Частное учреждение образования

«Колледж бизнеса и права»

АВТОМАТИЗАЦИЯ УЧЕТА АРЕНДЫ МАШИН

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по предмету

«Базы данных и системы управления базами данных»

КП Т.995013.401

Руководитель проекта (К.О. Якимович )

Учащийся (Н.И. Пархомчик)

2023

2023

ТЗ

Изм.

Лист

№ докум.

Подпись

дА

Лист

У 3

КП Т.995013.401 ПЗ

Разраб.

*Пархомчик Н.И.*

Провер.

Якимович К.О.

Т. Контр.

Н. Контр.

Утверд.

Автоматизация учета аренды машин

Лит.

Листов

41

КБП

СОДЕРЖАНИЕ

[Введение 4](#_Toc129888697)

[1 Описание задачи 5](#_Toc129888698)

[1.1 Анализ предметной области 5](#_Toc129888699)

[1.2 Постановка задачи 7](#_Toc129888700)

[2 Проектирование системы 8](#_Toc129888701)

[2.1 Требование к приложению 8](#_Toc129888702)

[2.2 Проектирование модели 8](#_Toc129888703)

[2.3 Проектирование структуры базы данных 10](#_Toc129888704)

[2.4 Концептуальный прототип 11](#_Toc129888705)

[3 Описание реализации программного средства 14](#_Toc129888706)

[3.1 Инструменты разработки и применяемые технологии 14](#_Toc129888707)

[3.2 Организация данных 15](#_Toc129888708)

[3.3 Функции: логическая и физическая организация 18](#_Toc129888709)

[3.4 Входные и выходные данные 20](#_Toc129888710)

[3.5 Функциональное тестирование 21](#_Toc129888711)

[3.6 Описание справочной системы 24](#_Toc129888712)

[4 Применение 25](#_Toc129888713)

[4.1 Назначение программного средства 25](#_Toc129888714)

[4.2 Условия применения 25](#_Toc129888715)

[Заключение 26](#_Toc129888716)

[Список информационных источников 27](#_Toc129888717)

[ПРИЛОЖЕНИЕ А Текст программных модулей 28](#_Toc129888718)

[ПРИЛОЖЕНИЕ Б Результаты работы приложения 36](#_Toc129888719)

Дата

# Введение

Современный мир стремительно движется к оцифровке и автоматизации всех возможных процессов. Однако традиционный процесс аренды автомобиля может быть громоздким и неудобным, особенно в часы пик.

Целью данного курсового проекта на тему «Автоматизация учета аренды машин» является создание программного средства «CarRent.exe», в котором будет организовано ведение базы данных машин, клиентов, и арендованных машин. Будет организовано отображение описанных ранее в введении объектов, а также поиск, фильтрация и сортировка машин по различным категориям в зависимости от специфики машины.

Пояснительная записка к курсовому проекту состоит из четырех разделов, которые содержат решение поставленных задач, а также необходимую информацию по использованию программного средства «CarRent».

В первом разделе «Описание задачи» описан анализ предметной области поставленной задачи и сформулированы задачи, подлежащие автоматизации.

Второй раздел «Проектирование системы» содержит требования к приложению, моделированию всего программного средства и структуры базы данных, представлена модель внешнего пользовательского интерфейса.

Третий раздел «Описание реализации программного средства» содержит описание инструментов разработки и применяемых технологий, описана инфраструктура базы данных, а также логическая и физическая организация данных реализованных функций. Представлено проведение испытаний разработанного программного средства с описанием организации входных и выходных данных.

В четвертом разделе «Применение» предназначен для описания сведений о назначении программного средства и области его применения.

В заключении сформулированы краткие выводы по результатам выполненной работы. Приведены количественные и качественные характеристики реализации базы данных: решенные бизнес-задачи, результаты применения выбранной технологии и т. п.

Приложение А будет содержать текст программы.

Приложение Б будет содержать результаты работы программы.

В графической части будут представлены диаграммы деятельности, классов, вариантов использования.

1. Описание задачи
   1. Анализ предметной области

Темой курсового проекта является «Автоматизация учета аренды машин».

Предметной областью является услуги аренды машин.

Услуги аренды машин является одним из самых востребованных во всех странах мира. Зачастую аренда машин работает в формате салона, предлагая клиентам большой выбор машин различных марок и моделей. Для того, чтобы работа в салоне была максимально эффективной, необходимо внедрить автоматизацию процессов.

Объектами исследования являются:

* машины;
* клиенты;
* аренда.

При исследовании предметной области выделены три основных объекта которые перечислены выше. Так как автоматизация идет от лица сотрудника, с помощью объектов исследования «Машины», «Клиенты» и «Аренда» будет отображён основной функционал услуг аренды машин.

До автоматизации процесс аренды автомобиля был более сложным и требовал больше времени и усилий от потенциального арендатора и арендодателя. Обычно процесс аренды автомобиля выглядел следующим образом:

1. Поиск арендодателя: арендаторы обычно искали арендодателей через местные журналы или рекламные буклеты, а также через рекомендации от друзей и знакомых.
2. Связь с арендодателем: после нахождения подходящего арендодателя, арендатор связывался с ним по телефону или лично, чтобы обсудить детали аренды и договориться о дате и времени получения автомобиля.
3. Оформление документов: при получении автомобиля арендатору необходимо было заполнить ряд документов, включая арендный договор, в котором указывались условия аренды, страховые полисы, права и удостоверения личности. Пример договора аренды автомобиля представлен на рисунке 1.
4. Оплата: арендаторы обычно оплачивали аренду наличными или банковской картой в момент получения автомобиля.
5. Возврат автомобиля: после окончания срока аренды арендатор должен был вернуть автомобиль в том же состоянии, в котором он был получен, и оплатить дополнительные расходы, если таковые возникли.

Этот процесс мог занимать много времени и потребовать много усилий, особенно если арендаторам нужно было искать арендодателя в другом городе или стране. Однако с появлением онлайн-аренды автомобилей процесс стал значительно проще и быстрее.

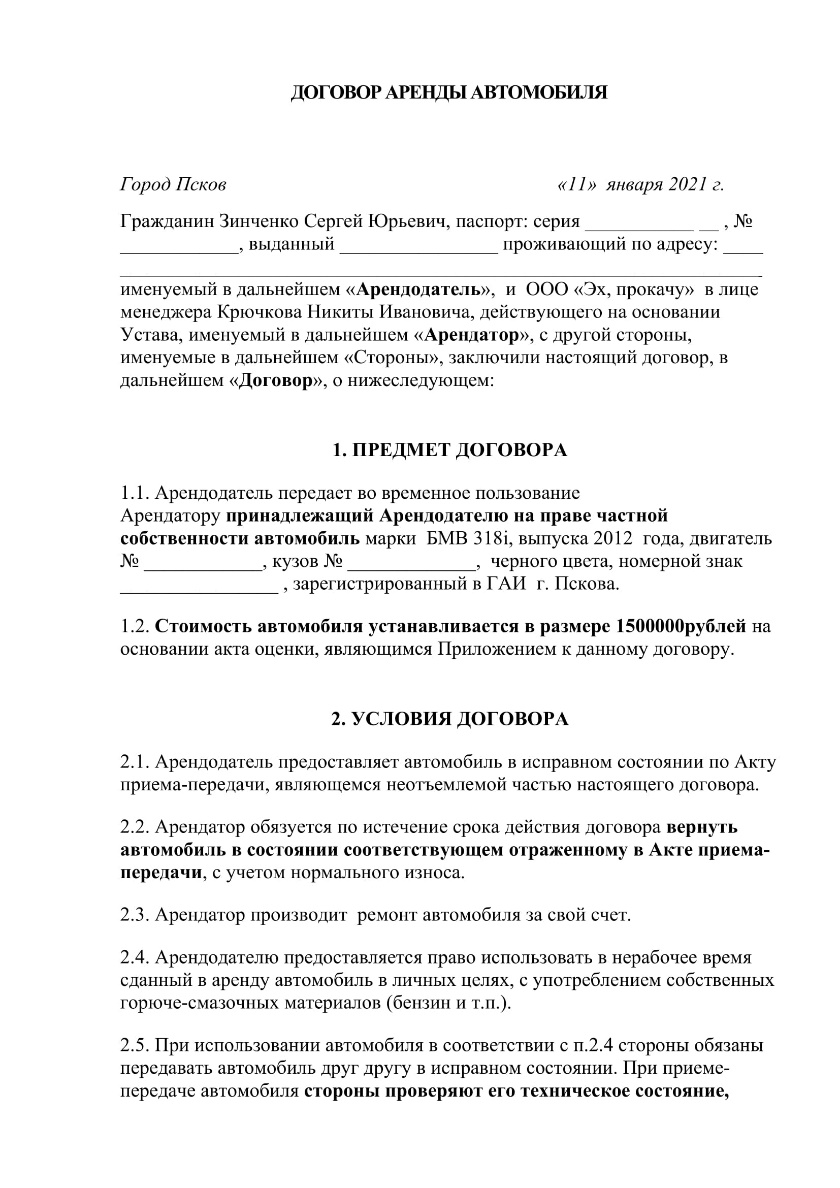


Рисунок 1

При разработке программного средства услуги аренды машин будут отображены сведения о машинах, их марки, модели, список желаемого, архив арендованного и история аренды. Это позволит работнику салона отслеживать услуги аренды используя при этом одно программное средство. Так же, для простоты использования будут добавлены функции, отслеживающие качественный контроль, при аренде и возврате менеджер салона будет получать информацию о том, что машина возвращена. На каждую аренду или возврат будет формирование архивной записи, в которой будет указана информация о клиенте, информация о аренде и информация о выбранном автомобиле. Не мало важными являются функции поиска, фильтрации и сортировки.

* 1. Постановка задачи

Исходя из анализа предметной области можно выделить следующие задачи подлежащие автоматизации:

* ведение базы данных машин, клиентов, списка арендованных машин;
* контроль арендного состояния при аренде автомобилей;
* организовать поиск, фильтрацию и сортировку машин по различным критериям;
* формирование отчета с самыми рентабельными машинами;

Аналогом разрабатываемого программного средства является программа для автоматизации услуг аренды машины «auto-sale». Она предоставляет аренду автомобилей, ведение базы данных автомобилей и отображение марок автомобилей.

Разрабатываемая программа упростит услуги аренды автомобилей, менеджер сможет не только вести учет арендованных автомобилей, а также отслеживать популярность автомобилей, проводить поиск и фильтрацию тех или иных автомобилей используя при этом одно программное средство.

1. Проектирование системы
   1. Требование к приложению

Стилистическое оформление приложения должно соответствовать техническому заданию.

[Требования к интерфейсу программного средства](#_heading=h.3as4poj):

* дизайн приложения должен соответствовать техническому заданию;
* дизайн приложения должен быть лаконичным и в то же время выглядеть стильно, современно.

[Требования к шрифтовому оформлению:](#_heading=h.1pxezwc)

* [размер (кегль) шрифтов должен обеспечивать удобство восприятия текста;](#_heading=h.1pxezwc)
* [предпочтителен шрифт без засечек.](#_heading=h.1pxezwc)

Система управления контентом приложения должна обеспечить пользователю приложения возможность выполнения следующих действий:

* реализовать наглядное отображение всех объектов базы данных;
* осуществить возможность добавления записей в объекты базы данных;
* реализовать просмотр записей;
* организовать сохранение добавленных записей в базу данных;
* обеспечить поиск, фильтрацию и сортировку товаров по различным категориям товара;
* реализовать ограничения на таблицы в которых ведется работа с числовыми полями, вводимые значения должны быть больше нуля;
* реализовать формирование платежного документа на каждую покупку.

Требования к аппаратным и операционным ресурсам:

* клавиатура;
* мышь;
* монитор;
* жесткий диск;
* операционная система Windows 10;
* процессор 11th Gen Intel(R) Core(TM) i7-11800H 4.60 ГГц;
* оперативная память 16,00 Гбайт;
* графический процессор NVIDIA GeForce RTX 3060 Dual OC 12,00 Гбайт.
  1. Проектирование модели

Диаграмма вариантов использования описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Она является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы.

В данной проектируемой системе в качестве актера выступает пользователь, который служит источником воздействия на моделируемую систему.

К основным функциям разрабатываемой программы относятся следующие функции:

* осуществить работу с автомобилями;
* оформить аренду;
* осуществить формирование отчета.

Функция осуществить работу с автомобилями содержит в себе вспомогательные функции:

* добавить;
* удалить;
* изменить.

Диаграмма вариантов использования представлена на листе 1 графической части.

Диаграмма классов представляет собой некоторый граф, вершинами которого являются элементы типа «классификатор» и которые связаны различными типами структурных отношений. Следует заметить, что диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи. Когда говорят о данной диаграмме, имеют в виду статическую структурную модель проектируемой системы. Поэтому диаграмму классов принято считать графическим представлением таких структурных взаимосвязей логической модели системы, которые не зависят от времени.

К основным классам относятся: клиенты, машины, характеристики, аренды, пользователи. В классе «Клиент» хранятся данные, описывающие клиента, а именно «Имя», «Электронная почта», «Номер телефон» и «Номер паспорта». В классе «Аренды» хранятся данные «Время аренды», «Дата аренды», «Дата конца аренды», «Итоговая цена». В классе «Характеристики» хранятся следующие данные «Марка», «Модель», «Мощность двигателя», «Коробка передач», «Привод» и «Тип топлива». В «Машины» хранится «Государственный номер», «Пробег», «Год создания», «Цвет», «Стоимость за час». В классе «Пользователи» хранится «Имя», «Электронная почта», «Логин» и «Пароль». Диаграмма классов представлена на листе 2 графической части.

Диаграммы деятельности – частный случай диаграмм состояний. Основная цель использования таких диаграмм – визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности описывает собой процесс заказа товара и оформления платежного документа, он включает в себя проверку на наличие товара. Диаграмма деятельности представлена на листе 3 графической части.

* 1. Проектирование структуры базы данных

Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая стока таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности.

Схема «сущность-связь» (ER-диаграмма) – это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ER-диаграммы (ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи – глаголов. Диаграмма «сущность-связь» представлена на рисунке 2.

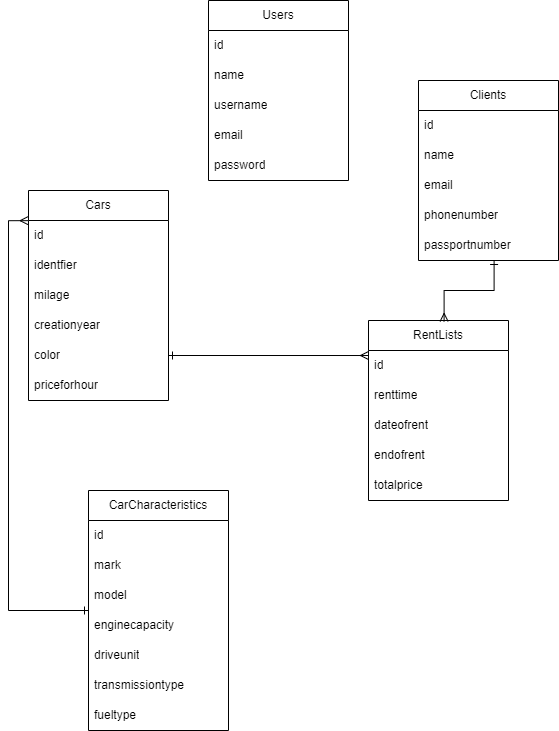


Рисунок 2

Исходя из предметной области можно выделить следующие сущности разработки: «CarCharacteristics», «Cars», «Clients», «RentList», «Users».

Для сущности «CarCharacteristics» атрибутами будут являться:

* mark;
* model;
* enginecapacity;
* driveunit;
* transmissiontype;
* fueltype.

Для сущности «Cars» атрибутами будут являться:

* milage;
* creationyear;
* priceforhour;
* color.

Для сущности «Clients» атрибутами будут являться:

* name;
* email;
* phonenumber;
* passportnumber.

Для сущности «RentLists» атрибутами будут являться:

* renttime;
* dateofrent;
* enndofrent;
* totalprice.

Для сущности «Users» атрибутами будут являться:

* name;
* username;
* email;
* password.
  1. Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – систему кнопок и форм.

Главное меню программы будет представлено формой Main, на которой находятся основные элементы управления программой. С помощью кнопок на главной форме можно получить доступ к другим формам, в которых отображаются сведения о объектах базы данных.

Главная форма приложения представляет собой окно в которой располагаются основные кнопки, при нажатии на них открываются формы, содержащие в себе таблицы и кнопки управления объектами базы данных.

В зависимости от выбранного элемента управления рабочей области, предназначенной для сопровождения основных таблиц базы данных, либо для организации форм отчетов, результатов фильтрации и поиска изменяется и само отображение. Макет формы представлен на рисунке 3.

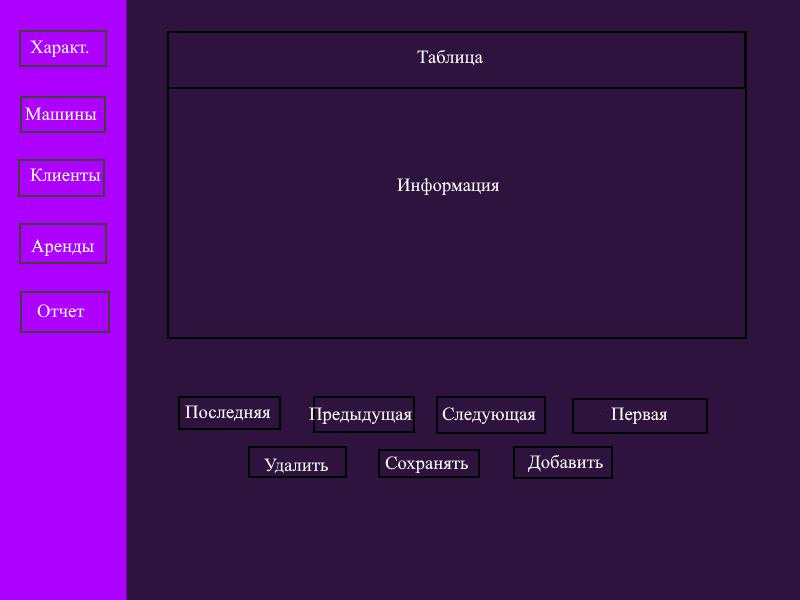


Рисунок 3

Для удобного выполнения поиска и фильтрации по интересующей пользователя информации создана отдельная форма. На данной форме располагаются кнопки управления фильтрами, таблица с данными и поле с критериями, по которым будет выполняться поиск и фильтрация. Макет формы с поиском и фильтрацией представлен на рисунке 4.

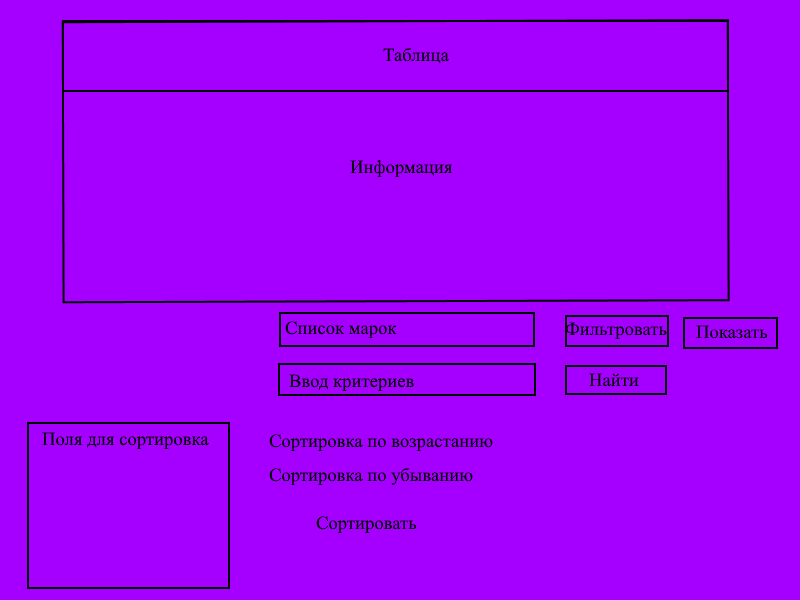


Рисунок 4

1. Описание реализации программного средства
   1. Инструменты разработки и применяемые технологии

Программное обеспечение, используемое для разработки программного средства:

* среда разработки Visual Studio 2022 Community;
* операционная система Windows 10;
* пакет программ Microsoft Office;
* программа для создания файлов справки Dr.Explain;
* программа для разработки диаграмм draw.io;
* структурированный язык запросов SQL;
* среда для управления SQL Server Management Studio;
* ядро реляционной база данных MSSQL 2019 Express (версия 15.0.2000.5).

Microsoft Visual Studio 2022 Community – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio [12].

Windows 10 – операционная система семейства Windows NT производства корпорации Microsoft. Предназначена для рабочих станций, персональных компьютеров и портативных устройств; версия, предназначенная для решения серверных задач [8].

Microsoft Office – это совокупность программных средств автоматизации офисной деятельности. В состав пакета входит множество приложений, каждое из которых предназначено для выполнения определенных функций и может быть использовано автономно и независимо от остальных [11]*.*

Dr.Explain – это приложение для быстрого создания файлов справки, справочных систем, онлайн-руководств пользователя, пособий и технической документации к программному обеспечению и техническим системам [10]*.*

Программа «draw.io» – это интернет-ресурс, позволяющий разрабатывать различные диаграммы. Имеет огромный функционал и все необходимые инструменты, необходимые для разработки диаграмм [9]*.*

SQL – это структурированный язык запросов, созданный для того, чтобы получать из базы данных необходимую информацию. Если описать схему работы SQL простыми словами, то специалист формирует запрос и направляет его в базу. Та в свою очередь обрабатывает эту информацию, «понимает», что именно нужно специалисту, и отправляет ответ [4].

SQL Server Management Studio (SSMS) – это интегрированная среда для управления любой инфраструктурой SQL, которая используется для доступа, настройки, администрирования и разработки всех компонентов SQL Server [2]*.*

SQL Server Express – это бесплатная база данных начального уровня, идеально подходящая для обучения и создания классических и небольших серверных приложений, управляемых данными. Этот выпуск – лучший выбор для независимых поставщиков программного обеспечения, непрофессиональных разработчиков и любителей, создающих клиентские приложения [4].

* 1. Организация данных

Реляционная база данных – это пространство, в котором связанную информацию хранят в нескольких таблицах. При этом есть возможность запрашивать информацию в нескольких таблицах одновременно.

Несомненно, хранилище данных – один из основ­ных компонентов, определяющих производи­тельность и доступность больших и малых экзем­пляров SQL Server. В условиях, возросших вычис­лительных возможностей физических и вирту­альных серверов и поддержки объемной памяти хранилища данных и подсистема ввода-вывода могут оказаться узки­ми местами, снижающими общую пропускную способ­ность. Неприятностей можно избежать, если иметь общее представление о том, как SQL Server использует хранилища данных, и знать основные приемы оптимальной организа­ции хранилищ SQL Server.

На основании ER-модели была создана база данных. Структура данных и их краткое описание приводятся в таблицах 1-5.

Таблица «CarCharacteristics» хранит информацию о характеристиках, имеющихся у машин в салоне, структура приведена в таблице 1.

Таблица 1 – Структура таблицы «CarCharacteristics»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | int | 4 | Первичный ключ в таблице «CarCharacteristics» |
| mark | nvarchar | 50 | Марка машины |
| model | nvarchar | 50 | Модель машины |
| enginecapacity | float | 8 | Мощность двигателя машины |
| driveunit | nvarchar | 50 | Привод машины |
| transmissiontype | nvarchar | 50 | Тип передачи машины |
| fueltype | nvarchar | 50 | Тип топлива |

Таблица «Cars» хранит информацию о машинах, структура приведена в таблице 2.

Таблица 2 – Структура таблицы «Cars»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | int | 4 | Первичный ключ в таблице «Cars» |
| characteristicsid | int | 4 | Внешний ключ с помощью которого определяются характеристики машины |
| identfier | nvarchar | 50 | Государственный номер машины |
| milage | int | 4 | Пробег машины |
| creationYear | int | 4 | Год создания машины |
| color | nvarchar | 50 | Цвет машины |
| priceforhour | float | 8 | Стоимость машины за час |

Таблица «Clients» хранит информацию о клиентах, структура приведена в таблице 3.

Таблица 3 – Структура таблицы «Clients»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | int | 4 | Первичный ключ в таблице «Clients» |
| name | nvarchar | 50 | Имя клиента |
| email | nvarchar | 50 | Электронная почта клиента |
| phonenumber | nvarchar | 50 | Номер телефона клиента |
| passportnumber | nvarchar | 50 | Номер паспорта клиента |

Таблица «RentLists» хранит информацию о арендованных машинах в текущий момент, структура приведена в таблице 4.

Таблица 4 – Структура таблицы «RentLists»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | int | 4 | Первичный ключ в таблице «RentLists» |
| clientid | int | 4 | Внешний ключ с помощью которого определяется клиента арендовавший машину |
| carid | int | 4 | Внешний ключ с помощью которого определяется арендованная машина |
| renttime | int | 4 | Время на которое арендована машина в часах |
| dateofrent | datetime | 8 | Дата и время аренды машины |
| endofrent | datetime | 8 | Дата и время окончания аренды |
| totalprice | float | 8 | Итоговая цена аренды |

Таблица «Users» хранит информацию о менеджерах, структура приведена в таблице 5.

Таблица 5 – Структура таблицы «Users»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | int | 4 | Первичный ключ в таблице «Users» |
| name | nvarchar | 50 | Имя менеджера |
| username | nvarchar | 50 | Псевдоним менеджера |
| email | nvarchar | 50 | Электронная почта менеджера |
| password | nvarchar | 50 | Пароль от учетной записи менеджера |

Все описанные таблицы выше созданы в реляционной базе данных SQL. Диаграмма базы данных представлена на рисунке 5.

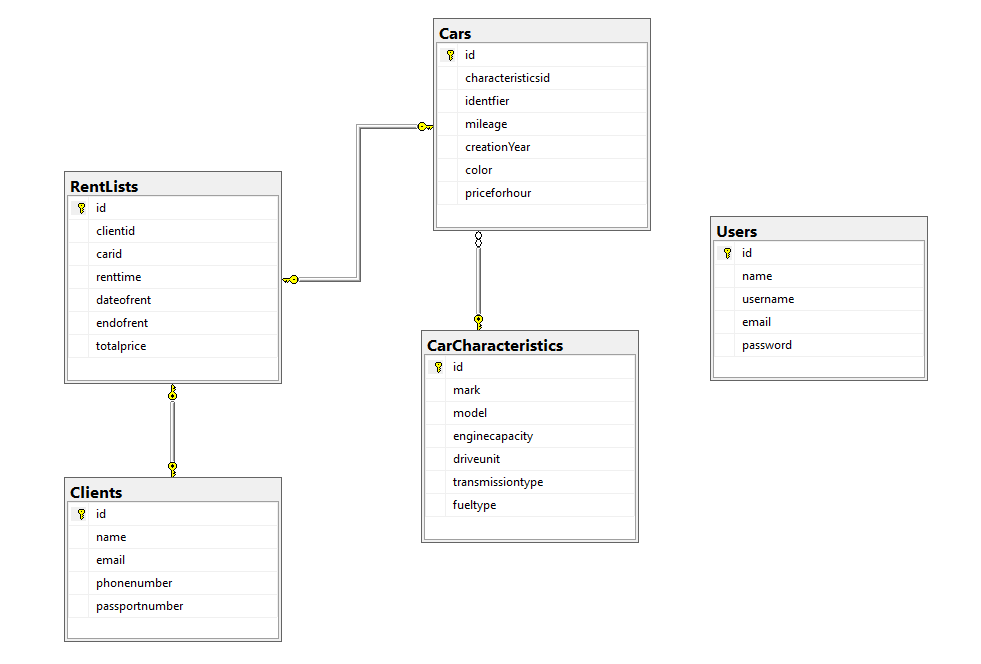


Рисунок 5

* 1. Функции: логическая и физическая организация

При разработке программного средства для автоматизации работы менеджера по продаже смартфонов были созданы следующие объекты базы данных:

* представления;
* хранимые процедуры;
* триггеры.

При реализации программного средства «CarRent» был создан триггер «OnRentListsInsert». Данный триггер срабатывает при добавлении записей в таблицу «RentLists». При добавлении пользователь указывает время аренды в часах, автомобиль и клиента. В случае добавления записи, в которой выбран арендованный автомобиль или клиент, который уже арендовал автомобиль пользователь получает сообщение «Данный автомобиль уже арендован или данный клиент уже арендовал другой автомобиль».

GO

CREATE OR ALTER TRIGGER OnRentListsInsert

ON RentLists

INSTEAD OF INSERT

AS

BEGIN

INSERT INTO RentLists(carid, clientid, renttime, dateofrent, endofrent, totalprice)

SELECT i.carid, i.clientid, i.renttime, CURRENT\_TIMESTAMP, DATEADD(HOUR, i.renttime, CURRENT\_TIMESTAMP), c.priceforhour \* i.renttime

FROM inserted i

JOIN Cars c ON c.id = i.carid

END;

Для установки пересчитанной итоговой цены был разработан триггер. Данный триггер срабатывает при изменении записей в таблице «OnRentListsUpdate». При добавлении пользователь указывает государственный номер машины, номер паспорта клиента, и время аренды в часах. Триггеры на основе выбранной машины и времени аренды подсчитывают итоговую стоимость аренды после чего изменяет запись.

GO

CREATE OR ALTER TRIGGER OnRentListsUpdate

ON RentLists

INSTEAD OF UPDATE

AS

BEGIN

UPDATE rl SET

rl.carid = i.carid,

rl.clientid = i.clientid,

rl.renttime = i.renttime,

rl.endofrent = DATEADD(HOUR, i.renttime, CURRENT\_TIMESTAMP),

rl.totalprice = c.priceforhour \* i.renttime

FROM RentLists rl

JOIN INSERTED i ON i.id = rl.id

JOIN Cars c ON c.id = i.carid

END;

Для получения информации об арендованных машинах их марках, моделях, арендаторах, времени аренды, дате аренды, дате возврата, итоговой цене была разработана процедура «RentListsReport». Она включает в себя выборку данных из объекта базы данных «RentLists» и присоединение объектов базы данных: «Cars», «CarCharacteristics», «Clients». При использовании данной процедуры пользователю необходимо ввести дату аренды и конечную дату аренды. Входными параметрами для данной процедуры являются два дейттайма которые выбирает пользователь (начальная дата и конечная дата). Данное представление используется при формировании отчета об арендованных автомобилях на форме «Report».

CREATE OR ALTER PROCEDURE RentListReport(@startDate datetime, @endDate datetime)

AS

BEGIN

SET NOCOUNT ON;

SELECT cc.mark, cc.model, c.identfier, cl.name, cl.passportnumber, rl.renttime, rl.dateofrent, rl.endofrent, rl.totalprice

FROM RentLists rl

JOIN Cars c ON rl.carid = c.id

JOIN CarCharacteristics cc ON c.characteristicsid = cc.id

JOIN Clients cl ON rl.clientid = cl.id

WHERE CAST(@startDate AS date) = CAST(rl.dateofrent AS date) AND CAST(@endDate AS date) = CAST(rl.endofrent AS date);

END;

Для получения информации о поставках товара было разработано представление «RentListsView». Оно включает в себя выборку данных из объекта базы данных «RentLists» и присоединение объектов базы данных: «Cars», «CarCharacteristics», «Clients». Данное представление используется при формировании отчета о поставках на форме «SortRentLists».

CREATE OR ALTER VIEW RentListView(mark, model, identfier, name, passportnumber,

renttime, dateofrent, totalprice) AS SELECT cc.mark, cc.model, c.identfier, cl.name, cl.passportnumber, rl.renttime, rl.dateofrent, rl.totalprice

FROM RentLists rl

JOIN Cars c ON rl.carid = c.id

JOIN CarCharacteristics cc ON c.characteristicsid = cc.id

JOIN Clients cl ON rl.clientid = cl.id;

Полный код программы представлен в приложении А.

* 1. Входные и выходные данные

Входные данные – это данные, которые передаются в программу или функцию для обработки. Эти данные могут быть представлены в различных форматах, таких как числа, строки, булевы значения и т.д. Входные данные могут быть получены от пользователя, из файла, базы данных или других источников.

К входным данным относится информация, которая добавляется на формах: «Характеристики», «Машины», «Клиенты», «СпискиАренды», «Пользователи». Описание входных данных представлено в таблице 6.

Таблица 6 – Описание входных данных

|  |  |
| --- | --- |
| Наименование таблицы | Наименование полей для заполнения |
| Характеристики | Марка, модель, объем двигателя, тип передачи, тип привода, тип топлива. |
| Машины | Характеристика, государственный номер, пробег, год создания, цвет, стоимость за час. |
| Клиенты | Фио, электронная почта, номер телефона, номер паспорта. |
| СпискиАренды | Клиент, машина, время аренды. |
| Пользователи | Фио, логин, электронная почта, пароль. |

Выходные данные – это результат обработки входных данных программой или функцией. Эти данные также могут быть представлены в различных форматах, в зависимости от того, что программа или функция должна вернуть. Например, это может быть число, строка, булево значение, массив или объект.

К выходным данным относятся документы, которые формируются в результате работы программы. К выходным данным относятся: платежный документ, отчеты, результаты поиска и фильтрации. Результат формирования отчета по поставкам с входными параметрами представлен в приложении Б на рисунке Б.1. Результат получения выходных данных в результате поиска и фильтрации поставок телефонов представлен в приложении Б на рисунке Б.2.

* 1. Функциональное тестирование

В процессе написания программного средства необходимо производить тестирование на правильность работы программного средства. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование приложения будет оформлено в виде тест-кейсов. Все действия начинаются с главной формы приложения. Макет главной формы представлен в приложении Б рисунок Б.3.

В таблице 7 приводится тест-кейс для добавления нового автомобиля.

Таблица 7 – Добавление нового автомобиля

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 001 | Добавление нового автомобиля в базу данных | 1. На левой боковой панели нажать на кнопку «Машины» 2. На навигационной панели нажать на кнопку «Добавить» 3. Заполнить поля «characteristicsid» – «Porsche», «identfier» – «GG3Z», «milage» – «15000», «creation Year» – «2017», «color» – «Черный», «priceforhour» – «150» 4. Нажать на кнопку «Сохранить» | Ожидаемый результат: успешное добавление нового автомобиля в базу данных |
| Фактический результат: успешное добавление нового автомобиля «Porsche Panamera» в базу данных. Автомобиль добавлен и отображается в таблице. Результат о добавлении телефона представлен в приложении Б рисунок Б.6 |

В таблице 8 представлен тест-кейс для добавления аренды автомобиля.

Таблица 8 – Добавление аренды автомобиля

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 002 | Добавление аренды автомобиля | 1. На левой боковой панели нажать на кнопку «Аренды» 2. На навигационной панели нажать на кнопку «Добавить» 3. Заполнить поля «clientid» – «MPT567», «carid» – «GG3Z», «renttime» – «5», «dateofrent» – «22 марта 2023», «endofrent» – «24 марта 2023» 4. Нажать на кнопку «Сохранить» | Ожидаемый результат: успешное добавление аренды в таблицу |
| Фактический результат: успешное добавление телефона в таблицу. Результат о добавлении телефона в заказ представлен в приложении Б рисунок Б.7 |

В таблице 9 представлен тест-кейс для добавления аренды на арендованный автомобиль.

Таблица 9 – Добавление аренды на арендованный автомобиль

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 003 | Добавление аренды на арендованный автомобиль | 1. На левой боковой панели нажать на кнопку «Аренды» 2. На навигационной панели нажать на кнопку «Добавить» 3. Заполнить поля «clientid» – «MPT567», «carid» – «GG3Z», «renttime» – «5», «dateofrent» – «24 марта 2023», «endofrent» – «24 марта 2023» 4. Нажать на кнопку «Добавить в заказ» | Ожидаемый результат: появление всплывающего сообщения об ошибке |
| Фактический результат: появление всплывающего сообщения об ошибке. Результат о добавлении некорректного количества телефона в заказ представлен в приложении Б рисунок Б.8 |

В таблице 10 представлен тест-кейс для проверки фильтрации арендованного автомобиля по определённой марке.

Таблица 10 – Проверка фильтрации арендованного автомобиля по определённой марке

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 004 | Проверка фильтрации арендованного автомобиля по определённой марке | 1. На левой боковой панели нажать на кнопку «Аренда» 2. На навигационной панели нажать на кнопку «Сортировка» 3. В поле марки выбрать «Porsche» 4. Нажать на кнопку «Фильтровать» | Ожидаемый результат: на форме отобразились автомобили только с маркой «Porsche» |
| Фактический результат: на форме отобразились автомобили только с маркой «Porsche». Результат фильтрации представлен в приложении Б рисунок Б.9 |

В таблице 11 представлен тест-кейс для поиска арендованного автомобиля с существующими критериями.

Таблица 11 – Поиск арендованного автомобиля с существующими критериями

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 005 | Поиск арендованного автомобиля с существующими критериями | 1. На левой боковой панели нажать на кнопку «Аренда» 2. На навигационной панели нажать на кнопку «Сортировка» 3. В текстовое поле «Поиск» ввести «Porsche» 4. Нажать на кнопку «Найти» | Ожидаемый результат: на форме ячейки с нужными критериями выделены зеленым цветом |
| Фактический результат: на форме ячейки с нужными критериями выделены зеленым цветом. Результат поиска автомобиля представлен в приложении Б рисунок Б.10 |

В таблице 12 представлен тест-кейс для поиска арендованного автомобиля с несуществующими критериями.

Таблица 12 – Поиск арендованного автомобиля с несуществующими критериями

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 006 | Поиск арендованного автомобиля с несуществующими критериями | 1. На левой боковой панели нажать на кнопку «Аренда» 2. На навигационной панели нажать на кнопку «Сортировка» 3. В текстовое поле «Поиск» ввести «Pagani» 4. Нажать на кнопку «Найти» | Ожидаемый результат: на форме ячейки не изменяются |
| Фактический результат: на форме ячейки не изменяются. Результат поиска автомобиля представлен в приложении Б рисунок Б.11 |

* 1. Описание справочной системы

Справочная система – это программный компонент, выполняющий роль документации для программного пакета. В первую очередь он описывает и объясняет каждую функцию программы, кнопку, параметр панели инструментов или другой элемент в пользовательском интерфейсе.

Руководство использования программным средством «Spravka.chm» описывает работу со следующими формами: форма для сортировки и фильтрации, форма для управления объектами базы данных, форма для формирования отчета. Пример формы для сортировки и фильтрации представлен в приложении Б на рисунке Б.4. Пример формы для управления объектами базы данных представлен в приложении Б на рисунке Б.5. Пример формы для формирования отчета представлен в приложении Б на рисунке Б.12.

1. Применение
   1. Назначение программного средства

Программное средство «CarRent» включает в себя ведение базы данных характеристик, машин, аренд, клиентов. С помощью данного программного средства упрощается работа менеджера по оказанию услуг аренды автомобилей, облегчается поиск автомобилей и аренд, а также поиск и фильтрация по интересующей менеджера информации. В программном средстве предусмотрен учет аренды автомобилей и получение сообщения об ошибке в случае попытки аренды уже арендованных автомобилей.

Данное программное средство применимо во всех салонах аренды автомобилей.

* 1. Условия применения

Для применения данного программного средства необходимо установить SQL Server Management Studio (SSMS). С помощью SSMS необходимо развернуть базу данных с помощью приложенного скрипта «Create.sql». После создания базы данных программное средство работает автономно. Для экспорта отчетов и платежного документа необходимо иметь установленные на компьютере программные средства Microsoft Word и Microsoft Excel.

Для того чтобы запустить проект необходимо:

1. Перенести проект на компьютер
2. Развернуть базу данных с помощью запроса «Create.sql»
3. В проекте изменить строку подключения к базе данных в файле «App.config»

# Заключение

В рамках курсового проекта на тему «Автоматизация учета аренды машин» было разработано программное средство «CarRent». Позволяющее вести ведение базы данных характеристик, машин, клиентов, аренд. Предусмотрена организация поиска, фильтрации и сортировки арендованных автомобилей по различным категориям в зависимости от специфики автомобиля, реализован учет аренды.

Для достижения цели курсового проекта были решены следующие задачи:

* по модели выполнено проектирование задачи;
* разработана физическая и логическая модель данных;
* разработано программное средство;
* описано созданное программное средство;
* выбрана методика испытаний;
* описан процесс тестирования;
* приведены примеры области применения.

Разработка имеет интуитивно понятный графический интерфейс, позволяющий даже с минимальным знанием использования персонального компьютера достичь максимум производительности в работе.

Программное средство реализовано в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

Программное средство готово к практическому использованию и может быть дополнено и модернизировано.

В процессе разработки программного средства использовался в большом объеме материал базам данных и системами управления базами данных, что способствовало закреплению наработанных навыков и умений в этих областях знаний. При разработке программного средства наибольшее внимание уделялось максимальному созданию лаконичного интерфейса.

Для разработки программного средства использовался язык программирования C#, а также ядро реляционной базы данных SQL Server Express.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – М.: КБП, 2013. – 29 c.
2. Бондарь, А.Г. Microsoft SQL Server 2012 / А.Г. Бондарь. – СПб.: БХВ-Петербург, 2013. – 608 с.
3. Мацяшек, Лешек А. Анализ и проектирование информационных систем с помощью UML 2.0 / Лешек А. Мацяшек. – 3-е изд. – М.: ООО «И.Д. Вильямс», 2008. – 816 с.
4. Митин, А.И. Работа с базами данных Microsoft SQL Server: сценарии практических занятий / А. И. Митин. – М.: Директ-Медиа, 2020. – 142 с.
5. Общие требования к тестовым документам: ГОСТ 2.105-95. – Введ. 01.01.1996. – М.: Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
6. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.301-2000. – Введ. 01.09.2001. – М.: Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
7. Текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – М.: Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
8. Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Microsoft, 2023 – Режим доступа: https://www.microsoft.com/ru-ru/windows/features. – Дата доступа: 17.02.2023.
9. Draw.io – бесплатное средство для создания блок-схем, инфографики, прототипов [Электронный ресурс]. – EL-BLOG.RU, 2023 – Режим доступа: https://el-blog.ru/draw-io/. – Дата доступа: 25.02.2023.
10. Dr.Explain – приложение для быстрого создания файлов справки [Электронный ресурс]. – Режим доступа: https://help-spravka.ru/blog/13-hat-programs/49-dr-explain-otechestvennaya-programma-dlya-razrabotki-spravki – Дата доступа 01.03.2023.
11. Microsoft Office – офисный пакет приложений, созданных корпорацией Microsoft для операционных систем Windows [Электронный ресурс]. – ru.wikipedia.org, 2023. – Режим доступа: [https://ru.wikipedia.org/wiki/Microsoft\_Office.](https://office-guru.ru/excel/microsoft-office-excel-chto-eto-59.html.) – Дата доступа: 13.03.2023.
12. Visual Studio 2022 [Электронный ресурс]. – Microsoft, 2023. – Режим доступа: https://visualstudio.microsoft.com/ru/vs. – Дата доступа 17.02.2023.

ПРИЛОЖЕНИЕ А

**(обязательное)**

**Текст программных модулей**

CREATE DATABASE autodb;

USE autodb;

CREATE TABLE [Users] (

id int PRIMARY KEY IDENTITY,

[name] nvarchar(50) NOT NULL,

username nvarchar(50) NOT NULL,

email nvarchar(50) NOT NULL,

[password] nvarchar(50) NOT NULL

);

CREATE TABLE CarCharacteristics (

id int NOT NULL PRIMARY KEY IDENTITY(1,1),

mark varchar(50) NOT NULL,

model varchar(50) NOT NULL,

enginecapacity float NOT NULL,

driveunit varchar(50) NOT NULL,

transmissiontype varchar(50) NOT NULL,

fueltype varchar(50) NOT NULL

);

CREATE TABLE Cars (

id int PRIMARY KEY IDENTITY(1,1),

characteristicsid int,

identfier NVARCHAR(50) NOT NULL,

mileage int NOT NULL,

creationYear int NOT NULL,

color varchar(50) NOT NULL,

priceforhour FLOAT NOT NULL,

FOREIGN KEY (characteristicsid) REFERENCES CarCharacteristics(id)

);

CREATE TABLE Clients (

id int PRIMARY KEY IDENTITY,

[name] nvarchar(50) NOT NULL,

email nvarchar(50) NOT NULL,

phonenumber nvarchar(50) NOT NULL,

passportnumber nvarchar(50) UNIQUE NOT NULL,

);

CREATE TABLE RentLists (

id int PRIMARY KEY IDENTITY,

clientid INT NOT NULL UNIQUE,

carid INT NOT NULL UNIQUE,

renttime INT NOT NULL,

dateofrent DATETIME,

endofrent DATETIME,

totalprice float,

FOREIGN KEY (clientid) REFERENCES Clients(id),

FOREIGN KEY (carid) REFERENCES Cars(id)

);

CREATE TABLE ArchiveRentLists (

id int PRIMARY KEY IDENTITY,

[name] nvarchar(50) NOT NULL,

email nvarchar(50) NOT NULL,

phonenumber nvarchar(50) NOT NULL,

passportnumber nvarchar(50) NOT NULL,

mileage int NOT NULL,

creationYear int NOT NULL,

color varchar(50) NOT NULL,

priceforhour FLOAT NOT NULL,

mark varchar(50) NOT NULL,

model varchar(50) NOT NULL,

enginecapacity float NOT NULL,

driveunit varchar(50) NOT NULL,

transmissiontype varchar(50) NOT NULL,

fueltype varchar(50) NOT NULL,

renttime INT NOT NULL,

totalprice float,

);

CREATE TABLE ArchiveUsers (

id int PRIMARY KEY IDENTITY,

[name] nvarchar(50) NOT NULL,

username nvarchar(50) NOT NULL,

email nvarchar(50) NOT NULL,

[password] nvarchar(50) NOT NULL

);

CREATE TABLE ArchiveCars (

id int PRIMARY KEY IDENTITY,

mark varchar(50) NOT NULL,

model varchar(50) NOT NULL,

enginecapacity float NOT NULL,

driveunit varchar(50) NOT NULL,

transmissiontype varchar(50) NOT NULL,

fueltype varchar(50) NOT NULL,

mileage int NOT NULL,

creationYear int NOT NULL,

color varchar(50) NOT NULL,

priceforhour FLOAT NOT NULL,

);

GO

CREATE OR ALTER TRIGGER OnUserDelete

ON [Users]

FOR DELETE

AS

INSERT INTO ArchiveUsers ([name], username, email, [password])

SELECT [name], username, email, [password]

FROM [Users]

GO

CREATE OR ALTER TRIGGER OnRentListsInsert

ON RentLists

INSTEAD OF INSERT

AS

BEGIN

INSERT INTO RentLists(carid, clientid, renttime, dateofrent, endofrent, totalprice)

SELECT i.carid, i.clientid, i.renttime, CURRENT\_TIMESTAMP, DATEADD(HOUR, i.renttime, CURRENT\_TIMESTAMP), c.priceforhour \* i.renttime

FROM inserted i

JOIN Cars c ON c.id = i.carid

END

GO

CREATE OR ALTER TRIGGER OnRentListsUpdate

ON RentLists

INSTEAD OF UPDATE

AS

BEGIN

UPDATE rl SET

rl.carid = i.carid,

rl.clientid = i.clientid,

rl.renttime = i.renttime,

rl.endofrent = DATEADD(HOUR, i.renttime, CURRENT\_TIMESTAMP),

rl.totalprice = c.priceforhour \* i.renttime

FROM RentLists rl

JOIN INSERTED i ON i.id = rl.id

JOIN Cars c ON c.id = i.carid

END

GO

CREATE OR ALTER TRIGGER OnRentListDelete

ON RentLists

INSTEAD OF DELETE

AS

INSERT INTO ArchiveRentLists ([name], email, phonenumber, passportnumber, mark, model, enginecapacity, driveunit, transmissiontype, fueltype,

mileage, creationYear, color, priceforhour, renttime, totalprice)

SELECT Clients.[name], Clients.email, Clients.phonenumber, Clients.passportnumber, CarCharacteristics.mark, CarCharacteristics.model,

CarCharacteristics.enginecapacity, CarCharacteristics.driveunit,

CarCharacteristics.transmissiontype, CarCharacteristics.fueltype, Cars.mileage, Cars.creationYear, Cars.color, Cars.priceforhour, renttime, totalprice

FROM RentLists

JOIN Clients ON Clients.id = RentLists.clientid

JOIN Cars ON Cars.id = RentLists.carid

JOIN CarCharacteristics ON CarCharacteristics.id = Cars.characteristicsid

DELETE FROM RentLists WHERE id IN (SELECT id FROM DELETED);

GO

CREATE OR ALTER TRIGGER OnCarDelete

ON Cars

INSTEAD OF DELETE

AS

INSERT INTO ArchiveCars (mark, model, enginecapacity, driveunit, transmissiontype, fueltype, mileage, creationYear, color, priceforhour)

SELECT CarCharacteristics.mark, CarCharacteristics.model, CarCharacteristics.enginecapacity, CarCharacteristics.driveunit,

CarCharacteristics.transmissiontype, CarCharacteristics.fueltype, mileage, creationYear, color, priceforhour

FROM Cars

JOIN CarCharacteristics ON CarCharacteristics.id = Cars.characteristicsid

DELETE FROM RentLists WHERE carid IN (SELECT id FROM DELETED);

DELETE FROM Cars WHERE id IN (SELECT id FROM DELETED);

CREATE OR ALTER VIEW RentListView(mark, model, identfier, name, passportnumber,

renttime, dateofrent, totalprice) AS SELECT cc.mark, cc.model, c.identfier, cl.name, cl.passportnumber, rl.renttime, rl.dateofrent, rl.totalprice

FROM RentLists rl

JOIN Cars c ON rl.carid = c.id

JOIN CarCharacteristics cc ON c.characteristicsid = cc.id

JOIN Clients cl ON rl.clientid = cl.id;

CREATE OR ALTER PROCEDURE RentListReport(@startDate datetime, @endDate datetime)

AS

BEGIN

SET NOCOUNT ON;

SELECT cc.mark, cc.model, c.identfier, cl.name, cl.passportnumber, rl.renttime, rl.dateofrent, rl.endofrent, rl.totalprice

FROM RentLists rl

JOIN Cars c ON rl.carid = c.id

JOIN CarCharacteristics cc ON c.characteristicsid = cc.id

JOIN Clients cl ON rl.clientid = cl.id

WHERE CAST(@startDate AS date) = CAST(rl.dateofrent AS date) AND CAST(@endDate AS date) = CAST(rl.endofrent AS date);

END;

GO

CREATE OR ALTER TRIGGER OnCarCharacteristicsDelete

ON CarCharacteristics

INSTEAD OF DELETE

AS

DELETE FROM Cars WHERE characteristicsid IN (SELECT id FROM DELETED);

DELETE FROM CarCharacteristics WHERE id IN (SELECT id FROM DELETED);

INSERT INTO CarCharacteristics (mark, model, enginecapacity, driveunit, transmissiontype, fueltype)

VALUES

('Toyota', 'Corolla', 1.6, 'Передний привод', 'Автоматическая', 'Бензин'),

('Honda', 'Civic', 1.5, 'Передний привод', 'Механическая ', 'Бензин'),

('Volkswagen', 'Golf', 1.4, 'Передний привод', 'Автоматическая', 'Бензин'),

('Ford', 'Fiesta', 1.0, 'Передний привод', 'Механическая ', 'Бензин'),

('BMW', '3 Series', 2.0, 'Задний привод', 'Автоматическая', 'Дизель'),

('Mercedes-Benz', 'C-Class', 2.0, 'Задний привод', 'Автоматическая', 'Бензин'),

('Audi', 'A4', 2.0, 'Передний привод', 'Автоматическая', 'Дизель'),

('Lexus', 'ES', 2.5, 'Передний привод', 'Автоматическая', 'Бензин'),

('Nissan', 'Altima', 2.5, 'Передний привод', 'Вариаторная', 'Бензин'),

('Hyundai', 'Elantra', 2.0, 'Передний привод', 'Автоматическая', 'Бензин'),

('Kia', 'Optima', 2.0, 'Передний привод', 'Автоматическая ', 'Дизель'),

('Mazda', '6', 2.5, 'Передний привод', 'Автоматическая', 'Бензин'),

('Subaru', 'Impreza', 2.0, 'Полный привод', 'Механическая ', 'Бензин'),

('Jeep', 'Wrangler', 3.6, 'Полный привод', 'Механическая ', 'Бензин'),

('Chevrolet', 'Camaro', 3.6, 'Задний привод', 'Автоматическая', 'Бензин'),

('Dodge', 'Challenger', 3.6, 'Задний привод', 'Автоматическая', 'Бензин'),

('Tesla', 'Model S', 100.0, 'Полный привод', 'Автоматическая', 'Электричество'),

('Porsche', '911', 3.0, 'Задний привод', 'Автоматическая', 'Бензин'),

('Ferrari', '458 Italia', 4.5, 'Задний привод', 'Автоматическая', 'Бензин'),

('Lamborghini', 'Aventador', 6.5, 'Полный привод', 'Автоматическая', 'Бензин');

INSERT INTO Cars (characteristicsid, identfier ,mileage, creationYear, color, priceforhour)

VALUES

(1, '7B1A', 50000, 2018, 'Красный', 10.0),

(2, 'SB8A', 70000, 2017, 'Синий', 8.5),

(3, '4B2V', 60000, 2016, 'Серебрянный', 9.0),

(4, '5BAS', 80000, 2015, 'Белый', 7.5),

(5, 'YZ29', 90000, 2014, 'Черный', 12.0),

(6, '92TR', 40000, 2019, 'Серый', 14.5),

(7, '77GH', 45000, 2020, 'Белый', 13.0),

(8, '438F', 30000, 2021, 'Красный', 16.0),

(9, '09OP', 55000, 2018, 'Серебрянный', 7.0),

(10, '1111', 65000, 2017, 'Черный', 8.0),

(11, 'FFFF', 50000, 2016, 'Белый', 6.5),

(12, 'HT34', 60000, 2015, 'Красный', 9.5),

(13, 'RETU', 75000, 2014, 'Синий', 10.5),

(14, 'RT43', 40000, 2019, 'Серый', 15.0),

(15, '324F', 35000, 2020, 'Белый', 12.5),

(16, 'GFH2', 55000, 2021, 'Черный', 17.0),

(17, '23JH', 80000, 2018, 'Красный', 20.0),

(18, '89FH', 60000, 2017, 'Серебрянный', 18.0),

(19, 'JJU2', 70000, 2016, 'Белый', 16.5),

(20, '2KSM', 90000, 2015, 'Черный', 21.0);

INSERT INTO [Users] ([name], username, email, [password])

VALUES

('Иван Петров', 'ivanp', 'ivan.petrov@example.com', 'пароль123'),

('Анна Иванова', 'annai', 'anna.ivanova@example.com', '12345678'),

('Михаил Сидоров', 'mihails', 'msidorov@example.com', 'пароль0'),

('Елена Козлова', 'elenak', 'ekozlova@example.com', 'дайвойти'),

('Дмитрий Николаев', 'dmitryn', 'dnikolaev@example.com', 'qwerty'),

('Ольга Смирнова', 'olgas', 'osmirnova@example.com', 'пароль1'),

('Алексей Васильев', 'alexeyv', 'avasiliev@example.com', 'добро'),

('Наталья Иванова', 'nataliai', 'natalia.ivanova@example.com', 'пароль2'),

('Игорь Кузнецов', 'igork', 'ikuznetsov@example.com', 'qazwsx'),

('Андрей Попов', 'andreyp', 'apopov@example.com', 'пароль3'),

('Светлана Михайлова', 'svetlanam', 'smikhailova@example.com', '111111'),

('Алена Зайцева', 'alenaz', 'azaitseva@example.com', 'password'),

('Владислав Крылов', 'vladislavk', 'vkrylov@example.com', 'parol123'),

('Кристина Новикова', 'kristinan', 'knovikova@example.com', 'p@ssword'),

('Евгений Соколов', 'evgeniys', 'esokolov@example.com', 'qwerty123'),

('Надежда Кузьмина', 'nadezhda\_k', 'nkuzmina@example.com', 'пароль4'),

('Глеб Волков', 'glebv', 'gvolkov@example.com', 'password123'),

('Алина Петрова', 'alinap', 'apetrova@example.com', 'пароль5'),

('Денис Смирнов', 'deniss', 'dsmirnov@example.com', '123qwe'),

('Оксана Ильина', 'oksana\_i', 'oilina@example.com', 'пароль6');

INSERT INTO Clients ([name], email, phonenumber, passportnumber)

VALUES

('Иван Иванов', 'ivan.ivanov@example.com', '555-1234', 'ABC123'),

('Анна Смирнова', 'anna.smirnova@example.com', '555-5678', 'DEF456'),

('Дмитрий Петров', 'dmitry.petrov@example.com', '555-9876', 'GHI789'),

('Елена Кузнецова', 'elena.kuznetsova@example.com', '555-4321', 'JKL012'),

('Петр Сидоров', 'peter.sidirov@example.com', '555-8765', 'MNO345'),

('Любовь Чернова', 'lyubov.chernova@example.com', '555-2345', 'PQR678'),

('Тимофей Новиков', 'timofey.novikov@example.com', '555-6789', 'STU901'),

('Ольга Морозова', 'olga.morozova@example.com', '555-3456', 'VWX234'),

('Александр Козлов', 'alexander.kozlov@example.com', '555-7890', 'YZA567'),

('Марина Волкова', 'marina.volkova@example.com', '555-3210', 'BCD890'),

('Владимир Андреев', 'vladimir.andreev@example.com', '555-7654', 'EFG123'),

('Светлана Ковалева', 'svetlana.kovaleva@example.com', '555-2109', 'HIJ456'),

('Максим Лебедев', 'maxim.lebedev@example.com', '555-1593', 'KLM789'),

('Надежда Белова', 'nadezhda.belova@example.com', '555-4680', 'NOP012'),

('Илья Макаров', 'ilya.makarov@example.com', '555-8027', 'QRS345'),

('Евгения Алексеева', 'evgeniya.alekseeva@example.com', '555-2468', 'TUV678'),

('Григорий Смирнов', 'grigory.smirnov@example.com', '555-6802', 'WXY901'),

('Екатерина Борисова', 'ekaterina.borisova@example.com', '555-0246', 'ZAB234'),

('Артем Чернышев', 'artem.chernyshev@example.com', '555-4680', 'CDE567'),

('Нина Васильева', 'nina.vasilieva@example.com', '555-8027', 'FGH890');

INSERT INTO RentLists (clientid, carid, renttime)

VALUES

(1, 3, 5),

(2, 2, 3),

(3, 5, 2),

(4, 1, 1),

(5, 6, 4),

(6, 4, 6),

(7, 8, 8),

(8, 9, 3),

(9, 7, 2),

(10, 12, 5),

(11, 10, 3),

(12, 15, 1),

(13, 13, 4),

(14, 18, 6),

(15, 11, 8),

(16, 14, 2),

(17, 20, 3),

(18, 17, 5),

(19, 16, 7),

(20, 19, 2);

namespace CarRent

{

public partial class RentListsForm : Form

{

public RentListsForm()

{

InitializeComponent();

}

private void RentListsForm\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "autodbDataSet.Clients". При необходимости она может быть перемещена или удалена.

this.clientsTableAdapter.Fill(this.autodbDataSet.Clients);

// TODO: данная строка кода позволяет загрузить данные в таблицу "autodbDataSet.Cars". При необходимости она может быть перемещена или удалена.

this.carsTableAdapter.Fill(this.autodbDataSet.Cars);

// TODO: данная строка кода позволяет загрузить данные в таблицу "autodbDataSet.RentLists". При необходимости она может быть перемещена или удалена.

this.rentListsTableAdapter.Fill(this.autodbDataSet.RentLists);

}

private void Первая\_Click(object sender, EventArgs e)

{

this.rentListsBindingSource.MoveFirst();

}

private void Предыдущая\_Click(object sender, EventArgs e)

{

this.rentListsBindingSource.MovePrevious();

}

private void Следущая\_Click(object sender, EventArgs e)

{

this.rentListsBindingSource.MoveNext();

}

private void Последняя\_Click(object sender, EventArgs e)

{

this.rentListsBindingSource.MoveLast();

}

private void Добавить\_Click(object sender, EventArgs e)

{

try

{

this.rentListsBindingSource.AddNew();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка");

}

}

private void Удалить\_Click(object sender, EventArgs e)

{

try

{

this.rentListsBindingSource.RemoveCurrent();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка");

}

}

private void Сохранять\_Click(object sender, EventArgs e)

{

try

{

this.Validate();

this.rentListsBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.autodbDataSet);

}

catch (SqlException ex)

{

MessageBox.Show("Данный автомобиль уже арендован или данный клиент уже арендовал другой автомобиль", "Ошибка");

}

}

private void Sort\_Click(object sender, EventArgs e)

{

Program.form.ShowForm(new SortRentListForm());

}

private void rentListsBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

this.Validate();

this.rentListsBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.autodbDataSet);

}

private void RentListsForm\_Load\_1(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "autodbDataSet.Cars". При необходимости она может быть перемещена или удалена.

this.carsTableAdapter.Fill(this.autodbDataSet.Cars);

// TODO: данная строка кода позволяет загрузить данные в таблицу "autodbDataSet.Clients". При необходимости она может быть перемещена или удалена.

this.clientsTableAdapter.Fill(this.autodbDataSet.Clients);

// TODO: данная строка кода позволяет загрузить данные в таблицу "autodbDataSet.RentLists". При необходимости она может быть перемещена или удалена.

this.rentListsTableAdapter.Fill(this.autodbDataSet.RentLists);

}

}

}

namespace CarRent

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

ShowForm(new CarsForm());

}

public void ShowForm(Form form)

{

MainPanel.Controls.Clear();

form.TopLevel = false;

form.AutoScroll = true;

form.WindowState = FormWindowState.Maximized;

MainPanel.Controls.Add(form);

form.Show();

}

private void button1\_Click(object sender, EventArgs e) // Характеристики

{

ShowForm(new CarCharacteristicsForm());

}

private void button2\_Click(object sender, EventArgs e) // Машины

{

ShowForm(new CarsForm());

}

private void button3\_Click(object sender, EventArgs e) // Клиенты

{

ShowForm(new ClientsForm());

}

private void button4\_Click(object sender, EventArgs e) // Пользователи

{

ShowForm(new UsersForm());

}

private void button5\_Click(object sender, EventArgs e) // Аренды

{

ShowForm(new RentListsForm());

}

private void Отчет\_Click(object sender, EventArgs e)

{

ShowForm(new ReportForm());

}

private void button6\_Click(object sender, EventArgs e)

{

Process.Start(Directory.GetCurrentDirectory() + @"\Spravka.chm");

}

}

}

namespace CarRent

{

public partial class SortRentListForm : Form

{

public SortRentListForm()

{

InitializeComponent();

}

private void SortRentListForm\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "autodbDataSet.Cars". При необходимости она может быть перемещена или удалена.

this.carsTableAdapter.Fill(this.autodbDataSet.Cars);

// TODO: данная строка кода позволяет загрузить данные в таблицу "autodbDataSet.CarCharacteristics". При необходимости она может быть перемещена или удалена.

this.carCharacteristicsTableAdapter.Fill(this.autodbDataSet.CarCharacteristics);

// TODO: данная строка кода позволяет загрузить данные в таблицу "autodbDataSet.RentListView". При необходимости она может быть перемещена или удалена.

this.rentListViewTableAdapter.Fill(this.autodbDataSet.RentListView);

}

private void button5\_Click(object sender, EventArgs e)

{

DataGridViewColumn dataGridViewColumn = new DataGridViewColumn();

switch (listBox1.SelectedIndex)

{

case 0: { dataGridViewColumn = dataGridViewTextBoxColumn1; } break;

case 1: { dataGridViewColumn = dataGridViewTextBoxColumn2; } break;

case 2: { dataGridViewColumn = dataGridViewTextBoxColumn3; } break;

case 3: { dataGridViewColumn = dataGridViewTextBoxColumn4; } break;

case 4: { dataGridViewColumn = dataGridViewTextBoxColumn5; } break;

case 5: { dataGridViewColumn = dataGridViewTextBoxColumn6; } break;

case 6: { dataGridViewColumn = dataGridViewTextBoxColumn7; } break;

case 7: { dataGridViewColumn = dataGridViewTextBoxColumn8; } break;

}

if (radioButton1.Checked)

{

rentListViewDataGridView.Sort(dataGridViewColumn, System.ComponentModel.ListSortDirection.Ascending);

}

if (radioButton2.Checked)

{

rentListViewDataGridView.Sort(dataGridViewColumn, System.ComponentModel.ListSortDirection.Descending);

}

}

private void button6\_Click(object sender, EventArgs e)

{

Program.form.ShowForm(new RentListsForm());

}

private void button1\_Click(object sender, EventArgs e)

{

rentListViewBindingSource.Filter = "[Mark]='" + comboBox1.Text + "'";

}

private void button2\_Click(object sender, EventArgs e)

{

rentListViewBindingSource.Filter = "";

}

private void button3\_Click(object sender, EventArgs e)

{

for (int i = 0; i < rentListViewDataGridView.Columns.Count; i++)

{

for (int j = 0; j < rentListViewDataGridView.Rows.Count; j++)

{

rentListViewDataGridView[i, j].Style.BackColor = Color.White;

rentListViewDataGridView[i, j].Style.ForeColor = Color.Black;

}

}

for (int i = 0; i < rentListViewDataGridView.Columns.Count; i++)

{

for (int j = 0; j < rentListViewDataGridView.Rows.Count; j++)

{

if (rentListViewDataGridView[i, j].Value != null)

if (rentListViewDataGridView[i, j].Value.ToString().ToLower().Contains(textBox1.Text.ToLower()))

{

rentListViewDataGridView[i, j].Style.BackColor = Color.LightSeaGreen;

rentListViewDataGridView[i, j].Style.ForeColor = Color.White;

}

}

}

}

}

}

ПРИЛОЖЕНИЕ Б

**(справочное)**

**Результаты работы приложения**

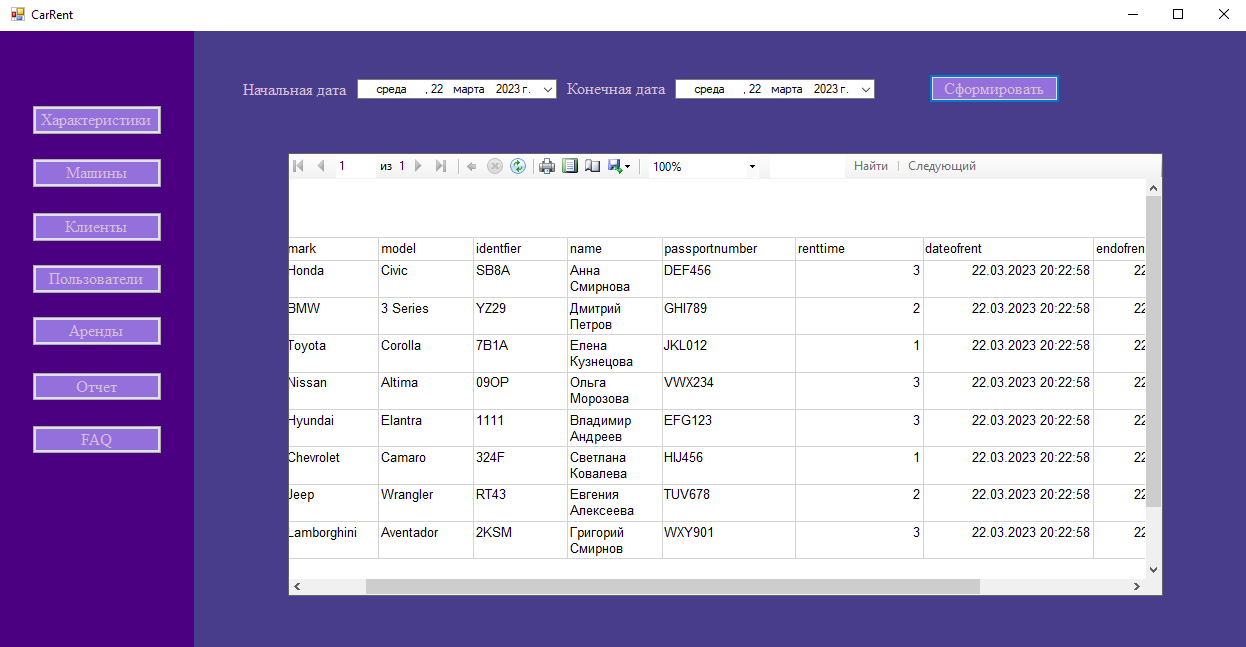


Рисунок Б.1

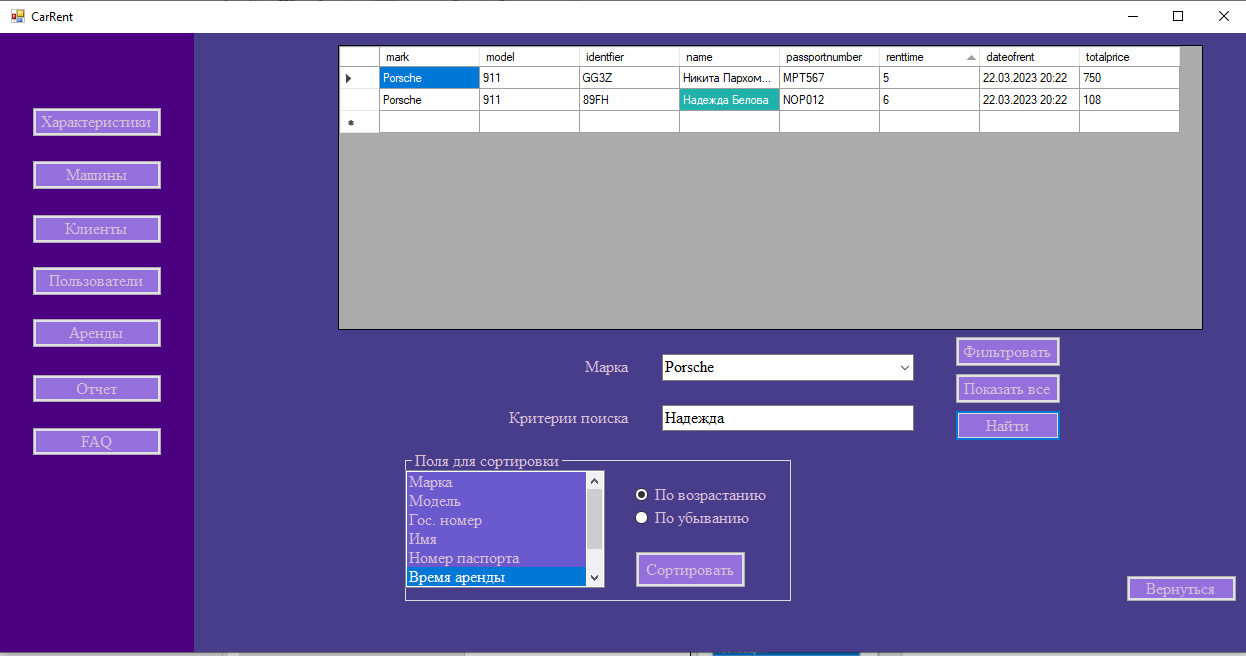


Рисунок Б.2



Рисунок Б.3

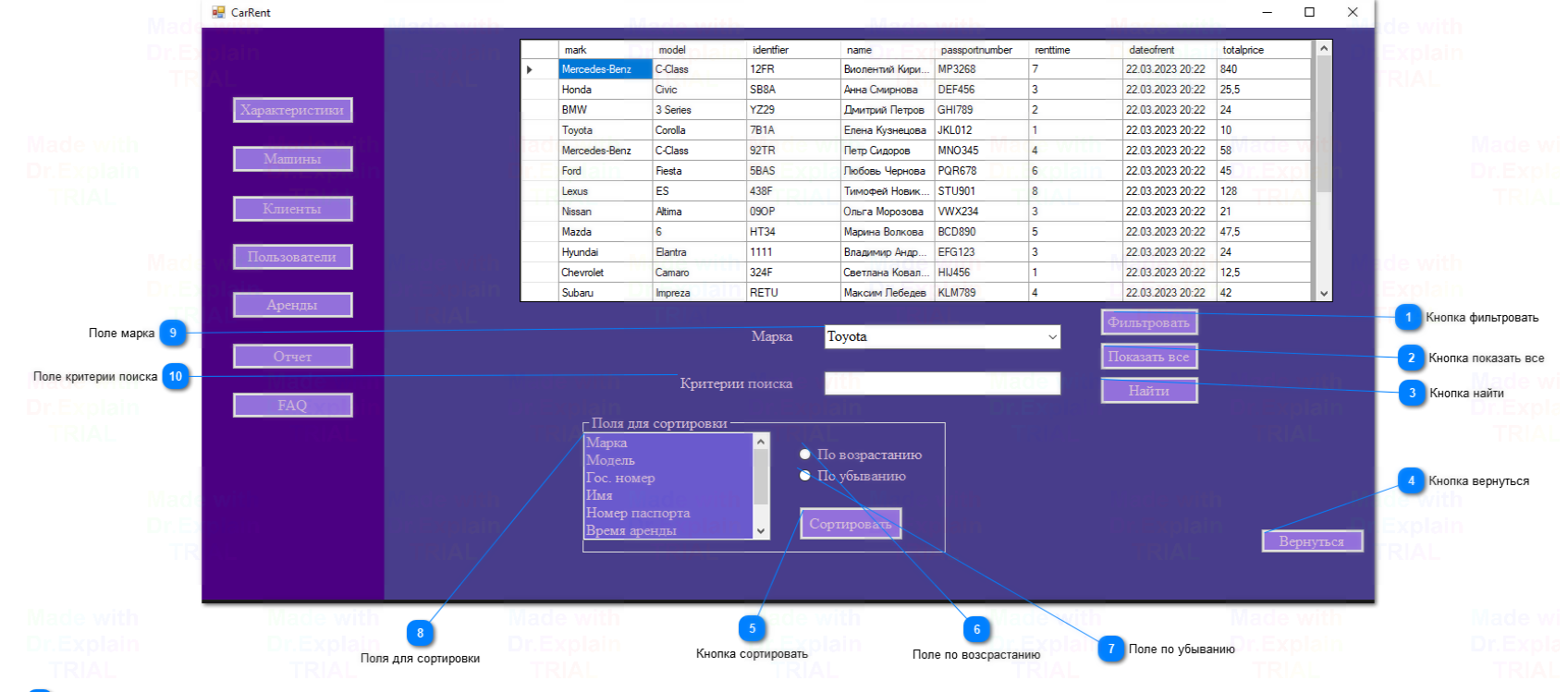


Рисунок Б.4

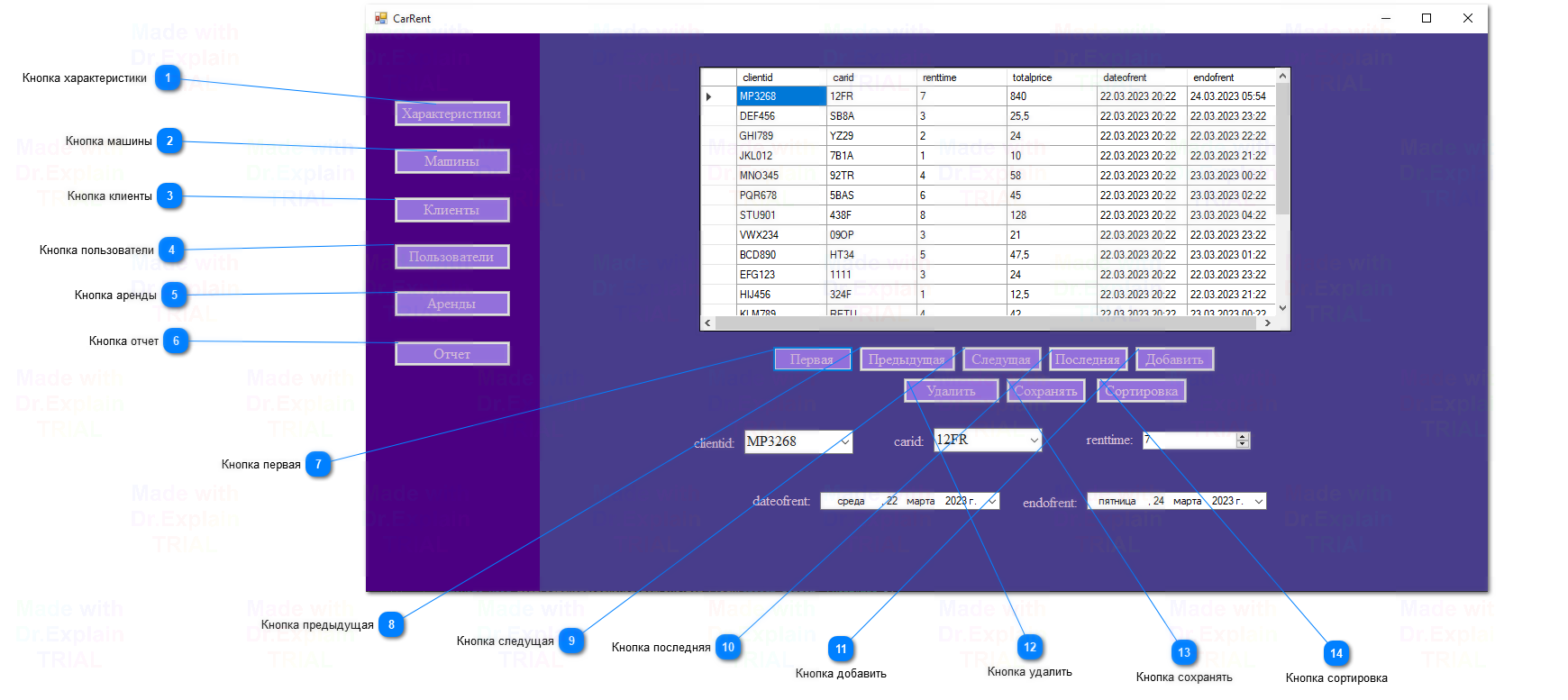


Рисунок Б.5

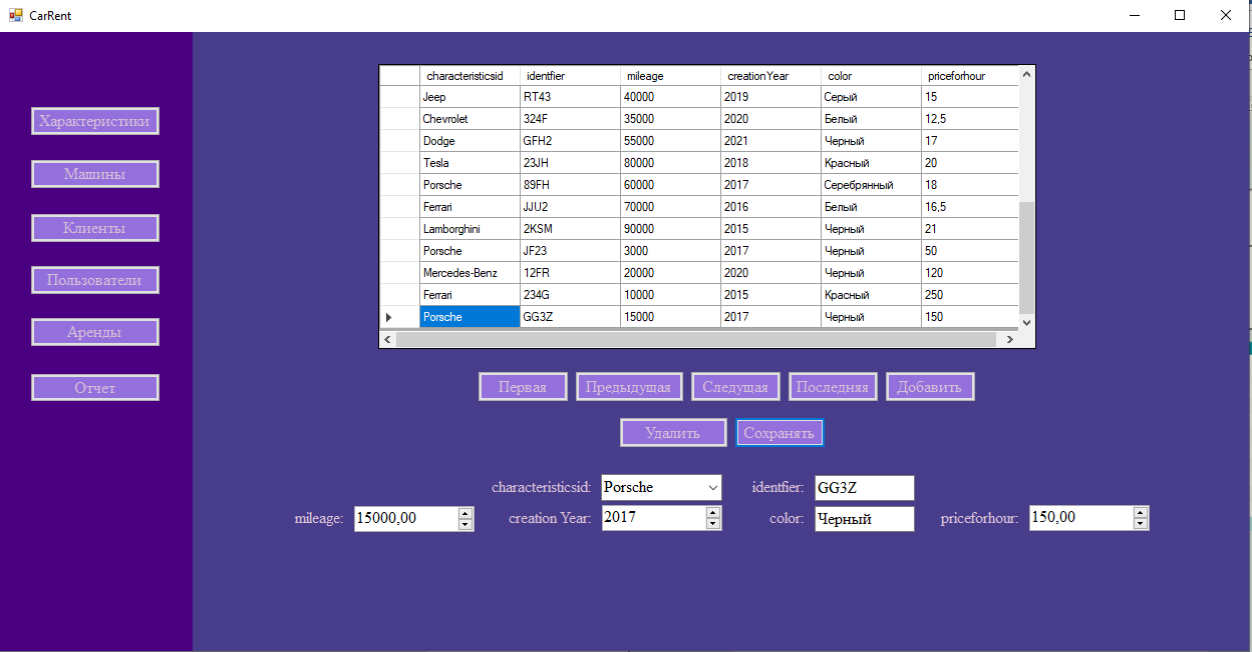


Рисунок Б.6

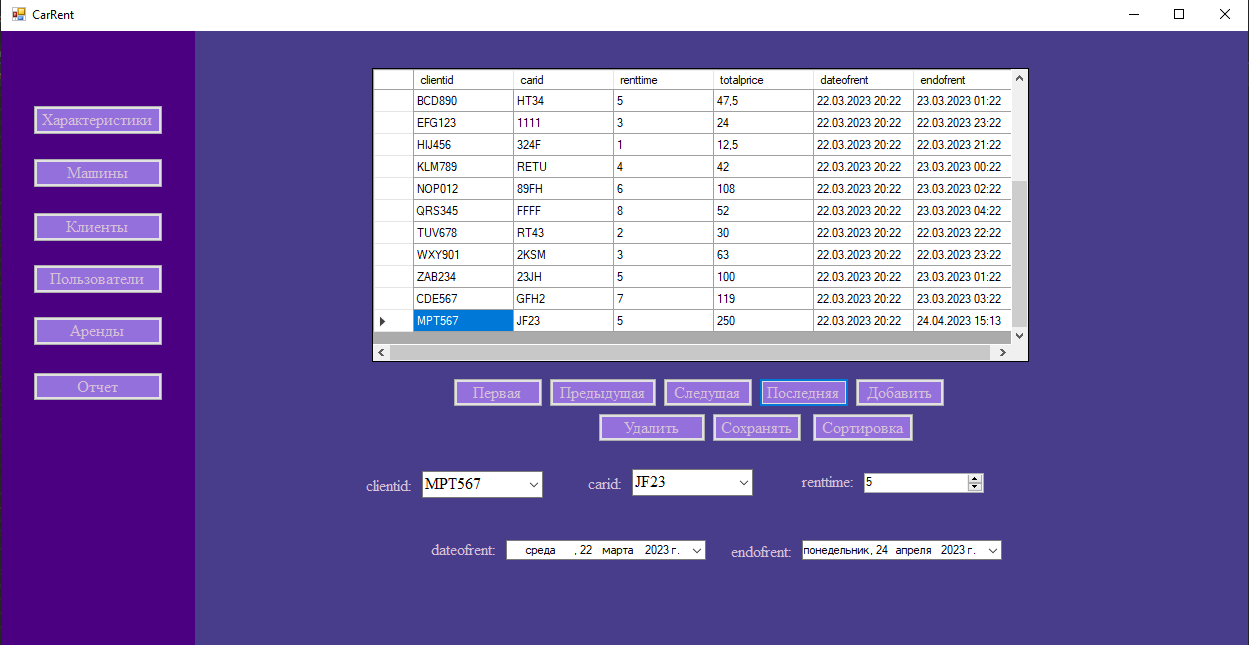


Рисунок Б.7

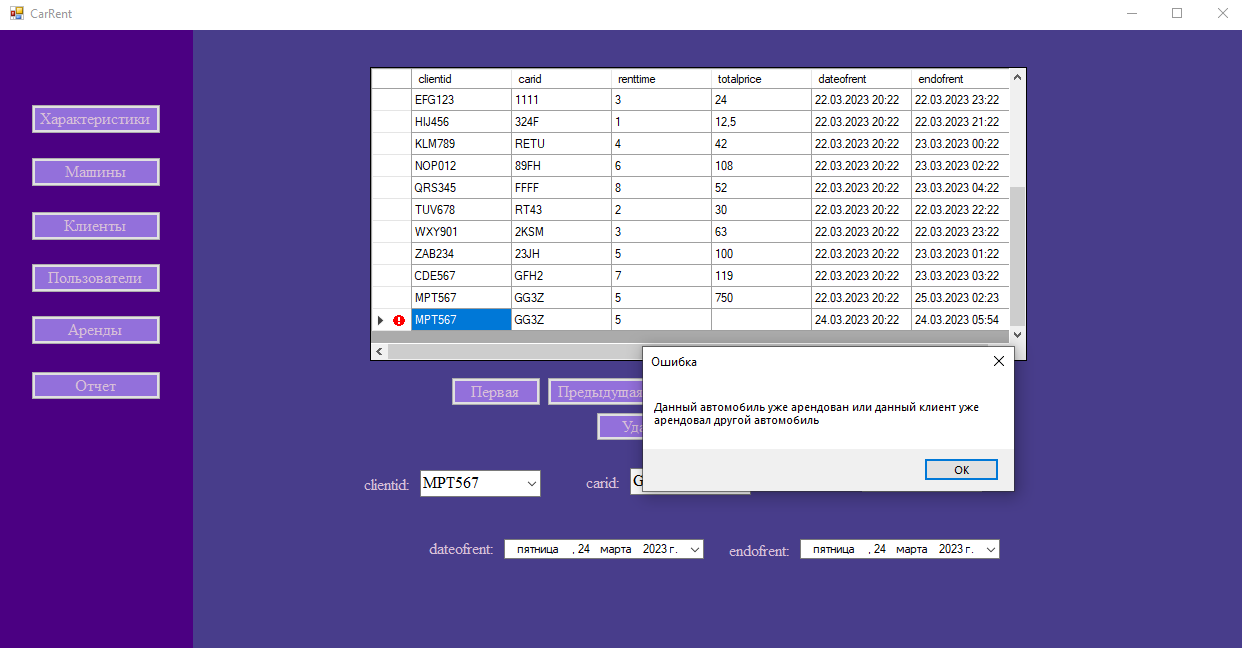


Рисунок Б.8

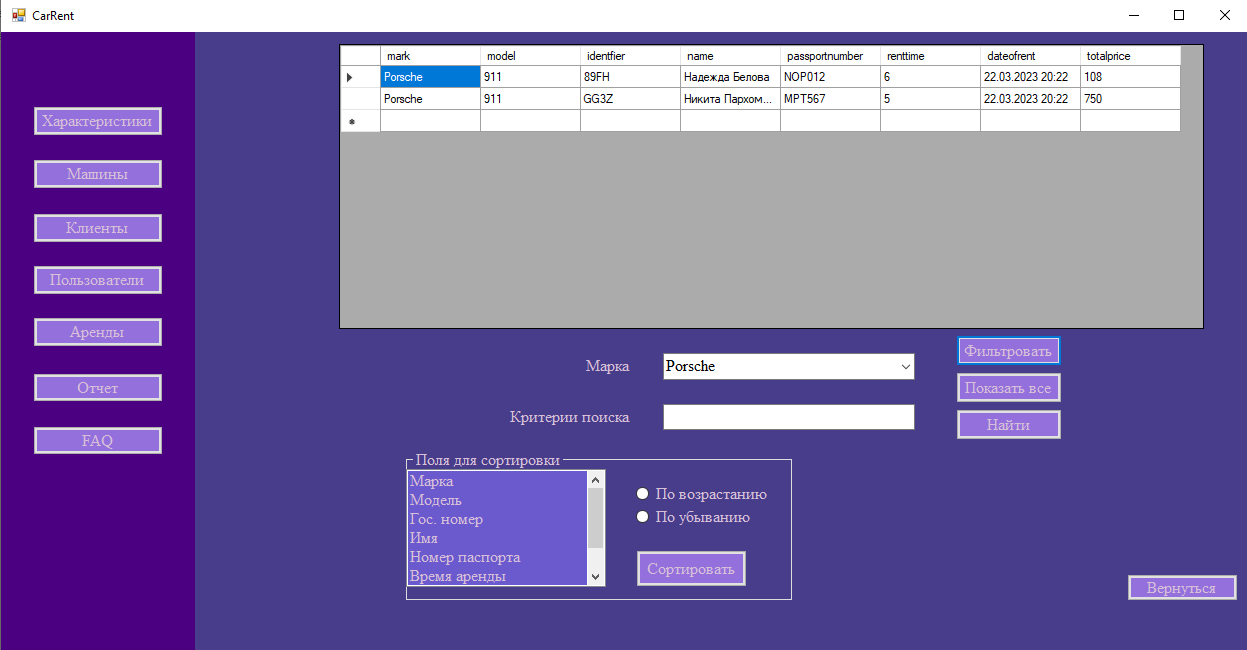


Рисунок Б.9

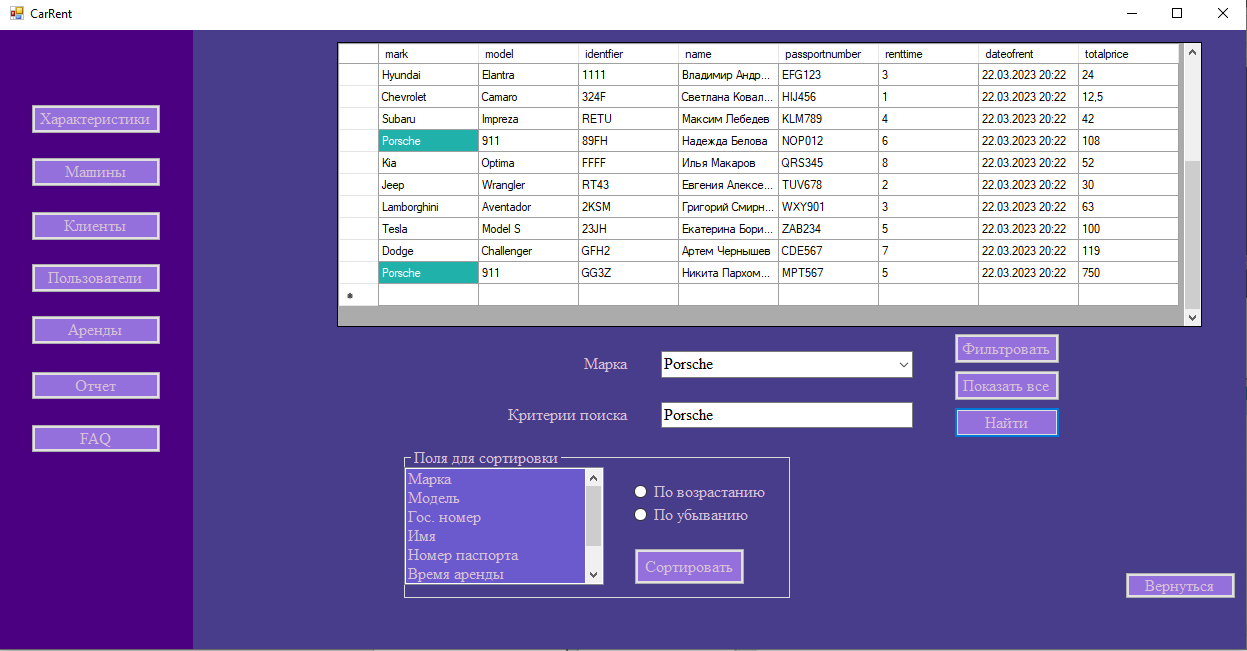


Рисунок Б.10

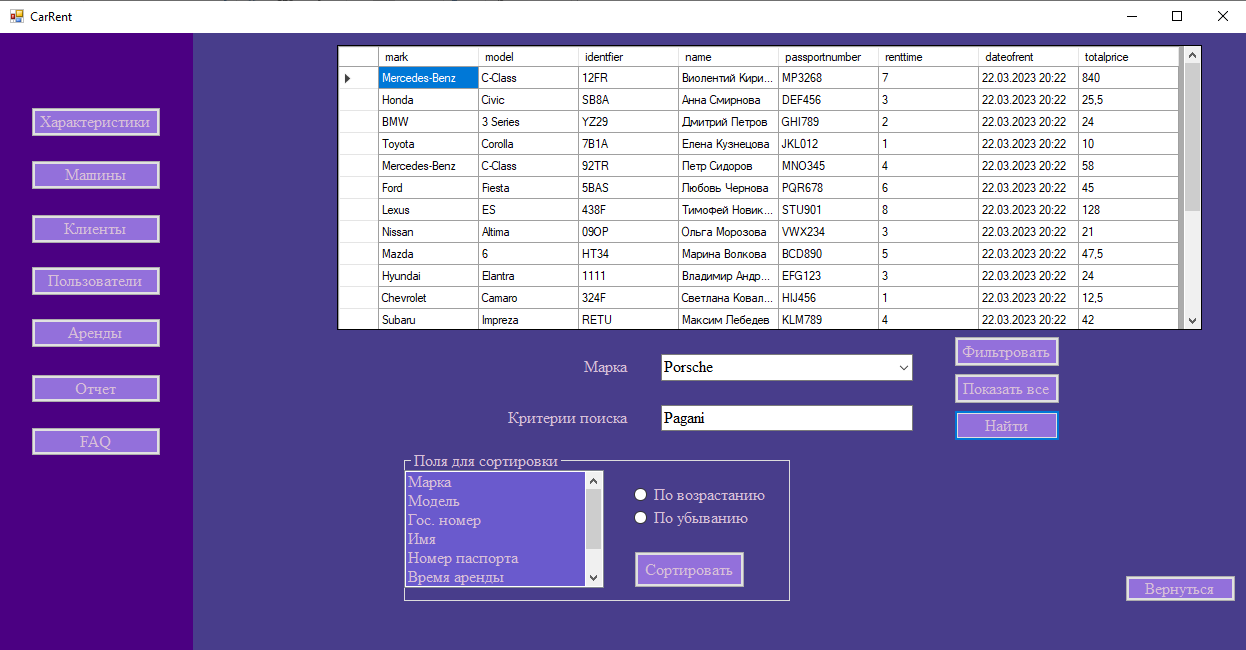


Рисунок Б.11

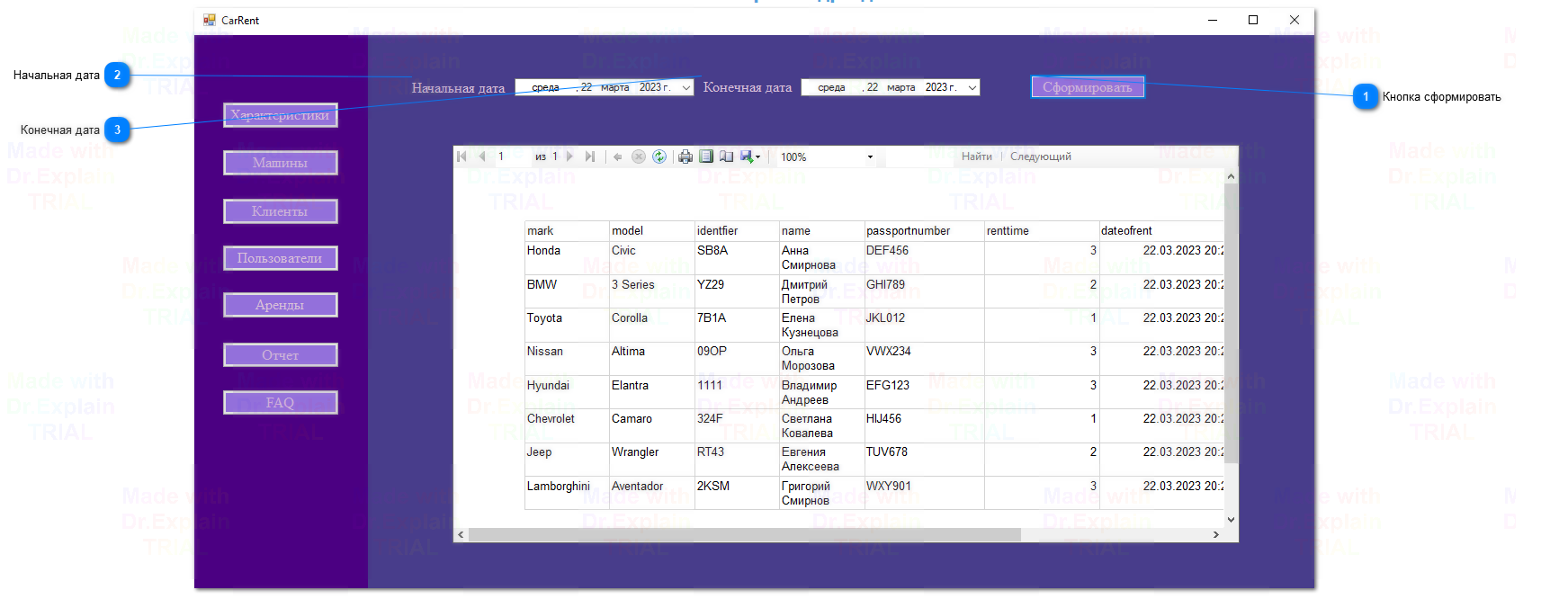


Рисунок Б.12