**Содержание**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | МИВУ 10.03.01-12ПЗ | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум | Подпись | Дата |
| Разраб | | Фадеев Н. Д. |  |  | АИС салона по аренде автомобилей | Литера | | | Лист | Листов |
| Пров | | Колпаков А. А. |  |  |  | y |  | 3 |  |
|  | |  |  |  | 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 ИС для работы с 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.

1. **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.

Гибкость: предоставляет детальный контроль над аппаратным обеспечением.

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

1. Характеристики C#:

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

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

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

1. Характеристики Visual Studio:

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

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

Использование: очень удобная и интуитивно понятная среда.

4. Характеристики Eclipse:

Открытый исходный код: бесплатная и доступная, поддерживаемая сообществом.

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

Плагины: огромное количество доступных плагинов для расширения функциональности.

5. Характеристики SQLite

Легковесность: не требует установки и настройки сервера.

Простота: легка в использовании и имеет компактную архитектуру.

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

6. Характеристики MySQL

Производительность: высокая производительность и масштабируемость для крупных проектов.

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

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

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 |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| 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 | - | - |
| 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.

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

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

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

* 1. **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 покупок. Предполагаемая база данных должна обеспечивать работу автосалона по учету аренды автомобилей различных видов, автоматизированную выдачу отчетов по операциям за определённые периоды и по категориям автомобилей.

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 главный менеджер или 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 или главный менеджер.

В автосалоне есть свой спектр сотрудников: менеджеры по продажам, главный менеджер, продавцы-консультанты. Главный менеджер автосалона контролирует работу отдела.

Клиент, приходя в автосалон взаимодействует с продавцом, и исходя из его запроса продавец 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.

* 1. **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

1. **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:

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.

* 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.

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.

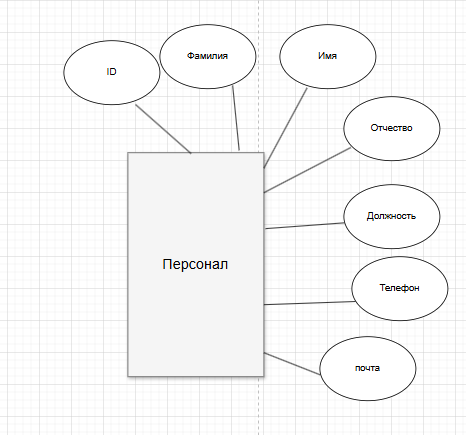


Рис. 1 – Атрибуты персонала

2) Автомобили.

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

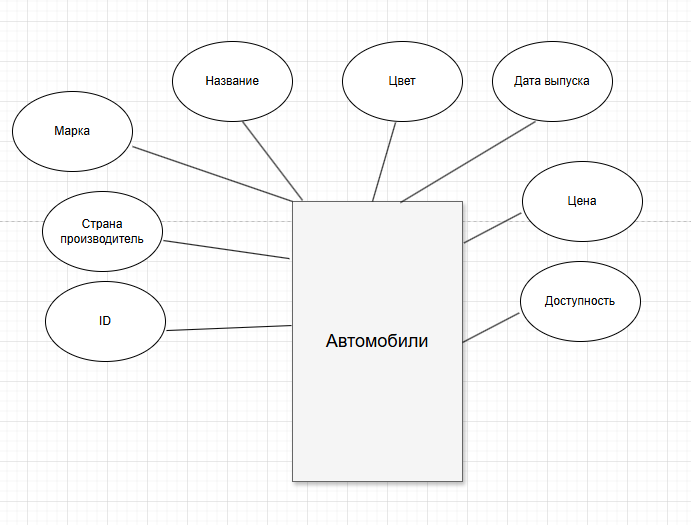


Рис. 2 – Атрибуты автомобилей

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.

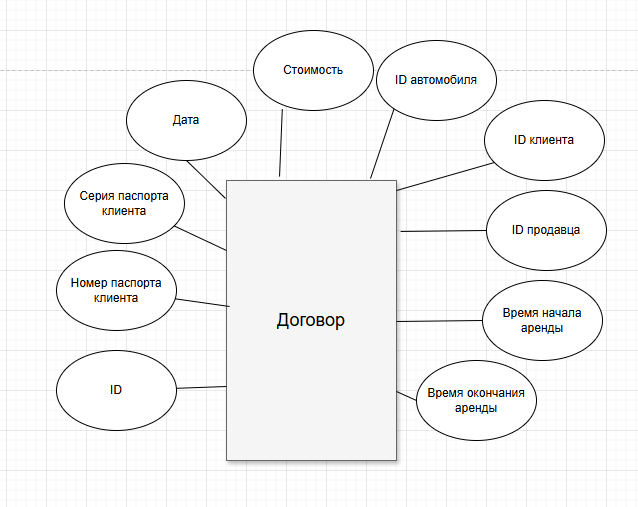


Рис. 3 – Атрибуты договора

4) Клиенты.

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

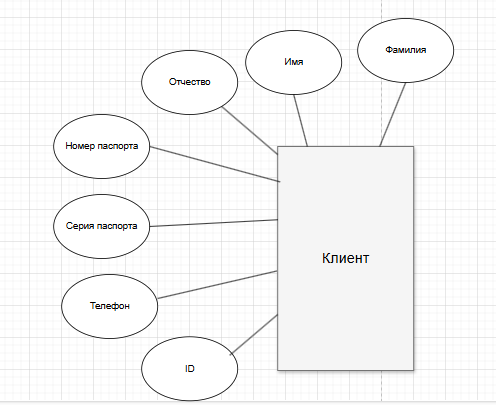


Рис. 4 – Атрибуты клиента

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

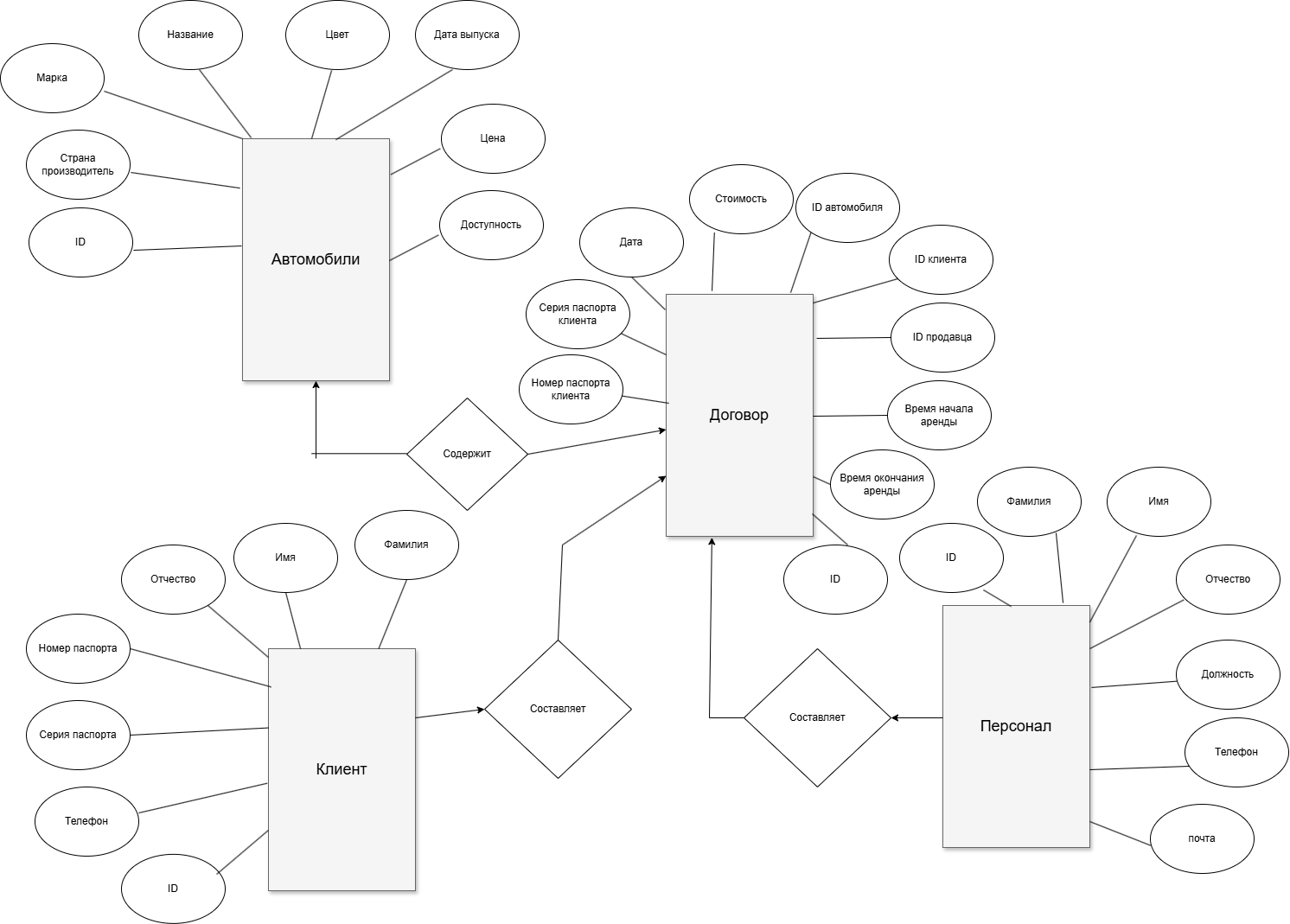


Рис. 5 – Концептуальная модель данных.

Можно выделить следующие связи между сущностями:

Между всеми таблицами существуют связи один ко многим.

* 1. **Логическая модель**

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

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:

* + 1. Клиенты:
  + ID – Уникальный идентификатор клиента, первичный ключ
  + Фамилия
  + Имя
  + Отчество
  + Серия\_паспорта
  + Номер\_паспорта
  + Телефон
    1. 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
    1. Автомобили:
  + ID – уникальный идентификатор, первичный ключ
  + ID\_Страны – уникальный идентификатор страны, внешний ключ
  + ID\_авто – уникальный идентификатор авто, внешний ключ
  + Цена
  + Доступность
    1. Страна\_производитель
* ID – уникальный идентификатор, первичный ключ
* Страна
  + 1. Характеристики
* ID – уникальный идентификатор, первичный ключ
* Марка
* Название
* Цвет
  + 1. Персонал:
  + ID – уникальный идентификатор, первичный ключ
  + Фамилия
  + Имя
  + Отчество
  + Должность
  + Почта
  + Телефон

На основе этих данных была составлена логическая модель данных, представленная на рисунке 7.

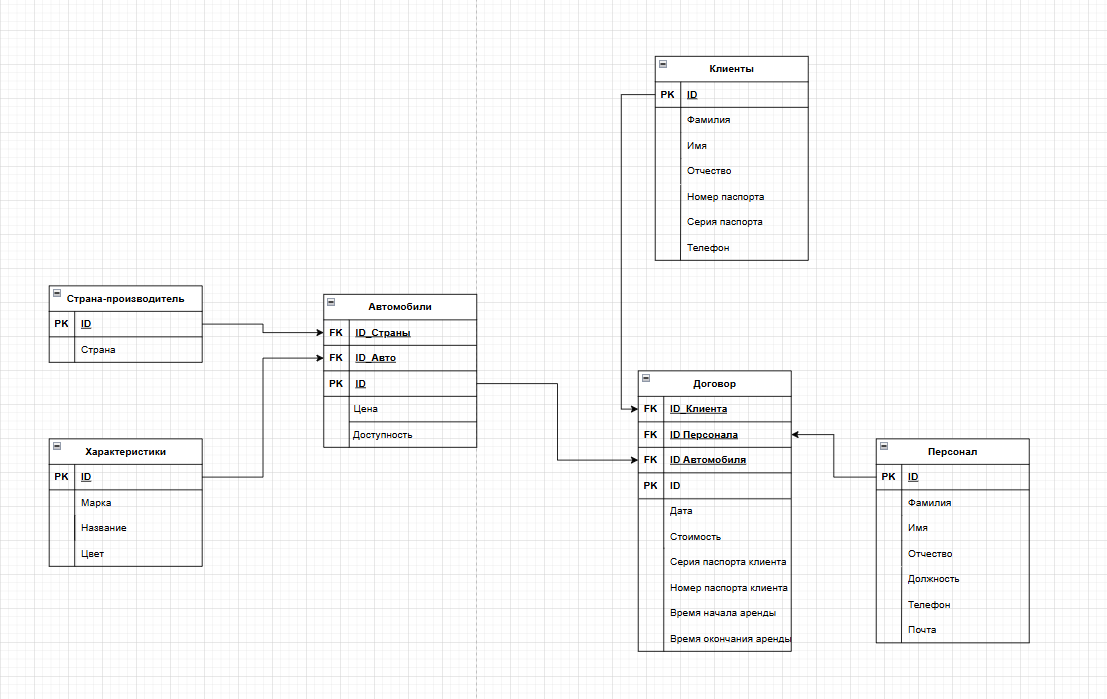


Рис. 7 – Логическая модель данных

* 1. **Приведение логической модели к физической.**

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

К основные задачи физической модели данных относится:

2.3.1 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)

2.3.2 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

2.3.3 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 8.

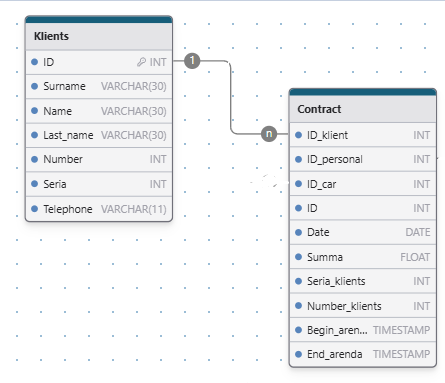


Рис. 8 – Связь между таблицами «Klients» и «Conract»

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 9.

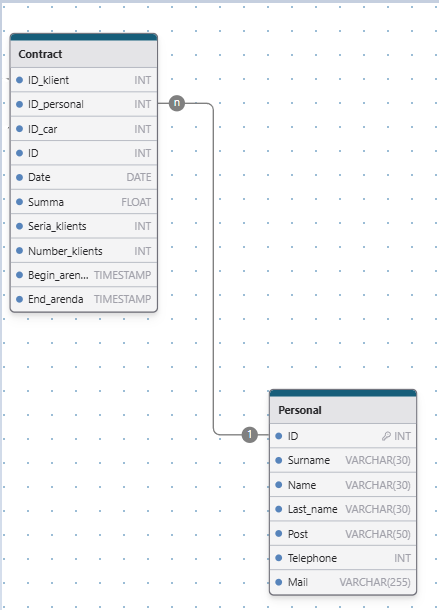


Рис. 9 – Связь между таблицами «Personal» и «Contract»

Связь между таблицами «Cars» и «Conract» организована через поля «ID» и «ID\_car» соответственно. Это можно увидеть на рисунке 10.

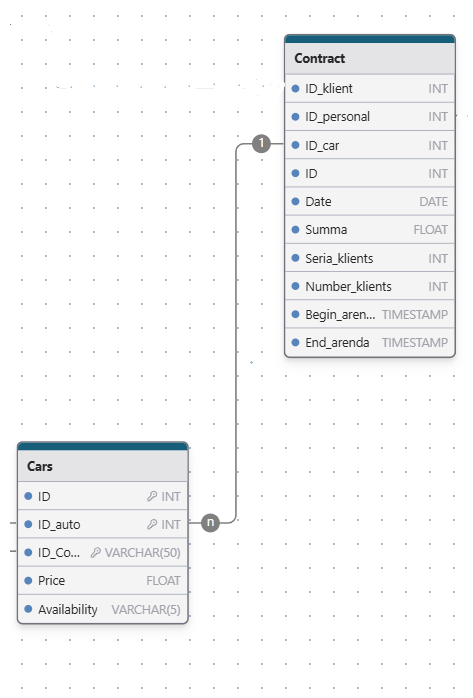


Рис. 10 – Связь между таблицами «Cars» и «Contract»

Также таблица «Cars» связана с еще 2 таблицами: «Specifications» и «Country of origin», связаны они через поля ID в таблицах «Specifications» и «Country of origin», и ID\_auto и ID\_coгntry в таблице «Cars». Это можно увидеть на рисунке 11.

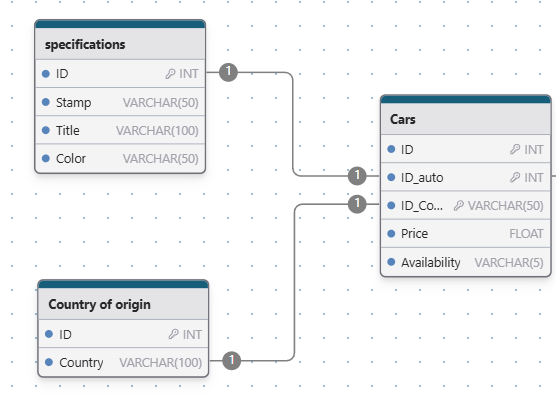


Рис. 11 – Связь между таблицами «Cars», «Specifications» и «Country of origins»

Полная схема физической модели данных представлена на рисунке 12.

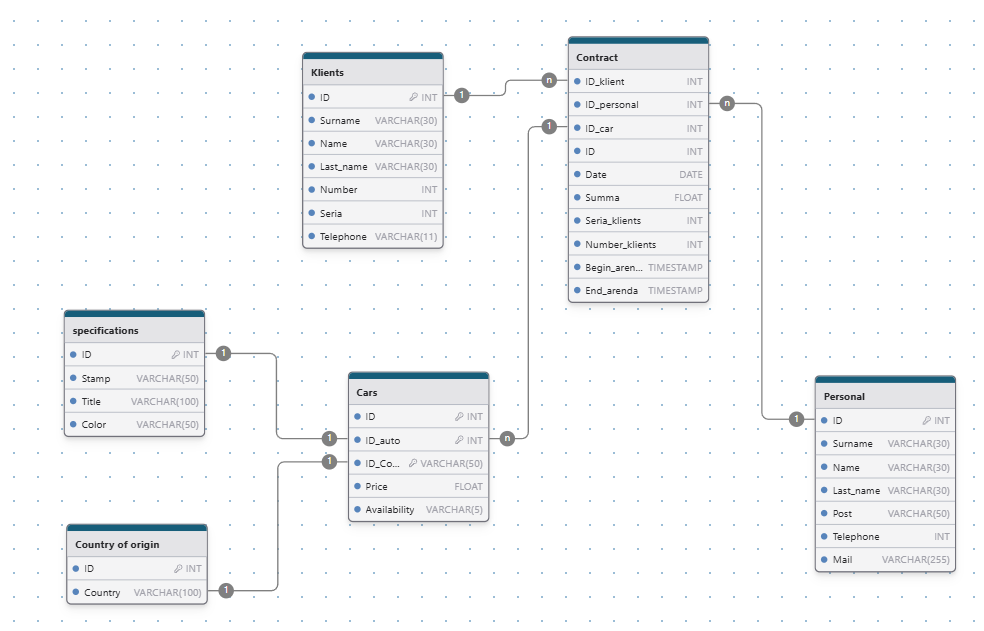
****

Рис. 12 – Физическая модель данных

2.3.4 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

2.3.5 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)

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;

2.3.6 Таблица "Cars":

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

ID\_country (integer) - уникальный идентификатор, внешний ключ

ID\_auto – уникальный идентификатор, внешний ключ

Price (float)

Availability (varchar(5))

Ключевые особенности:

-первичный ключ ID для уникальной идентификации записей;

2.3.7 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

2.3.8 Таблица «Country of origin»

ID\_FK – Уникальный идентификатор, внешний ключ

Country(varchar(50))

Ключевые особенности:

-ограничение Country на уникальность

2.3.9 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.

**2.4. Создание таблиц**

Для создания таблиц будет использоваться СУБД SQLite. Ее преимущества приведены в пункте 1.

Сами таблицы будут создаваться в соответствии с физической моделью данных.

На рисунках 13-18 представлены структуры каждой таблицы с указанием типа данных и особенностей.

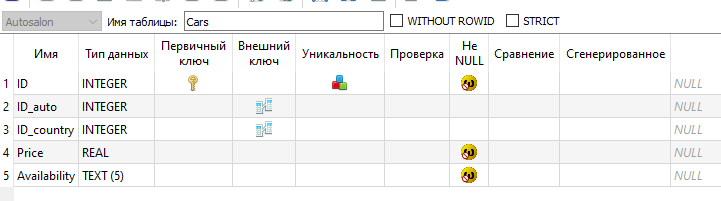


Рис. 13 – Структура таблицы «Cars»

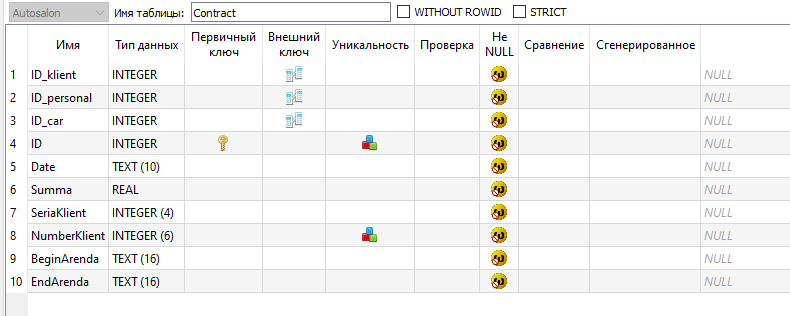


Рис. 14 – Структура таблицы «Contract»

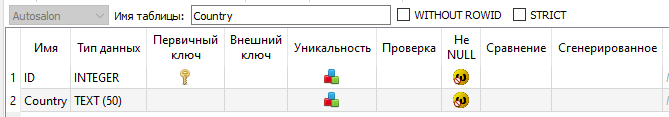


Рис. 15 – Структура таблицы «Country»

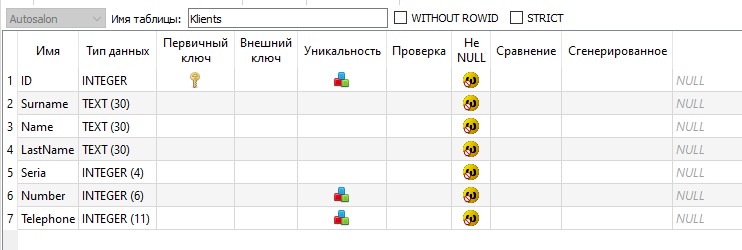


Рис. 16 – Структура таблицы «Klients»

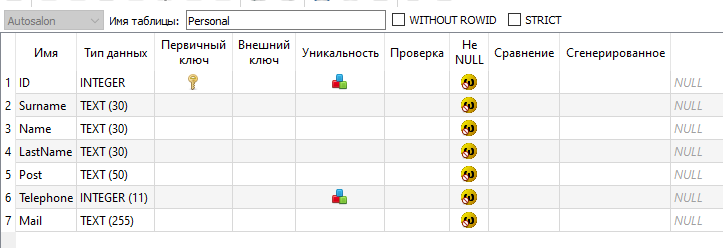


Рис. 17 – Структура таблицы «Personal»

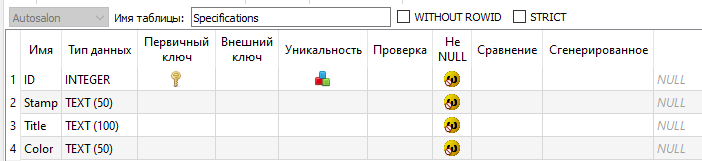


Рис. 18 – Структура таблицы «Specifications»

Также необходимо отметить, что для корректной работы БД, необходимо указать связи между таблицами с помощью персональных и внутренних ключей. На рисунках 19-20 показаны необходимые связи между таблицами.

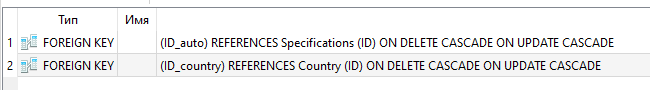


Рис. 19 – Связь таблицы «Cars» с таблицами «Specifications» и «Country»

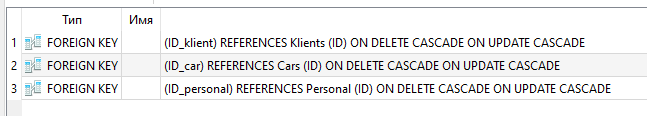


Рис. 20 – Связь таблицы «Contract» с таблицами «Klients», «Cars» и «Personal»

1. **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.

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

Создадим запросы, позволяющие вывести все данные из каждой таблицы.

* + 1. SELECT \* FROM Cars – Вывод всех данных из таблицы «Cars»
    2. SELECT \* FROM Contract – Вывод всех данных из таблицы «Contract»
    3. SELECT \* FROM Country – Вывод всех данных из таблицы «Country»
    4. SELECT \* FROM Klients – Вывод всех данных из таблицы «Klients»
    5. SELECT \* FROM Personal – Вывод всех данных из таблицы «Personal»
    6. SELECT \* FROM Specifications – Вывод всех данных из таблицы « Specifications»

Представления этих запросов показаны на рисунках 21-26.

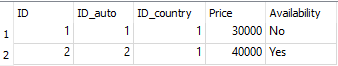


Рис. 21 – Представление таблицы «Cars»



Рис. 22 – Представление к таблице «Contract»



Рис. 23 – Представление к таблице «Country»



Рис. 24 – Представление к таблице «Klients»



Рис. 25 – Представление к таблице «Personal»

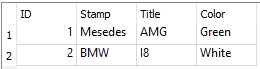


Рис. 26 – Представление к таблице «Specifications»

Также необходимо создать запрос, который будет выводить данные из нескольких таблиц для демонстрирования клиенту имеющихся автомобилей, их характеристик и стоимости.

SELECT s.Stamp, s.Title, s.Color, co.Country AS CountryName, c.Price, c.Availability FROM Specifications s JOIN Cars c ON s.ID = c.ID\_auto JOIN Country co ON c.ID\_country = co.ID

Его представление показано на рисунке 27.

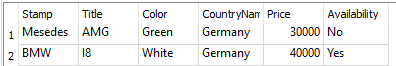


Рис. 27 – Представление данных пользователю

Далее были созданы запросы, необходимые для отображения в программе нужных для выбора данных.

* 1. SELECT ID, Title FROM specifications – Вывод названия автомобиля
  2. SELECT ID, Country FROM country – Вывод названия страны
  3. SELECT ID, Surname || ' ' || Name || ' ' || LastName AS FullName, Seria, Number FROM klients – Вывод ФИО и данных клиента
  4. SELECT ID, Surname || ' ' || Name || ' ' || LastName AS FullName FROM personal – Вывод ФИО сотрудника
  5. SELECT cars.ID, specifications.Title FROM cars JOIN specifications ON cars.ID\_auto = specifications.ID – Вывод характеристик автомобилей по названию

Представления показаны на рисунках 28-32.

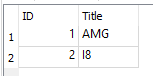


Рис. 28 – Представление 1



Рис. 29 – Представление 2



Рис. 30 – Представление 3



Рис. 31 – Представление 4

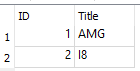


Рис. 32 – Представление 5

1. **Создание клиентского приложения**

Приложение должно состоять из 3 форм: форма авторизации, форма пользователя и форма администратора.

Каждая будет выполнять свою задачу. Форма авторизации позволяет ввести логин и пароль, чтобы определить, кто взаимодействует с приложением. Ее макет показан на рисунке 33.

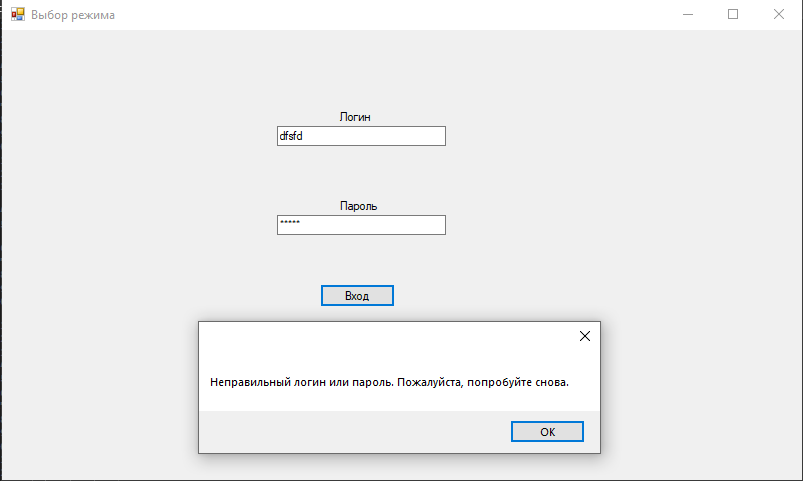


Рис. 33 – Форма авторизации

В случае ввода неправильного логина или пароля будет выдаваться ошибка, соответствующая этому.

Форма пользователя предоставляет возможность просмотра списка автомобилей, имеющихся в салоне, их характеристики, стоимость, а также доступность. Макет представлен на рисунке 34.

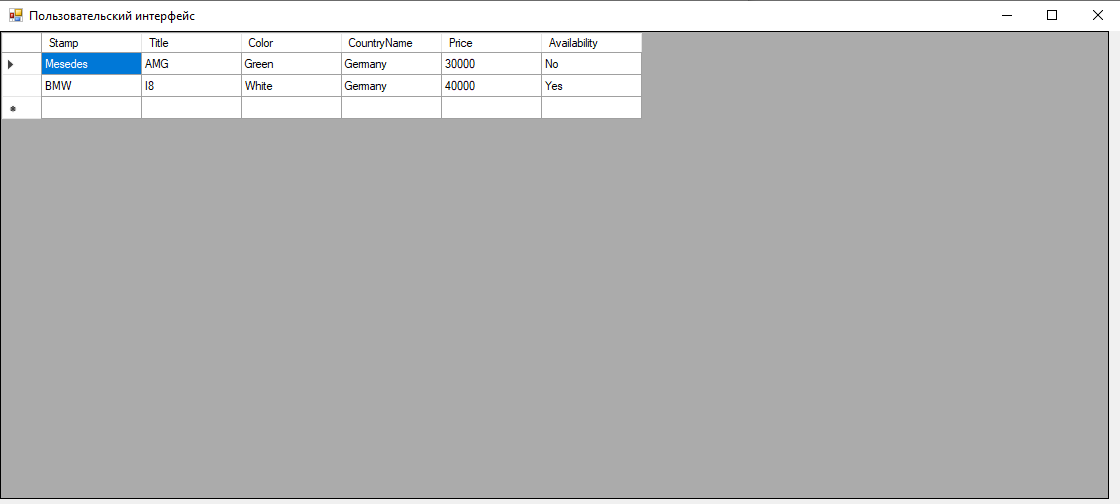


Рис. 34 – Форма пользователя

Форма администратора служит для редактирования БД, добавления новых полей, редактирование или удаление имеющихся. Она представляет собой страницы, каждая из которых служит для своей таблицы. На рисунках 35-41 представлена каждая страница формы.

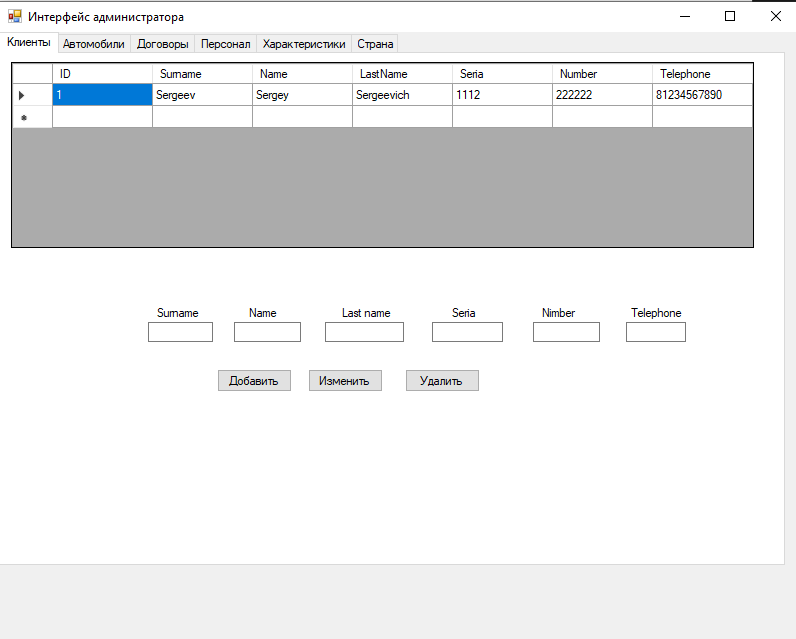


Рис. 35 – Страница для таблицы «Klients»

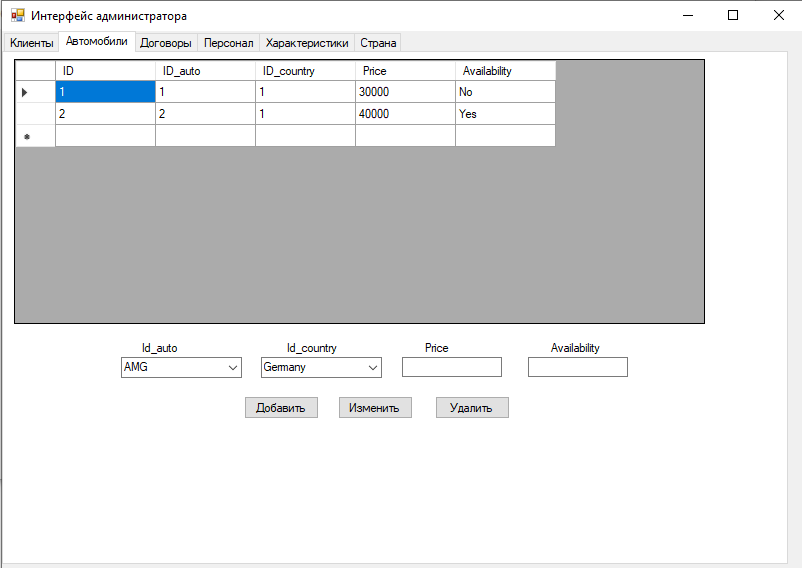
****

Рис. 36 – Страница для таблицы «Cars»

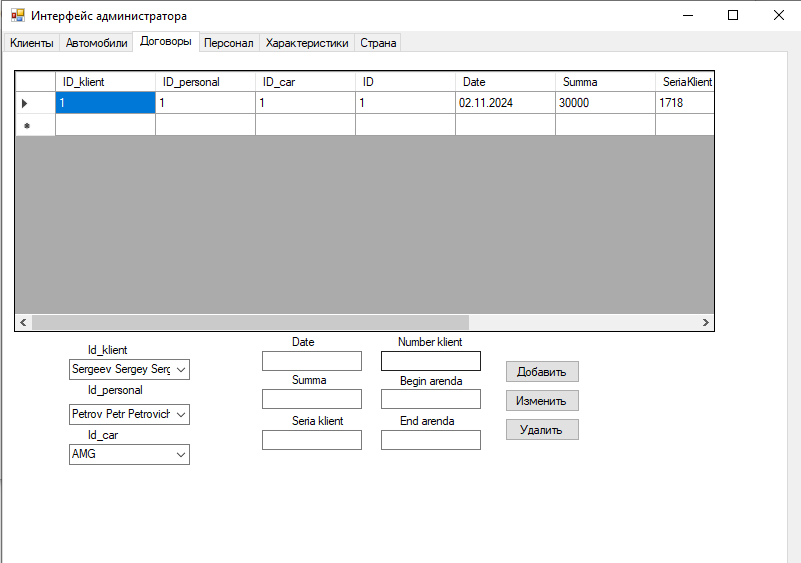


Рис. 37 – Страница для таблицы «Contract»

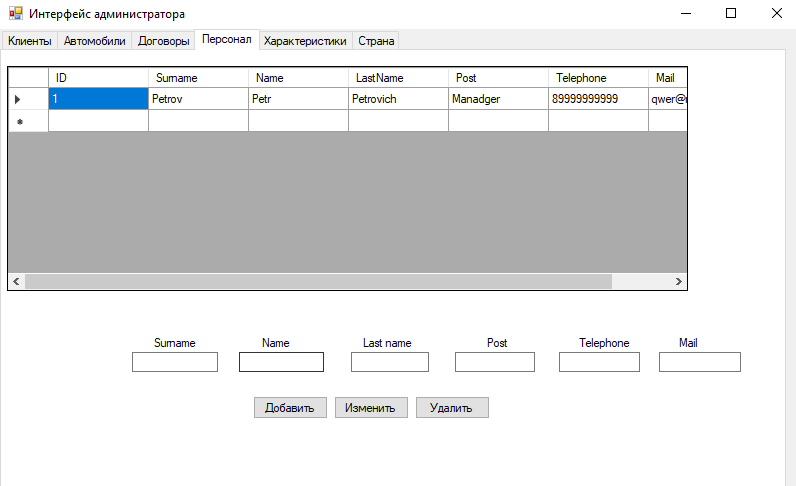


Рис. 38 – Страница для таблицы «Personal»

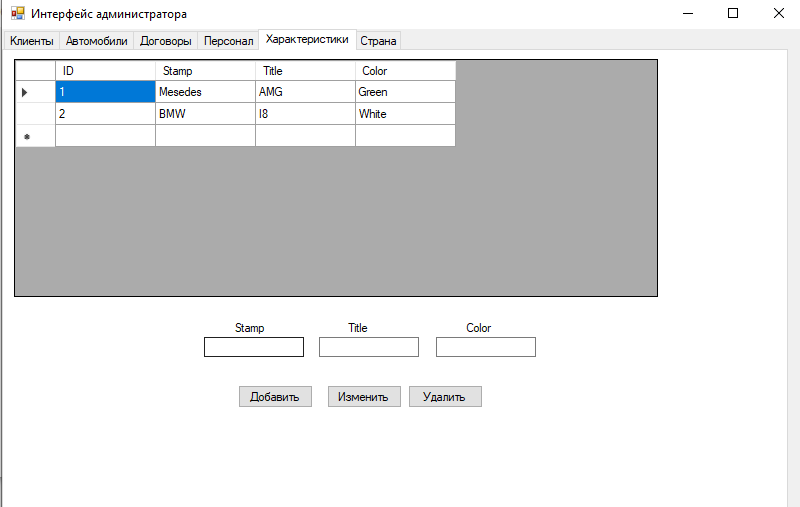


Рис. 39 – Страница для таблицы «Specifications»

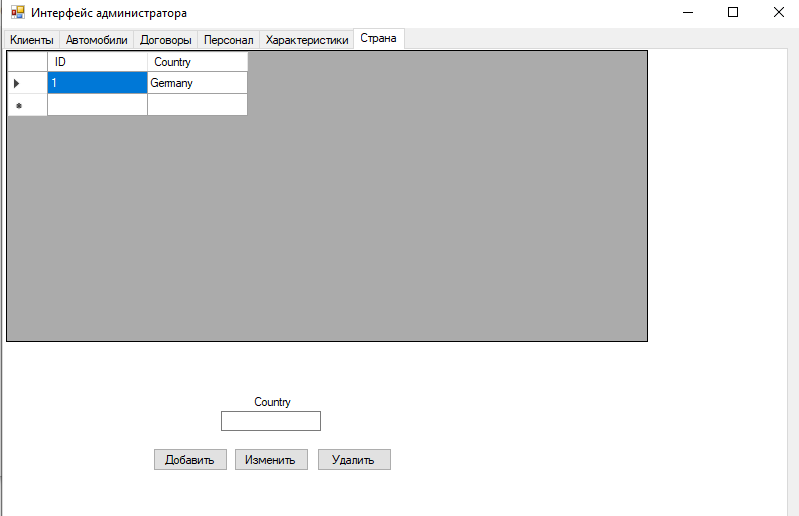


Рис. 40 – Страница для таблицы «Country»

Здесь можно ввести в поля новые значения, которые при нажатии кнопки «Добавить» будут занесены в БД. Также при выборе нужной строки и нажатии кнопки «Изменить» или «Удалить» будет произведена соответствующая операция.

**Заключение**

**Приложение А: «Модели данных»**

**Приложение Б: «Ссылка на GitHub»**

**Список литературы**