Client)

INSERT INTO Contract (ID\_Klient, ID\_Personal, ID\_Car, Summa, Date, BeginArenda, EndArenda, SeriaKlient, NumberKlient) – эта часть запроса перечисляет столбцы, в которые будут вставляться данные, и задает сам факт вставки данных.

VALUES (@ID\_Klient, @ID\_Personal, @ID\_Car, @Summa, @Date, @BeginArenda, @EndArenda, @Seria\_Client, @Number\_Client) – данная часть перечисляет значения, которые будут вставлены в соответствующие столбцы. При выполнении запроса все переменные будут заменены на конкретные значения.

* + 1. Запрос для обновления записей (UPDATE)

Обновление данных в таблице Personal:

UPDATE Personal SET Surname=@Surname, Name=@Name, LastName=@LastName, Post=@Post, Telephone=@Telephone, Mail=@Mail WHERE ID=@ID

UPDATE Personal SET – эта часть запроса указывает на факт обновления данных в таблице Personal.

Surname=@Surname, Name=@Name, LastName=@LastName, Post=@Post, Telephone=@Telephone, Mail=@Mail WHERE ID=@ID – данная часть показывает столбец и новое значение этого столбца. При выполнении запроса, все значения будут заменены на конкретные.

* + 1. Запрос для удаления записей (DELETE)

Удаление данных из таблицы Specifications:

DELETE FROM Specifications WHERE ID=@ID

DELETE FROM Specifications – эта часть запроса указывает на факт удаления данных из таблицы Specifications.

WHERE ID=@ID – данная часть показывает, что будет произведено удаление строки с конкретным значением поля ID. При выполнении запроса значение будет заменено на конкретное.

## Руководство пользователя

При запуске приложения первым шагом для пользователя служит окно авторизации, показанное на рисунке 14. Оно служит для идентификации и аутентификации пользователей.

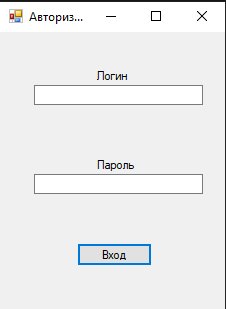


Рис. 14 – Окно авторизации

В окне авторизации пользователь вводит логин и пароль от своей учетной записи, что показано на рисунках 15 и 16. Если логин и пароль правильные, то он попадает в рабочее окно, что показано рисунках 17 и 18.

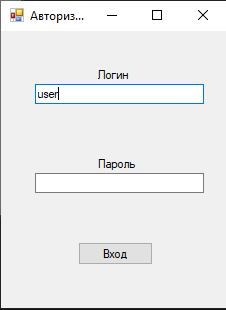


Рис. 15 – Ввод данных от учетной записи менеджера

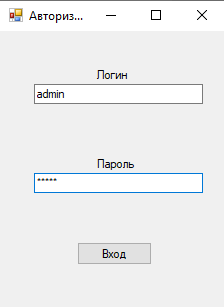


Рис. 16 – Ввод данных от учетной записи администратора

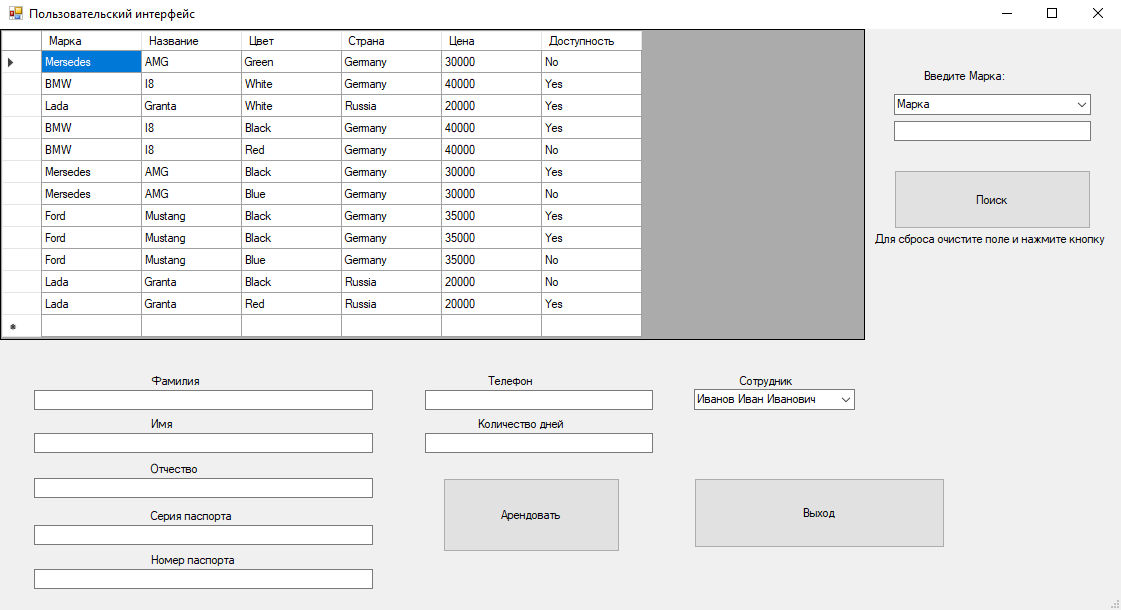


Рис. 17 – Рабочее окно менеджера

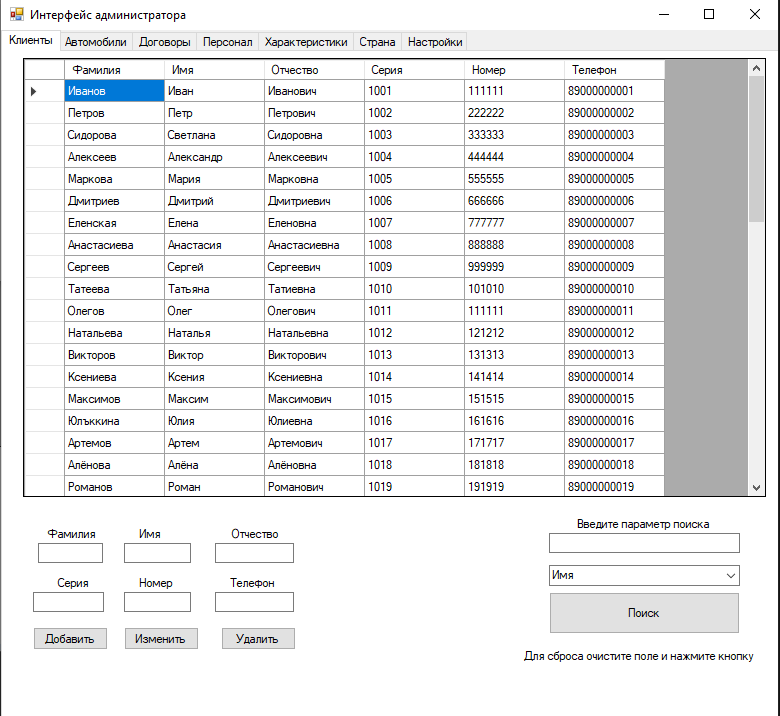


Рис. 18 – Рабочее окно администратора

Рабочее окно менеджера или окно аренды, показанное на рисунке 17 имеет следующие функции:

* 1. Просмотр доступных автомобилей, их характеристик и стоимости
  2. Текстовые поля для ввода данных клиента, желающего арендовать автомобиль
  3. Кнопка аренды для записи данных нового клиента в базу и оформления договора об аренде
  4. Поле поиска автомобиля по любой из характеристик
  5. Кнопка выхода из окна и возвращение на форму авторизации

Форма администратора, представленная на рисунке 18 состоит из нескольких страниц для каждой из таблиц в базе. Она имеет следующие функции:

1. Возможность просмотра необходимых данных, они показаны на рисунках 19 – 24

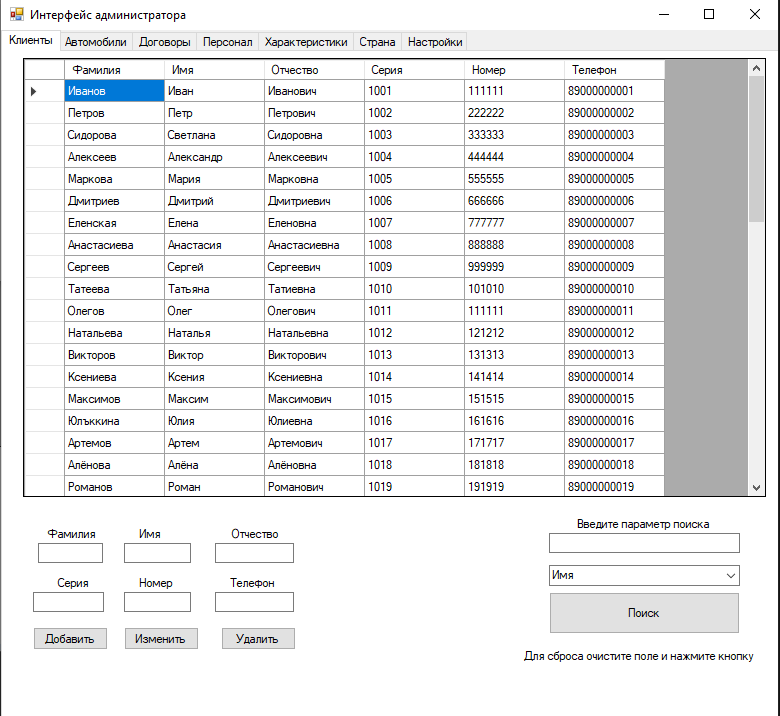


Рис. 19 – Данные из таблицы Klients

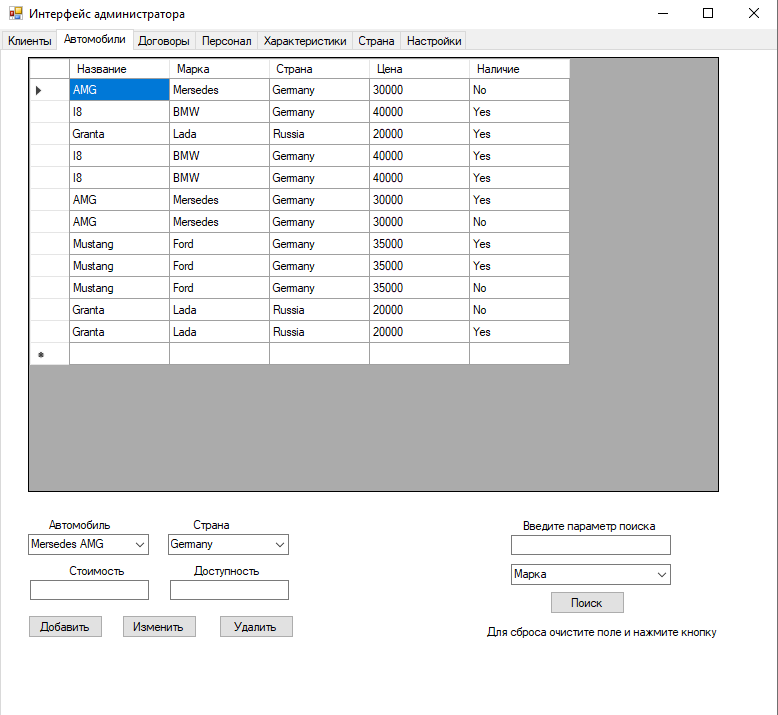


Рис. 20 – Данные из таблицы Cars

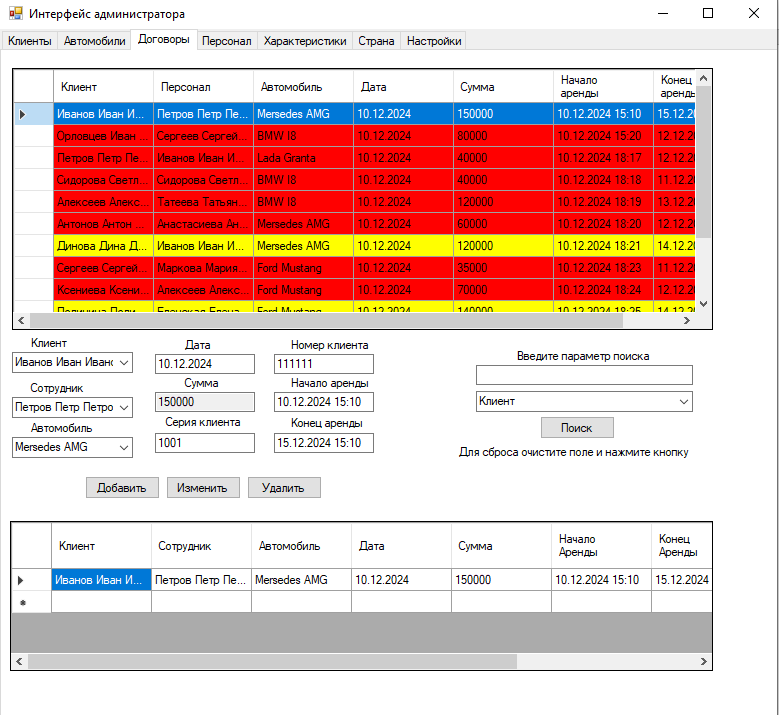


Рис. 21 – Данные из таблицы Contract

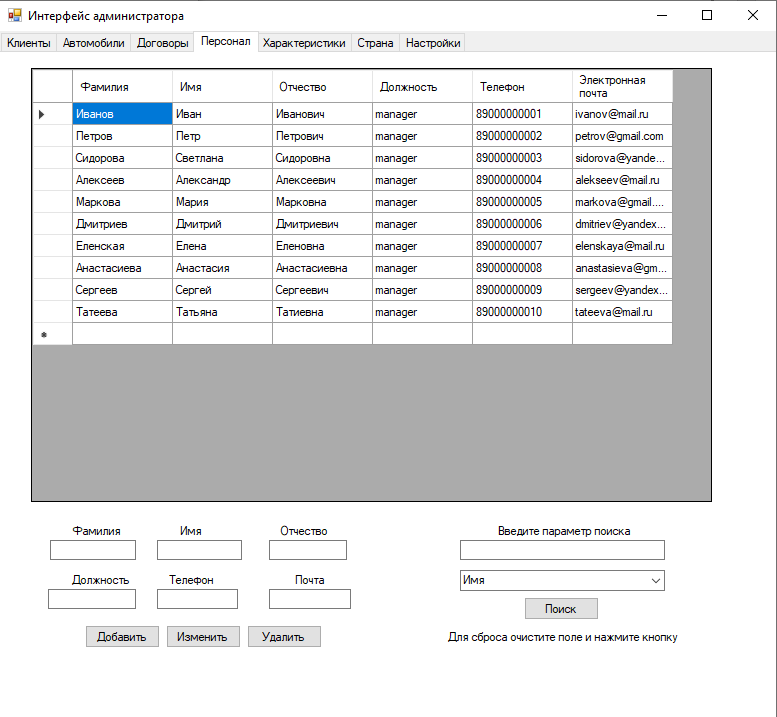


Рис. 22 – Данные из таблицы Personal

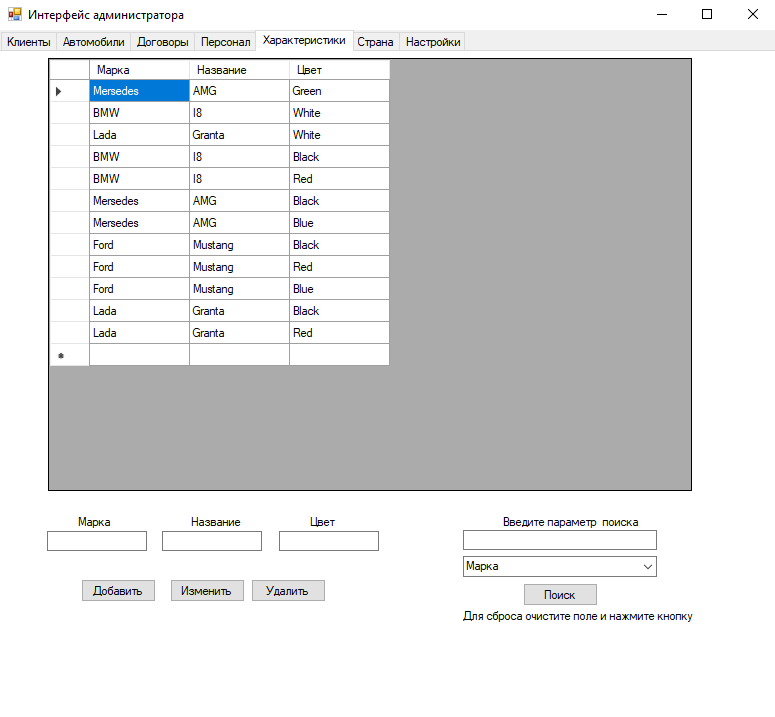


Рис. 23 – Данные из таблицы Specifications

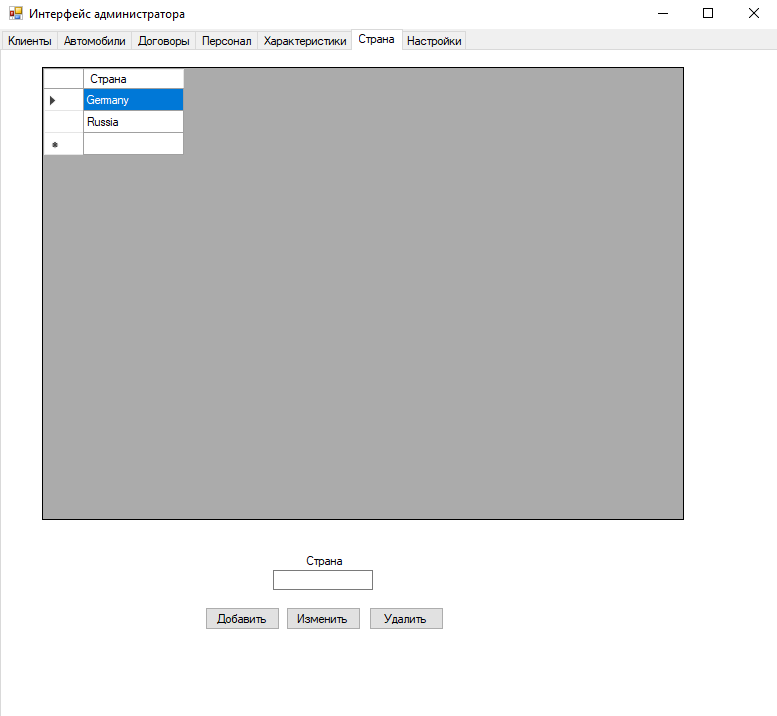


Рис. 24 – Данные из таблицы Country

1. Каждая из страниц имеет текстовые поля для ввода данных, а также кнопки для их добавления, обновления или удаления. Для добавления данных в таблицу необходимо заполнить все поля, как показано на рисунке 25 и нажать на кнопку. Результат показан на рисунке 26.

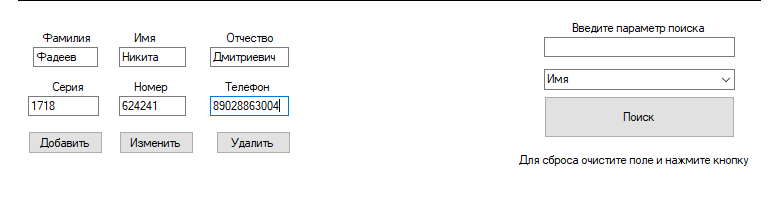


Рис. 25 – Заполнение полей



Рис. 26 – Запись добавлена в таблицу

1. Для изменения данных необходимо выбрать запись, все данные перенесутся в текстовые поля, изменить данные и нажать на кнопку. Результат показан на рисунках 27-28.

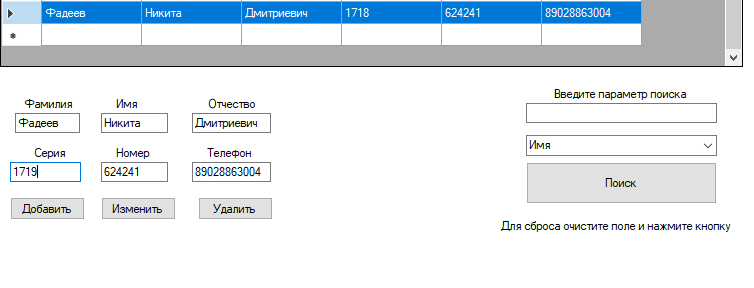


Рис. 27 – Изменение данных



Рис. 28 – Запись обновляется

1. Для удаления данных необходимо также выбрать запись и нажать на кнопку удаления. Результат представлен на рисунках 29-30.

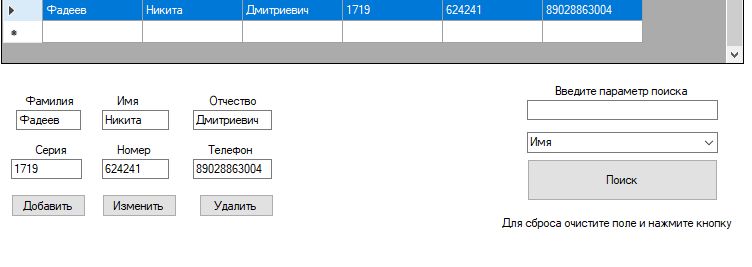


Рис. 29 – Выбор записи

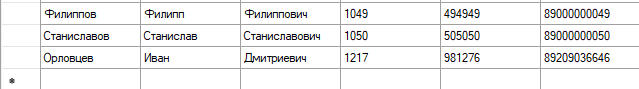


Рис. 30 - Запись удалена

1. Также каждая из страниц обладает возможностью поиска данных по параметрам. Для этого необходимо выбрать параметр из выпадающего списка, ввести нужное значение в текстовое поле и нажать кнопку поиска. Результаты показаны на рисунках 31-32.

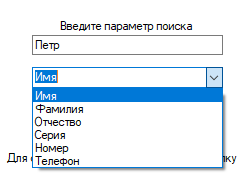


Рис. 31 – Выбор параметра

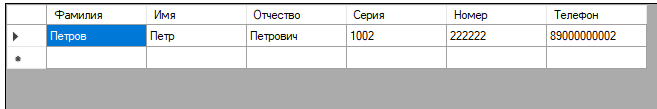


Рис. 32 – Результат поиска

1. Последняя страница представляет собой настройки, она показана на рисунке 33. Там находятся кнопки для перехода в архив контрактов, он представлен на рисунке 34, обновления данных и возвращения в форму авторизации.

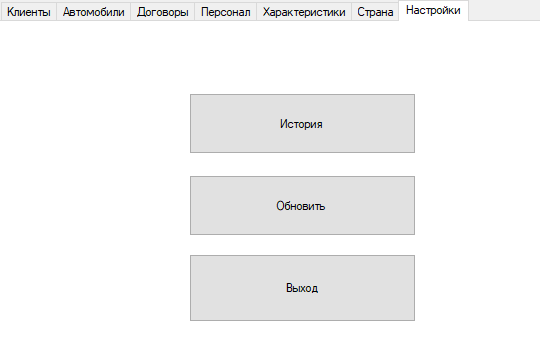


Рис. 33 – Страница настроек

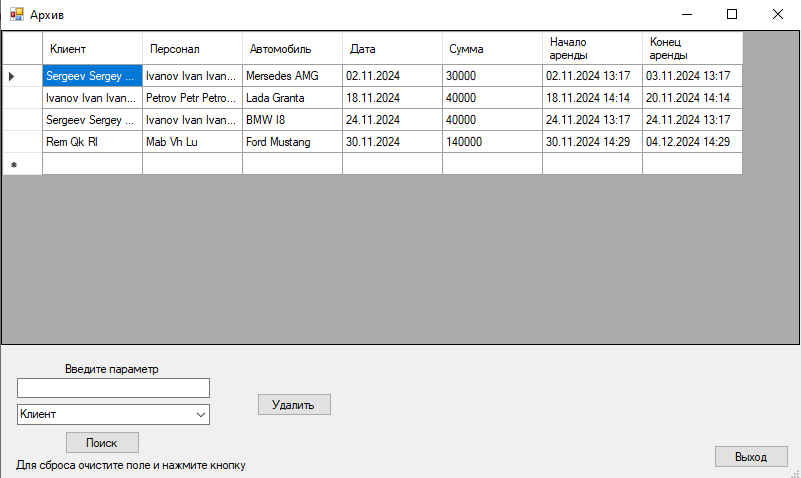


Рис. 34 – Форма архива

Форма архива также обладает возможностью поиска, он происходит аналогично поиску, представленному выше. А также тут расположена кнопка для возвращения в форму администратора.

## Руководство программиста

3.3.1 Введение

Приложение разработано на языке программирования C# в среде Visual Studio с использованием СУБД SQLite. Программа включает в себя несколько форм: форма авторизации, рабочая форма администратора, рабочая форма менеджера по аренде и форма архива.

3.3.2 Установка и настройка

Необходимые компоненты:

* Visual Studio 2019 или новее
* .NET Framework 4.7.2 или новее
* СУБД SQLite

Установка:

1. Откройте Visual Studio
2. Запустите проект
3. Установите пакет SQLite через пункт управление NuGet пакетами

3.3.3 Структура приложения

Проект состоит из следующих файлов:

* Form1.cs – Форма авторизации
* Form2.cs – Форма менеджера по аренде
* Form3.cs – Форма администратора
* Form4.cs – Форма архива

3.3.4 Описание форм

1) Form1.cs

* Конструктор Form1: инициализирует форму.
* Метод button1\_Click: обрабатывает нажатие кнопки входа, проверяет введенные в текстовые поля логин и пароль и открывает соответствующую форму.
* Метод textBox2\_TextChanged: скрывает ввод пароля с помощью \*.

2) Form2.cs

* Конструктор Form2: инициализирует форму, устанавливает соединение с базой данных, заполняет ComboBox1 данными об автомобилях, ComboBox4 – данными о сотрудниках.
* Метод LoadAutomobileData: выполняет SQL-запрос и загружает данные об имеющихся автомобилях и выводит их в dataGridView1.
* Метод button1\_Click: обрабатывает нажатие кнопки аренды, проверяет доступность автомобиля, и если он доступен, а также заполнены текстовые поля о данных клиенты выполняет SQL-запрос, заносит запись в таблицу Klients и в таблицу Contract, а также обновляет доступность автомобиля с Yes на No.
* Метод textBox1\_Validating: проверяет корректность ввода фамилии в textBox1.
* Метод textBox2\_Validating: проверяет корректность ввода имени в textBox2.
* Метод textBox3\_Validating: проверяет корректность ввода отчества в textBox3.
* Метод textBox4\_Validating: проверяет корректность ввода серии паспорта в textBox4.
* Метод textBox5\_Validating: проверяет корректность ввода номера паспорта в textBox5.
* Метод textBox6\_Validating: проверяет корректность ввода телефона в textBox6.
* Метод textBox7\_Validating: проверяет корректность ввода количества дней в textBox7.
* Метод IsValidName: проверяет выбрана ли строка.
* Метод IsNumeric: проверяет, что введено числовое значение, используется при проверке серии, номера паспорта и количества дней аренды.
* Метод button2\_Click: обрабатывает нажатие кнопки поиска, выполняет SQL-запрос и выводит результат в dataGridView1.
* Метод button3\_Click: обрабатывает нажатие на кнопку выхода, закрывает Form2 и возвращается на Form1.

3) Form3.cs

* Конструктор Form3: инициализирует форму, устанавливает соединение с базой данных и заполняет ComboBox 1-10 необходимыми данными.
* Метод LoadKlientData: выполняет SQL-запрос и заполняет dataGridView1 данными о клиентах.
* Метод LoadAutomobileData: выполняет SQL-запрос и заполняет dataGridView2 данными об автомобилях.
* Метод LoadContractData: выполняет SQL-запрос и заполняет dataGridView3 данными о контрактах.
* Метод DataGridView3\_RowPrePaint: устанавливает цвета выделения записей в dataGridView3, а также автоматическую смену доступности автомобиля при истечении аренды.
* Метод UpdateCarAvailabilityInDatabase: обновляет доступность автомобилей в БД.
* Метод LoadPersonalData: выполняет SQL-запрос и заполняет dataGridView4 данными о сотрудниках.
* Метод LoadCountryData: выполняет SQL-запрос и заполняет dataGridView6 данными о странах-производителях.
* Метод LoadSpecificationsData: выполняет SQL-запрос и заполняет dataGridView7 данными о характеристиках автомобилей.
* Метод button1\_Click: обрабатывает нажатие кнопки добавления записей в dataGridView1, выполняет SQL-запрос и добавляет запись в зависимости от введенных данных.
* Метод button2\_Click: обрабатывает нажатие кнопки обновления записей в dataGridView1, выполняет SQL-запрос и обновляет запись в зависимости от введенных данных.
* Метод button4\_Click: обрабатывает нажатие кнопки удаления записей из dataGridView1, выполняет SQL-запрос и удаляет запись.
* Метод button5\_Click: обрабатывает нажатие кнопки добавления записей в dataGridView2, выполняет SQL-запрос и добавляет запись в зависимости от введенных данных.
* Метод button6\_Click: обрабатывает нажатие кнопки обновления записей в dataGridView2, выполняет SQL-запрос и обновляет запись в зависимости от введенных данных.
* Метод button8\_Click: обрабатывает нажатие кнопки удаления записей из dataGridView2, выполняет SQL-запрос и удаляет запись.
* Метод button9\_Click: обрабатывает нажатие кнопки добавления записей в dataGridView3, выполняет SQL-запрос и добавляет запись в зависимости от введенных данных.
* Метод button10\_Click: обрабатывает нажатие кнопки обновления записей в dataGridView3, выполняет SQL-запрос и обновляет запись в зависимости от введенных данных.
* Метод button12\_Click: обрабатывает нажатие кнопки удаления записей из dataGridView3, выполняет SQL-запрос и удаляет запись.
* Метод button13\_Click: обрабатывает нажатие кнопки добавления записей в dataGridView4, выполняет SQL-запрос и добавляет запись в зависимости от введенных данных.
* Метод button15\_Click: обрабатывает нажатие кнопки обновления записей в dataGridView4, выполняет SQL-запрос и обновляет запись в зависимости от введенных данных.
* Метод button16\_Click: обрабатывает нажатие кнопки удаления записей из dataGridView4, выполняет SQL-запрос и удаляет запись.
* Метод button17\_Click: обрабатывает нажатие кнопки добавления записей в dataGridView5, выполняет SQL-запрос и добавляет запись в зависимости от введенных данных.
* Метод button18\_Click: обрабатывает нажатие кнопки обновления записей в dataGridView5, выполняет SQL-запрос и обновляет запись в зависимости от введенных данных.
* Метод button20\_Click: обрабатывает нажатие кнопки удаления записей из dataGridView5, выполняет SQL-запрос и удаляет запись.
* Метод button21\_Click: обрабатывает нажатие кнопки добавления записей в dataGridView6, выполняет SQL-запрос и добавляет запись в зависимости от введенных данных.
* Метод button22\_Click: обрабатывает нажатие кнопки обновления записей в dataGridView6, выполняет SQL-запрос и обновляет запись в зависимости от введенных данных.
* Метод button24\_Click: обрабатывает нажатие кнопки удаления записей из dataGridView6, выполняет SQL-запрос и удаляет запись.
* Метод comboBox3\_SelectedIndexChanged: переносит данные о серии и номере клиента в textbox 17, 20 при выборе его в comboBox3.
* Метод сomboBox5\_SelectedIndexChanged: переносит данные о цене автомобиля в textbox 18 при выборе его в comboBox5.
* Метод UpdateRentSum: подсчитывает общую стоимость в результате анализа дат начала и конца аренды.
* Метод button3\_Click: обрабатывает нажатие кнопки поиска, выполняет SQL-запрос и выводит результат в dataGridView1.
* Метод button7\_Click: обрабатывает нажатие кнопки поиска, выполняет SQL-запрос и выводит результат в dataGridView4.
* Метод button11\_Click: обрабатывает нажатие кнопки поиска, выполняет SQL-запрос и выводит результат в dataGridView5.
* Метод button14\_Click: обрабатывает нажатие кнопки поиска, выполняет SQL-запрос и выводит результат в dataGridView2.
* Метод button19\_Click: обрабатывает нажатие кнопки поиска, выполняет SQL-запрос и выводит результат в dataGridView3.
* Метод dataGridView1\_SelectionChanged: обрабатывает выбор строки в dataGridView1 и переносит данные в textbox.
* Метод dataGridView2\_SelectionChanged: обрабатывает выбор строки в dataGridView2 и переносит данные в textbox.
* Метод dataGridView3\_SelectionChanged: обрабатывает выбор строки в dataGridView3 и переносит данные в textbox.
* Метод dataGridView4\_SelectionChanged: обрабатывает выбор строки в dataGridView4 и переносит данные в textbox.
* Метод dataGridView5\_SelectionChanged: обрабатывает выбор строки в dataGridView5 и переносит данные в textbox.
* Метод dataGridView6\_SelectionChanged: обрабатывает выбор строки в dataGridView6 и переносит данные в textbox.
* Метод button25\_Click: обрабатывает нажатие кнопки обновления и снова выполняет методы по загрузке данных.
* Метод button26\_Click: обрабатывает нажатие кнопки выхода, закрывает текущую форму и возвращается на Form1.
* Метод button23\_Click: обрабатывает нажатие кнопки перехода в архив, скрывает текущую форму и открывает Form4.
* Метод IsValidName: проверяет корректность ввода ФИО.
* Метод IsUniqueSeriaNumber: проверяет уникальность ввода серии + номер в таблице Klients.
* Метод IsUniqueTelephone: проверяет уникальность ввода телефона в таблице Klients.
* Метод IsUniqueTelephonePer: проверяет уникальность ввода телефона в таблице Personal.
* Метод IsValidMail: проверяет корректность ввода электронной почты.
* Метод IsUniqueMail: проверяет уникальность ввода электронной посты в таблице Personal.
* Метод IsNumeric: проверяет ввод на соответствие числовому параметру, используется для проверки на содержание посторонних символов.
* Метод IsYesOrNo: проверяет корректность ввода доступности автомобиля.
* Метод IsTextOnly: проверяет ввод на отсутствие цифр, используется для проверки ввода ФИО.
* Метод IsUniqueCountry: проверяет уникальность ввода страны.
* Метод IsDateValid: проверяет корректность ввода даты.
* Метод IsDateTimeValid: проверяет корректность ввода времени начала и окончания аренды.
* Метод LoadKlientComboBoxData: загружает в ComboBox3 данные из таблицы Klients с помощью SQL-запроса.
* Метод LoadAutomobileComboBoxData: загружает в ComboBox5 данные из таблицы Cars с помощью SQL-запроса.
* Метод LoadPersonalComboBoxData: загружает в ComboBox4 данные из таблицы Personal с помощью SQL-запроса.
* Метод LoadCountryComboBoxData: загружает в ComboBox2 данные из таблицы Country с помощью SQL-запроса.

1. Form3.cs

* Конструктор Form4: инициализирует форму, устанавливает соединение с базой данных и заполняет ComboBox 1 необходимыми данными.
* Метод LoadDeletedContractData: выполняет SQL-запрос и заполняет dataGridView1 данными об удаленных договорах.
* Метод button1\_Click: обрабатывает нажатие кнопки поиска, выполняет SQL-запрос и выводит результат в dataGridView1.
* Метод button2\_Click: обрабатывает нажатие кнопки удаления записей из dataGridView2, выполняет SQL-запрос и удаляет запись.
* Метод button3\_Click: обрабатывает нажатие кнопки выхода, закрывает текущую форму и возвращается к Form3.

# Тестирование АИС

Тестирование автоматизированной информационной системы является неотъемлемой частью процесса разработки, направленной на обеспечение ее корректной работы, надежности и безопасности. При открытии приложения первым шагом для пользователя является окно авторизации, показанное на рисунке 35, которое служит для идентификации и аутентификации пользователей. Это окно играет важную роль в защите данных и управлении доступом к функционалу системы, что делает его критически важным элементом для тестирования.

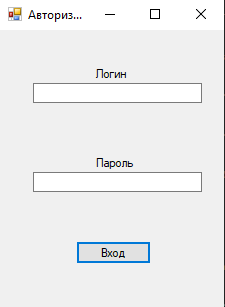


Рис. 35 – Окно авторизации

В окне авторизации пользователь вводит логин и пароль от своей учетной записи, если логин и пароль правильные, то пользователь попадает в рабочее окно, а если логин или пароль, который был введен не корректен, то приложение выдаст ошибку, что логин и пароль не правильный, это показано на рисунке 36.

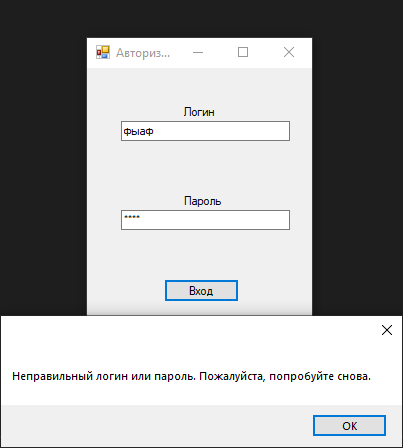


Рис. 36 – Ошибка входа при неправильном логине или пароле

Если логин и пароль правильные, то пользователь попадает в одно из рабочих окон. Начнем с формы менеджера. Здесь пользователь может найти интересующий его автомобиль с помощью поиска. Если запись не нашлась, то будет выведено сообщение. Результат представлен на рисунках 37-38.

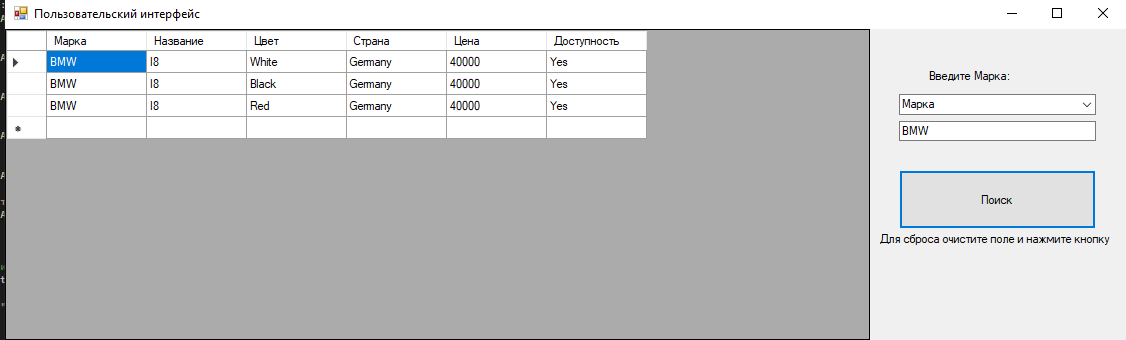


Рис. 37 – Корректный поиск

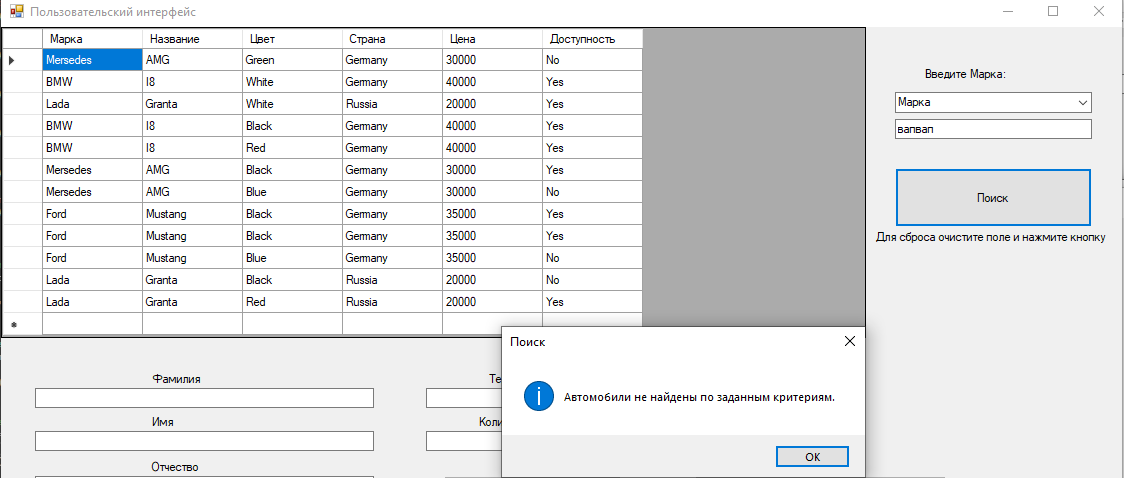


Рис. 38 – Записи не найдены

Найдя автомобиль, пользователь может его арендовать. Для этого ему необходимо выбрать автомобиль, заполнить текстовые поля и ввести свои данные. Если одно из полей будет неправильно заполнено, то приложение выведет сообщение, показанное на рисунке 39.

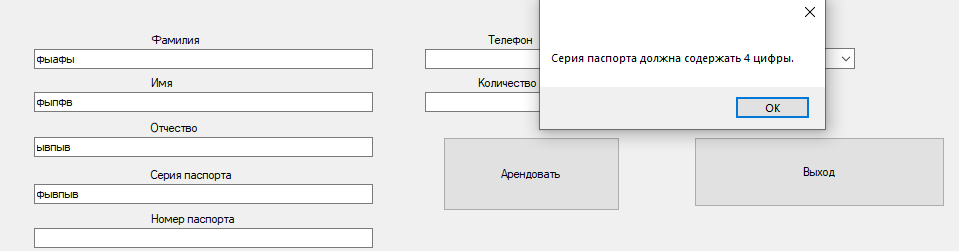


Рис. 39 – Сообщение при неправильном заполнении полей

Если все данные корректны, то при нажатии на кнопку будет выведено сообщение об успешной аренде, представленное на рисунке 40.

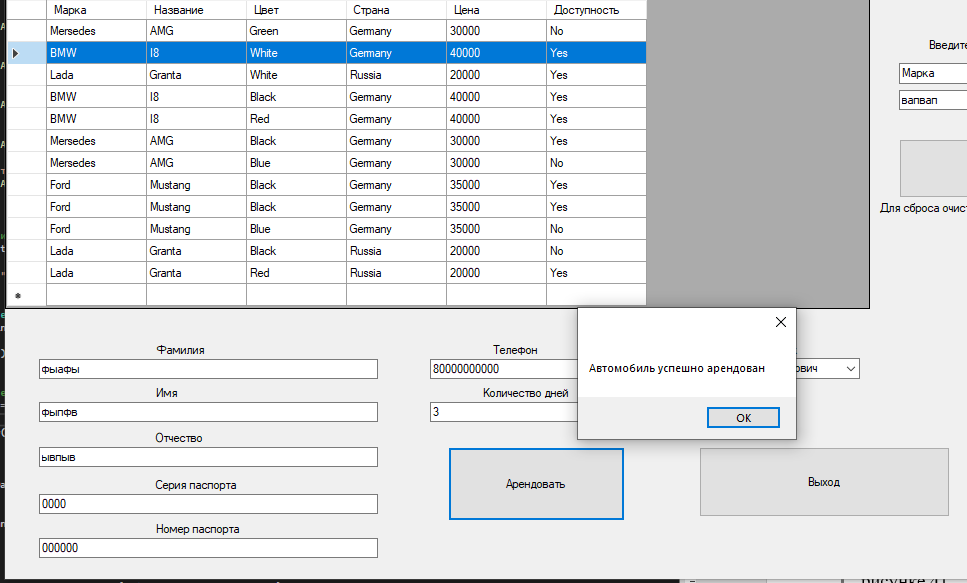


Рис. 40 – Сообщение об успешной аренде

Теперь перейдем к форме администратора. Он может найти нужные ему данные с помощью поиска. Если запись не нашлась, то будет выведено сообщение. Результат представлен на рисунках 41-42.

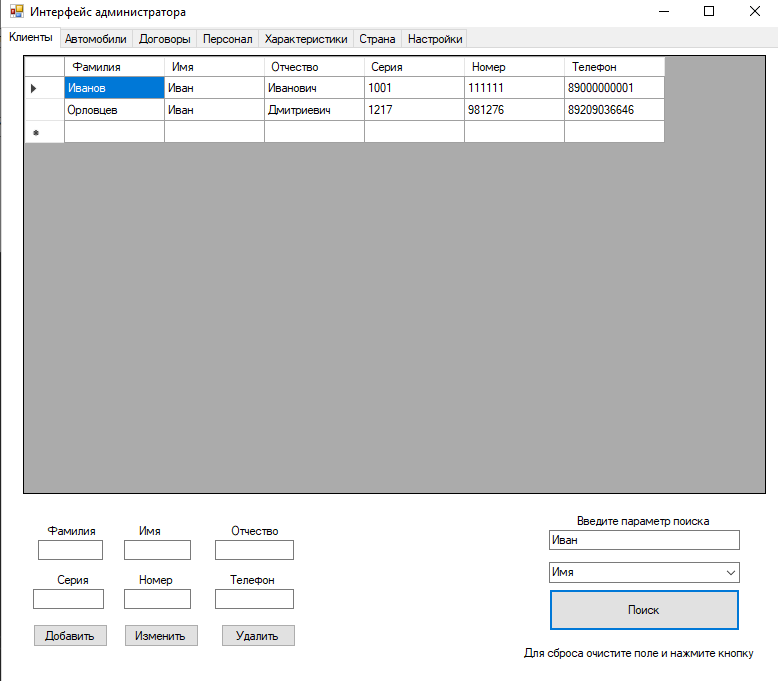


Рис. 41 – Пример корректного поиска

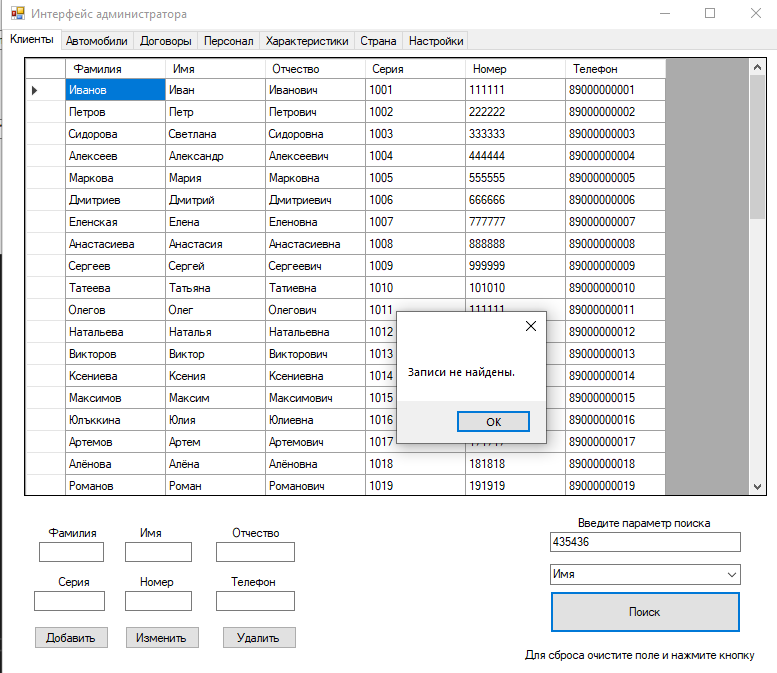


Рис. 42 – Записи не найдены

В данном случае показан пример поиска на таблице Klients. Для всех остальных таблиц он работает аналогично.

Также администратор может редактировать или удалять записи. Для этого ему нужно выбрать запись, данные будут перенесены в текстовые поля. При нажатии на кнопку они будут изменены или удалены. Результат показан на рисунках 43-44.

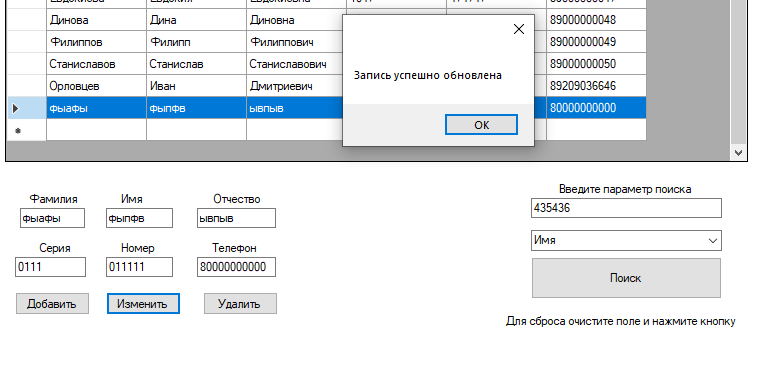


Рис. 43 – Обновление записи

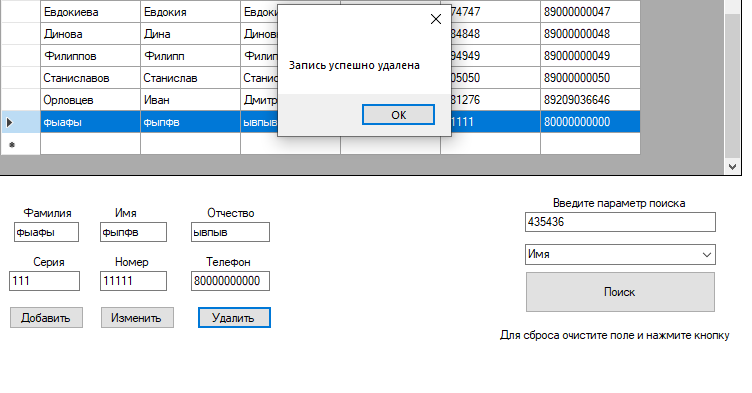


Рис. 44 – Удаление записи

В данном случае показан пример поиска на таблице Klients. Для всех остальных таблиц он работает аналогично.

На вкладке «Настройки» при нажатии на кнопку «Обновить» администратор обновит все таблицы. При завершении будет выведено сообщение, показанное на рисунке 45.

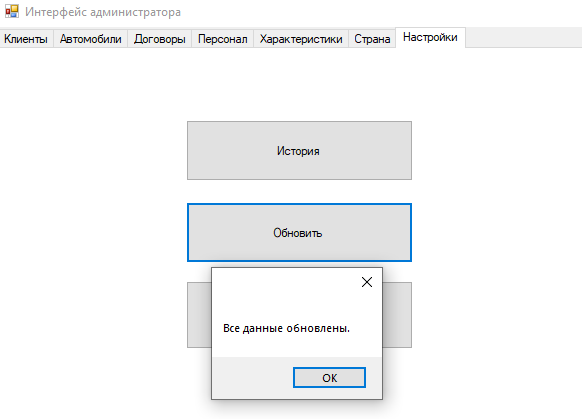


Рис. 45 – Обновление данных

На этой же вкладке при нажатии на кнопку «История» администратор перейдет на форму архива удаленных договоров.

Здесь также можно произвести поиск или удаление записи. Если при поиске запись не будет найдена, то выведется сообщения. Результаты показаны на рисунках 46-48.

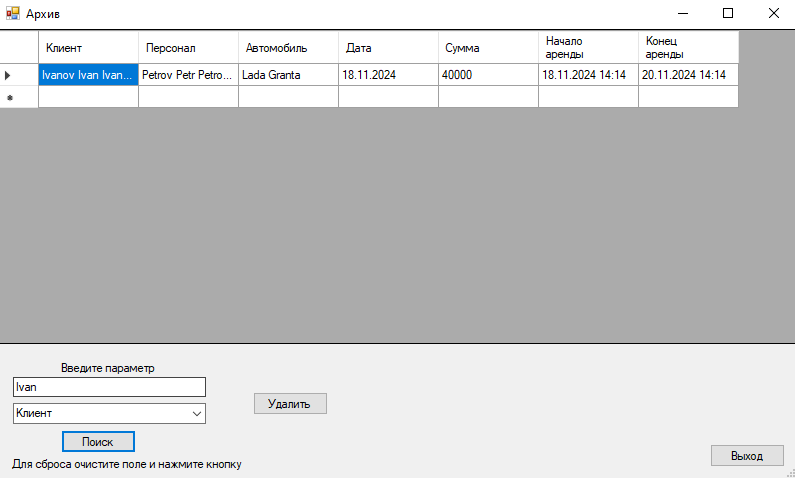


Рис. 46 – Корректный поиск

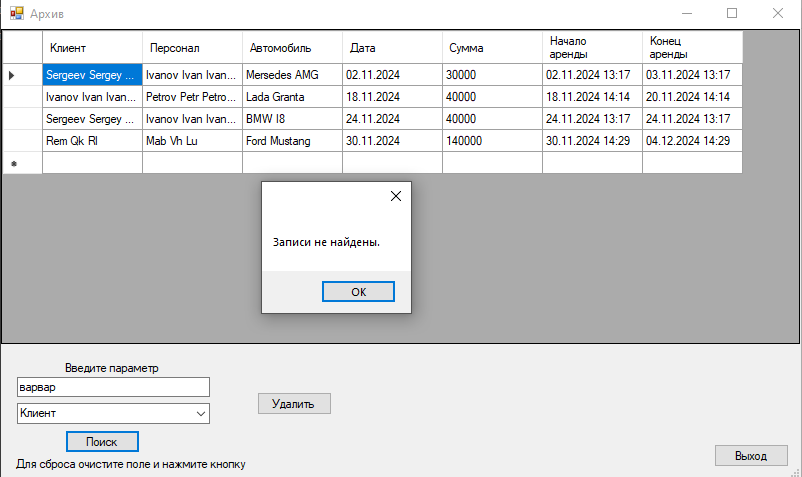


Рис. 47 – Записи не найдены

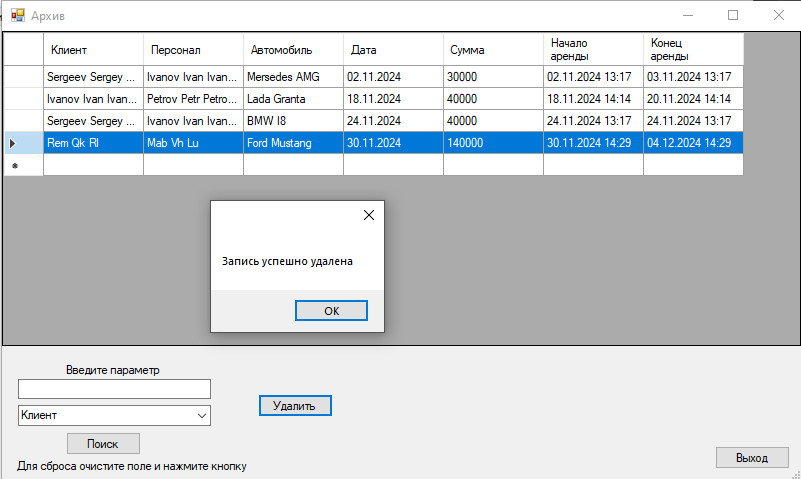


Рис. 48 – Удаление записи

# Заключение

В ходе данной курсовой работы были решены поставленные задачи, направленные на создание Автоматизированной Информационной Системы (АИС) для автосалона по аренде автомобилей. Основное внимание уделялось разработке базы данных, обеспечивающей эффективное управление данными. Использование такой системы позволяет автоматизировать процессы аренды автомобилей и оформления договоров.

Разработанная АИС также значительно улучшила взаимодействие пользователей и персонала автосалона. Каждый клиент теперь имеет возможность оперативно получать информацию об интересующем его автомобиле, что повышает уровень удовлетворенности и доверия к услугам автосалона.

На основании выполненного сравнительного анализа производительности различных СУБД и средств программирования были выбраны оптимальные решения для разработки данной базы данных и многопользовательского приложения. Это позволило создать надежную, производительную и масштабируемую систему, готовую к дальнейшему расширению.

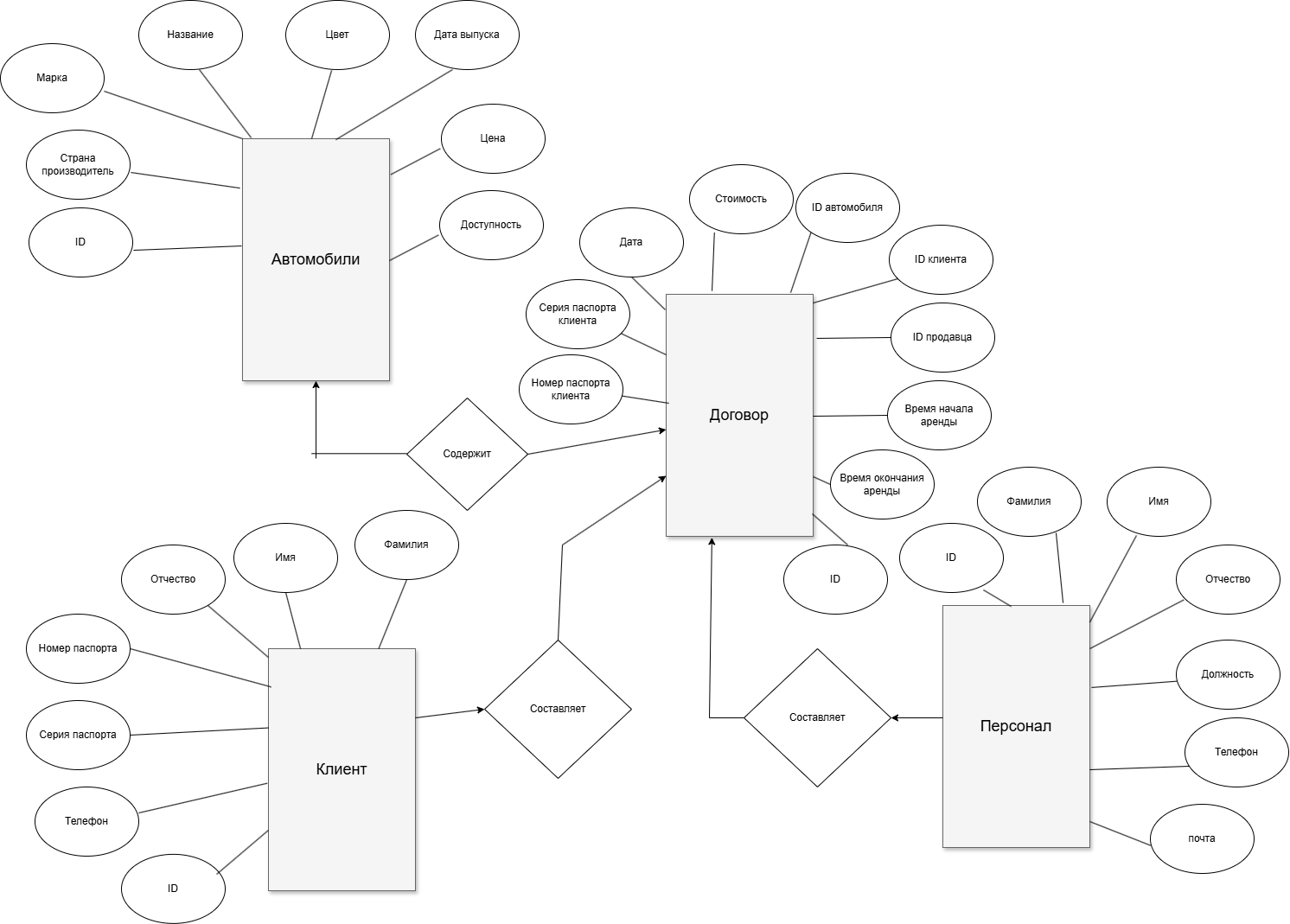
# Список литературы

* 1. Бьерн Страуструп, Язык программирования C++. Специальное издание. Пер. с англ. – М.: Издательство Бином, 2011 г. – 1136 с.: ил.
  2. Казарин С.А., Клишин А.П. К 143 Среда разработки Java-приложений Eclipse: (ПО для объектно-ориентированного программирования и разработки приложений на языке Java): Учебное пособие. Москва 2008. — 77 с.
  3. Медведев, М. А. М42 Программирование на СИ# : учеб. пособие / М. А. Медведев, А. Н. Медведев. — Екатеринбург : Изд-во Урал. ун-та, 2015. — 64 с.
  4. Мотев А. А. М85 Уроки MySQL. Самоучитель. — СПб.: БХВ-Петербург, 2006. — 208 с.: ил.
  5. Рындина С. В.. – Пенза : Изд-во ПГУ, 2023. – 82 с. , Цифровые технологии управления получением, хранением, передачей и обработкой больших данных: SQLite : учеб.-метод. пособие.
  6. Сапаров А.Ю., Разработка Windows Forms приложений на языке программирования C#: учебно-методическое пособие / Сост.: А.Ю. Сапаров, Ижевск, 2020. 61 с.

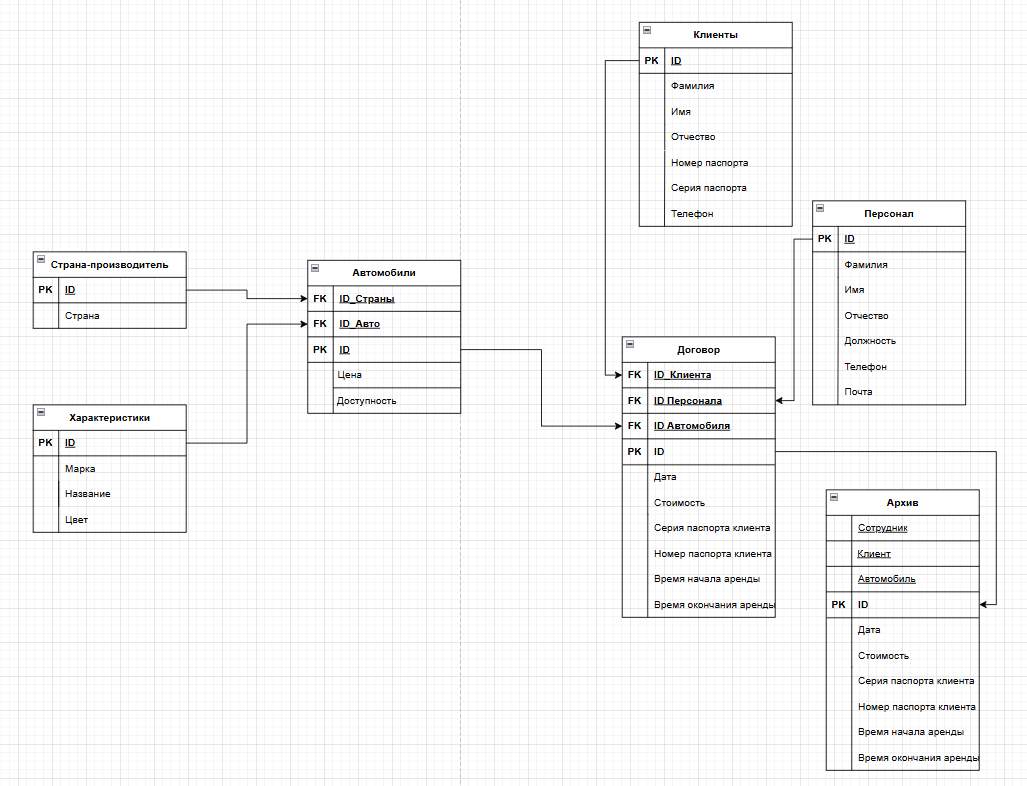
1. Сергеева Т.И. Базы данных: модели данных, проектирование, язык SQL: учеб. пособие / Т.И. Сергеева, М.Ю. Сергеев. Воронеж: ФГБОУ ВПО «Воронежский государственный технический университет», 2012. 233 с.

# Приложение А: «Модели данных»

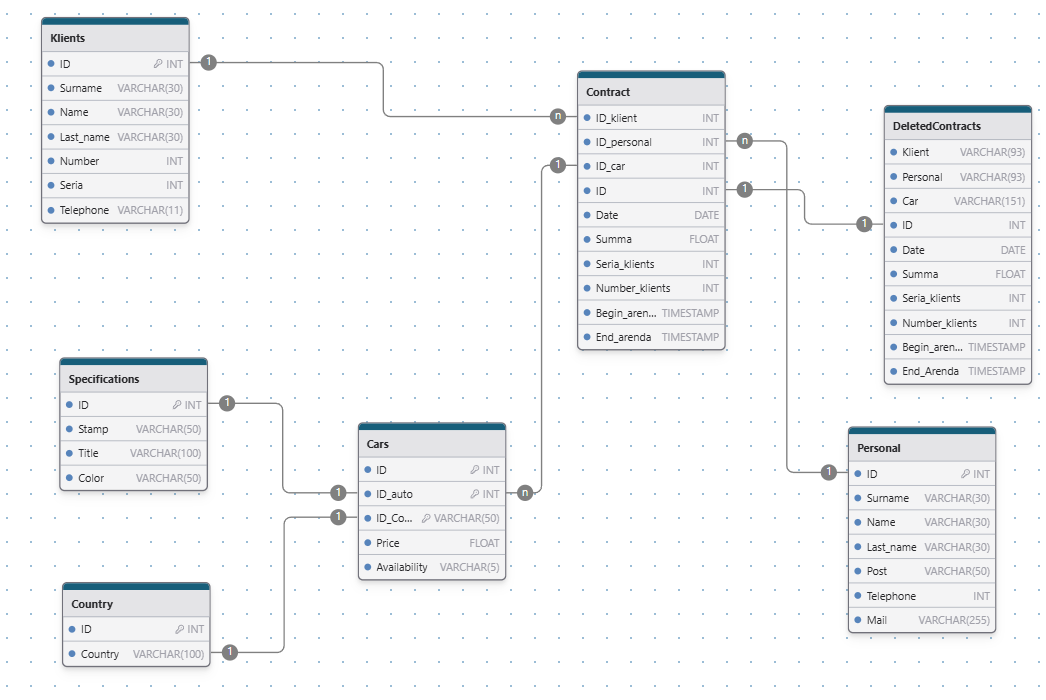
Концептуальная модель.



Логическая модель.



Физическая модель.



# Приложение Б: Текст кода

Весь исходный код представлен по ссылке: [**https://github.com/Nikiita12/Autosalon**](https://github.com/Nikiita12/Autosalon)