Учреждение высшего образования

«Университет управления «ТИСБИ»

Факультет информационных технологий

Кафедра информационных технологий

КУРСОВАЯ РАБОТА

по дисциплине «Базы Данных»

на тему: «Проектирование и разработка базы данных “Строительная  
компания ”»

Выполнил: студент гр. ПИ-022

Кадриев Р.З.

Проверил: Кандидат пед. Наук, доцент

Таренко Л.Б.

Казань 2022

Оглавление

[1. Постановка задачи 3](#_Toc120742799)

[1.1 Теоретические сведения по проектированию реляционных баз данных. 5](#_Toc120742800)

[1.2 Анализ предметной области: выявление основных сущностей и их связей 7](#_Toc120742801)

[1.3. Модель сущность-связь для базы данных задачи. 9](#_Toc120742802)

[1.4. Схема реляционной базы данных задачи 10](#_Toc120742803)

[2. Разработка программного приложения 15](#_Toc120742804)

[2.1 Описание структуры и основных методов (процедур и функций) приложения 21](#_Toc120742805)

# Постановка задачи

Цель курсовой работы - спроектировать и разработать базу данных, хранящую договоры строительной компании, и разработать приложение для заключения, изменения и аннулирования договоров. База данных будет хранить информацию о:

* Договорах приобретения объекта
* Объектах
* Сотрудниках
* Бригадах
* Должностях рабочих

Пользователем приложения является сотрудник компании, он заносит информацию о клиентах в базу данных, так же заключает и выполняет необходимые действия с договорами.

Данная система позволит выполнять добавление, удаление, редактирование договоров, объектов, покупателей, рабочий, бригад, должностей.

Курсовая работа будет выполнена при помощи следующих инструментальных средств:

* СУБД: MySQL
* Среда разработки: Visual Studio 2019.
* Языки разработки: C#, SQL

Проектирование и разработка базы данных включает в себя:

* Исследование предметной области (объекта исследования).
* На основе анализа предметной области сформулировать перечень задач.
* Построить инфологическую модель объекта учета
* Построить логическую модель объекта.
* Спроектировать и построить в среде, выбранной СУБД физическую компьютерную структуру данных.
* Разработать и отладить программной обеспечение для работы с БД.
* Заполнить таблицы контрольными данными и провести тестирование.
* Реализовать необходимые функции добавления, удаления, поиска и т.д.

Разработка выполняется с учетом следующих требований:

* Количество таблиц – не менее 4.
* Наличие связанных таблиц.
* Наличие вычисляемых столбцов.
* Размеры таблиц – не менее 10 записей.

Разработка должна выполняться по этапам с обязательным отчетом по каждому этапу

* Этап 1: Постановка задач
* Этап 2: Проектирование и разработка структуры базы данных информационной системы
* Этап 3: Программная реализация и отладка приложения
* Этап 4: Полностью оформленная пояснительная записка

# 1.1 Теоретические сведения по проектированию реляционных баз данных.

Автоматизированные базы данных можно понимать как технические системы, состоящие из комбинации баз данных, пользователей, технологий и программных средств.

Чтобы спроектировать базу данных, сначала необходимо проанализировать набор описаний объектов реального мира, относящихся к целевой области, или области применения, и отношений между ними.

Существует несколько уровней понимания данных:

* Инфологический
* Даталогическая модель
* Физическая модель

Инфологической модель понимается как описание целевой области на основе взаимодействия объектов и создается для понимания того, что представляет собой целевая область конкретной базы данных.

Даталогическая модель данных является доработанной инфологической моделью и учитывает среду управления базой данных, используемую для создания базы данных. Модель отображает логические связи между элементами данных.

Физическая модель данных уже содержит таблицы, атрибуты и отношения, которые физически реализованы в базе данных, т.е. имена полей, первичные ключи, определения внешних ключей, типы полей в таблице и т.д.

Реляционные базы данных используют ключи, которые являются уникальными данными для каждой строки таблицы, чтобы СУБД могла различать практически идентичные строки. Например, в строительной компании есть несколько одинаковых объектов, в этом случае при выполнении некоторых операций над базой данных СУБД может различать их на основе уникального ключа.

Для взаимодействия таблиц данных, используются связи, они бывают:

* Многие ко многим
* Один ко многим
* Один к одному

Связь “Многие ко многим” реализуется, когда нескольким объектам таблицы 1, может принадлежать несколько объектов из таблицы 2.

Связь “Один ко многим” используется, если объекту из таблицы 1, могут принадлежать несколько объектов из таблицы 2.

Связь “Один к одному” самая редко использующаяся связь, так как нужна, только когда объекту из таблицы 1, может принадлежать только один объект из таблицы 2 и одному объекту из таблицы 2 принадлежит объект таблицы 1.

# Анализ предметной области: выявление основных сущностей и их связей

Предметная область: Строительная компания

Для работы строительной компании необходимы строительные объекты, которые будут проданы, бригада и рабочие в ней, которые будут строить объект, клиенты, которые будут покупать недвижимость, а также стоимость строительного объекта и дата совершения купли-продажи. Когда основные моменты работы фирмы выявлены, следует провести подробный анализ всех сущностей, которые были упомянуты выше.

Договор купли-продажи включает в себя предмет сделки, которым выступает Строительный Объект, построенный данной компанией, сторону покупающую предмет договора, именуемая далее как Покупатель, а также Дату совершения договора и Сумму сделки.

Строительный объект имеет Название объекта, Адрес, по которому находится недвижимость, Даты начала и конца строительства, а также Бригаду, которая производила строительство данного объекта.

Бригада включается в себя, сотрудников и бригадира.

Все сотрудники имеют ФИО, номер телефона, бригаду, в которую они входят, и должность, которую они занимают.

Должность включается в себя название должности и оклад.

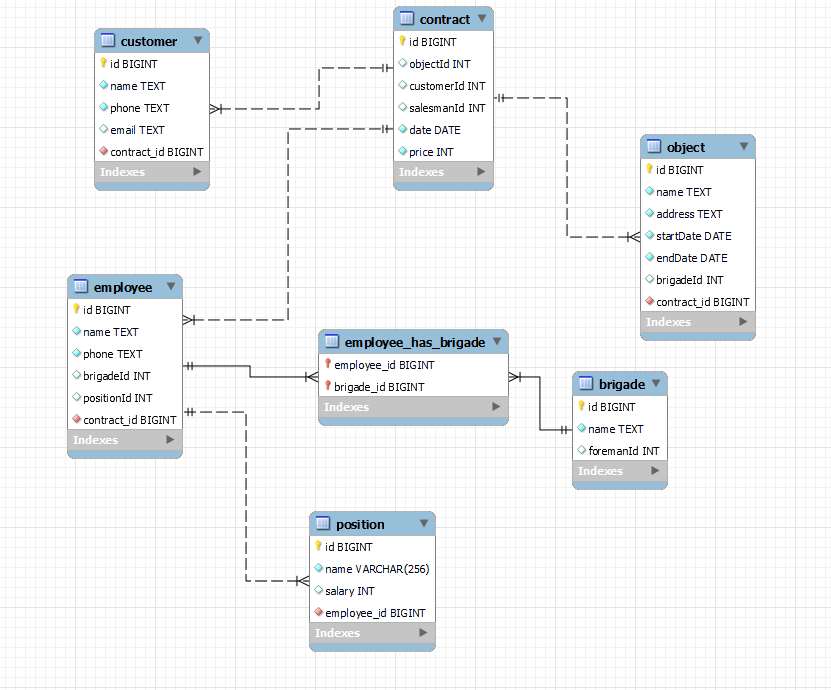
Покупатель имеет ФИО, номер телефона, адрес электронной почты.

# 1.3. Модель сущность-связь для базы данных задачи.

Реляционные базы данных используют отношения, поэтому для того, чтобы правильно реализовать приложение и базу данных, необходима модель отношений сущность-сущность между таблицами, определенными в ходе анализа целевой области.

В результате анализа выявленных сущностей, я выделил связи:

* Объект – Договор (Один-ко-многим)
* Покупатель – Договор (Один-ко-многим)
* Продавец – Договор (Один-ко-многим)
* Рабочий – Должность (Один-ко-многим)
* Рабочий – Бригада (Многие-ко-многим)



(Рис. 1 ER-диаграмма базы данных)

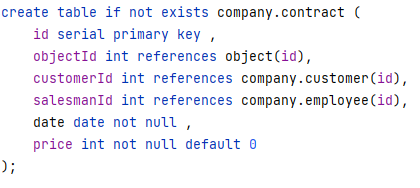
# 1.4. Схема реляционной базы данных задачи

Используя внешние ключи модель сущность-связь преобразуется в реляционную модель базы данных. В данной БД используются связи Один ко многим (1: N) и Многие ко многим (N:N).

Сущность “Contract”

Атрибуты:

* id
* objectId
* customerId
* salesmanId
* date
* price

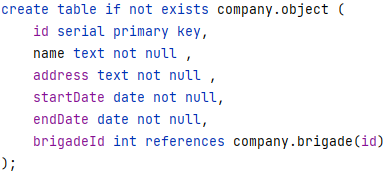


(Рис. 2 Сущность “Договор”)

Сущность “Object”

Атрибуты:

* id
* name
* address
* startDate
* endDate
* brigadeId

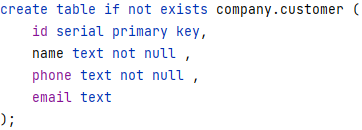


(Рис 3. Сущность “Объект”)

Сущность “Customer”

Атрибуты:

* id
* name
* phone
* email

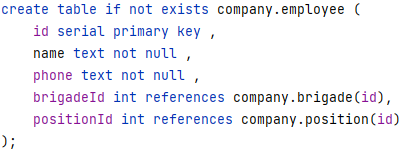


(Рис. 4 Сущность “Покупатель ”)

Сущность “Employee”

Атрибуты:

* id
* name
* phone
* brigadeId
* positionId

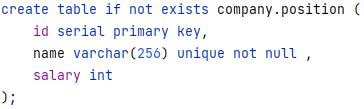


(Рис. 5 Сущность “Работники ”)

Сущность “Position”

Атрибуты:

* id
* name
* salary

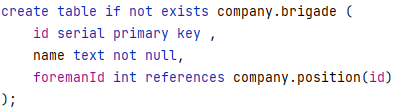


(Рис. 6 Сущность “Должность ”)

Сущность “Brigade”

Атрибуты:

* id
* name
* foremanId



(Рис. 7 Сущность “Бригада ”)

# Разработка программного приложения

С точки зрения программной реализации приложение имеет 4 основных формы: Главная форма: Договоры, форма для заключения новых договоров, форма для управления клиентами (Добавление новых клиентов и редактирование информации о клиентах) и форма для изменения информации в договорах.

На формах есть такие элементы, как:

* DataGridView – Для вывода информации из базы данных в таблицу.
* Textbox- Для вывода информации по отдельному выбранному договору из таблицы, так же для осуществления поиска договоров.
* Button- Для осуществления переходов на другие формы, а так же удаления записей и сохранения базы данных.
* Label- Используется для подписей необходимых объектов на форме.
* Picturebox- Элементы дизайна, некоторые также работают как кнопки.
* Groupbox- Объединяет элементы в группы по их функциям.
* Combobox- Позволяет делать выбор класса автомобиля.
* Maskedtextbox- Реализован для правильного заполнения дат.

Форма “Договор”

* Datagridview1 – Выводит в таблицу информацию из базы данных.
* Textbox\_search – Поиск в таблице записей.
* Textbox\_ID и остальные – Выводят информацию по выбранному договору.
* PictureBox2 – Обновляет таблицу.
* Picturebox1 – Очищает textbox`ы.
* Button\_add – Осуществляет переход в форму заключения договора.
* Button\_delete – Осуществляет удаление выбранной записи.
* Button\_change – Выполняет переход в форму изменения договора.
* Button\_save – Сохраняет информацию в базу данных.
* Button\_ClientControl – Выполняет переход на форму управления клиентами.

Форма “Управление клиентами”

* DataGridView2 – Вывод информации о клиентах из бд в таблицу.
* Textbox`ы – заполняются выбором из таблицы или самостоятельно.
* Button\_add – Добавляет нового клиента в бд.
* Button\_change – Изменяет информацию о клиенте.

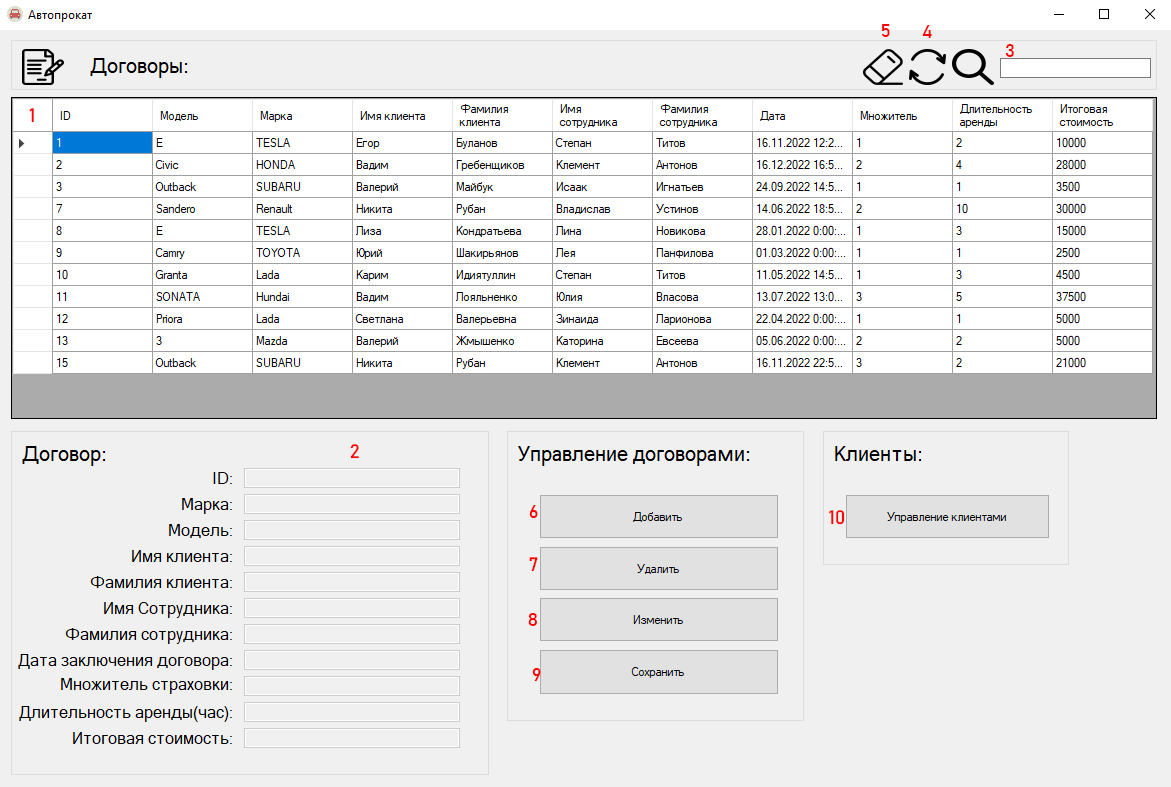
Форма “Заключение договора”

* DataGridView1 – Выводит в таблицу информацию об автомобилях из бд.
* DataGridView2 – Таблица клиентов.
* DataGridView3 – Таблица сотрудников.
* Combobox – Выбор класса автомобиля.
* Textbox – Заполняется самостоятельно, кроме textbox\_TotalCost, который вычисляется по кнопке.
* Button\_mult – Выполняет подсчёт итоговой стоимости аренды.
* Button\_add2 – Выполняет добавление нового договора в бд.

Форма “Изменение договора”

* DataGridView1-4 – Вывод информации по договорам, автомобилям, клиентам и сотудникам.
* Combobox – Выбор класса автомобиля.
* Textbox – Заполняется самостоятельно, кроме textbox\_TotalCost, который вычисляется по кнопке.
* Button\_mult – Выполняет подсчёт итоговой стоимости аренды.
* Button\_change – Выполняет изменение договора в бд.

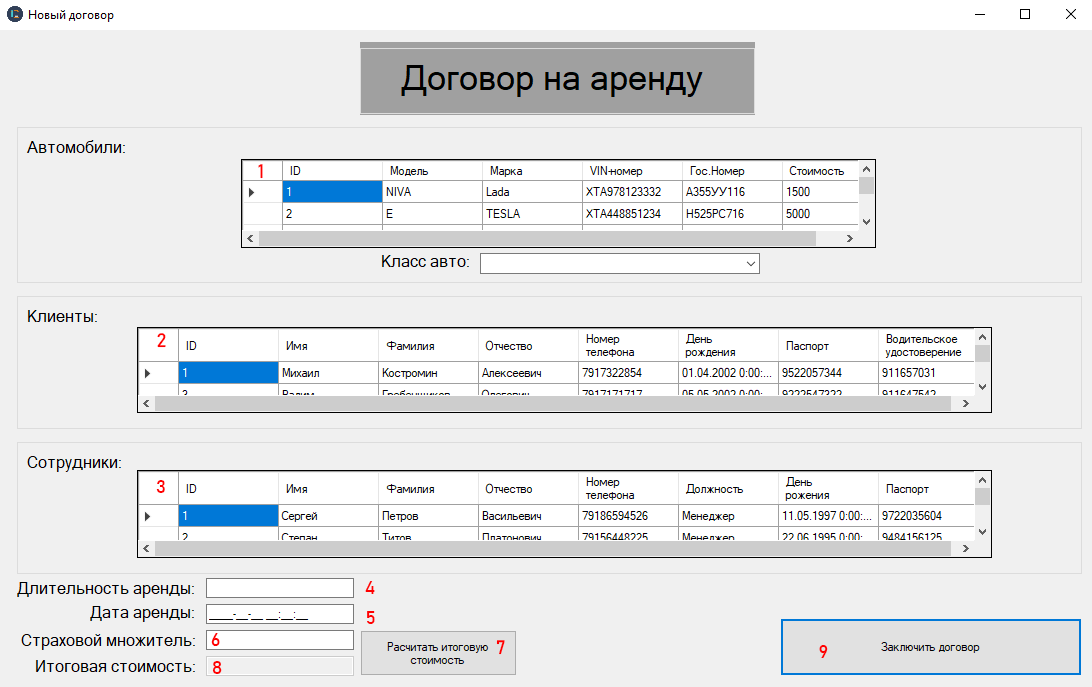
С точки зрения использования приложения можно привести инструкцию по использованию приложения в каждой форме.



(Рис.8 Главная форма)

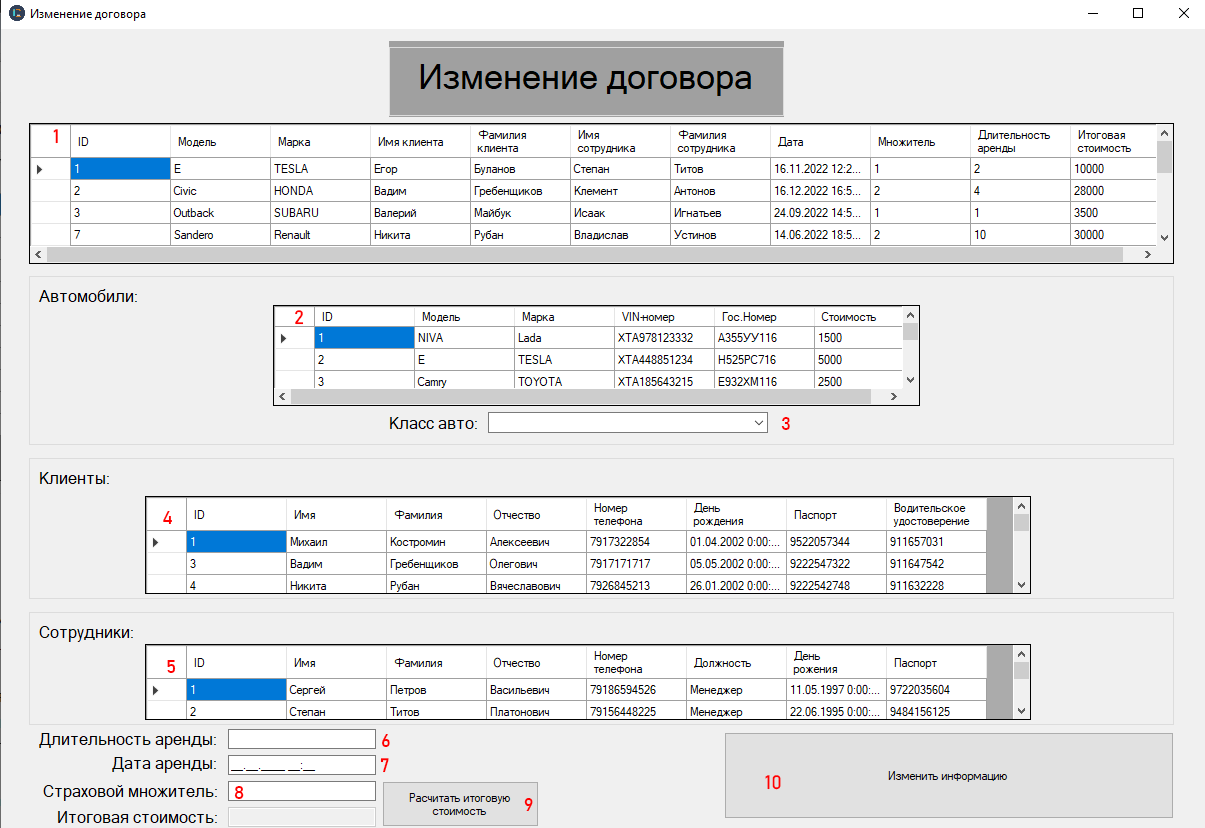
Форма предназначена для просмотра договоров, а также перехода в другие формы.

“1” в данной области пользователь приложения (сотрудник) может просматривать все заключенные договора, а также выбирать их, при выборе договора вся информация выводится в “2” для поиска договоров можно использовать “3, картинка “4” отвечает за обновление таблицы договоров, нажав “5” можно стереть информацию из “2”, кнопка “6” отвечает за переход на форму заключения нового договора, кнопка “7” позволяет удалить выбранную запись договора, “8” выполняет переход на форму изменения договора, после удаления договора таблицу необходимо сохранить и перенести информацию в бд за это отвечает кнопка “9” и кнопка “10” переводит пользователя на форму управления клиентами.



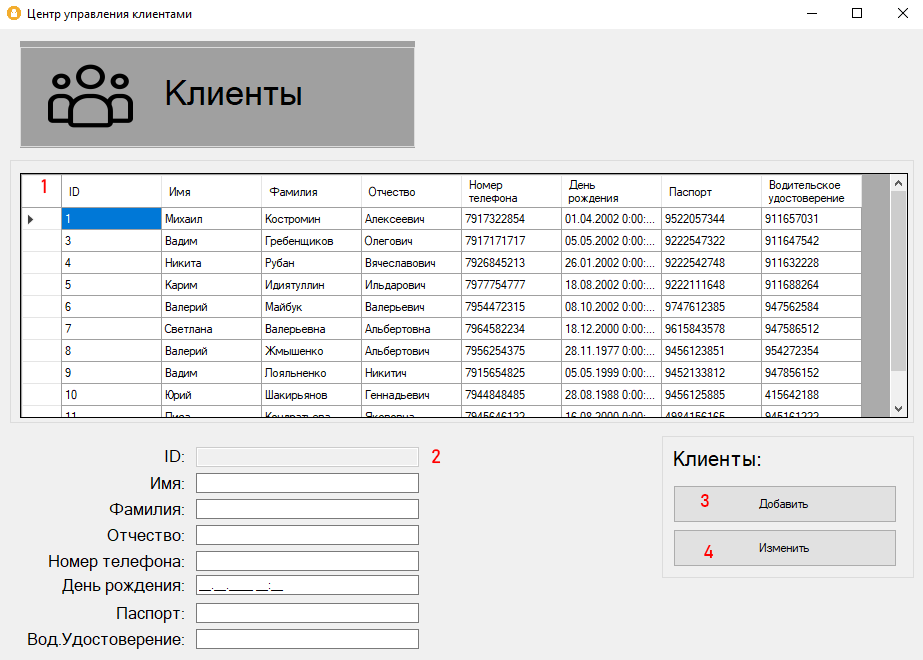
(Рис. 9 Форма заключения нового договора)

На данной форме сотрудник может выбрать заключать новые договоры, для начала необходимо выбрать какой автомобиль арендуют в “1”, далее выбор клиента в “2” и выбор сотрудника, оформляющего договор в “3”, следует заполнить поля “4,5,6” далее можно нажать “7” для расчета стоимости аренды, это необязательный шаг, так как при нажатии “9” заключается новый договор, который автоматически заносится в таблицу, расчет стоимости проведётся без участия сотрудника.



(Рис. 10 Форма изменения договора)

На данной форме сотрудник выбирает договор, который необходимо изменить в “1” далее выбирает клиента, сотрудника и автомобиль в “2,4,5” для удобства в “3” можно выбрать класс автомобиля, после этого если необходимо заполняется информация в “6,7,8” и для обновления информации в договоре необходимо нажать “10”, кнопка “9” как и в прошлой форме отвечает за подсчёт итоговой стоимости.



(Рис. 11 Форма управления клиентами)

На данной форме сотрудник может выбрать в “1” клиента, о котором необходимо обновить информацию, далее изменить информацию, которая занесётся автоматически в поля “2” и для изменения нажать кнопку “4”, для добавления просто необходимо заполнить поля “2” и нажать “3” кнопку.

# 2.1 Описание структуры и основных методов (процедур и функций) приложения

Главная форма (Form1) имеет методы:

1. Метод заполнения названий столбцов в datagridview (createcolumns)
2. Метод построчного занесения информации из бд в таблицу (ReadSingleRow)
3. Метод обновления таблицы (RefreshDataGrid)
4. Метод для вывода информации выбранной записи в текстбоксы (dataGridView1\_CellClick)
5. Методы для открытия форм по кнопкам (button\_add\_Click и т.п.)
6. Метод поиска в таблице при вводе информации в текстбокс поиска (search)
7. Метод удаления договора из таблицы и бд (deleteRow)
8. Метод обработки удаления (Update)
9. Метод очистки текстбоксов по нажатию на картинку (pictureBox1\_Click)

Форма добавления нового договора (AddForm) имеет методы:

1. Имеет методы заполнения названий столбцов в dataGridView (createcolumnsCar, createColumnsClient, createColumnsWorkers)
2. Методы построчного занесения информации из бд в таблицу (ReadSingleRowCar, ReadSingleRowClient, ReadSingleRowWorker)
3. Методы обновления таблицы (RefreshDataGridCar, RefreshDataGridClient, RefreshDataGridWorker)
4. Метод загрузки данных из бд в comboBox (comboBoxRefill)
5. Метод добавления нового договора в бд (button\_add\_Click)
6. Метод подсчёта итоговой стоимости аренды (button\_mult\_Click)

Форма изменения информации договора (ChangeForm) имеет методы:

1.Те же методы, что и раньше для работы с таблицами

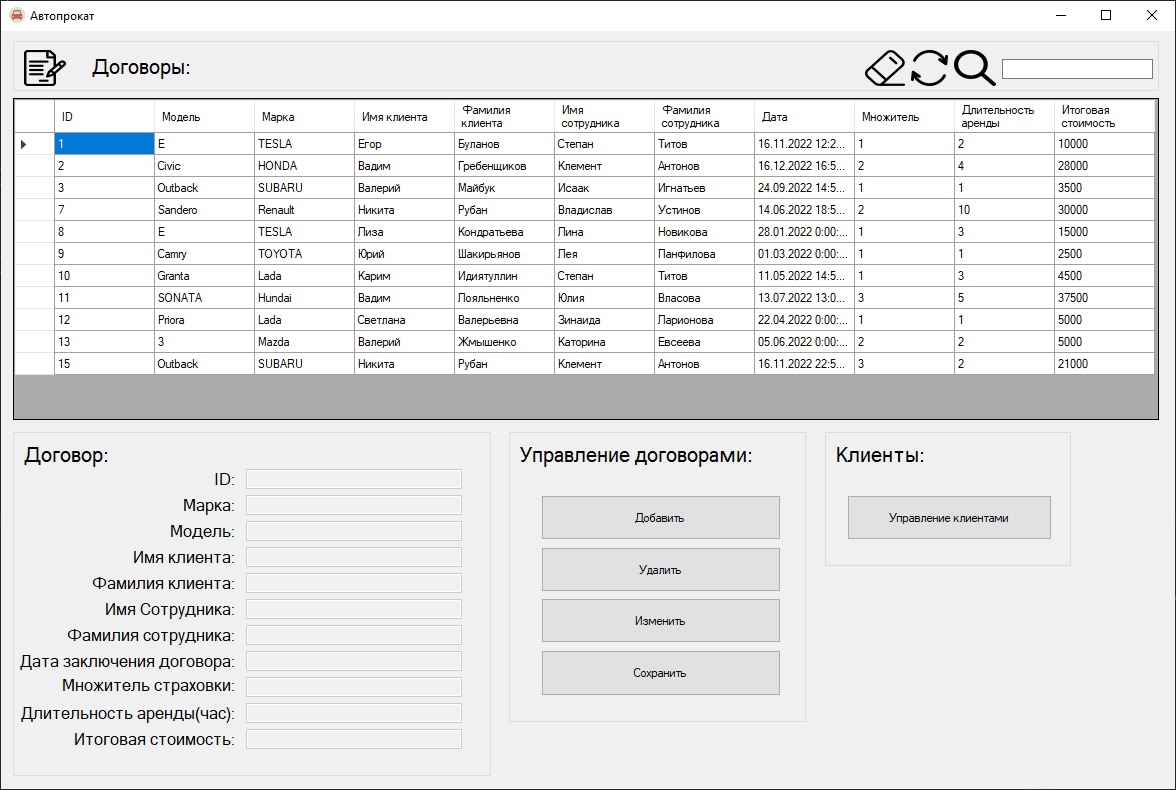
2. Метод изменения информации о договоре в бд (button\_ChangeInfo\_Click)

Форма Управления клиентами(ClientControl) имеет методы:

1. Те же методы, что и раньше для работы с таблицами
2. Метод добавления нового клиента в бд (button\_ClientAdd\_Click)
3. Метод изменения информации о клиенте (change)

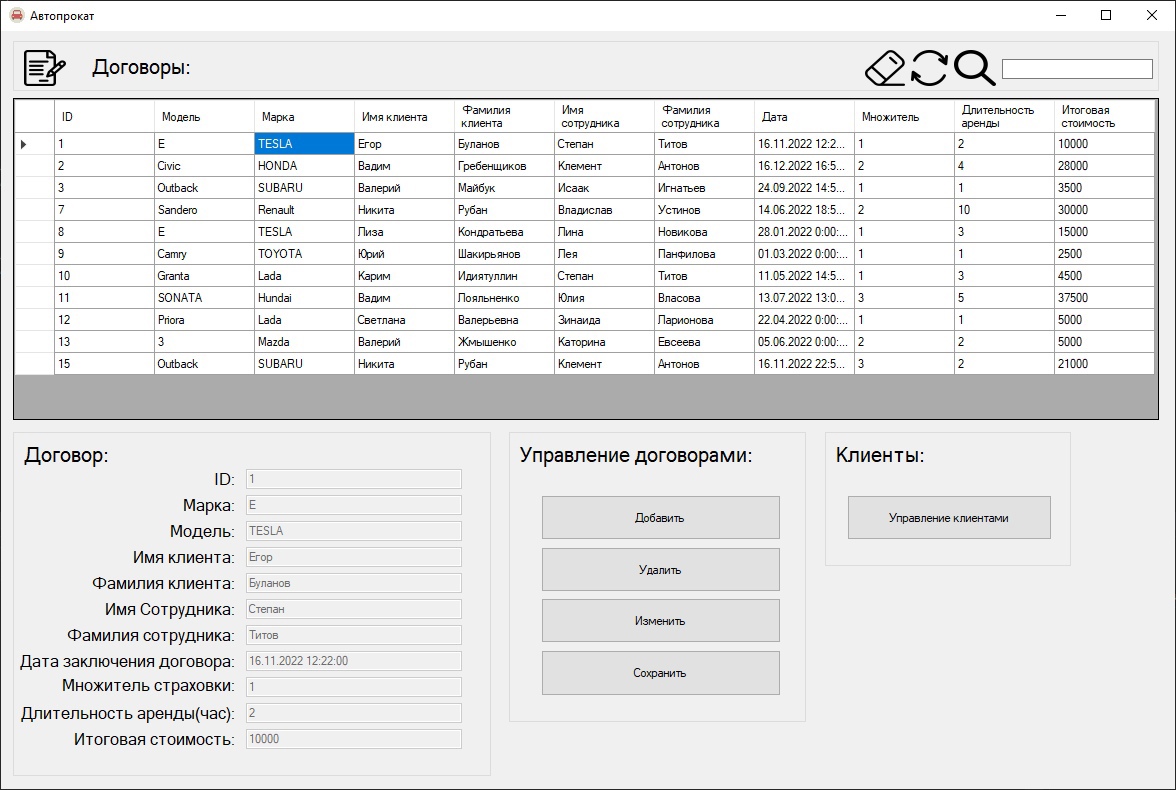
# Результаты тестирования с примерами визуальных форм приложения

Ниже представлена начальная форма, которая загружается при запуске приложения.



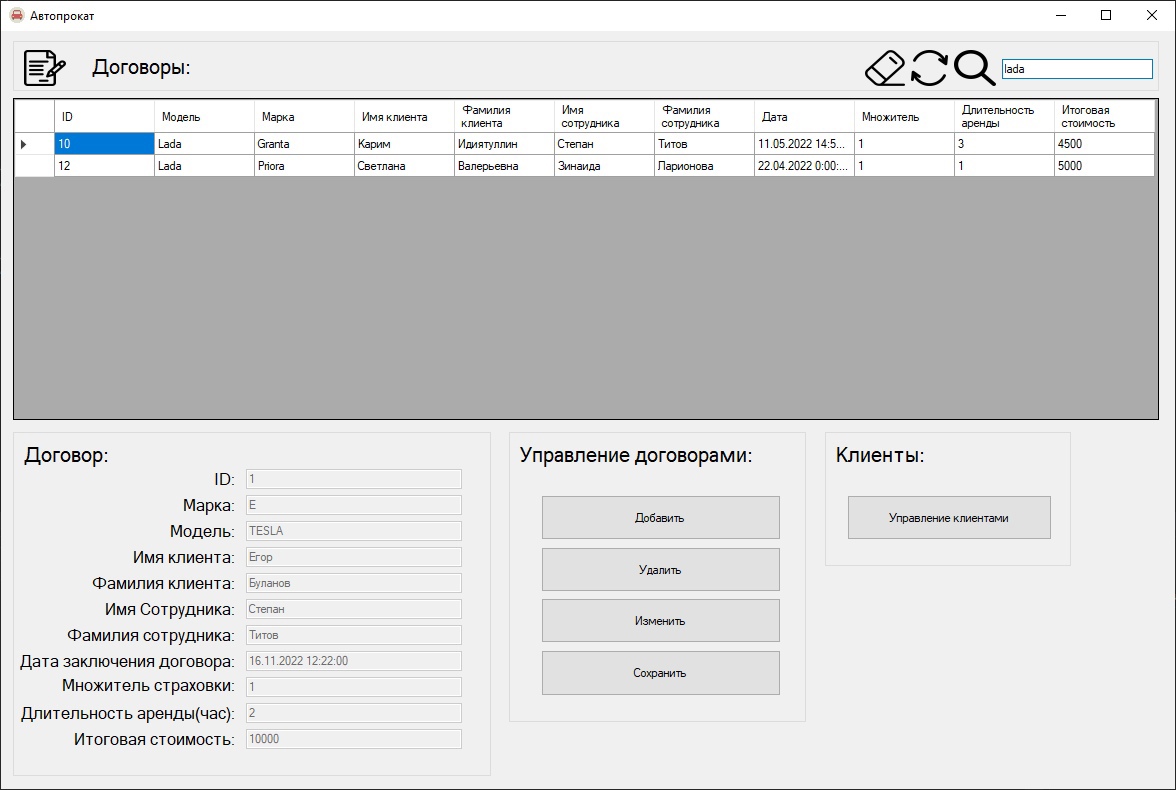
(Рис. 12 Изначальная форма)

Выберем в таблице первую запись, таким образом все данные о выбранном договоре перенесутся в поля textbox.



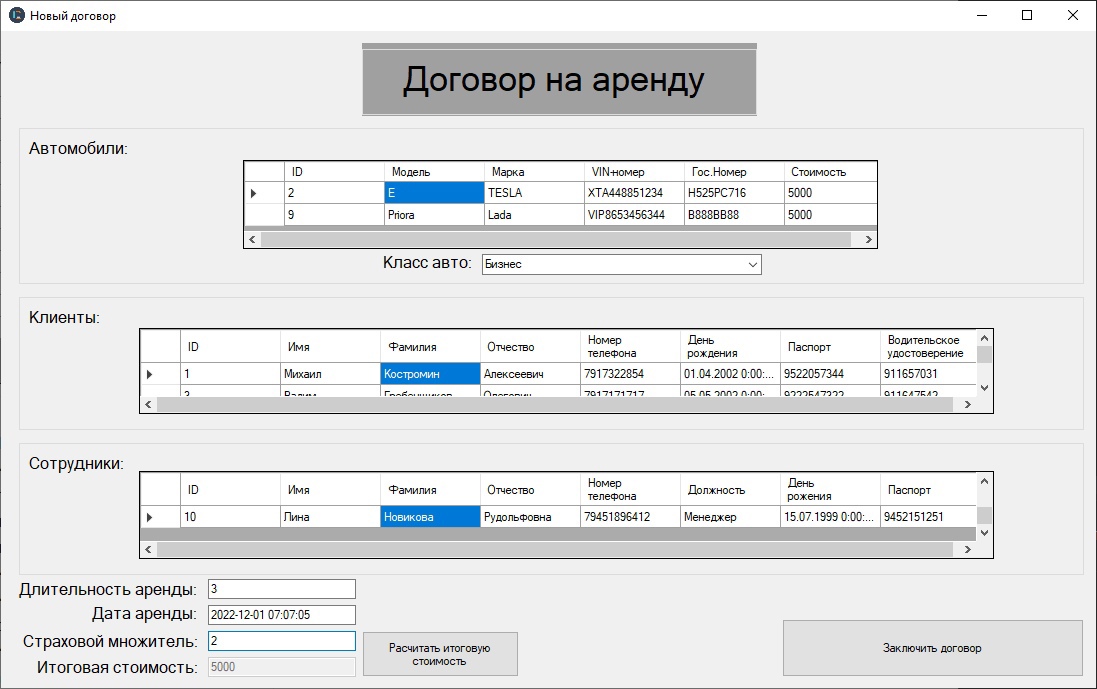
(Рис. 13 Тест выбора и вывода в textbox)

Так же можно выполнить поиск договора, начав писать в поле поиска любую информацию из договора.



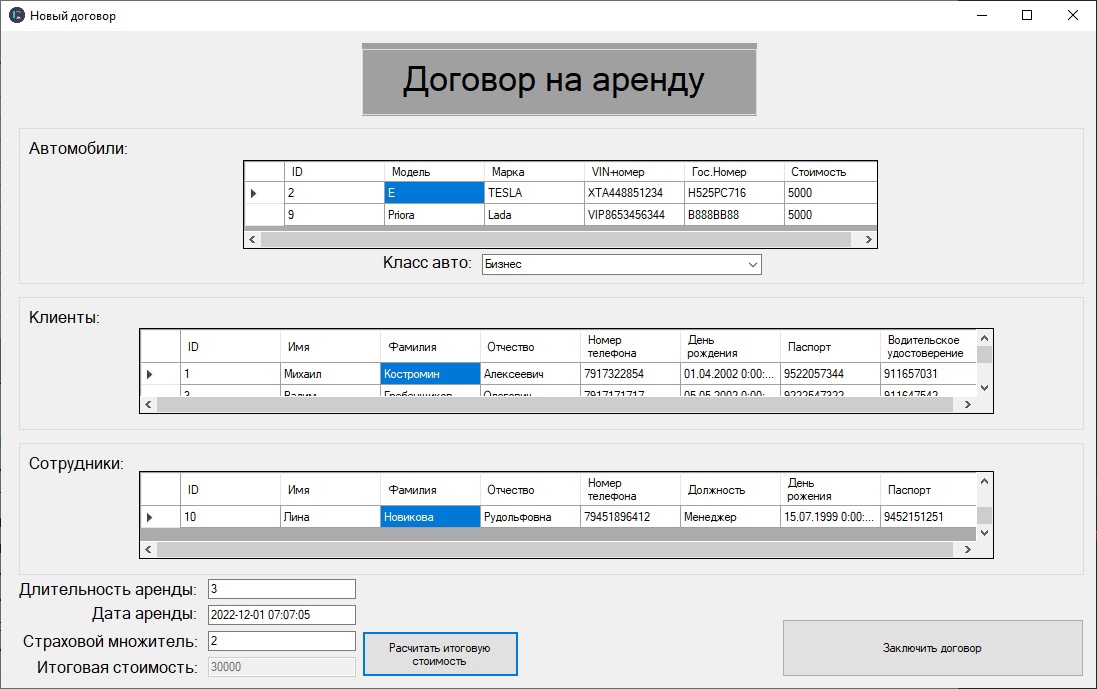
(Рис. 14 Тест поиска)

Далее перейдём в форму добавления договора по кнопке из меню на главной форме. Выглядит форма заключения договора так:



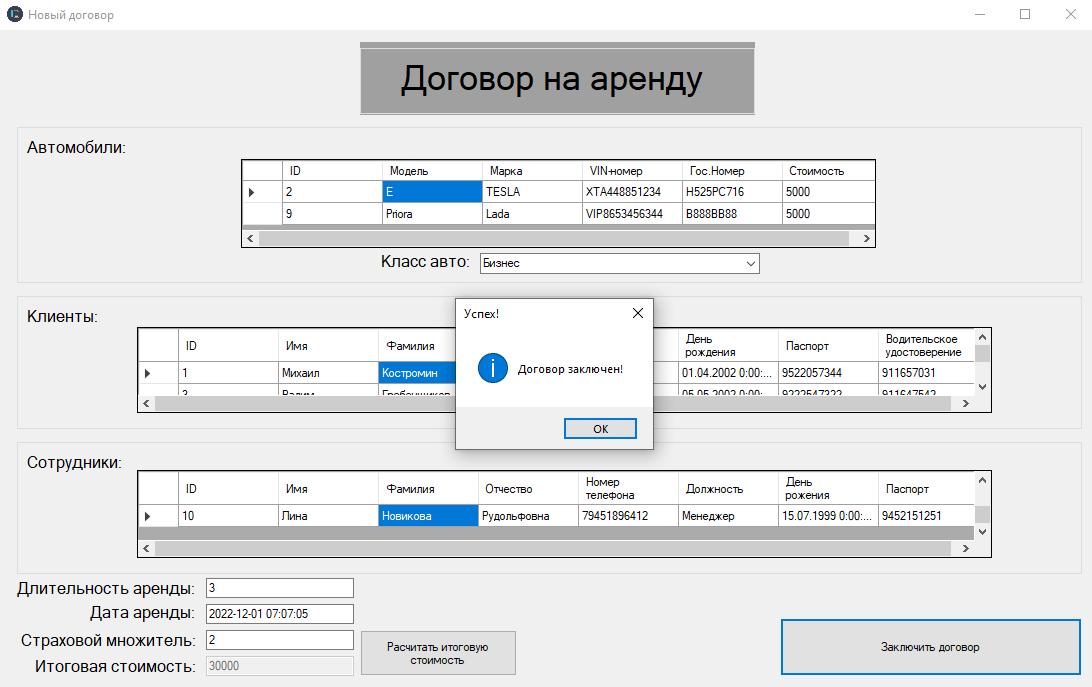
(Рис. 15 Изначальная форма нового договора)

На данной форме, как и на прошлой можно выбрать из таблиц автомобиль, клиента и сотрудника, который будет заключать договор, после этого заполнив все поля, можно нажать на кнопку расчёта стоимости аренды.



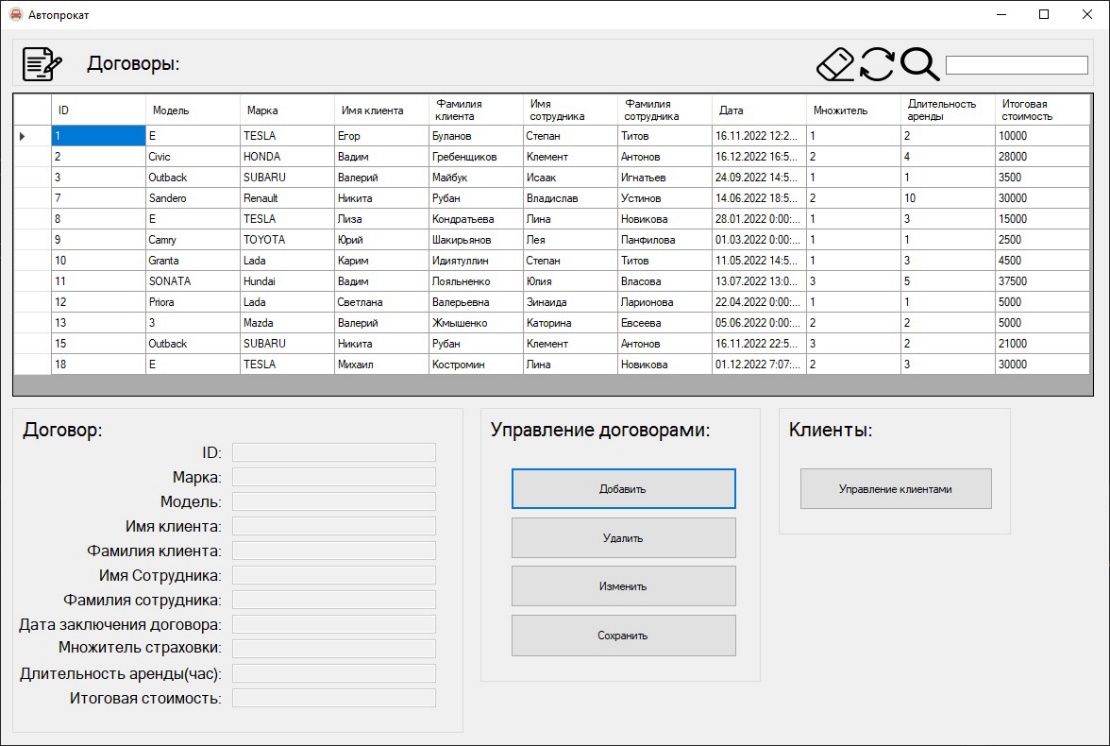
(Рис. 16 Тест расчета стоимости)

После этого можно нажать на кнопку заключить договор, появится окошко об успешном добавлении договора.



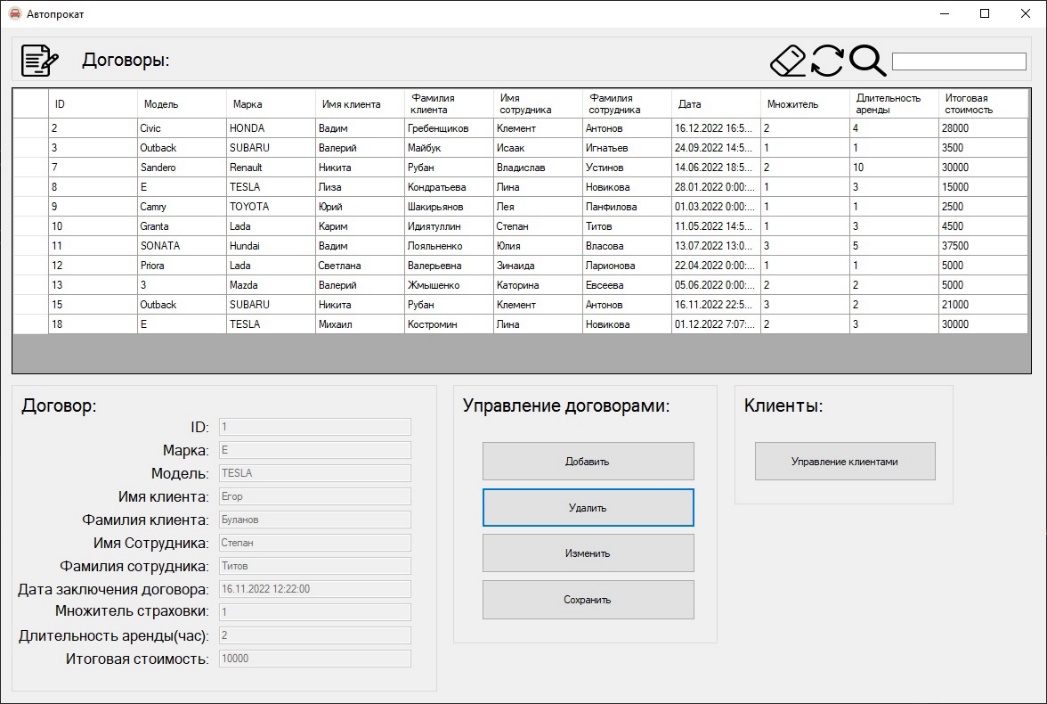
(Рис. 17 Тест добавления нового договора)

Далее попробуем удалить договор с ID 1 на главной форме.



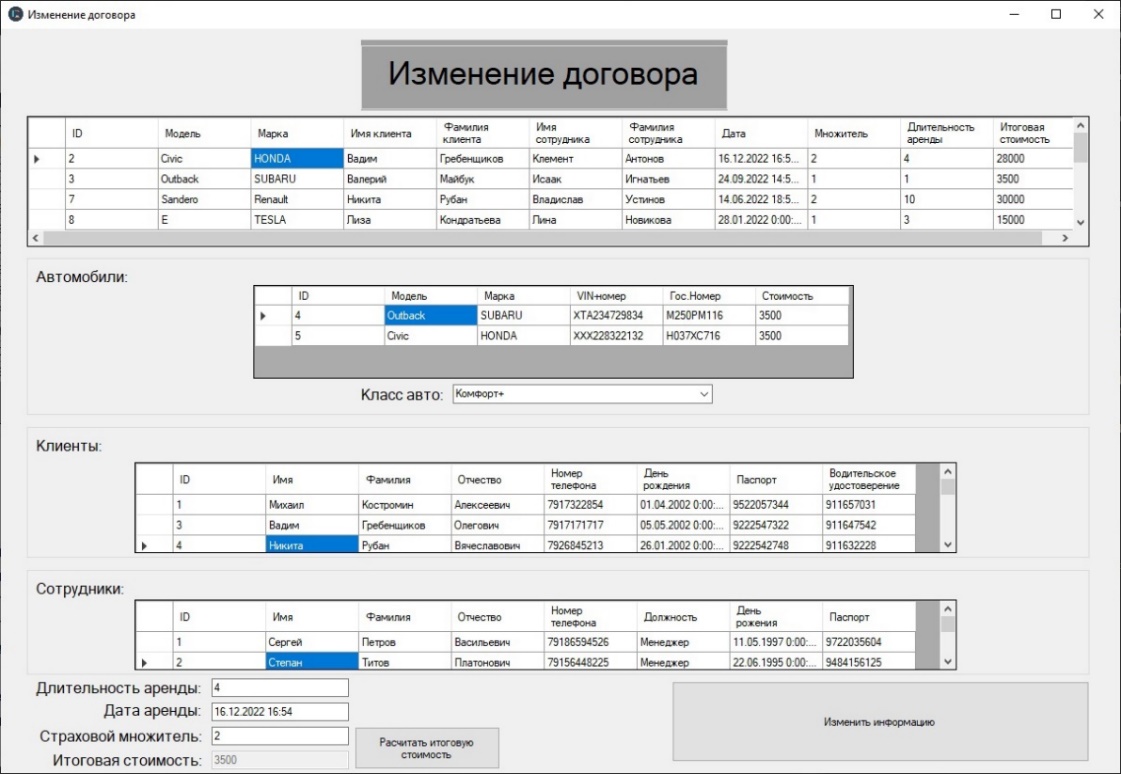
(Рис. 18 Таблица перед удалением записи)

Договор с ID 1 удалён, после удаления необходимо перенести изменения в БД с помощью кнопки сохранить.



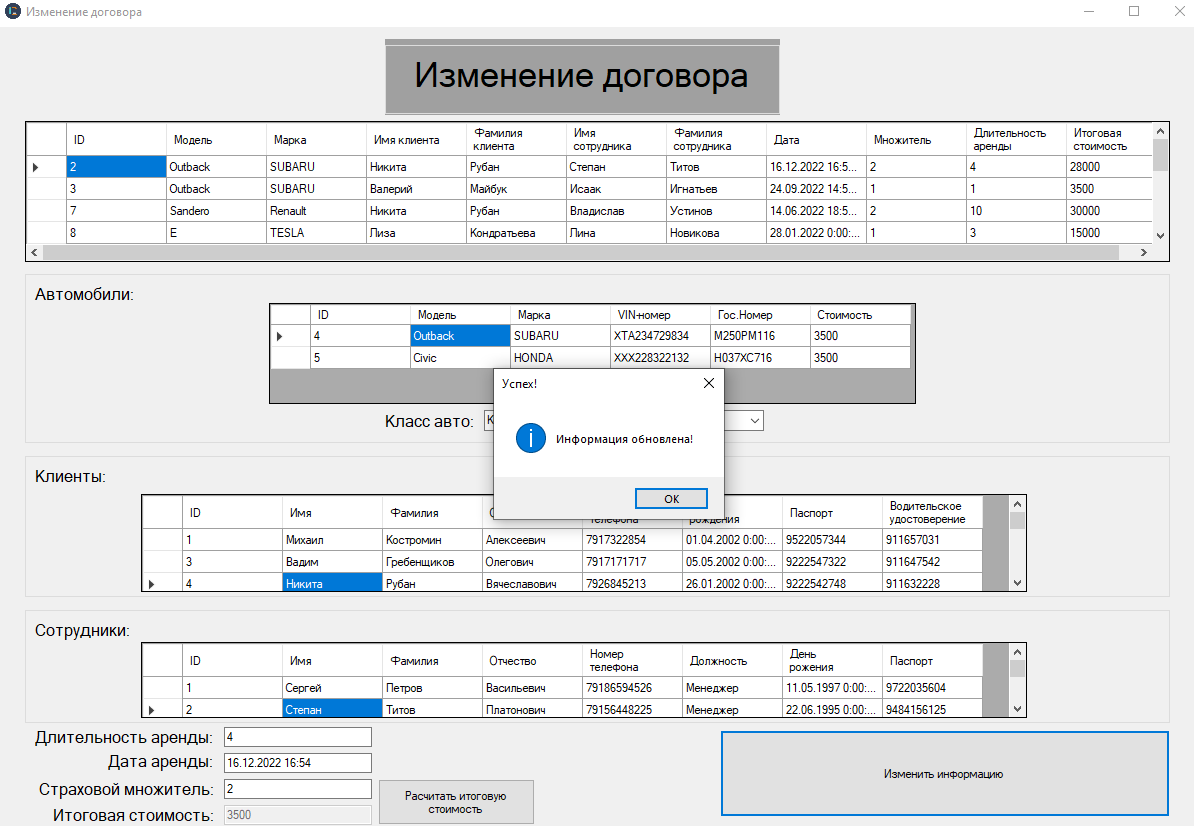
(Рис. 19 Тест удаления (договор с ID 1 удалён)

Перейдём в форму изменения договора, тут всё точно так же, как и добавлением нового договора, но есть выбор договора, который необходимо изменить



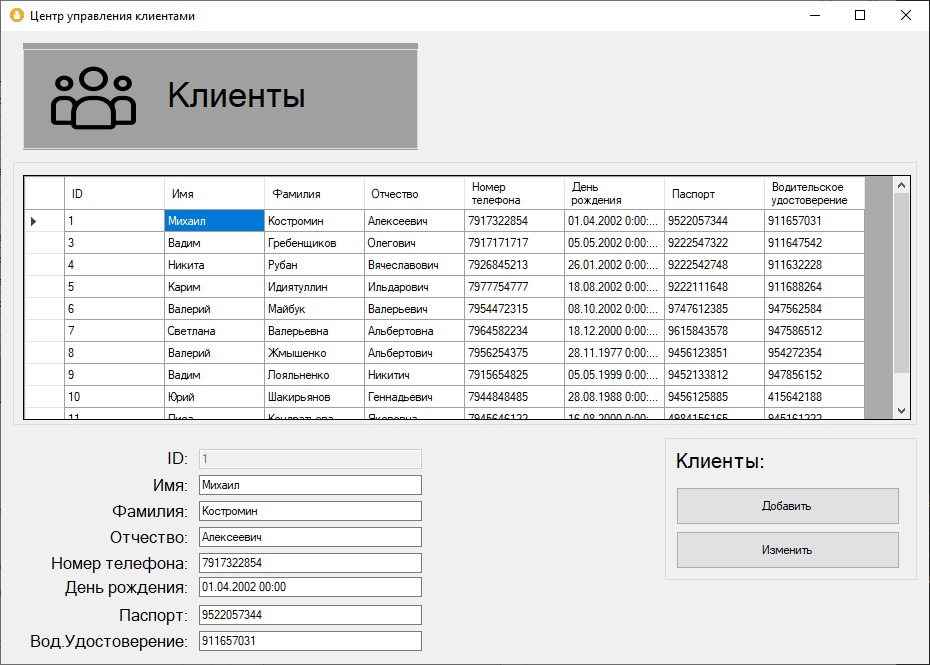
(Рис. 20 Изначальная форма изменения договора)

После занесения и выбора необходимых изменений нажимаем на кнопку изменить информацию и появляется окно с успешным изменением договора.



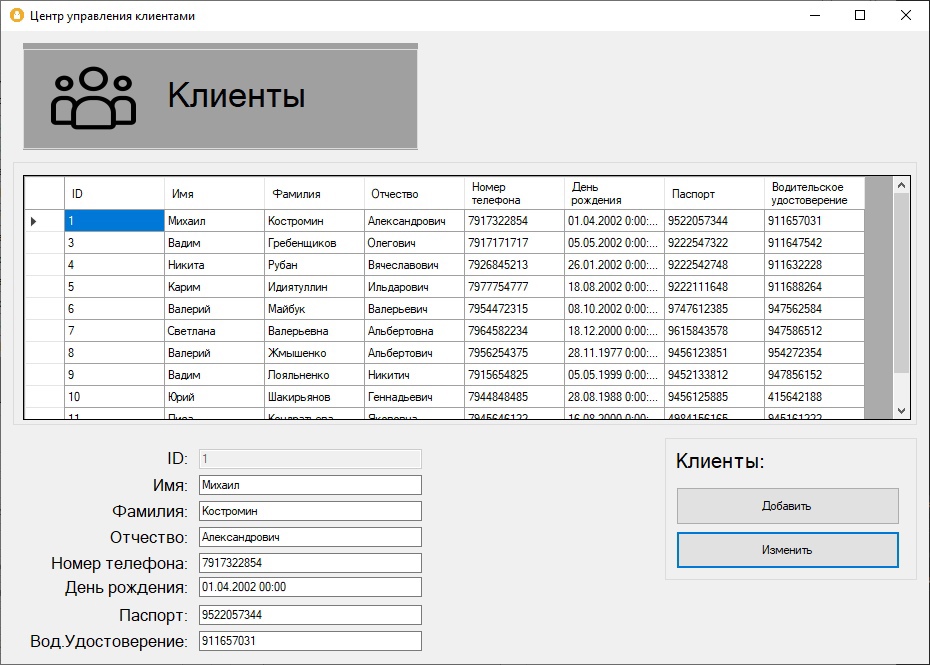
(Рис. 21 Тест изменения информации в договоре)

Перейдём на форму управления клиентами, перейти на неё можно из главной формы по кнопке.



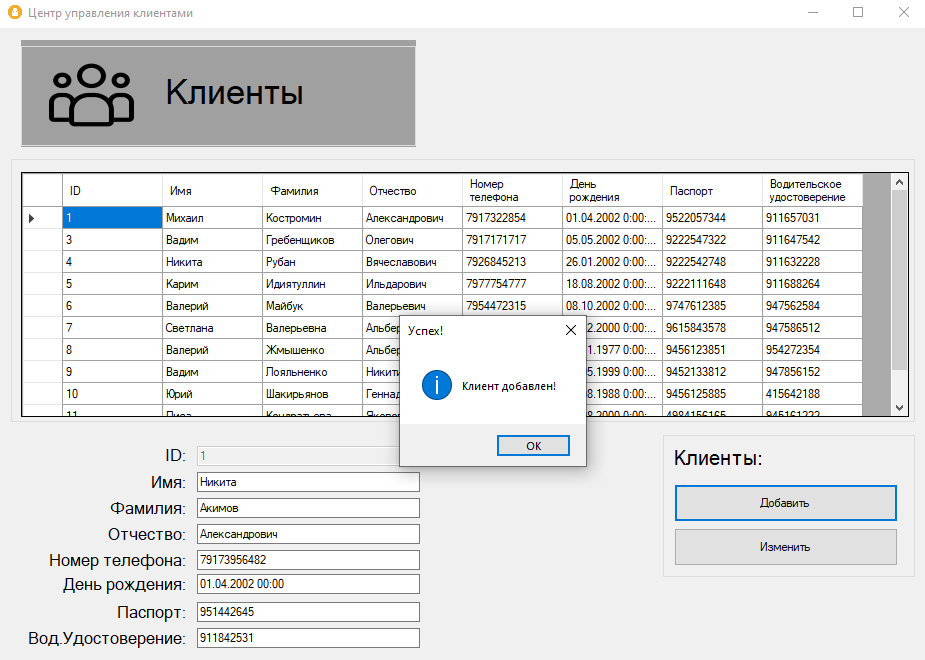
(Рис. 22 Изначальная форма клиенты)

Выберем клиента и изменим о нем информацию в полях textbox.



(Рис. 23 Тест изменения информации о клиенте)

Далее можно заполнить поля информацией о новом клиенте, и нажав на кнопку добавить, появится окно об успешном добавлении.



(Рис. 24 Тест добавления нового клиента)

# Список литературы

1. Виейра, Р. Программирование баз данных Microsoft SQL Server 2005 для профессионалов / Р. Виейра. - М.: Диалектика, **2021**. - **958** c.
2. Глушаков, С.В. Базы данных / С.В. Глушаков, Д.В. Ломотько. - М.: Харьков: Фолио, **2022**. - 504 c.
3. Нильсен, П. Microsoft SQL Server 2005: Библия пользователя / П. Нильсен. - М.: Диалектика, **2018**. - **726** c.
4. Уилтон SQL для начинающих / Уилтон, Колби Пол; , Джон. - М.: Вильямс, **2020**. - 496 c.
5. Браст, Э.Дж. Разработка приложений на основе Microsoft SQL Server 2008 / Э.Дж. Браст. - М.: Русская Редакция, **2020**. - **156** c.
6. Жилинский, А. Самоучитель Microsoft SQL Server 2005 / А. Жилинский. - М.: БХВ-Петербург, **2021**. - 224 c.
7. Петкович, Д. Microsoft SQL Server 2012. Руководство для начинающих / Д. Петкович. - М.: БХВ-Петербург, **2022**. - 816 c.
8. Тернстрем, Т. Microsoft SQL Server 2008. Разработка баз данных. Учебный курс Microsoft (+ CD-ROM) / Т. Тернстрем. - М.: Русская Редакция, **2022**. - **277** c.
9. Голицына Базы данных / Голицына, О.Л. и. - М.: Форум; Инфра-М, **2022**. - 399 c.
10. Гетц Access 2000. Руководство разработчика. Том 1. Настольные приложения. том 1 / Гетц и др. - М.: Киев: BHV, **2018**. - **272** c.
11. Маркин, А. В. СУБД «Ред База Данных». Основы SQL : учебное пособие / А. В. Маркин. — Москва : Ай Пи Ар Медиа, 2022. — 460 c. — ISBN 978-5-4497-1605-7. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/119617.html (дата обращения: 01.12.2022).
12. Базы данных : учебное пособие для обучающихся по направлению подготовки 09.03.03 «Прикладная информатика» / составители Т. Ж. Базаржапова, О. А. Гармаева, А. Ю. Хаптахаев. — Улан-Удэ : Бурятская государственная сельскохозяйственная академия им. В.Р. Филиппова, 2022. — 84 c. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/125200.html (дата обращения: 21.10.2022).
13. Введение в СУБД MySQL : учебное пособие. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 228 c. — ISBN 978-5-4497-0912-7. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/102004.html (дата обращения: 22.09.2022).
14. Тарасов, С. В. СУБД для программиста. Базы данных изнутри / С. В. Тарасов. — Москва : СОЛОН-Пресс, 2018. — 320 c. — ISBN 978-2-7466-7383-0. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/90409.html (дата обращения: 01.12.2022).
15. Кузнецов, С. Д. Введение в модель данных SQL : учебное пособие / С. Д. Кузнецов. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 350 c. — ISBN 978-5-4497-0873-1. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/101995.html (дата обращения: 22.09.2022).

# Приложение 1

Класс database

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

namespace Carsharing

{

class database

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source=DESKTOP\SQLEXPRESS;Initial Catalog=Carsharing;Integrated Security=True");

public void openConnection()//Установка соединения с бд

{

if (sqlConnection.State == System.Data.ConnectionState.Closed)

{

sqlConnection.Open();

}

}

public void closeConnection()//Закрытие соединения с бд

{

if (sqlConnection.State == System.Data.ConnectionState.Open)

{

sqlConnection.Close();

}

}

public SqlConnection getConnection()//Запрос к бд

{

return sqlConnection;

}

}

}

Главная форма Form1

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Carsharing

{

enum RowState

{

Existed,

New,

Modified,

ModifiedNew,

Deleted

}

public partial class Form1 : Form

{

database dataBase = new database();

int selectedRow;

public Form1()

{

InitializeComponent();

}

private void CreateColumns()//Создание названий столбцов

{

dataGridView1.Columns.Add("Contract.ID\_Contract", "ID");

dataGridView1.Columns.Add("Cars.Model", "Модель");

dataGridView1.Columns.Add("Cars.Mark", "Марка");

dataGridView1.Columns.Add("Client.Name", "Имя клиента");

dataGridView1.Columns.Add("Client.Surname", "Фамилия клиента");

dataGridView1.Columns.Add("Workers.Name", "Имя сотрудника");

dataGridView1.Columns.Add("Workers.Surname", "Фамилия сотрудника");

dataGridView1.Columns.Add("Contract.Date", "Дата");

dataGridView1.Columns.Add("Contract.Insurance\_coefficient", "Множитель");

dataGridView1.Columns.Add("Contract.Rent\_hours", "Длительность аренды");

dataGridView1.Columns.Add("Contract.Total\_cost", "Итоговая стоимость");

dataGridView1.Columns.Add("IsNew", String.Empty);

}

private void ReadSingleRow(DataGridView dgw, IDataRecord record)//Заполнение таблиц

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetString(4), record.GetString(5), record.GetString(6), record.GetDateTime(7), record.GetDouble(8),record.GetDouble(9), record.GetDouble(10), RowState.ModifiedNew);

}

private void RefreshDataGrid(DataGridView dgw)//Обновление таблиц

{

dgw.Rows.Clear();

string queryString = $"SELECT Contract.ID\_Contract, Cars.Model, Cars.Mark, Client.Name, Client.Surname, Workers.Name, Workers.Surname, Contract.Date, Contract.Insurance\_coefficient, Contract.Rent\_hours, Contract.Total\_cost FROM Contract INNER JOIN Client ON Contract.ID\_Client = Client.ID\_Client INNER JOIN Workers ON Contract.ID\_Worker = Workers.ID\_Worker INNER JOIN Cars ON Contract.ID\_Car = Cars.ID\_Car";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRow(dgw, reader);

}

reader.Close();

}

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

{

CreateColumns();

RefreshDataGrid(dataGridView1);

dataGridView1.Columns[11].Visible = false;

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)//Вывод в текстбоксы при выборе записи в таблице

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox\_ID.Text = row.Cells[0].Value.ToString();

textBox\_Mark.Text = row.Cells[1].Value.ToString();

textBox\_Model.Text = row.Cells[2].Value.ToString();

textBox\_cName.Text = row.Cells[3].Value.ToString();

textBox\_cSurname.Text = row.Cells[4].Value.ToString();

textBox\_wName.Text = row.Cells[5].Value.ToString();

textBox\_wSurname.Text = row.Cells[6].Value.ToString();

textBox\_Date.Text = row.Cells[7].Value.ToString();

textBox\_Insurance.Text = row.Cells[8].Value.ToString();

textBox\_RentHours.Text = row.Cells[9].Value.ToString();

textBox\_TotalCost.Text = row.Cells[10].Value.ToString();

}

}

private void pictureBox2\_Click(object sender, EventArgs e)//Обновление таблиц

{

RefreshDataGrid(dataGridView1);

}

private void button\_add\_Click(object sender, EventArgs e)//Открытие формы

{

AddForm addFRM = new AddForm();

addFRM.Show();

}

private void search(DataGridView dgw)//Поиск

{

dgw.Rows.Clear();

string searchString = $"SELECT Contract.ID\_Contract, Cars.Mark, Cars.Model, Client.Name, Client.Surname, Workers.Name AS wName, Workers.Surname AS wSurname, Contract.Date, Contract.Insurance\_coefficient, Contract.Rent\_hours, Contract.Total\_cost FROM Contract INNER JOIN Client ON Contract.ID\_Client = Client.ID\_Client INNER JOIN Workers ON Contract.ID\_Worker = Workers.ID\_Worker INNER JOIN Cars ON Contract.ID\_Car = Cars.ID\_Car WHERE ('Contract.ID\_Contract' + Cars.Mark + Cars.Model + Client.Name + Client.Surname + Workers.Name + Workers.Surname + 'Contract.Date' + 'Contract.Insurance\_coefficient' + 'Contract.Rent\_hours' + 'Contract.Total\_cost' ) like '%" + textBox\_search.Text + "%'";

SqlCommand com = new SqlCommand(searchString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader read = com.ExecuteReader();

while (read.Read())

{

ReadSingleRow(dgw, read);

}

read.Close();

}

private void textBox\_search\_TextChanged(object sender, EventArgs e)

{

search(dataGridView1);

}

private void deleteRow()//Удаление договора из таблицы

{

int index = dataGridView1.CurrentCell.RowIndex;

dataGridView1.Rows[index].Visible = false;

if (dataGridView1.Rows[index].Cells[0].Value.ToString() == String.Empty)

{

dataGridView1.Rows[index].Cells[11].Value = RowState.Deleted;

return;

}

dataGridView1.Rows[index].Cells[11].Value = RowState.Deleted;

}

private void button\_delete\_Click(object sender, EventArgs e)

{

deleteRow();

}

private void Update()//Удаление договора из бд

{

dataBase.openConnection();

for (int index = 0; index < dataGridView1.Rows.Count; index++)

{

var rowState = (RowState)dataGridView1.Rows[index].Cells[11].Value;

if (rowState == RowState.Existed)

continue;

if (rowState == RowState.Deleted)

{

var id = Convert.ToInt32(dataGridView1.Rows[index].Cells[0].Value);

var deleteQuery = $"DELETE FROM CONTRACT WHERE CONTRACT.ID\_Contract ={id}";

var command = new SqlCommand(deleteQuery, dataBase.getConnection());

command.ExecuteNonQuery();

}

}

dataBase.closeConnection();

}

private void button\_save\_Click(object sender, EventArgs e)//Сохранение

{

Update();

}

private void button\_change\_Click(object sender, EventArgs e)//Открытие формы

{

ChangeForm chngFRM = new ChangeForm();

chngFRM.Show();

}

private void button\_ClientControl\_Click(object sender, EventArgs e)//Открытие формы

{

ClientControl clientFRM = new ClientControl();

clientFRM.Show();

}

private void pictureBox1\_Click(object sender, EventArgs e)//Очистка текстбоксов

{

textBox\_cName.Text = "";

textBox\_cSurname.Text = "";

textBox\_Date.Text = "";

textBox\_ID.Text = "";

textBox\_Insurance.Text = "";

textBox\_Mark.Text = "";

textBox\_Model.Text = "";

textBox\_RentHours.Text = "";

textBox\_TotalCost.Text = "";

textBox\_wName.Text = "";

textBox\_wSurname.Text = "";

}

}

}

Форма Управления клиентами ClientControl

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Carsharing

{

public partial class ClientControl : Form

{

database dataBase = new database();//Подключение класса для работы с бд

int selectedRow,ID\_Client;//Переменные для работы методов

public ClientControl()

{

InitializeComponent();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void CreateColumnsClient()//Создание названий столбцов

{

dataGridView2.Columns.Add("Client.ID\_Client", "ID");

dataGridView2.Columns.Add("Client.Name", "Имя");

dataGridView2.Columns.Add("Client.Surname", "Фамилия");

dataGridView2.Columns.Add("Client.Otchestvo", "Отчество");

dataGridView2.Columns.Add("Client.Phone\_Number", "Номер телефона");

dataGridView2.Columns.Add("Client.Birthday", "День рождения");

dataGridView2.Columns.Add("Client.Pass", "Паспорт");

dataGridView2.Columns.Add("Client.Licence", "Водительское удостоверение");

}

private void ReadSingleRowClient(DataGridView dgw, IDataRecord record)//Заполнение таблиц

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetDouble(4), record.GetDateTime(5), record.GetString(6), record.GetString(7));

}

private void RefreshDataGridClient(DataGridView dgw)//Обновление таблиц

{

dgw.Rows.Clear();

string queryString = $"SELECT Client.ID\_Client,Client.Name,Client.Surname,Client.Otchestvo,Client.Phone\_Number,Birthday,Client.Pass,Client.Licence FROM Client";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowClient(dgw, reader);

}

reader.Close();

}

private void ClientControl\_Load(object sender, EventArgs e)//методы при загрузке формы

{

CreateColumnsClient();

RefreshDataGridClient(dataGridView2);

}

private void button\_ClientAdd\_Click(object sender, EventArgs e)//Добавление клиента

{

double phone;

if ((textBox\_cName.Text != "") && (textBox\_cSurname.Text != "") && (textBox\_cOtch.Text != "") && (textBox\_cPhone.Text != "") && (textBox\_cPass.Text != "") && (textBox\_cLicence.Text != "")&&(double.TryParse(textBox\_cPhone.Text,out phone)))

{

dataBase.openConnection();

DateTime Date = Convert.ToDateTime(textBox\_cBirth.Text);

phone = Convert.ToDouble(textBox\_cPhone.Text);

string name = textBox\_cName.Text;

string surname = textBox\_cSurname.Text;

string otch = textBox\_cOtch.Text;

string pass = textBox\_cPass.Text;

string lic = textBox\_cLicence.Text;

var addQuery = $"INSERT INTO Client (Name,Surname,Otchestvo,Phone\_Number,Birthday,Pass,Licence) VALUES ('{name}','{surname}','{otch}','{phone}','{Date}','{pass}','{lic}')";

var command = new SqlCommand(addQuery, dataBase.getConnection());

command.ExecuteNonQuery();

MessageBox.Show("Клиент добавлен!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Заполните поля!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

dataBase.closeConnection();

RefreshDataGridClient(dataGridView2);

}

private void button\_ClientChange\_Click(object sender, EventArgs e)//Изменение информации о клиенте кнопка

{

change();

}

private void change()//Изменение информации о клиенте

{

var selectedRowIndex = dataGridView2.CurrentCell.RowIndex;

var id = textBox\_cID.Text;

var name = textBox\_cName.Text;

var surname = textBox\_cSurname.Text;

var otch = textBox\_cOtch.Text;

double phone = Convert.ToDouble(textBox\_cPhone.Text);

DateTime birth = Convert.ToDateTime(textBox\_cBirth.Text);

var pass = textBox\_cPass.Text;

var lic = textBox\_cLicence.Text;

if (dataGridView2.Rows[selectedRowIndex].Cells[0].Value.ToString()!= String.Empty)

{

if (double.TryParse(textBox\_cPhone.Text,out phone))

{

var changeQuery = $"UPDATE Client SET Name = '{name}', Surname = '{surname}', Otchestvo = '{otch}', Phone\_Number = '{phone}', Birthday = '{birth}', Pass = '{pass}', Licence = '{lic}' WHERE ID\_Client = '{id}'";

dataBase.openConnection();

var command = new SqlCommand(changeQuery, dataBase.getConnection());

command.ExecuteNonQuery();

dataBase.closeConnection();

RefreshDataGridClient(dataGridView2);

}

else

{

MessageBox.Show("Поле заполнено неверно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)//Вывод в текстбоксы при выборе записи в таблице

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView2.Rows[selectedRow];

textBox\_cID.Text = row.Cells[0].Value.ToString();

textBox\_cName.Text = row.Cells[1].Value.ToString();

textBox\_cSurname.Text = row.Cells[2].Value.ToString();

textBox\_cOtch.Text = row.Cells[3].Value.ToString();

textBox\_cPhone.Text = row.Cells[4].Value.ToString();

textBox\_cBirth.Text = row.Cells[5].Value.ToString();

textBox\_cPass.Text = row.Cells[6].Value.ToString();

textBox\_cLicence.Text = row.Cells[7].Value.ToString();

}

}

}

}

Форма добавление нового договора AddForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Carsharing

{

public partial class AddForm : Form

{

database dataBase = new database();//Подключение класса для работы с бд

int selectedRow;//Переменные для работы методов

int TotalMult;//Переменные для работы методов

int ID\_Car, ID\_Client, ID\_Worker;//Переменные для работы методов

public AddForm()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

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

{

CreateColumnsCar();

RefreshDataGridCar(dataGridView1);

comboBoxRefill();

CreateColumnsClient();

RefreshDataGridClient(dataGridView2);

CreateColumnsWorkers();

RefreshDataGridWorker(dataGridView3);

}

private void CreateColumnsCar()//Создание названий столбцов

{

dataGridView1.Columns.Add("Cars.ID\_Car", "ID");

dataGridView1.Columns.Add("Cars.Model", "Модель");

dataGridView1.Columns.Add("Cars.Mark", "Марка");

dataGridView1.Columns.Add("Cars.VIN", "VIN-номер");

dataGridView1.Columns.Add("Cars.GosNumber", "Гос.Номер");

dataGridView1.Columns.Add("CarClass.Price", "Стоимость");

}

private void CreateColumnsClient()//Создание названий столбцов

{

dataGridView2.Columns.Add("Client.ID\_Client", "ID");

dataGridView2.Columns.Add("Client.Name", "Имя");

dataGridView2.Columns.Add("Client.Surname", "Фамилия");

dataGridView2.Columns.Add("Client.Otchestvo", "Отчество");

dataGridView2.Columns.Add("Client.Phone\_Number", "Номер телефона");

dataGridView2.Columns.Add("Client.Birthday", "День рождения");

dataGridView2.Columns.Add("Client.Pass", "Паспорт");

dataGridView2.Columns.Add("Client.Licence", "Водительское удостоверение");

}

private void CreateColumnsWorkers()//Создание названий столбцов

{

dataGridView3.Columns.Add("Workers.ID\_Worker", "ID");

dataGridView3.Columns.Add("Workers.Name", "Имя");

dataGridView3.Columns.Add("Workers.Surname", "Фамилия");

dataGridView3.Columns.Add("Workers.Otchestvo", "Отчество");

dataGridView3.Columns.Add("Workers.Phone\_Number", "Номер телефона");

dataGridView3.Columns.Add("Workers.Post", "Должность");

dataGridView3.Columns.Add("Workers.Birthday", "День рожения");

dataGridView3.Columns.Add("Workers.Pass", "Паспорт");

}

private void ReadSingleRowWorker(DataGridView dgw, IDataRecord record)//Заполнение таблиц

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetDouble(4), record.GetString(5), record.GetDateTime(6), record.GetString(7));

}

private void ReadSingleRowClient(DataGridView dgw, IDataRecord record)//Заполнение таблиц

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetDouble(4), record.GetDateTime(5), record.GetString(6), record.GetString(7));

}

private void RefreshDataGridClient(DataGridView dgw)//Обновление таблиц

{

dgw.Rows.Clear();

string queryString = $"SELECT Client.ID\_Client,Client.Name,Client.Surname,Client.Otchestvo,Client.Phone\_Number,Birthday,Client.Pass,Client.Licence FROM Client";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowClient(dgw, reader);

}

reader.Close();

}

private void RefreshDataGridWorker(DataGridView dgw)//Обновление таблиц

{

dgw.Rows.Clear();

string queryString = $"SELECT Workers.ID\_Worker,Workers.Name,Workers.Surname,Workers.Otchestvo,Workers.Phone\_Number,Workers.Post,Workers.Birthday,Workers.Pass FROM Workers";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowWorker(dgw, reader);

}

reader.Close();

}

private void comboBoxRefill()//Заполнение ComboBox

{

comboBox\_Class.Items.Clear();

string queryString = $"SELECT CarClass.Title FROM CarClass";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox\_Class.Items.Add(reader.GetString(0));

}

reader.Close();

}

private void ReadSingleRowCar(DataGridView dgw, IDataRecord record)//Заполнение таблиц

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetString(4), record.GetDouble(5));

}

private void RefreshDataGridCar(DataGridView dgw)//Обновление таблиц

{

dgw.Rows.Clear();

if (comboBox\_Class.SelectedItem == null)

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

else if (Convert.ToString(comboBox\_Class.SelectedItem) == "Эконом")

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class WHERE CarClass.Title like 'Эконом'";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

else if (Convert.ToString(comboBox\_Class.SelectedItem) == "Комфорт")

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class WHERE CarClass.Title like 'Комфорт'";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

else if (Convert.ToString(comboBox\_Class.SelectedItem) == "Комфорт+")

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class WHERE CarClass.Title like 'Комфорт+'";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

else if (Convert.ToString(comboBox\_Class.SelectedItem) == "Бизнес")

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class WHERE CarClass.Title like 'Бизнес'";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

}

private void button\_add2\_Click(object sender, EventArgs e)//Добавление нового договора в бд

{

if ((ID\_Car>0)&&(ID\_Client > 0) && (ID\_Worker > 0)&&(textBox\_RentDate.Text!=null))

{

dataBase.openConnection();

DateTime Date = Convert.ToDateTime(textBox\_RentDate.Text);

double InsuranceMult;

double RentHrs;

double TotalCost;

if (double.TryParse(textBox\_Insurance.Text, out InsuranceMult) && (double.TryParse(textBox\_Rent.Text, out RentHrs)) && (double.TryParse(textBox\_Total.Text, out TotalCost)))

{

if ((textBox\_Rent.Text != "") && (textBox\_Insurance.Text != ""))

{

double rent;

double Insurance;

if ((textBox\_Total.Text != "") && (double.TryParse(textBox\_Insurance.Text, out Insurance)) && (double.TryParse(textBox\_Rent.Text, out rent)))

{

rent = Convert.ToDouble(textBox\_Rent.Text);

Insurance = Convert.ToDouble(textBox\_Insurance.Text);

TotalCost = Convert.ToInt32(textBox\_Rent.Text) \* Convert.ToInt32(textBox\_Insurance.Text) \* TotalMult;

var addQuery = $"INSERT INTO Contract (ID\_Car,ID\_Client,ID\_Worker,Date,Insurance\_coefficient,Total\_cost,Rent\_hours) VALUES ('{ID\_Car}','{ID\_Client}','{ID\_Worker}','{Date}','{InsuranceMult}','{TotalCost}','{RentHrs}')";

var command = new SqlCommand(addQuery, dataBase.getConnection());

command.ExecuteNonQuery();

MessageBox.Show("Договор заключен!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Проверьте формат данных!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Проверьте формат данных или заполните поля!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Проверьте формат данных или заполните поля!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

dataBase.closeConnection();

}

else

{

MessageBox.Show("Выберите авто,клиента и сотрудника!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void comboBox\_Class\_SelectedIndexChanged(object sender, EventArgs e)//Обновление таблиц

{

RefreshDataGridCar(dataGridView1);

}

private void textBox\_Insurance\_TextChanged(object sender, EventArgs e)

{

}

private void textBox\_Rent\_TextChanged(object sender, EventArgs e)

{

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)//Вывод в текстбоксы при выборе записи в таблице

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView2.Rows[selectedRow];

ID\_Client = Convert.ToInt32(row.Cells[0].Value.ToString());

}

}

private void dataGridView3\_CellClick(object sender, DataGridViewCellEventArgs e)//Вывод в текстбоксы при выборе записи в таблице

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView3.Rows[selectedRow];

ID\_Worker = Convert.ToInt32(row.Cells[0].Value.ToString());

}

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)//Вывод в текстбоксы при выборе записи в таблице

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox\_Total.Text = row.Cells[5].Value.ToString();

TotalMult = Convert.ToInt32(row.Cells[5].Value.ToString());

ID\_Car = Convert.ToInt32(row.Cells[0].Value.ToString());

}

}

private void button\_mult\_Click(object sender, EventArgs e)//Подсчет итоговой стоимости

{

if ((textBox\_Rent.Text != "") && (textBox\_Insurance.Text != ""))

{

double rent;

double Insurance;

if ((textBox\_Total.Text != "") && (double.TryParse(textBox\_Insurance.Text, out Insurance)) && (double.TryParse(textBox\_Rent.Text, out rent)))

{

rent = Convert.ToDouble(textBox\_Rent.Text);

Insurance = Convert.ToDouble(textBox\_Insurance.Text);

int total = 0;

total = Convert.ToInt32(textBox\_Rent.Text) \* Convert.ToInt32(textBox\_Insurance.Text) \* TotalMult;

textBox\_Total.Text = Convert.ToString(total);

}

else

{

MessageBox.Show("Проверьте формат данных!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Проверьте формат данных или заполните поля!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

}

Форма изменения договора ChangeForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Carsharing

{

public partial class ChangeForm : Form

{

database dataBase = new database();//Подключение класса для работы с бд

int selectedRow;//Переменные для работы методов

int id\_con, id\_car, id\_client, id\_work,TotalMult;//Переменные для работы методов

public ChangeForm()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

private void ChangeForm\_Load(object sender, EventArgs e)//методы при загрузке формы

{

CreateColumns();

RefreshDataGrid(dataGridView1);

CreateColumnsCar();

RefreshDataGridCar(dataGridView4);

comboBoxRefill();

CreateColumnsClient();

RefreshDataGridClient(dataGridView2);

CreateColumnsWorkers();

RefreshDataGridWorker(dataGridView3);

}

private void CreateColumns()//Создание названий столбцов

{

dataGridView1.Columns.Add("Contract.ID\_Contract", "ID");

dataGridView1.Columns.Add("Cars.Model", "Модель");

dataGridView1.Columns.Add("Cars.Mark", "Марка");

dataGridView1.Columns.Add("Client.Name", "Имя клиента");

dataGridView1.Columns.Add("Client.Surname", "Фамилия клиента");

dataGridView1.Columns.Add("Workers.Name", "Имя сотрудника");

dataGridView1.Columns.Add("Workers.Surname", "Фамилия сотрудника");

dataGridView1.Columns.Add("Contract.Date", "Дата");

dataGridView1.Columns.Add("Contract.Insurance\_coefficient", "Множитель");

dataGridView1.Columns.Add("Contract.Rent\_hours", "Длительность аренды");

dataGridView1.Columns.Add("Contract.Total\_cost", "Итоговая стоимость");

}

private void CreateColumnsCar()//Создание названий столбцов

{

dataGridView4.Columns.Add("Cars.ID\_Car", "ID");

dataGridView4.Columns.Add("Cars.Model", "Модель");

dataGridView4.Columns.Add("Cars.Mark", "Марка");

dataGridView4.Columns.Add("Cars.VIN", "VIN-номер");

dataGridView4.Columns.Add("Cars.GosNumber", "Гос.Номер");

dataGridView4.Columns.Add("CarClass.Price", "Стоимость");

}

private void CreateColumnsClient()//Создание названий столбцов

{

dataGridView2.Columns.Add("Client.ID\_Client", "ID");

dataGridView2.Columns.Add("Client.Name", "Имя");

dataGridView2.Columns.Add("Client.Surname", "Фамилия");

dataGridView2.Columns.Add("Client.Otchestvo", "Отчество");

dataGridView2.Columns.Add("Client.Phone\_Number", "Номер телефона");

dataGridView2.Columns.Add("Client.Birthday", "День рождения");

dataGridView2.Columns.Add("Client.Pass", "Паспорт");

dataGridView2.Columns.Add("Client.Licence", "Водительское удостоверение");

}

private void CreateColumnsWorkers()//Создание названий столбцов

{

dataGridView3.Columns.Add("Workers.ID\_Worker", "ID");

dataGridView3.Columns.Add("Workers.Name", "Имя");

dataGridView3.Columns.Add("Workers.Surname", "Фамилия");

dataGridView3.Columns.Add("Workers.Otchestvo", "Отчество");

dataGridView3.Columns.Add("Workers.Phone\_Number", "Номер телефона");

dataGridView3.Columns.Add("Workers.Post", "Должность");

dataGridView3.Columns.Add("Workers.Birthday", "День рожения");

dataGridView3.Columns.Add("Workers.Pass", "Паспорт");

}

private void ReadSingleRowWorker(DataGridView dgw, IDataRecord record)//Заполнение таблиц

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetDouble(4), record.GetString(5), record.GetDateTime(6), record.GetString(7));

}

private void ReadSingleRowClient(DataGridView dgw, IDataRecord record)//Заполнение таблиц

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetDouble(4), record.GetDateTime(5), record.GetString(6), record.GetString(7));

}

private void RefreshDataGridClient(DataGridView dgw)//Обновление таблиц

{

dgw.Rows.Clear();

string queryString = $"SELECT Client.ID\_Client,Client.Name,Client.Surname,Client.Otchestvo,Client.Phone\_Number,Birthday,Client.Pass,Client.Licence FROM Client";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();//Обновление таблиц

while (reader.Read())

{

ReadSingleRowClient(dgw, reader);

}

reader.Close();

}

private void RefreshDataGridWorker(DataGridView dgw)//Обновление таблиц

{

dgw.Rows.Clear();

string queryString = $"SELECT Workers.ID\_Worker,Workers.Name,Workers.Surname,Workers.Otchestvo,Workers.Phone\_Number,Workers.Post,Workers.Birthday,Workers.Pass FROM Workers";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowWorker(dgw, reader);

}

reader.Close();

}

private void comboBoxRefill()//Заполнение выбора в comboBox

{

comboBox\_Class.Items.Clear();

string queryString = $"SELECT CarClass.Title FROM CarClass";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox\_Class.Items.Add(reader.GetString(0));

}

reader.Close();

}

private void ReadSingleRowCar(DataGridView dgw, IDataRecord record)//Заполнение таблиц

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetString(4), record.GetDouble(5));

}

private void RefreshDataGridCar(DataGridView dgw)//Обновление таблиц

{

dgw.Rows.Clear();

if (comboBox\_Class.SelectedItem == null)

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

else if (Convert.ToString(comboBox\_Class.SelectedItem) == "Эконом")

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class WHERE CarClass.Title like 'Эконом'";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

else if (Convert.ToString(comboBox\_Class.SelectedItem) == "Комфорт")

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class WHERE CarClass.Title like 'Комфорт'";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

else if (Convert.ToString(comboBox\_Class.SelectedItem) == "Комфорт+")

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class WHERE CarClass.Title like 'Комфорт+'";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

else if (Convert.ToString(comboBox\_Class.SelectedItem) == "Бизнес")

{

string queryString = $"SELECT Cars.ID\_Car,Cars.Model,Cars.Mark,Cars.VIN,Cars.GosNumber,CarClass.Price FROM Cars INNER JOIN CarClass on Cars.ID\_Class = CarClass.ID\_Class WHERE CarClass.Title like 'Бизнес'";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRowCar(dgw, reader);

}

reader.Close();

}

}

private void ReadSingleRow(DataGridView dgw, IDataRecord record)//Заполнение таблиц

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetString(3), record.GetString(4), record.GetString(5), record.GetString(6), record.GetDateTime(7), record.GetDouble(8), record.GetDouble(9), record.GetDouble(10));

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)//Вывод в текстбоксы при выборе записи в таблице

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

id\_con = Convert.ToInt32(row.Cells[0].Value.ToString());

textBox\_RentDate.Text = row.Cells[7].Value.ToString();

textBox\_Insurance.Text = row.Cells[8].Value.ToString();

textBox\_Rent.Text = row.Cells[9].Value.ToString();

}

}

private void dataGridView4\_CellClick(object sender, DataGridViewCellEventArgs e)//Вывод в текстбоксы при выборе записи в таблице

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView4.Rows[selectedRow];

textBox\_Total.Text = row.Cells[5].Value.ToString();

TotalMult = Convert.ToInt32(row.Cells[5].Value.ToString());

id\_car = Convert.ToInt32(row.Cells[0].Value.ToString());

}

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)//Вывод в текстбоксы при выборе записи в таблице

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView2.Rows[selectedRow];

id\_client = Convert.ToInt32(row.Cells[0].Value.ToString());

}

}

private void dataGridView3\_CellClick(object sender, DataGridViewCellEventArgs e)//Вывод в текстбоксы при выборе записи в таблице

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView3.Rows[selectedRow];

id\_work = Convert.ToInt32(row.Cells[0].Value.ToString());

}

}

private void button\_mult\_Click(object sender, EventArgs e)//Подсчет итоговой стоимости

{

if ((textBox\_Rent.Text != "") && (textBox\_Insurance.Text != ""))

{

double rent;

double Insurance;

if ((textBox\_Total.Text != "") && (double.TryParse(textBox\_Insurance.Text, out Insurance)) && (double.TryParse(textBox\_Rent.Text, out rent)))

{

rent = Convert.ToDouble(textBox\_Rent.Text);

Insurance = Convert.ToDouble(textBox\_Insurance.Text);

int total = 0;

total = Convert.ToInt32(textBox\_Rent.Text) \* Convert.ToInt32(textBox\_Insurance.Text) \* TotalMult;

textBox\_Total.Text = Convert.ToString(total);

}

else

{

MessageBox.Show("Проверьте формат данных!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Проверьте формат данных или заполните поля!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void RefreshDataGrid(DataGridView dgw)//Обновление таблиц

{

dgw.Rows.Clear();

string queryString = $"SELECT Contract.ID\_Contract, Cars.Model, Cars.Mark, Client.Name, Client.Surname, Workers.Name, Workers.Surname, Contract.Date, Contract.Insurance\_coefficient, Contract.Rent\_hours, Contract.Total\_cost FROM Contract INNER JOIN Client ON Contract.ID\_Client = Client.ID\_Client INNER JOIN Workers ON Contract.ID\_Worker = Workers.ID\_Worker INNER JOIN Cars ON Contract.ID\_Car = Cars.ID\_Car";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

dataBase.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSingleRow(dgw, reader);

}

reader.Close();

}

private void comboBox\_Class\_SelectedIndexChanged(object sender, EventArgs e)//Обновление таблиц

{

RefreshDataGridCar(dataGridView4);

}

private void button\_ChangeInfo\_Click(object sender, EventArgs e)//Изменение информации договора в бд

{

if ((id\_car > 0) && (id\_client > 0) && (id\_con > 0) && (id\_work > 0))

{

DateTime Date = Convert.ToDateTime(textBox\_RentDate.Text);

double insurance;

double rent;

double TotalCost;

if ((textBox\_Total.Text != "") && (double.TryParse(textBox\_Insurance.Text, out insurance)) && (double.TryParse(textBox\_Rent.Text, out rent)))

{

rent = Convert.ToDouble(textBox\_Rent.Text);

insurance = Convert.ToDouble(textBox\_Insurance.Text);

TotalCost = Convert.ToInt32(textBox\_Rent.Text) \* Convert.ToInt32(textBox\_Insurance.Text) \* TotalMult;

var changeQuery = $"UPDATE Contract SET ID\_Car ='{id\_car}', ID\_Client = '{id\_client}', ID\_Worker = '{id\_work}', Date = '{Date}', Insurance\_coefficient = '{insurance}', Total\_cost = '{TotalCost}', Rent\_hours = '{rent}' WHERE ID\_Contract = {id\_con}";

dataBase.openConnection();

var command = new SqlCommand(changeQuery, dataBase.getConnection());

command.ExecuteNonQuery();

dataBase.closeConnection();

RefreshDataGrid(dataGridView1);

MessageBox.Show("Информация обновлена!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Проверьте формат данных!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Проверьте формат данных, заполните поля либо выберите данные!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

}