**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых**»

(ВлГУ)

Колледж инновационных технологий и предпринимательства

**КАФЕДРА ФИЗИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

по дисциплине «Технология разработки и защиты баз данных»

Тема: «Туристический оператор: разработка и администрирование базы данных, разработка клиентского приложения»

Выполнил студент

группы ПКсп-120

Агафонов Д. А.

Приняла Кабанова М. Ю.

Владимир, 2023

ВЛАДИМИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «**Физики и прикладной математики**»

УТВЕРЖДАЮ

Зав. кафедрой С.И. Абрахин

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

по дисциплине «Технология разработки и защиты баз данных»

Выдано студенту **Родионову Юрию Руслановичу** группы **ПКсп-120**

Направление подготовки СПО 09.02.03 «Программирование в компьютерных системах»

1. Тема работы **«Туристический оператор: разработка и администрирование базы данных, разработка клиентского приложения»**

2. Срок сдачи законченной работы **00.00.2023**

3. Исходные данные к курсовой работе постановка задачи, MS SQL Server 2008 и выше.

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

Введение

1. Проектирование базы данных
2. Подготовка базы данных
3. Разработка клиентского приложения

Заключение

5. Постановка задачи

ЧАСТЬ 1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

1. Выполнить описание предметной области.

2. На основании описания предметной области выполнить разработку ER-модели, реляционной модели данных. При проектировании придерживаться нормализации данных не ниже нормальной формы Бойса-Кодда.

ЧАСТЬ 2. ПОДГОТОВКА БАЗЫ ДАННЫХ

1. Написать sql-скрипт создания структуры базы данных согласно схеме из варианта индивидуального задания.

2. Написать sql-скрипт заполнения таблиц тестовыми данными (не менее 10 строк в каждой таблице).

3. Написать sql-скрипт удаления данных из всех таблиц.

4. Написать sql-скрипт создания дополнительных объектов базы данных, необходимых для обеспечения работы клиентского приложения: триггеры, хранимые процедуры, представления и т.д.

ЧАСТЬ 3. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

1. Выполнить проектирование приложения (использовать нотацию UML).

2. Выполнить разработку приложения.

3. Выполнить тестирование.

Дата выдачи задания 00.00.2023

Руководитель М.Ю. Кабанова

Задание приняли к исполнению Д.А. Агафонов

Содержание

[Введение 4](#_Toc131377722)

[1 Проектирование базы данных 5](#_Toc131377723)

[1.1 ER-диаграмма 5](#_Toc131377724)

[2. Подготовка базы данных 8](#_Toc131377725)

[2.1 Словарь данных 8](#_Toc131377726)

[2.2 Нормализация полученных отношений 12](#_Toc131377727)

[2.3 Процедуры, функции, представления, триггеры 13](#_Toc131377728)

[3. Резервное копирование и восстановление 16](#_Toc131377729)

[4. УПРАВЛЕНИЕ ДОСТУПОМ 18](#_Toc131377730)

[5. Разработка клиентского приложения 21](#_Toc131377731)

[5.1 Структура приложения 21](#_Toc131377732)

[5.2 Пользовательский интерфейс 27](#_Toc131377733)

[Заключение 36](#_Toc131377734)

[Список использованных источников 37](#_Toc131377735)

[Приложение А 38](#_Toc131377736)

[Приложение Б 41](#_Toc131377737)

[Приложение В 46](#_Toc131377738)

[Приложение Г 49](#_Toc131377739)

# Введение

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД [https://metanit.com/sql/sqlserver/1.1.php].

База данных по теме «Туристический оператор: клиенты» используется для автоматизации работы туристического оператора. А именно для хранения, редактирования, получения и отправки данных.

# 1 Проектирование базы данных

## 1.1 ER-диаграмма

В базе данных «Туристический оператор: клиенты» содержится следующий набор сущностей с присущими им атрибутами:

– Клиент (client): id\_клиент, фамилия, имя, отчество, серия паспорта, номер паспорта, номер телефона, номер визы;

– Туристическое агентство (travel agency): id\_тур\_агент, название, адрес, номер телефона;

– Путевка (travel package): id\_путевка, дата отправления, дата окончания, стоимость, оплачено;

– Трансфер (transfer): id\_трансфер, вид транспорта, адрес начала поездки, адрес конца поездки;

– Туристический оператор (tourist operator): id\_тур\_оператор, название, адрес, номер телефона;

– Сопровождающий (maintainer): id\_сопровождающий, фамилия, имя, отчество, номер телефона;

– Маршрут (route): id\_маршрут, название, точка отправления, точка прибытия, описание;

– Проживание (habitation): id\_проживание, вид, вид номера, номер комнаты;

– Экскурсия (excursion): id\_экскурсия, название, адрес, описание;

– Экскурсовод (guide): id\_экскурсовод, фамилия, имя, отчество, номер телефона;

– Питание (food): id\_питание, вид;

– Страховка (insurance): id\_страховка, вид.

Между этими сущностями организованы связи типа «один-ко-многим» и «многие-ко-многим», через ассоциативные таблицы.

Диаграмма в нотации Чена представлена на рисунке 1.

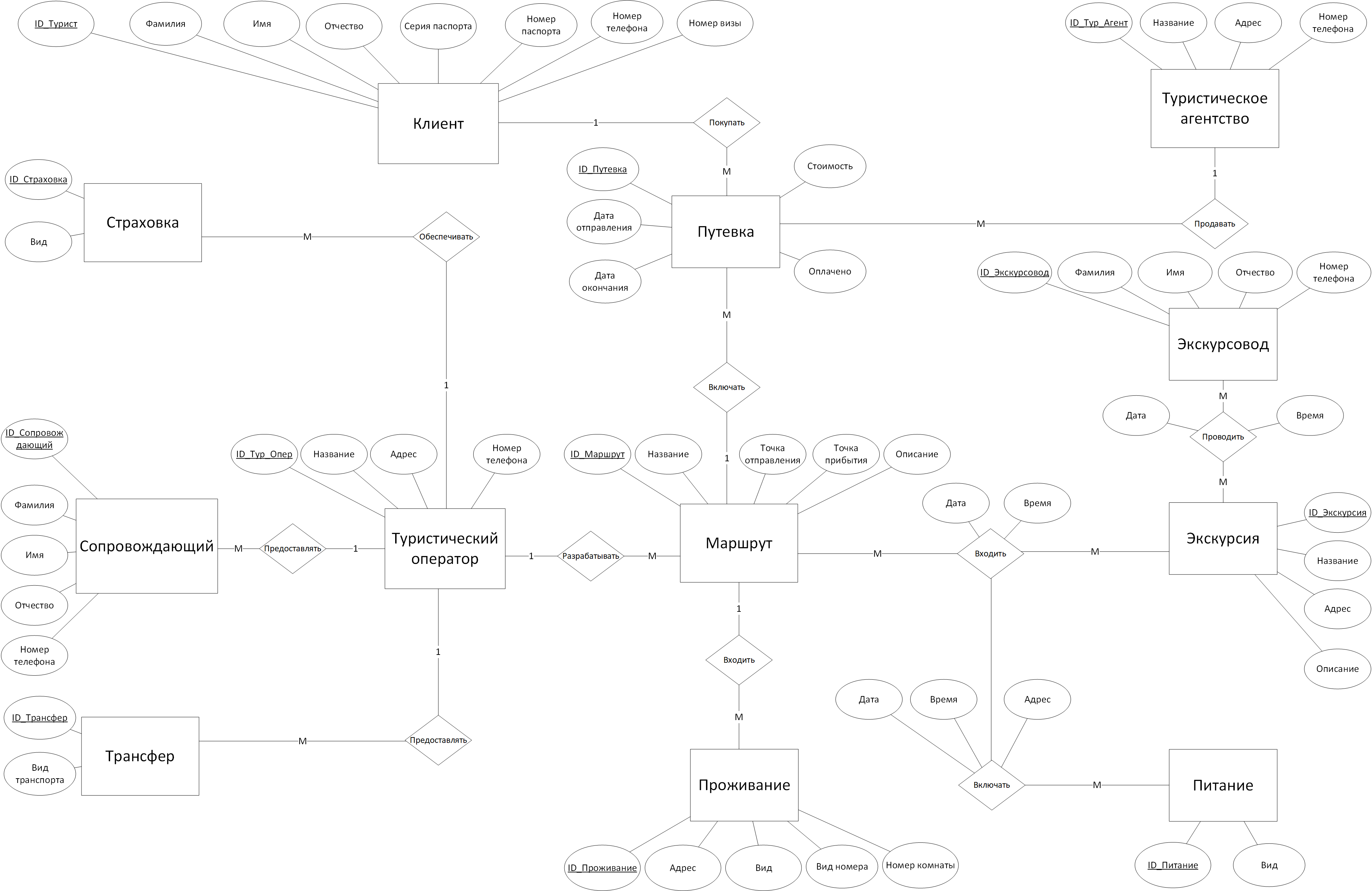


Рисунок 1 – Диаграмма в нотации Чена

Диаграмма в нотации Мартина представлена на рисунке 2.

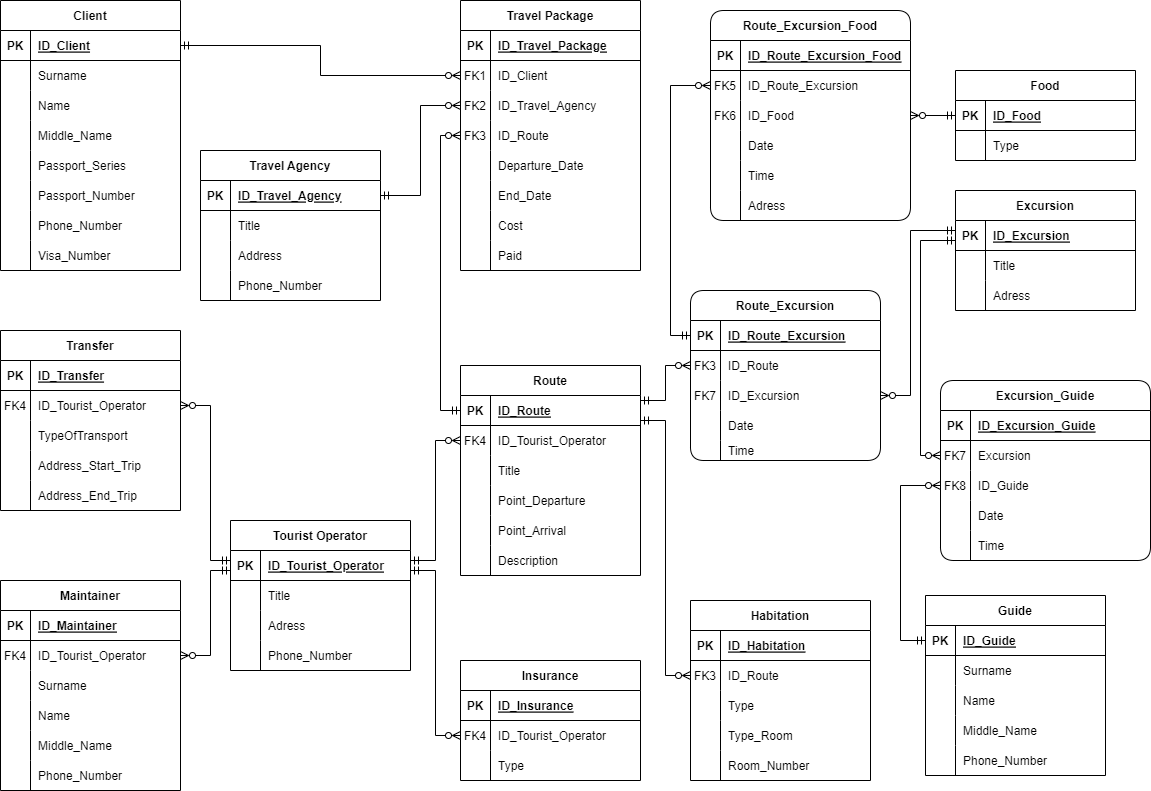


Рисунок 2 – Диаграмма в нотации Мартина

Результатом работы стала модель базы данных «Туристический оператор: клиенты».

# 2. Подготовка базы данных

## 2.1 Словарь данных

Словарь данных для создания базы данных представлен в таблице 1.

Таблица 1 – Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| client | | | | |
| Ключ | Поле | Тип | Обязательное  (да/нет) | Примечание |
| Первичный ключ | id\_client | Числовой | Да | Идентификатор клиента |
|  | surname\_client | Текстовый | Да | Фамилия клиента |
|  | name\_client | Текстовый | Да | Имя клиента |
|  | middle\_name\_client | Текстовый | Да | Отчество клиента |
|  | passport\_series\_client | Числовой | Да | Серия паспорта клиента |
|  | passport\_number\_client | Числовой | Да | Номер паспорта клиента |
|  | phone\_number\_client | Числовой | Да | Номер телефона клиента |
|  | visa\_number\_client | Числовой | Нет | Номер визы клиента |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| travel\_package | | | | | |
| Ключ | Поле | Тип | Обязательное  (да/нет) | | Примечание |
| Первичный ключ | id\_ travel\_package | Числовой | Да | | Идентификатор путевки |
| Внешний ключ | id\_client | Числовой | Да | | Внешний ключ на таблицу client |
| Внешний ключ | id\_travel\_agency | Числовой | Да | | Внешний ключ на таблицу travel\_agency |
| Внешний ключ | id\_route | Числовой | Да | | Внешний ключ на таблицу route |
|  | departure\_date\_travel\_package | Дата | Да | | Дата отправления |
|  | end\_date\_travel\_package | Дата | Да | | Дата окончания путевки |
|  | cost\_travel\_package | Денежный | Да | | Стоимость за путевку |
|  | paid\_travel\_package | Денежный | Да | | Оплачено за путевку |
| travel\_agency | | | | | |
| Ключ | Поле | Тип | | Обязательное  (да/нет) | Примечание |
| Первичный ключ | id\_travel\_agency | Числовой | | Да | Идентификатор туристического агентства |
|  | title\_travel\_agency | Текстовый | | Да | Название туристического агентства |
|  | address\_travel\_agency | Текстовый | | Да | Адрес туристического агентства |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | phone\_number\_travel\_agency | Числовой | Да | Номер телефона туристического агентства |
| route | | | | |
| Первичный ключ | id\_route | Числовой | Да | Идентификатор маршрута |
| Внешний ключ | id\_tourist\_operator | Числовой | Нет | Внешний ключ на таблицу tourist\_operator |
|  | title\_route | Текстовый | Да | Название маршрута |
|  | point\_departure\_route | Текстовый | Да | Точка отправления |
|  | point\_arrival\_route | Текстовый | Да | Точка прибытия |
|  | description\_route | Текстовый | Да | Описание маршрута |
| tourist\_operator | | | | |
| Первичный ключ | id\_tourist\_operator | Числовой | Да | Идентификатор туристического оператора |
|  | title\_tourist\_operator | Текстовый | Да | Название туристического оператора |
|  | adress\_tourist\_operator | Текстовый | Да | Адрес туристического оператора |
|  | phone\_number\_tourist\_operator | Числовой | Да | Номер телефона туристического оператора |
| transfer | | | | |
| Первичный ключ | id\_transfer | Числовой | Да | Идентификатор трансфера |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Внешний ключ | id\_tourist\_operator | Числовой | Нет | Внешний ключ на таблицу tourist\_operator |
|  | type\_transport\_transfer | Текстовый | Да | Вид транспорта |
|  | address\_start\_trip\_transfer | Текстовый | Да | Адрес начала поездки |
|  | address\_end\_trip\_transfer | Текстовый | Да | Адрес конца поездки |
| maintainer | | | | |
| Первичный ключ | id\_maintainer | Числовой | Да | Идентификатор сопровождающего |
| Внешний ключ | id\_tourist\_operator | Числовой | Нет | Внешний ключ на таблицу tourist\_operator |
|  | surname\_maintainer | Текстовый | Да | Фамилия сопровождающего |
|  | name\_maintainer | Текстовый | Да | Имя сопровождающего |
|  | middle\_name\_maintainer | Текстовый | Да | Отчество сопровождающего |
|  | phone\_number\_maintainer | Числовой | Да | Номер телефона сопровождающего |
| insurance | | | | |
| Первичный ключ | id\_insurance | Числовой | Да | Идентификатор страховки |
| Внешний ключ | id\_tourist\_operator | Числовой | Нет | Внешний ключ на таблицу tourist\_operator |
|  | type\_insurance | Текстовый | Да | Вид страховки |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| route\_excursion | | | | |
| Первичный ключ | id\_route\_excursion | Числовой | Да | Идентификатор таблицы |
| Внешний ключ | id\_route | Числовой | Да | Внешний ключ на таблицу route |
| Внешний ключ | id\_excursion | Числовой | Да | Внешний ключ на таблицу excursion |
|  | date\_route\_excursion | Дата | Да | Дата экскурсии |
|  | time\_route\_excursion | Время | Да | Время экскурсии |
| excursion | | | | |
| Первичный ключ | id\_excursion | Числовой | Да | Идентификатор экскурсии |
|  | title\_excursion | Текстовый | Да | Название экскурсии |
|  | adress\_excursion | Текстовый | Да | Адрес экскурсии |
| route\_excursion\_food | | | | |
| Первичный ключ | id\_route\_excursion\_food | Числовой | Да | Идентификатор таблицы |
| Внешний ключ | id\_route\_excursion | Числовой | Да | Внешний ключ на таблицу route\_excursion |
| Внешний ключ | id\_food | Числовой | Да | Внешний ключ на таблицу food |
|  | date\_route\_excursion\_food | Дата | Да | Дата питания |
|  | time\_route\_excursion\_food | Время | Да | Время питания |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | adress\_route\_excursion\_food | Текстовый | Да | Адрес питания |
| food | | | | |
| Первичный ключ | id\_food | Числовой | Да | Идентификатор питания |
|  | type\_food | Текстовый | Да | Вид питания |
| excursion\_guide | | | | |
| Первичный ключ | id\_excursion\_guide | Числовой | Да | Идентификатор таблицы |
| Внешний ключ | id\_excursion | Числовой | Да | Внешний ключ на таблицу excursion |
| Внешний ключ | id\_guide | Числовой | Да | Внешний ключ на таблицу guide |
|  | date\_excursion\_guide | Дата | Да | Дата экскурсии |
|  | time\_excursion\_guide | Время | Да | Время экскурсии |
| guide | | | | |
| Первичный ключ | id\_guide | Числовой | Да | Идентификатор экскурсовода |
|  | surname\_guide | Текстовый | Да | Фамилия экскурсовода |
|  | name\_guide | Текстовый | Да | Имя экскурсовода |
|  | middle\_name\_guide | Текстовый | Да | Отчество экскурсовода |
|  | phone\_number\_guide | Числовой | Да | Номер телефона экскурсовода |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| habitation | | | | |
| Первичный ключ | id\_habitation | Числовой | Да | Идентификатор проживания |
| Внешний ключ | id\_route | Числовой | Да | Внешний ключ на таблицу route |
|  | type\_habitation | Текстовый | Да | Вид проживания |
|  | type\_room\_habitation | Текстовый | Да | Вид номера |
|  | room\_number\_habitation | Числовой | Да | Номер комнаты |

Итоговая диаграмма базы данных, сгенерированная после создания всех таблиц представлена на рисунке 3.

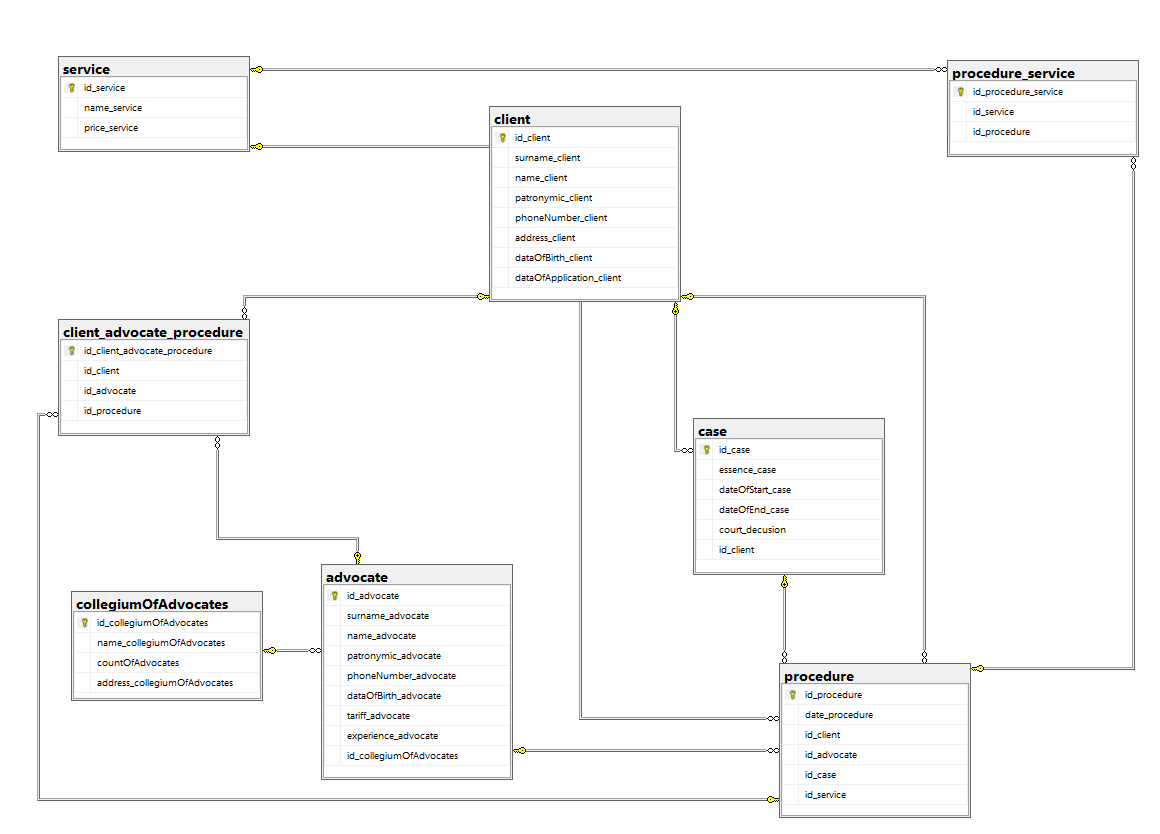


Рисунок 3 – Итоговая диаграмма базы данных

Код создания таблиц представлен в Приложении А. Код заполнения базы данных представлен в приложении Б.

## 2.2 Нормализация полученных отношений

Разработанная схема базы данных находится в (во):

– первой нормальной форме, так как в качестве доменов выступают только скалярные значения, и информация в таблице не дублируется. Во всех таблицах есть идентификатор id в качестве первичного ключа.

– второй и третьей нормальных формах, каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Схема базы данных находится в нормальной форме Бойса-Кода.

## 2.3 Процедуры, функции, представления, триггеры

В базе данных реализовано хранимая процедура, скалярная функция, представления.

Хранимая процедура имеет название InfoOfAll. Она предназначена для того, чтобы вывести в одну таблицу информацию о клиентах и делах с определенной сутью дела. Данная процедура путем объединения таблиц client и case облегчает поиск информации. На выходе формируется выборка с такими полями как name\_client, surname\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion

Пример работы хранимой процедуры представлен на рисунке 4.

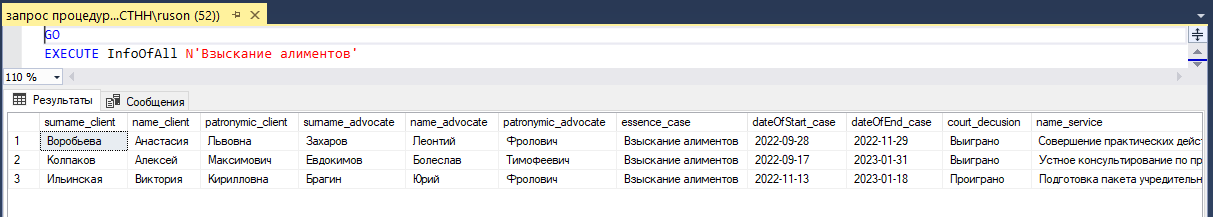


Рисунок 4 – Пример работы хранимой процедуры

В базе данных реализована скалярная функция с названием GSProcedure, которая рассчитывает сумму стоимости тарифа адвоката и стоимости услуги. Эти данные берутся из таблиц advocate и service. Выходным параметром данной функции является числовое значение, которое будет определять стоимость процедуры.

Пример работы скалярной функции представлен на рисунке 5.

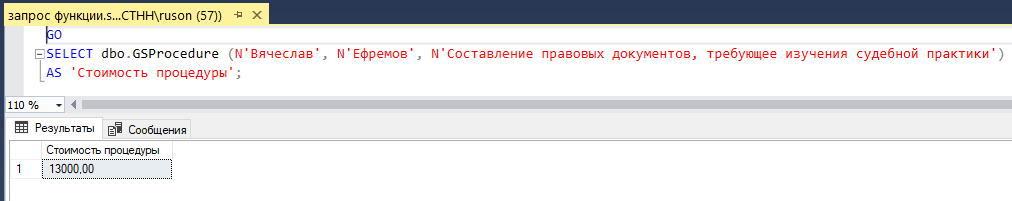


Рисунок 5 – Пример работы скалярной функции

В базе данных реализованы представления. Они предназначены для более удобной работы с данными.

Представление P1 выводит в одну таблицу информацию о клиенте, адвокате (который работает с данным клиентом) и процедуре (реализованной для данного клиента данным адвокатом). На выходе формируется таблица с такими полями как: name\_client, surname\_client, patronymic\_client, phoneNumber\_client, name\_advocate, surname\_advocate, patronymic\_advocate, phoneNumber\_advocate, p.id\_procedure, date\_procedure.

Пример работы представления P1 представлен на рисунке 6.

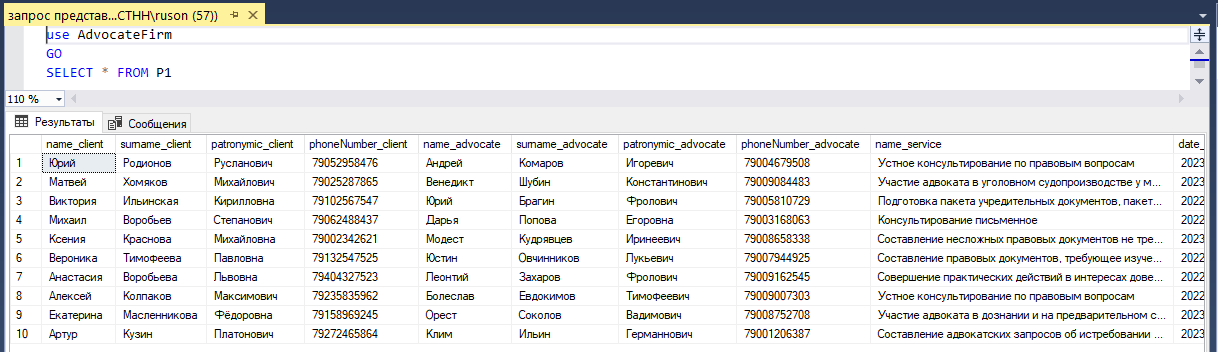


Рисунок 6 – Пример работы представления P1

Представление P2 выводит в одну таблицу информацию о процедуре, и услуге (которая была выполнена в процессе процедуры). На выходе формируется таблица с такими полями как: p.id\_procedure, date\_procedure, name\_service, price\_service.

Пример работы представления P1 представлен на рисунке 7.

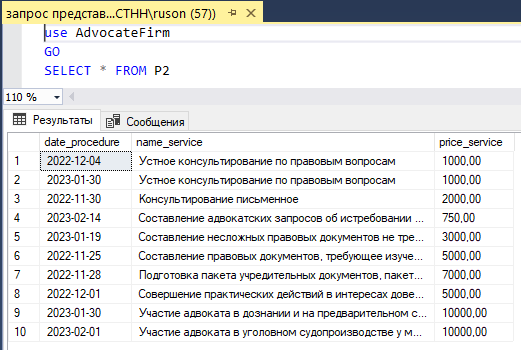


Рисунок 7 – Пример работы представления P2

Код создания хранимой процедуры, скалярной функции и представлений представлен в приложении В.

В базе данных реализованы следующие триггеры: Delete\_Client, Insert\_Client, Update\_Client.

Первый триггер не позволяет изменять данные в таблице service. Второй триггер при изменении данных в таблице advocate выводит сообщение: «Запись, в таблице advocate изменена». Третий триггер при изменении данных в таблице client выводит сообщение: «Запись, в таблице client изменена».

Созданные триггеры представлены на рисунке 8.

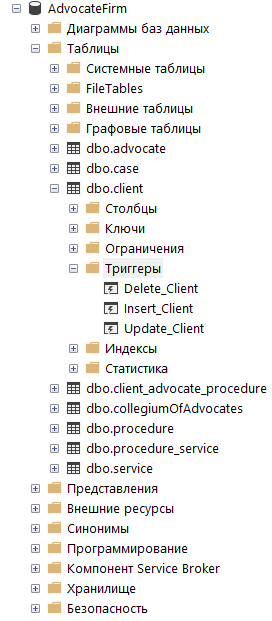


Рисунок 8 – Созданные триггеры в обозревателе объектов

Код создания триггеров, представлен в приложении В.

# 3. Резервное копирование и восстановление

Чтобы выполнить резервное копирование базы данных с использованием командного файла, нужно написать SQL-скрипт, который реализует резервное опирование базы, затем сохранить данный скрипт в файле с расширением (.sql). Скрипт для резервного копирования представлен в листинге 1.

Листинг 1 - SQL-скрипт для резервного копирования базы данных

BACKUP DATABASE [AdvocateFirm]

TO DISK = N'C:\Users\ruson\AdvocateFirm.bak', DISK = N'C:\Users\ruson\Desktop\Юра\KP\af.bak'

WITH NOFORMAT, NOINIT, NAME = N'AdvocateFirm-Полная База данных Резервное копирование', SKIP, NOREWIND, NOUNLOAD, STATS = 10

GO

Для запуска данного скрипта необходимо с помощью любого текстового редактора создать файл с расширением (.bat), внутри которого прописать код, представленный в листинге 2.

Листинг 2 - Код командного файла для запуска скрипта для резервного копирования базы данных

sqlcmd -s <(localdb)\MSSQLLocalDB> -I C:\Users\ruson\Desktop\Юра\KP\backup.sql

Теперь при запуске файла буде создаватся резервная копия базы данных в указанной папке.

Также был создан SQL-скрипт и командный файл для запуска восстановления базы данных из резервной копии. SQL-скрипт для восстановления представлен в листинге 3.

Листинг 3 - SQL-скрипт для восстановления базы данных из резервной копии

USE [master]

RESTORE DATABASE [AdvocateFirm]

FROM DISK = N'C:\Users\ruson\AdvocateFirm.bak', DISK = N'C:\Users\ruson\Desktop\Юра\KP\af.bak'

WITH FILE = 1, NOUNLOAD, STATS = 5

GO

Код командного файла представлен в листинге 4.

Листинг 4 - Код командного файла для запуска скрипта для восстановления базы данных из резервной копии

sqlcmd -s <(localdb)\MSSQLLocalDB> -I C:\Users\ruson\Desktop\Юра\KP\res.sql

И так, с помощью разработанных скриптов и командных файлов стала доступна возможность создавать резервную копию базы данных и восстанавливать ее.

# 4. УПРАВЛЕНИЕ ДОСТУПОМ

Для управления доступом в базе данных с помощью sql-скрипта создано две роли:

– ADMIN, которая не ограничена в работе с данными в базе данных, то есть может добавлять изменять и удалять данные.

– USER, которая может лишь просматривать данные в таблицах базы данных. Соответственно, для этой роли ограничена возможность добавления, удаления и редактирования данных.

Для каждой роли были созданы пользователи: ADMIN – Administrator, USER – Polzovatel. Каждый пользователь имеет свой логин и пароль. Код создания ролей пользователей представлен в приложении В.

Созданные роли и пользователи в базе данных показаны на рисунке 9.

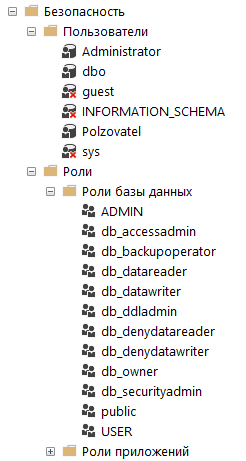


Рисунок 9 – Cозданные роли ADMIN и USER и пользователи Administrator и Polzovatel

Пользователи с ролью ADMIN представлены на рисунке 10.

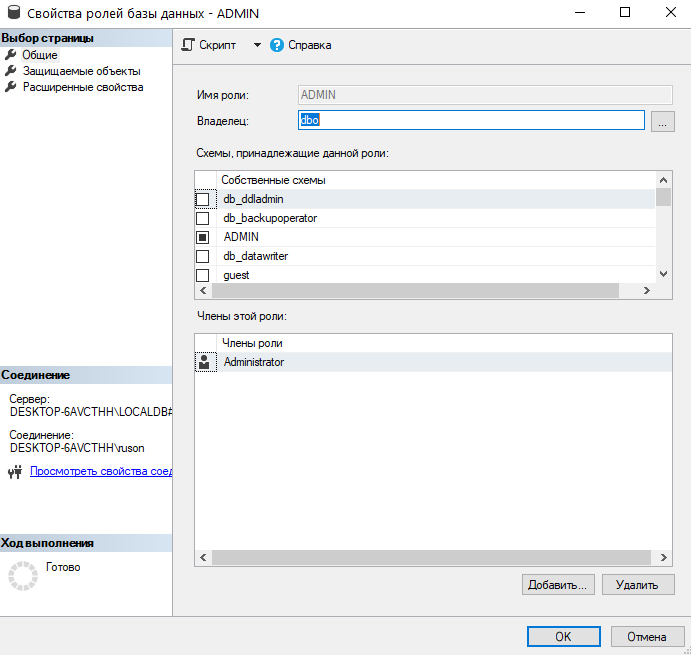


Рисунок 10 – Свойства роли ADMIN

Права роли ADMIN представлены на рисунке 11.

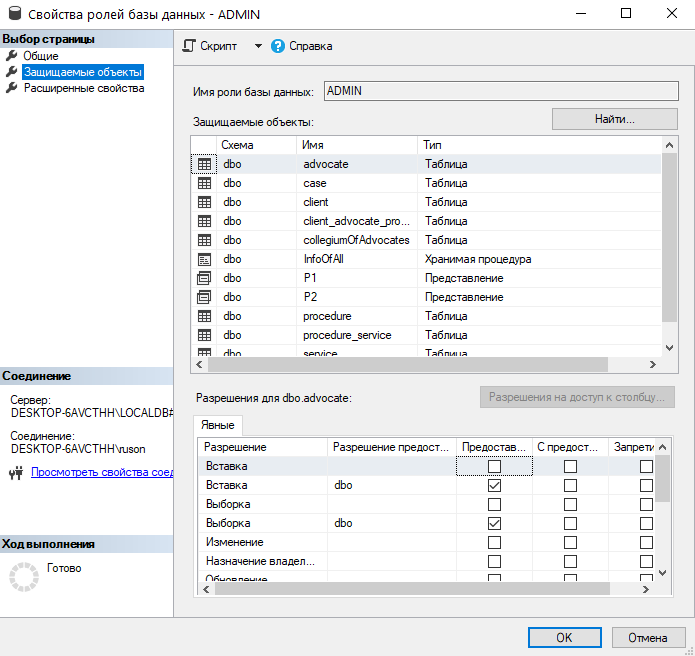


Рисунок 11 – Свойства роли ADMIN

Пользователи с ролью USER представлены на рисунке 12.

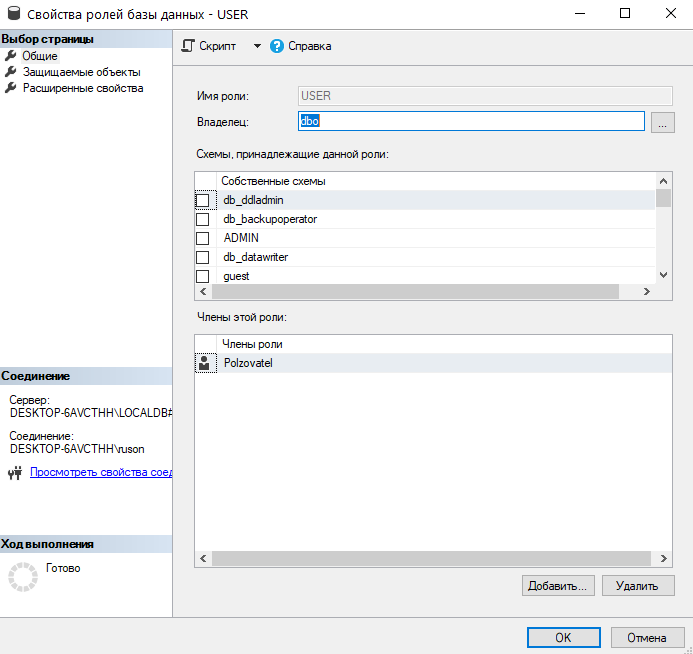


Рисунок 12 – Cвойства роли USER

Права роли USER представлены на рисунке 13.

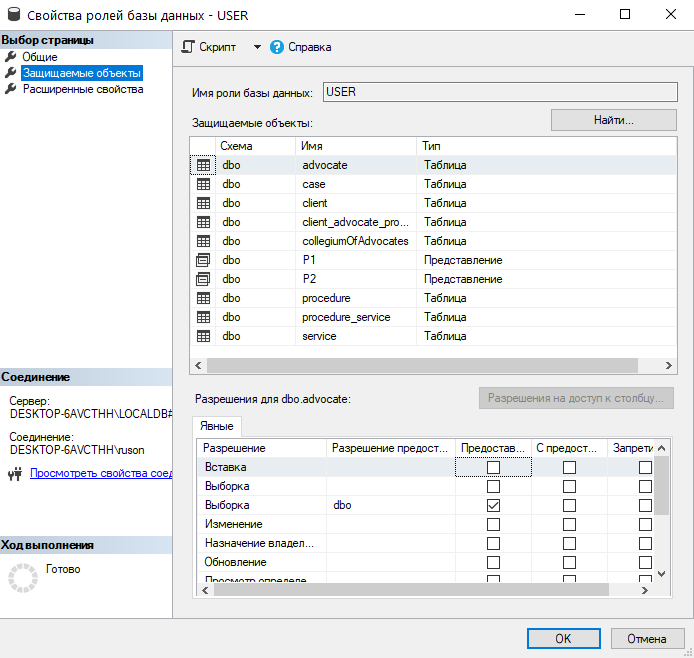


Рисунок 13 – Свойства роли USER

SQL-скрипт, выполняющий создание ролей и пользователей представлен в приложении В.

# 5. Разработка клиентского приложения

## 5.1 Структура приложения

Диаграмма, отражающая отношение между актерами и прецедентами, также позволяющая описать систему на концептуальном уровне представлена на рисунке 14.

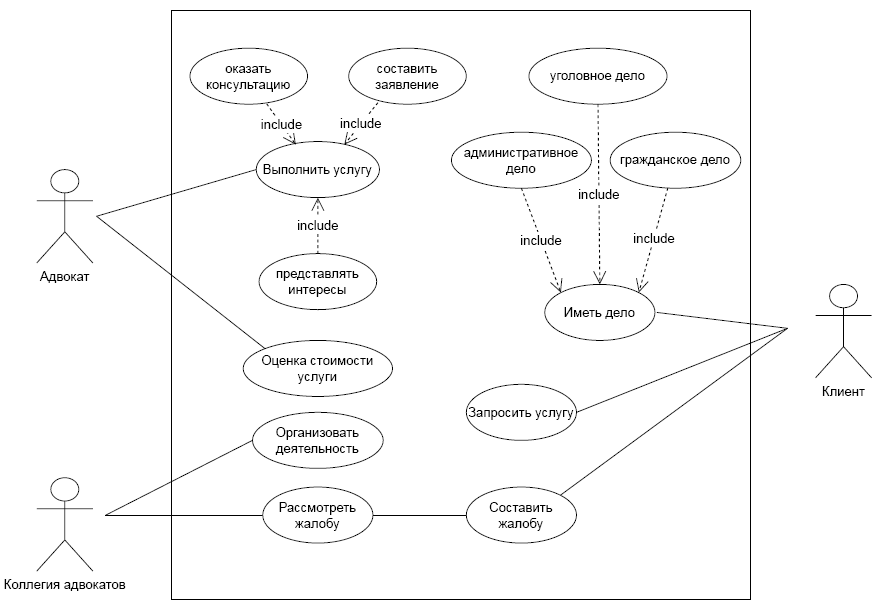


Рисунок 14 – Диаграмма прецедентов

Создано приложение, позволяющее получать доступ к данным в таблицах базы данных «AdvocateFirm» с использованием нескольких форм [4][5].

При запуске приложения пользователь попадает на форму авторизации, интерфейс которой представлен на рисунке 15.

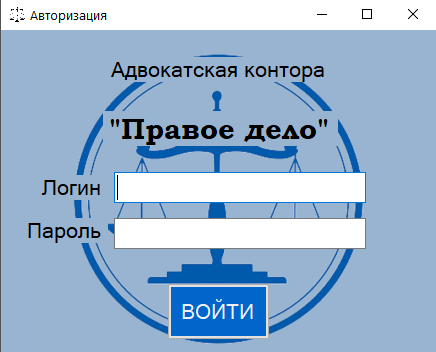


Рисунок 15 – Форма авторизации

В поля «Логин» и «Пароль» пользователь вводит соответствующие данные для попытки авторизации. Если введенные данные окажутся достоверными, то пользователь увидит сообщение «Подключение установлено» и получит такой доступ к приложению, какой предусматривает роль. В противном случае, появится сообщение об ошибке входа.

При входе под пользователем, которому назначена роль ADMIN, появится форма администратора, представленная на рисунке 16.

Она предлагает следующие возможности: просмотр (при помощи переключения страниц), добавление, редактирование и удаление информации; использование хранимой процедуры и скалярной функции.

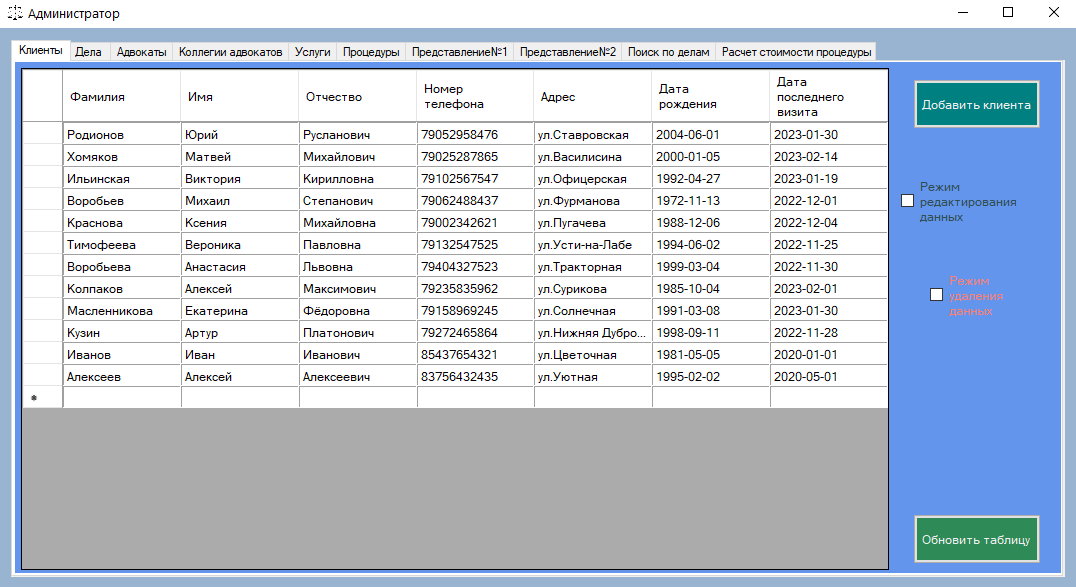


Рисунок 16 – Форма администратора

Данная форма позволяет просматривать данные в таблицах, при помощи переключения страниц; добавлять, изменять и удалять данные; использовать хранимую процедуру и скалярную функцию.

При нажатии на кнопку «Добавить клиента» появляется форма, представленная на рисунке 17.

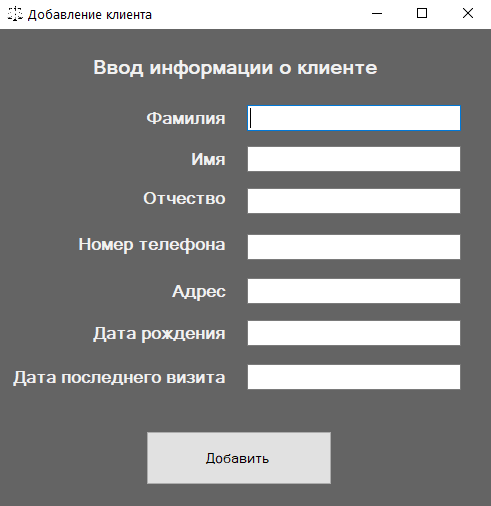


Рисунок 17 – Форма добавления клиента

Для добавления клиента необходимо ввести данные в соответствующие поля textBox, после чего нажать на кнопку «Добавить». Для отображения новых данных в таблице необходимо нажать на кнопку «Обновить таблицу».

Чтобы изменить данные в таблице нужно активировать «Режим редактирования данных» после чего сделать двойной щелчок левой кнопкой мыши по нужной записи, появиться форма, представленная на рисунке 18.

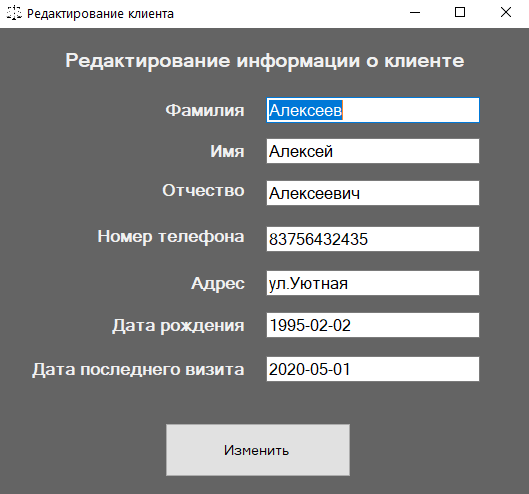


Рисунок 18 – Форма редактирования клиента

В соответствующих полях формы будет отражаться информация о выбранной записи. Необходимо ввести новые данные, после чего нажать на кнопку «Изменить». Для отображения новых данных в таблице необходимо нажать на кнопку «Обновить таблицу».

Для удаления данных в таблице нужно активировать «Режим удаления данных» после чего сделать двойной щелчок левой кнопкой мыши по нужной записи, появиться сообщение, представленное на рисунке 19.

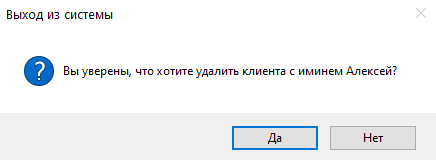


Рисунок 19 – Проверка на осознанность действия пользователя

При нажатии на кнопку «Да», запись будет удалена. Для отображения новых данных в таблице необходимо нажать на кнопку «Обновить таблицу».

При попытке одновременного использования режимов редактирования и удаления появиться сообщение об ошибке, представленное на рисунке 20.

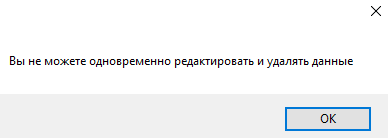


Рисунок 20 – Сообщение об ошибке

Пример работы с хранимой процедурой, в форме администратора представлен на рисунке 21.

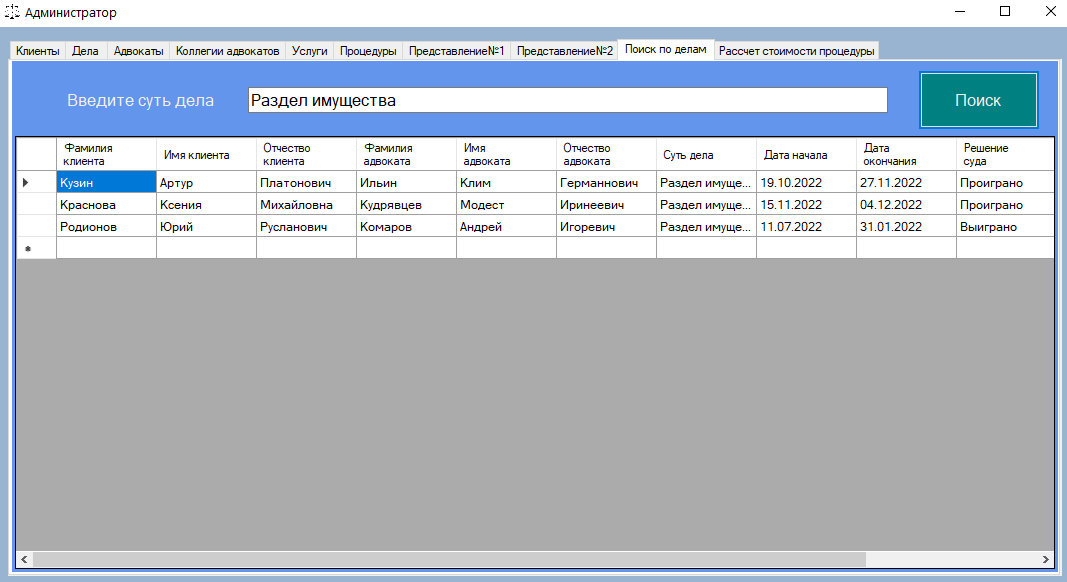


Рисунок 21 – Поиск по делам

Пример работы со скалярной функцией, в форме администратора представлен на рисунке 22.

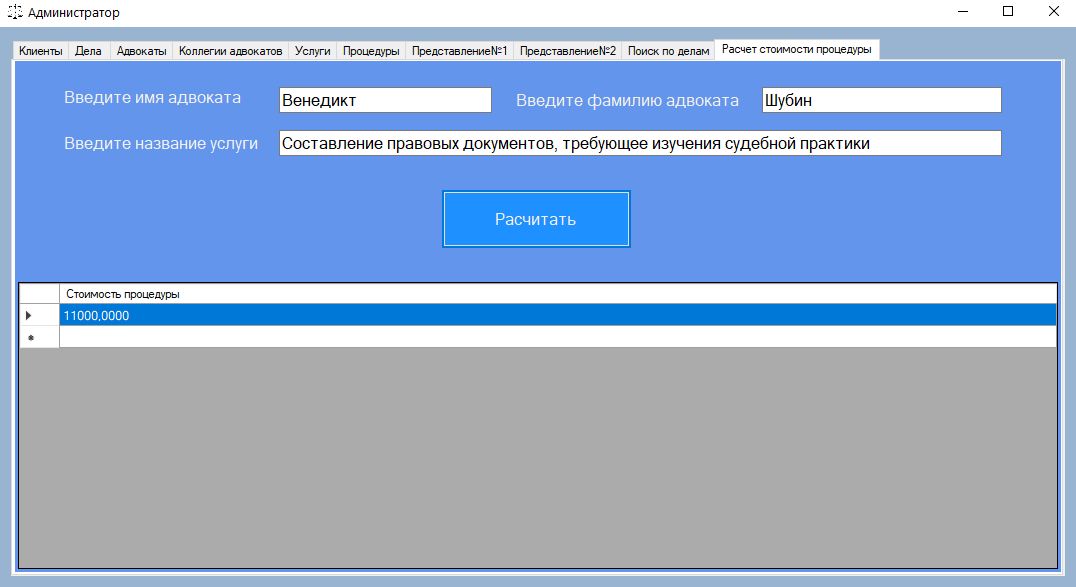


Рисунок 22 – Расчет стоимости процедуры

При входе под пользователем, которому назначена роль USER, появится форма администратора, представленная на рисунке 24.

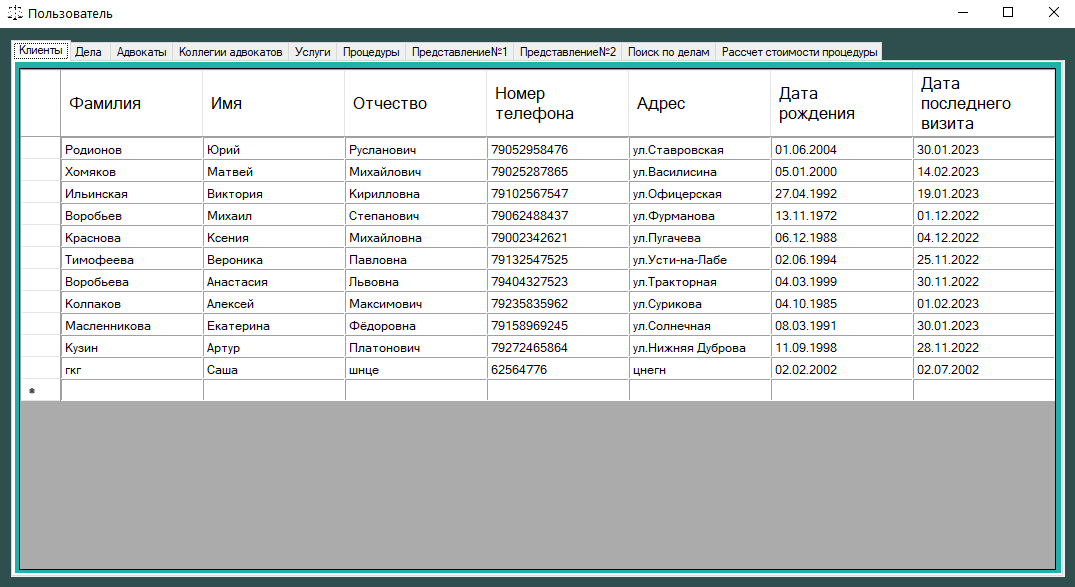


Рисунок 24 – Форма пользователя

Данная форма позволяет только просматривать данные в таблицах, при помощи переключения страниц; использовать хранимую процедуру и скалярную функцию.

Пример работы с хранимой процедурой, в форме пользователя представлен на рисунке 25.

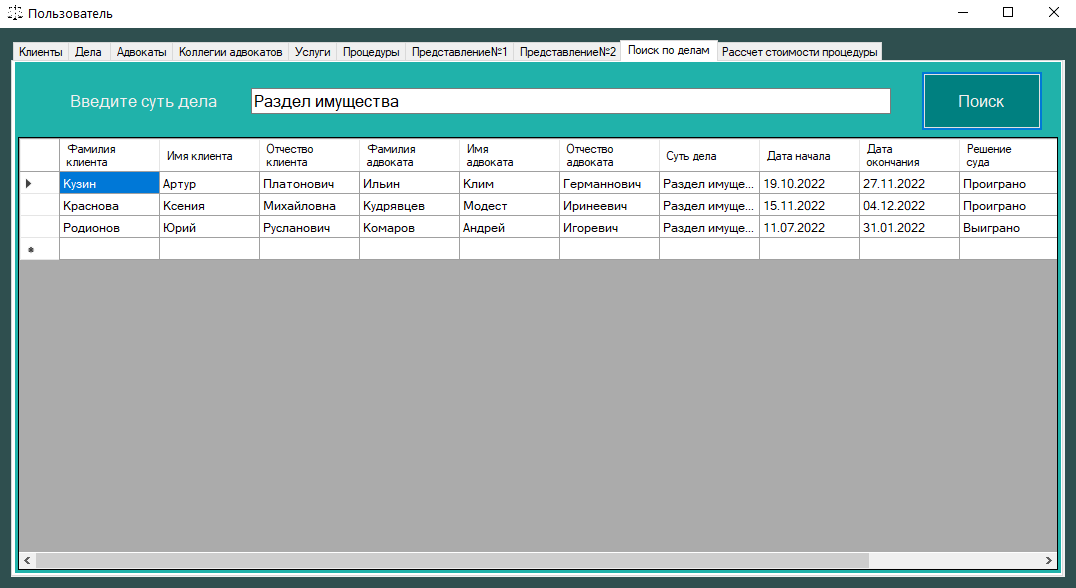


Рисунок 25 – Поиск по делам

Пример работы со скалярной функцией, в форме пользователя представлен на рисунке 26.

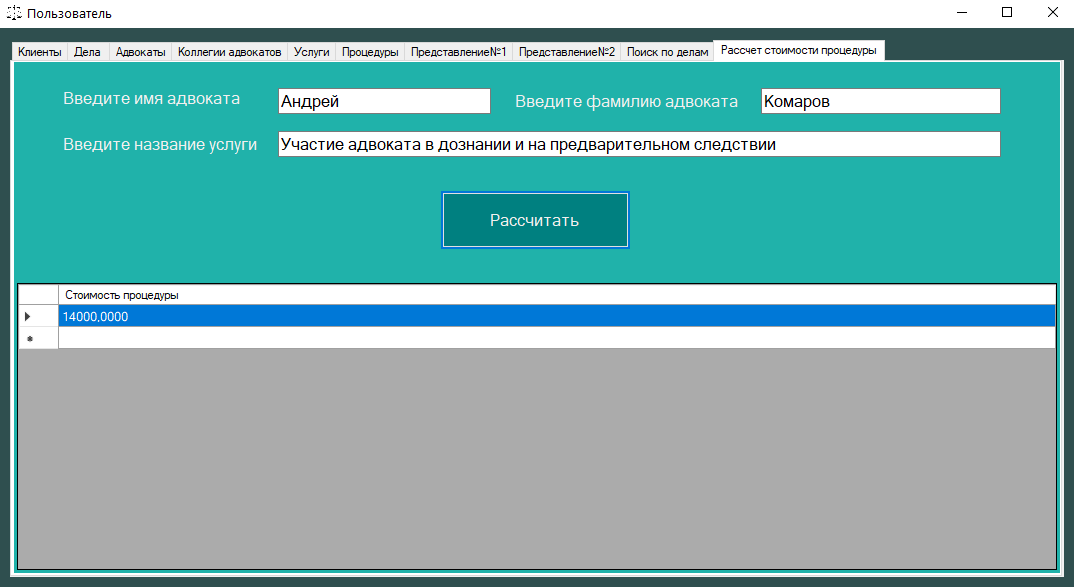


Рисунок 26 – Расчет стоимости процедуры

Код реализованных форм клиентского приложения представлен в приложении Г.

## 5.2 Пользовательский интерфейс

Пользовательский интерфейс состоит из графических форм, которые необходимы для работы с данными.

Для того, чтобы начать работу с данными, сначала необходимо пройти авторизацию, для этого была реализована форма авторизации, представленная на рисунке 27. В ней необходимо ввести данные, которые могут соответствовать либо администратору, либо обычному пользователю, после чего нажать на кнопку «ВОЙТИ».

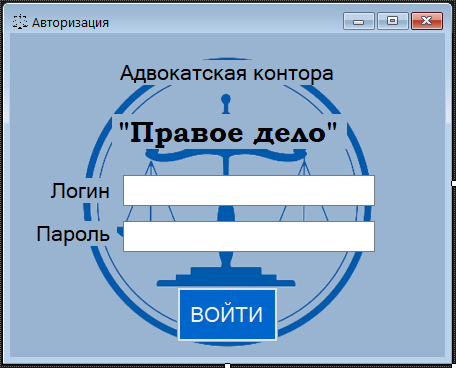


Рисунок 27 – Форма авторизации

При прохождении этапа авторизации, пользователь попадает на одну из двух основных графических форм, которые служат для взаимодействия с данными, это «Администратор» и «Пользователь», которые представлены на рисунках 28 – 29 соответственно. В них реализована возможность выбора конкретной таблицы для просмотра, а также возможность взаимодействия с хранимой процедурой и скалярной функцией.

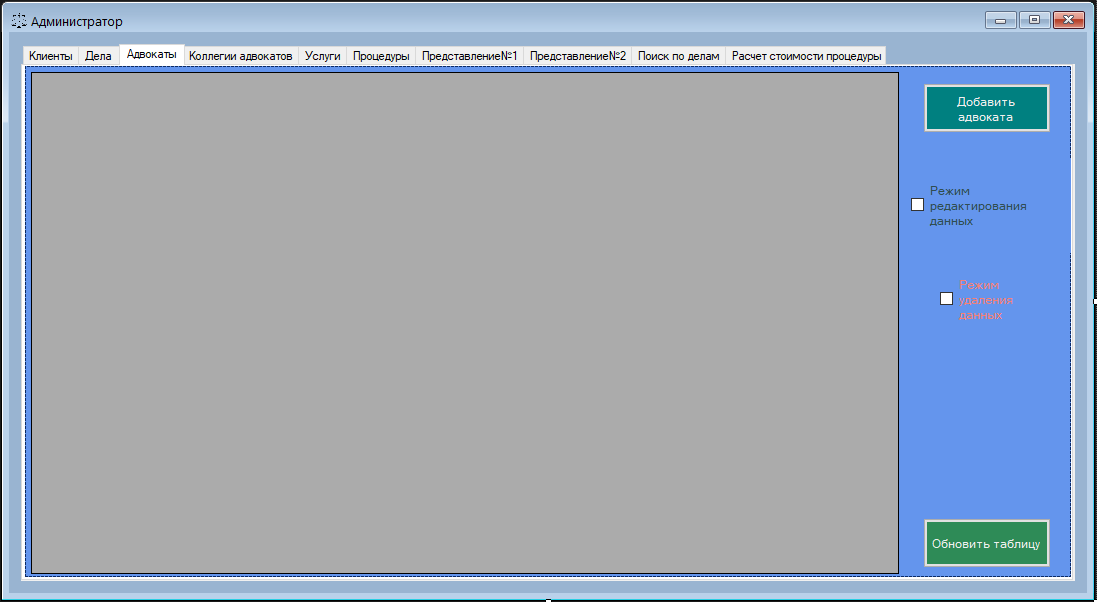


Рисунок 28 – Форма администратора

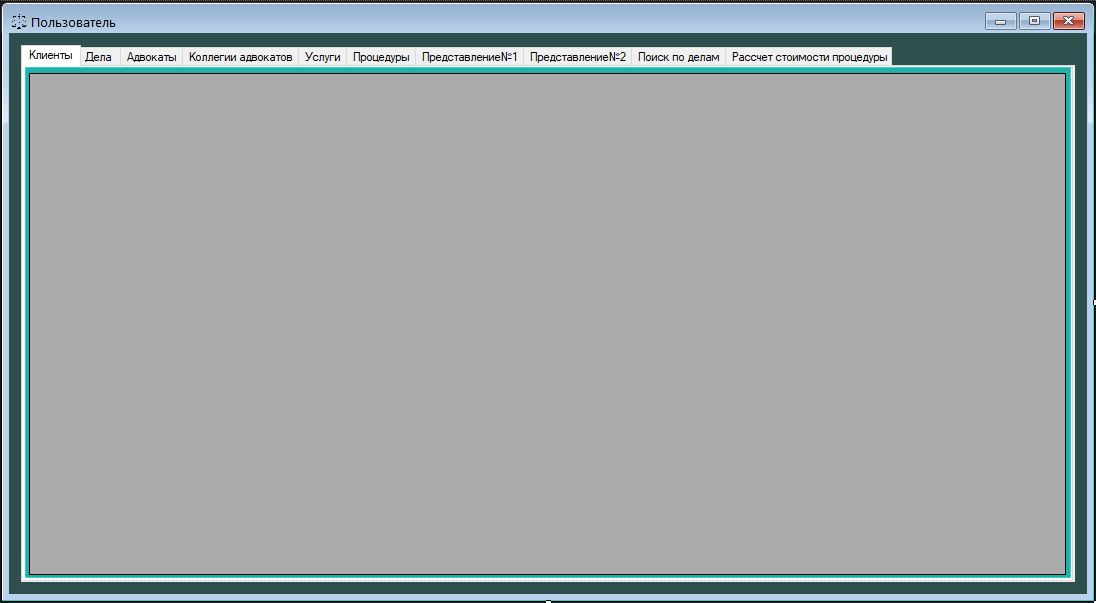


Рисунок 29 – Форма пользователя

Пользователь, попавший после авторизации на форму «Пользователь» ограничивается лишь просмотром данных, представленных в таблицах, а также возможностью использования хранимой процедуры и скалярной функции.

В свою очередь пользователь, попавший после авторизации на форму «Администратор» обладает такими возможностями как: просмотр, добавление, редактирование и удаление данных. Удаление данных реализовано двойным нажатием на запись в таблице, при этом должен быть включен режим удаления данных.

Добавление данных в таблицы реализовано при помощи кнопок, которые называются следующим образом «Добавить (название добавляемого элемента)», при нажатии на которые появляются формы, представленные на рисунках 30 – 35.

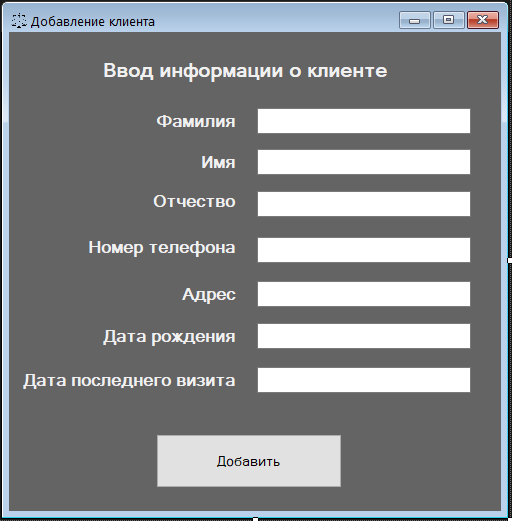


Рисунок 30 – Форма добавления клиента

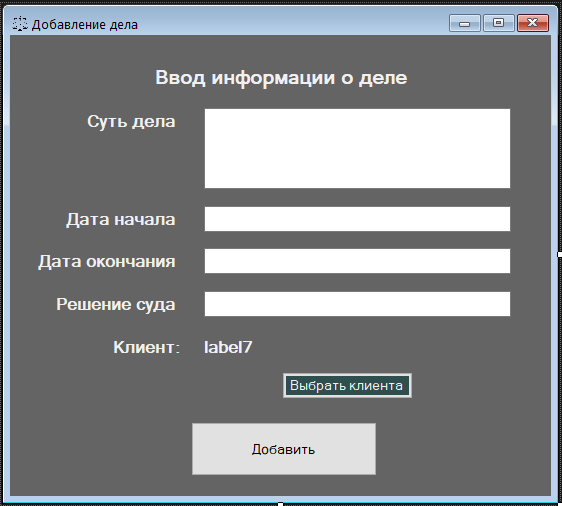


Рисунок 31 – Форма добавления дела

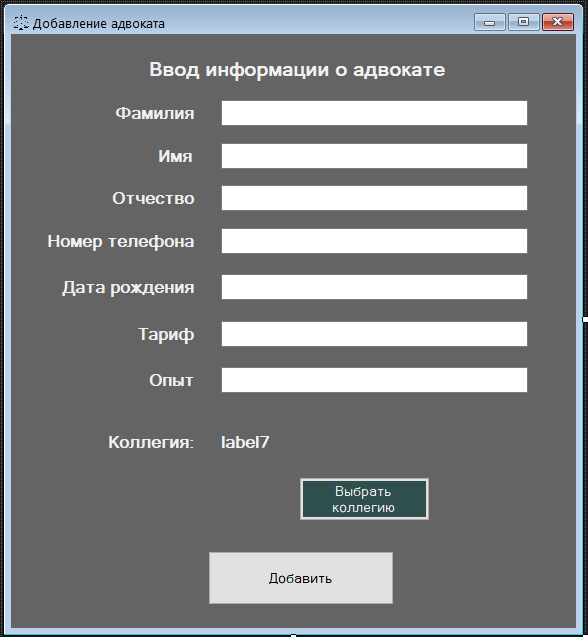


Рисунок 32 – Форма добавления адвоката

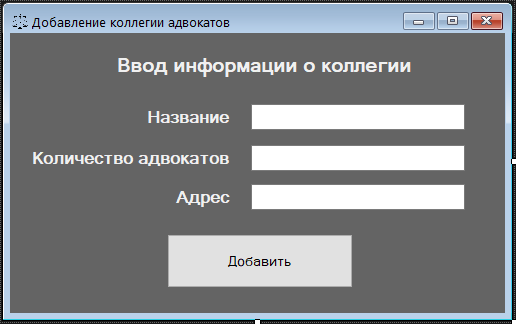


Рисунок 33 – Форма добавления коллегии адвокатов

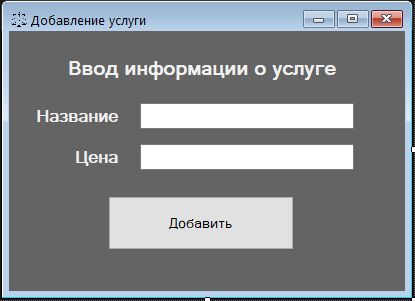


Рисунок 34 – Форма добавления услуги

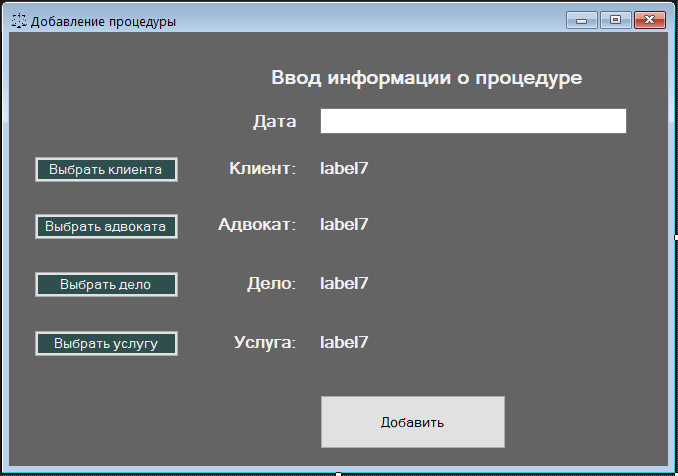


Рисунок 35 – Форма добавления процедуры

Редактирование данных в таблицы реализовано при помощи двойного нажатия на запись в таблице, после чего появляются следующие формы, которые представлены на рисунках 36 – 42. При этом должен быть активирован режим редактирования данных.

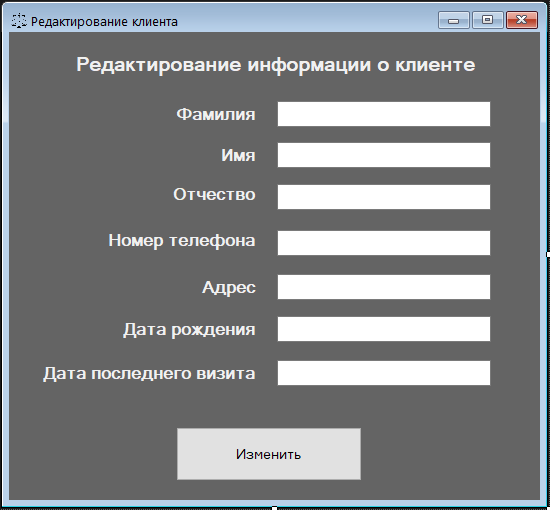


Рисунок 36 – Форма редактирования клиента

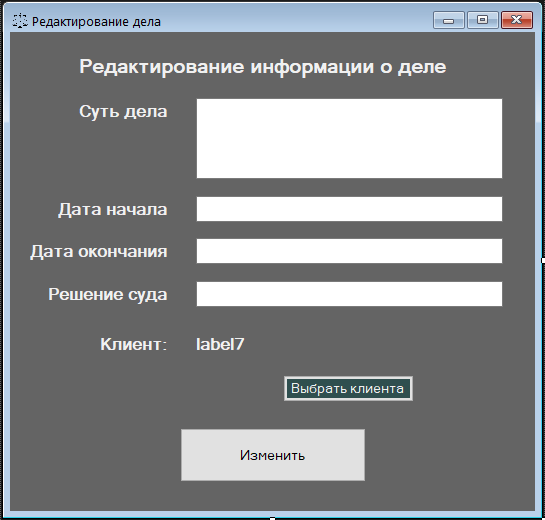


Рисунок 37 – Форма редактирования дела

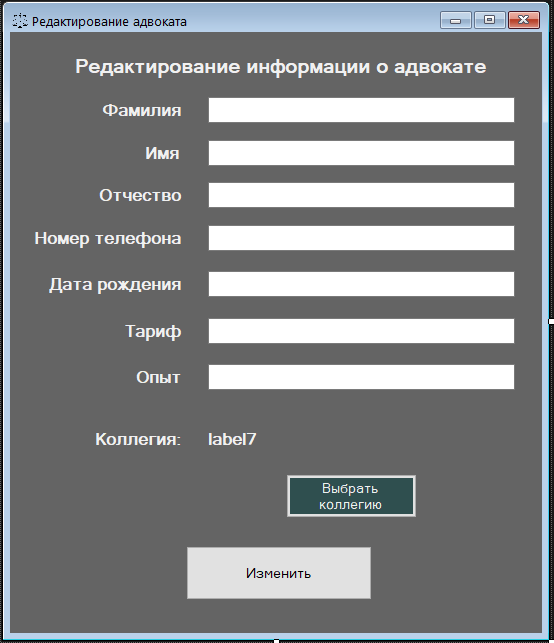


Рисунок 38 – Форма редактирования адвоката

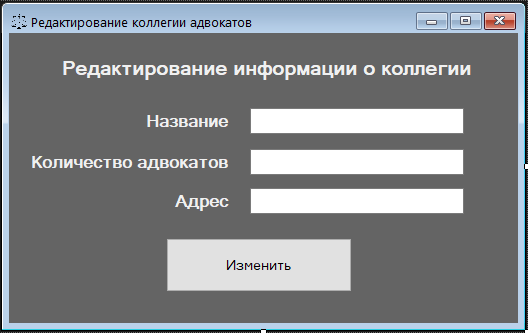


Рисунок 40 – Форма редактирования коллегии адвокатов

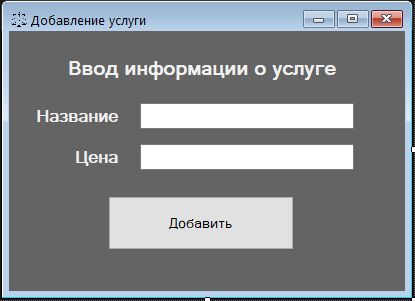


Рисунок 41 – Форма редактирования услуги

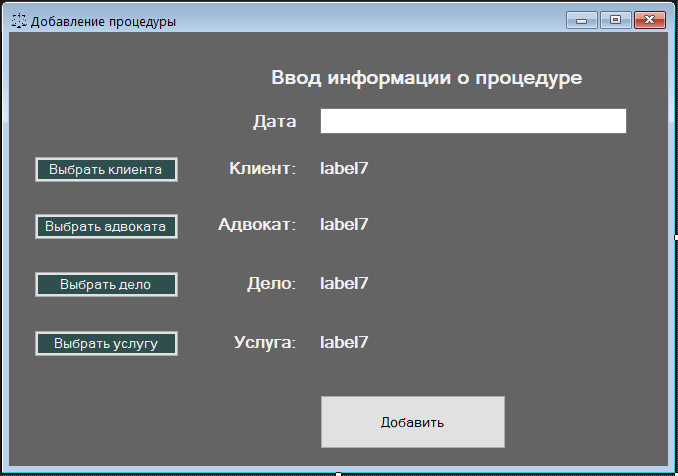


Рисунок 42 – Форма редактирования процедуры

В итоге было разработано приложение с возможностью взаимодействия с данными в базе данных «Адвокатская контора», а также разделением пользователей на «обычных» пользователей и администраторов.

# Заключение

В ходе выполнения курсовой работы была изучена и проанализирована учебно-методическая документация, учебная литература и интернет-ресурсы по заданной теме.

В результате выполнения работы была создана база данных для работников адвокатской конторы, а также приложение, обеспечивающее взаимодействие с базой данных.

Для этого были выполнены такие задачи как: разработка ER-диаграммы; проектирование базы дынных, а также создание хранимой процедуры, скалярной функции и триггеров; реализация клиентского приложения с графическим интерфейсом.

# Список использованных источников

1) Microsoft SQL Server — особенности и установка СУБД MS SQL // OTUS URL: <https://otus.ru/nest/post/1566/> (дата обращения 02.04.2023).

2) А. Бьюли Изучаем SQL. издательство Символ 2007. – 19с.

3) СЕРВЕР БАЗ ДАННЫХ SQL SERVER // Институт открытых программ дополнительного образования URL: <https://hsbi.hse.ru/articles/server-baz-dannykh-sql-server/> (дата обращения 02.04.2023).

4) Entity Framework Core подключение к MS SQL // InterestPrograms URL:<https://www.interestprograms.ru/source-codes-podklyuchenie-k-baze-ms-sql-cherez-entity-framework-core> (дата обращения 02.04.2023).

5) Все операции с БД в графическом приложении // METANIT URL: <https://metanit.com/sharp/adonet/3.5.php> (дата обращения 02.04.2023).

# Приложение А

Код создания таблицы client

USE [AdvocateFirm]

GO

/\*\*\*\*\*\* Object: Table [dbo].[client] Script Date: 11.02.2023 17:13:32 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[client](

[id\_client] [int] NOT NULL,

[surname\_client] [nvarchar](100) NOT NULL,

[name\_client] [nvarchar](100) NOT NULL,

[patronymic\_client] [nvarchar](100) NOT NULL,

[phoneNumber\_client] [nvarchar](11) NOT NULL,

[address\_client] [nvarchar](200) NOT NULL,

[dataOfBirth\_client] [date] NOT NULL,

[dataOfApplication\_client] [date] NULL,

CONSTRAINT [PK\_client] PRIMARY KEY CLUSTERED

(

[id\_client] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Код создания таблицы case

CREATE TABLE [dbo].[case](

[id\_case] [int] NOT NULL,

[essence\_case] [nvarchar](200) NOT NULL,

[dateOfStart\_case] [date] NULL,

[dateOfEnd\_case] [date] NULL,

[court\_decusion] [nvarchar](200) NULL,

[id\_client] [int] NOT NULL,

FOREIGN KEY (id\_client) REFERENCES client (id\_client),

CONSTRAINT [PK\_case] PRIMARY KEY CLUSTERED

(

[id\_case] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] GO

Код создания таблицы advocate

CREATE TABLE [dbo].[advocate](

[id\_advocate] [int] NOT NULL,

[surname\_advocate] [nvarchar](100) NOT NULL,

[name\_advocate] [nvarchar](100) NOT NULL,

[patronymic\_advocate] [nvarchar](100) NOT NULL,

[phoneNumber\_advocate] [nvarchar](11) NOT NULL,

[dataOfBirth\_advocate] [date] NOT NULL,

[tariff\_advocate] [money] NOT NULL,

[experience\_advocate] [nvarchar](30) NOT NULL,

[name\_collegiumOfAdvocates] [nvarchar](100) NOT NULL,

CONSTRAINT [PK\_advocate] PRIMARY KEY CLUSTERED

(

[id\_advocate] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON,

ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[advocate] WITH CHECK ADD FOREIGN KEY([name\_collegiumOfAdvocates])

REFERENCES [dbo].[collegiumOfAdvocates] ([name\_collegiumOfAdvocates])

GO

Код создания таблицы collegiumOfAdvocates

CREATE TABLE [dbo].[collegiumOfAdvocates](

[name\_collegiumOfAdvocates] [nvarchar](100) NOT NULL,

[countOfAdvocates] [int] NOT NULL,

[address\_collegiumOfAdvocates] [nvarchar](200) NOT NULL,

CONSTRAINT [PK\_collegiumOfAdvocates] PRIMARY KEY CLUSTERED

(

[name\_collegiumOfAdvocates] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Код создания таблицы procedure

CREATE TABLE [dbo].[procedure](

[id\_procedure] [int] NOT NULL,

[date\_procedure] [date] NOT NULL,

CONSTRAINT [PK\_procedure] PRIMARY KEY CLUSTERED

(

[id\_procedure] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Код создания таблицы service

CREATE TABLE [dbo].[service](

[id\_service] [int] NOT NULL,

[name\_service] [nvarchar](300) NOT NULL,

[price\_service] [money] NOT NULL,

CONSTRAINT [PK\_servise] PRIMARY KEY CLUSTERED

(

[id\_service] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Код создания таблицы procedure\_service

CREATE TABLE [dbo].[procedure\_service](

[id\_procedure\_service] [int] NOT NULL,

[id\_service] [int] NOT NULL,

[id\_procedure] [int] NOT NULL,

CONSTRAINT [PK\_procedure\_service] PRIMARY KEY CLUSTERED

(

[id\_procedure\_service] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[procedure\_service] WITH CHECK ADD FOREIGN KEY([id\_service])

REFERENCES [dbo].[service] ([id\_service])

GO

ALTER TABLE [dbo].[procedure\_service] WITH CHECK ADD FOREIGN KEY([id\_procedure])

REFERENCES [dbo].[procedure] ([id\_procedure])

GO

Код создания таблицы client\_advocate\_procedure

CREATE TABLE [dbo].[client\_advocate\_procedure](

[id\_client\_advocate\_procedure] [int] NOT NULL,

[id\_client] [int] NOT NULL,

[id\_advocate] [int] NOT NULL,

[id\_ procedure] [int] NOT NULL,

CONSTRAINT [PK\_client\_advocate\_procedure] PRIMARY KEY CLUSTERED

(

[id\_client\_advocate\_procedure] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[client\_advocate\_procedure] WITH CHECK ADD FOREIGN KEY([id\_client])

REFERENCES [dbo].[client] ([id\_client])

GO

ALTER TABLE [dbo].[client\_advocate\_procedure] WITH CHECK ADD FOREIGN KEY([id\_advocate])

REFERENCES [dbo].[advocate] ([id\_advocate])

GO

ALTER TABLE [dbo].[client\_advocate\_procedure] WITH CHECK ADD FOREIGN KEY([id\_ procedure])

REFERENCES [dbo].[procedure] ([id\_procedure])

GO

# Приложение Б

Код заполнения таблицы client

use AdvocateFirm

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (1, N'Родионов', N'Юрий', N'Русланович', '79052958476', N'ул.Ставровская', '2004-06-01', '2023-01-30');

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (2, N'Хомяков', N'Матвей', N'Михайлович', '79025287865', N'ул.Василисина', '2000-01-05', '2023-02-14');

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (3, N'Ильинская', N'Виктория', N'Кирилловна', '79102567547', N'ул.Офицерская', '1992-04-27', '2023-01-19');

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (4, N'Воробьев', N'Михаил', N'Степанович', '79062488437', N'ул.Фурманова', '1972-11-13', '2022-12-01');

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (5, N'Краснова', N'Ксения', N'Михайловна', '79002342621', N'ул.Пугачева', '1988-12-06', '2022-12-04');

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (6, N'Тимофеева', N'Вероника', N'Павловна', '79132547525', N'ул.Усти-на-Лабе', '1994-06-02', '2022-11-25');

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (7, N'Воробьева', N'Анастасия', N'Львовна', '79404327523', N'ул.Тракторная', '1999-03-04', '2022-11-30');

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (8, N'Колпаков', N'Алексей', N'Максимович', '79235835962', N'ул.Сурикова', '1985-10-04', '2023-02-01');

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (9, N'Масленникова', N'Екатерина', N'Фёдоровна', '79158969245', N'ул.Солнечная', '1991-03-08', '2023-01-30');

INSERT INTO client (id\_client, surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client)

VALUES (10, N'Кузин', N'Артур', N'Платонович', '79272465864', N'ул.Нижняя Дуброва', '1998-09-11', '2022-11-28');

Код заполнения таблицы procedure

use AdvocateFirm

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (1, '2023-01-30');

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (2, '2023-02-14');

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (3, '2023-01-19');

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (4, '2022-12-01');

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (5, '2022-12-04');

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (6, '2022-11-25');

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (7, '2022-11-30');

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (8, '2023-02-01');

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (9, '2023-01-30');

INSERT INTO [procedure] ([id\_procedure], [date\_procedure])

VALUES (10, '2022-11-28');

Код заполнения таблицы service

use AdvocateFirm

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (1, N'Устное консультирование по правовым вопросам', 1000);

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (2, N'Консультирование письменное', 2000);

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (3, N'Составление адвокатских запросов об истребовании сведений, необходимых для оказания юридической помощи', 750);

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (4, N'Составление несложных правовых документов не требующих изучения судебной практики', 3000);

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (5, N'Составление правовых документов, требующее изучения судебной практики', 5000);

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (6, N'Подготовка пакета учредительных документов, пакета документов для государственной регистрации прав', 7000);

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (7, N'Совершение практических действий в интересах доверителя', 5000);

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (8, N'Участие адвоката в дознании и на предварительном следствии', 10000);

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (9, N'Участие адвоката в уголовном судопроизводстве у мирового судьи', 10000);

INSERT INTO service (id\_service, name\_service, price\_service)

VALUES (10, N'Участие адвоката по гражданскому или административному делу в суде первой инстанции', 10000);

Код заполнения таблицы advocate

use AdvocateFirm

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (1, N'Попова', N'Дарья', N'Егоровна', '79003168063', '1985-05-16', '4000', N'8 лет', N'Линия права');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (2, N'Шубин', N'Венедикт', N'Константинович', '79009084483', '1974-08-22', '6000', N'16 лет', N'МИП');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (3, N'Брагин', N'Юрий', N'Фролович', '79005810729', '1977-02-05', '5000', N'12 лет', N'МИП');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (4, N'Соколов', N'Орест', N'Вадимович', '79008752708', '1988-07-26', '3000', N'5 лет', N'Линия права');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (5, N'Кудрявцев', N'Модест', N'Иринеевич', '79008658338', '1970-11-11', '8000', N'25 лет', N'Падва и партнеры');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (6, N'Комаров', N'Андрей', N'Игоревич', '79004679508', '1976-12-02', '4000', N'18 лет', N'Линия права');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (7, N'Евдокимов', N'Болеслав', N'Тимофеевич', '79009007303', '1980-05-04', '3000', N'10 лет', N'Падва и партнеры');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (8, N'Овчинников', N'Юстин', N'Лукьевич', '79007944925', '1969-11-09', '7000', N'27 лет', N'МИП');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (9, N'Захаров', N'Леонтий', N'Фролович', '79009162545', '1972-03-04', '6000', N'20 лет', N'Линия права');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (10, N'Ильин', N'Клим', N'Германнович', '79001206387', '1977-10-09', '4000', N'11 лет', N'Падва и партнеры');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (11, N'Шилов', N'Герман', N'Вадимович', '79009892833', '1978-01-14', '5000', N'15 лет', N'Падва и партнеры');

INSERT INTO advocate(id\_advocate, surname\_advocate, name\_advocate, patronymic\_advocate, phoneNumber\_advocate, dataOfBirth\_advocate, tariff\_advocate, experience\_advocate, name\_collegiumOfAdvocates)

VALUES (12, N'Ефремов', N'Вячеслав', N'Аркадьевич', '79007339873', '1971-05-16', '8000', N'22 лет', N'Падва и партнеры');

Код заполнения таблицы case

use AdvocateFirm

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (1, N'Раздел имущества', '2022-07-11', '2022-01-31', N'Выиграно', 1);

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (2, N'Взыскание задолженности по зарплате', '2022-08-02', '2023-02-13', N'Выиграно', 2);

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (3, N'Взыскание алиментов', '2022-11-13', '2023-01-18', N'Проиграно', 3);

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (4, N'Взыскание задолженности по зарплате', '2022-10-01', '2022-11-30', N'Выиграно', 4);

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (5, N'Раздел имущества', '2022-11-15', '2022-12-04', N'Проиграно', 5);

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (6, N'Взыскание задолженности по зарплате', '2022-10-20', '2022-11-24', N'Выиграно', 6);

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (7, N'Взыскание алиментов', '2022-09-28', '2022-11-29', N'Выиграно', 7);

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (8, N'Взыскание алиментов', '2022-09-17', '2023-01-31', N'Выиграно', 8);

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (9, N'Взыскание задолженности по зарплате', '2022-12-03', '2023-01-29', N'Проиграно', 9);

INSERT INTO [case] (id\_case, essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion, id\_client)

VALUES (10, N'Раздел имущества', '2022-10-19', '2022-11-27', N'Проиграно', 10);

Код заполнения таблицы collegiumOfAdvocates

use AdvocateFirm

INSERT INTO collegiumOfAdvocates (name\_collegiumOfAdvocates, countOfAdvocates, address\_collegiumOfAdvocates)

VALUES (N'Линия права', 4, N'Шлюзовая набережная, д. 4');

INSERT INTO collegiumOfAdvocates (name\_collegiumOfAdvocates, countOfAdvocates, address\_collegiumOfAdvocates)

VALUES (N'МИП', 3, N'Старопименовский переулок 18');

INSERT INTO collegiumOfAdvocates (name\_collegiumOfAdvocates, countOfAdvocates, address\_collegiumOfAdvocates)

VALUES (N'Падва и партнеры', 5, N'Большой Головин пер, д. 6');

Код заполнения таблицы client\_advocate\_procedure

use AdvocateFirm

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (1, 1, 1, 1);

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (2, 2, 2, 2);

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (3, 3, 3, 3);

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (4, 4, 4, 4);

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (5, 5, 5, 5);

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (6, 6, 6, 6);

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (7, 7, 7, 7);

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (8, 8, 8, 8);

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (9, 9, 9, 9);

INSERT INTO client\_advocate\_procedure (id\_client\_advocate\_procedure, id\_client, id\_advocate, id\_procedure)

VALUES (10, 10, 10, 10);

Код заполнения таблицы procedure\_service

use AdvocateFirm

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (1, 8, 1);

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (2, 3, 2);

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (3, 4, 3);

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (4, 7, 4);

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (5, 1, 5);

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (6, 5, 6);

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (7, 2, 7);

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (8, 9, 8);

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (9, 1, 9);

INSERT INTO procedure\_service (id\_procedure\_service, id\_service, id\_procedure)

VALUES (10, 6, 10);

# Приложение В

Код создания хранимой процедуры

use AdvocateFirm

GO

CREATE PROCEDURE InfoOfAll

@essence NVARCHAR(200)

AS

SELECT surname\_client, name\_client, patronymic\_client,

surname\_advocate, name\_advocate, patronymic\_advocate,

essence\_case, dateOfStart\_case, dateOfEnd\_case, court\_decusion,

name\_service, date\_procedure

FROM client, advocate, [case], [service], [procedure], client\_advocate\_procedure cap, procedure\_service ps

WHERE essence\_case = @essence AND cap.id\_client = client.id\_client AND client.id\_client = [case].id\_client AND

cap.id\_advocate = advocate.id\_advocate AND cap.id\_procedure = [procedure].id\_procedure AND

ps.id\_procedure = [procedure].id\_procedure AND ps.id\_service = [service].id\_service

Код создания скалярной функции

use AdvocateFirm

GO

CREATE FUNCTION GSProcedure (@name nvarchar (100), @sname nvarchar (100), @service nvarchar (300))

RETURNS money

BEGIN

DECLARE @summ money;

SELECT @summ = tariff\_advocate + price\_service

FROM advocate a, [service] s

WHERE name\_advocate = @name AND surname\_advocate = @sname AND name\_service = @service;

RETURN (@summ);

END;

Код создания первого представления

use AdvocateFirm

GO

CREATE VIEW P1

AS

SELECT

name\_client, surname\_client, patronymic\_client, phoneNumber\_client,

name\_advocate, surname\_advocate, patronymic\_advocate, s.name\_service, phoneNumber\_advocate,

p.id\_procedure, date\_procedure

FROM

client c, advocate a, [procedure] p, client\_advocate\_procedure cap

WHERE

cap.id\_client = c.id\_client AND cap.id\_advocate = a.id\_advocate AND cap.id\_procedure = p.id\_procedure AND s.id\_service = ps.id\_service AND p.id\_procedure = ps.id\_procedure

GROUP BY

cap.id\_client, cap.id\_advocate, cap.id\_procedure,

name\_client, surname\_client, patronymic\_client, phoneNumber\_client,

name\_advocate, surname\_advocate, patronymic\_advocate, phoneNumber\_advocate, .id\_procedure, date\_procedure, s.name\_service

Код создания второго представления

use AdvocateFirm

GO

CREATE VIEW P2

AS

SELECT

date\_procedure, name\_service, price\_service

FROM

[procedure] p, [service] s, procedure\_service ps

WHERE

ps.id\_procedure = p.id\_procedure AND ps.id\_service = s.id\_service

GROUP BY

ps.id\_service,

p.id\_procedure, date\_procedure, name\_service, price\_service

Код создания двух ролей

USE AdvocateFirm

EXEC sp\_addrole 'ADMIN'

EXEC sp\_addrole 'USER'

Код предоставления ролям прав

USE AdvocateFirm

GRANT DELETE, INSERT, UPDATE, SELECT ON advocate TO [ADMIN]

GRANT DELETE, INSERT, UPDATE, SELECT ON [case] TO [ADMIN]

GRANT DELETE, INSERT, UPDATE, SELECT ON client TO [ADMIN]

GRANT DELETE, INSERT, UPDATE, SELECT ON client\_advocate\_procedure TO [ADMIN]

GRANT DELETE, INSERT, UPDATE, SELECT ON collegiumOfAdvocates TO [ADMIN]

GRANT DELETE, INSERT, UPDATE, SELECT ON [procedure] TO [ADMIN]

GRANT DELETE, INSERT, UPDATE, SELECT ON procedure\_service TO [ADMIN]

GRANT DELETE, INSERT, UPDATE, SELECT ON [service] TO [ADMIN]

GRANT EXECUTE TO [ADMIN]

GRANT SELECT ON advocate TO [USER]

GRANT SELECT ON [case] TO [USER]

GRANT SELECT ON client TO [USER]

GRANT SELECT ON client\_advocate\_procedure TO [USER]

GRANT SELECT ON collegiumOfAdvocates TO [USER]

GRANT SELECT ON [procedure] TO [USER]

GRANT SELECT ON procedure\_service TO [USER]

GRANT SELECT ON [service] TO [USER]

GRANT EXECUTE TO [USER]

Код создания пользователей:

EXEC sp\_addlogin 'Administrator','admin182', 'AdvocateFirm'

USE AdvocateFirm

EXEC sp\_adduser 'Administrator','Administrator'

EXEC sp\_addrolemember 'ADMIN', 'Administrator'

GO

EXEC sp\_addlogin 'Polzovatel','polz270', 'AdvocateFirm'

USE AdvocateFirm

EXEC sp\_adduser 'Polzovatel','Polzovatel'

EXEC sp\_addrolemember 'USER', 'Polzovatel'

GO

Код создания триггера Update\_Client

use AdvocateFirm

GO

CREATE TRIGGER Update\_Client ON client

FOR UPDATE

AS

BEGIN

PRINT N'Запись, в таблице client изменена'

END

Код создания триггера Insert\_Client

use AdvocateFirm

GO

CREATE TRIGGER Insert\_Client ON client

FOR INSERT

AS

BEGIN

PRINT (N'В таблицу client была добавлена запись')

END

Код создания триггера Delete\_Client

use AdvocateFirm

GO

CREATE TRIGGER Delete\_Client ON client

FOR DELETE

AS

BEGIN

PRINT N'Запись, в таблице client удалена'

END

# Приложение Г

Код главной формы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Runtime.Remoting.Contexts;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ProgressBar;

namespace AdvocateFirm

{

public partial class AdminForm : Form

{

private SqlConnection connection;

private Authorization loginForm;

public AdminForm(SqlConnection connection, Authorization loginForm)

{

this.connection = connection;

this.loginForm = loginForm;

InitializeComponent();

}

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

{

SqlDataAdapter adapter0 = new SqlDataAdapter("SELECT \* FROM client", connection);

DataSet dataSet0 = new DataSet();

adapter0.Fill(dataSet0);

ClientDataGridView.DataSource = dataSet0.Tables[0];

ClientDataGridView.Columns[0].Visible = false;

ClientDataGridView.Columns[1].HeaderText = "Фамилия";

ClientDataGridView.Columns[2].HeaderText = "Имя";

ClientDataGridView.Columns[3].HeaderText = "Отчество";

ClientDataGridView.Columns[4].HeaderText = "Номер телефона";

ClientDataGridView.Columns[5].HeaderText = "Адрес";

ClientDataGridView.Columns[6].HeaderText = "Дата рождения";

ClientDataGridView.Columns[7].HeaderText = "Дата последнего визита";

SqlDataAdapter adapter1 = new SqlDataAdapter("SELECT \* FROM [case]", connection);

DataSet dataSet1 = new DataSet();

adapter1.Fill(dataSet1);

CaseDataGridView.DataSource = dataSet1.Tables[0];

CaseDataGridView.Columns[0].Visible = false;

CaseDataGridView.Columns[1].HeaderText = "Суть дела";

CaseDataGridView.Columns[2].HeaderText = "Дата начала";

CaseDataGridView.Columns[3].HeaderText = "Дата окончания";

CaseDataGridView.Columns[4].HeaderText = "Решение суда";

CaseDataGridView.Columns[5].Visible = false;

SqlDataAdapter adapter2 = new SqlDataAdapter("SELECT \* FROM advocate", connection);

DataSet dataSet2 = new DataSet();

adapter2.Fill(dataSet2);

ADataGridView.DataSource = dataSet2.Tables[0];

ADataGridView.Columns[0].Visible = false;

ADataGridView.Columns[1].HeaderText = "Фамилия";

ADataGridView.Columns[2].HeaderText = "Имя";

ADataGridView.Columns[3].HeaderText = "Отчество";

ADataGridView.Columns[4].HeaderText = "Номер телефона";

ADataGridView.Columns[5].HeaderText = "Дата рождения";

ADataGridView.Columns[6].HeaderText = "Тариф";

ADataGridView.Columns[7].HeaderText = "Опыт";

ADataGridView.Columns[8].Visible = false;

SqlDataAdapter adapter3 = new SqlDataAdapter("SELECT \* FROM collegiumOfAdvocates", connection);

DataSet dataSet3 = new DataSet();

adapter3.Fill(dataSet3);

CADataGridView.DataSource = dataSet3.Tables[0];

CADataGridView.Columns[0].Visible = false;

CADataGridView.Columns[1].HeaderText = "Название";

CADataGridView.Columns[2].HeaderText = "Количество адвокатов";

CADataGridView.Columns[3].HeaderText = "Адрес";

SqlDataAdapter adapter4 = new SqlDataAdapter("SELECT \* FROM [service]", connection);

DataSet dataSet4 = new DataSet();

adapter4.Fill(dataSet4);

SDataGridView.DataSource = dataSet4.Tables[0];

SDataGridView.Columns[0].Visible = false;

SDataGridView.Columns[1].HeaderText = "Название";

SDataGridView.Columns[2].HeaderText = "Цена";

SqlDataAdapter adapter5 = new SqlDataAdapter("SELECT \* FROM [procedure]", connection);

DataSet dataSet5 = new DataSet();

adapter5.Fill(dataSet5);

PDataGridView.DataSource = dataSet5.Tables[0];

PDataGridView.Columns[0].Visible = false;

PDataGridView.Columns[1].HeaderText = "Дата";

SqlDataAdapter adapter6 = new SqlDataAdapter("SELECT \* FROM P1", connection);

DataSet dataSet6 = new DataSet();

adapter6.Fill(dataSet6);

PredDataGridView1.DataSource = dataSet6.Tables[0];

PredDataGridView1.Columns[1].HeaderText = "Фамилия клиента";

PredDataGridView1.Columns[1].HeaderText = "Имя клиента";

PredDataGridView1.Columns[1].HeaderText = "Отчество клиента";

PredDataGridView1.Columns[1].HeaderText = "Номер тел. Клиента";

PredDataGridView1.Columns[1].HeaderText = "Фамилия Адвоката";

PredDataGridView1.Columns[1].HeaderText = "Имя Адвоката";

PredDataGridView1.Columns[1].HeaderText = "Отчество Адвоката";

PredDataGridView1.Columns[1].HeaderText = "Номер тел. Адвоката";

PredDataGridView1.Columns[1].Visible = false;

PredDataGridView1.Columns[1].HeaderText = "Дата процедуры";

SqlDataAdapter adapter7 = new SqlDataAdapter("SELECT \* FROM P2", connection);

DataSet dataSet7 = new DataSet();

adapter7.Fill(dataSet7);

PredDataGridView2.DataSource = dataSet7.Tables[0];

PredDataGridView2.Columns[1].Visible = false;

PredDataGridView2.Columns[1].HeaderText = "Дата процедуры";

PredDataGridView2.Columns[1].HeaderText = "Название услуги";

PredDataGridView2.Columns[1].HeaderText = "Стоимость услуги";

}

private void AdminForm\_FormClosing\_1(object sender, FormClosingEventArgs e)

{

var result = MessageBox.Show("Вы уверены, что хотите выйти?", "Выход из системы", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

loginForm.Close();

connection.Close();

}

else

e.Cancel = true;

}

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

{

new Dob\_Client(connection).Show();

}

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

{

SqlDataAdapter adapter0 = new SqlDataAdapter("SELECT \* FROM client", connection);

DataSet dataSet0 = new DataSet();

adapter0.Fill(dataSet0);

ClientDataGridView.DataSource = dataSet0.Tables[0];

ClientDataGridView.Columns[0].HeaderText = "ИД";

ClientDataGridView.Columns[1].HeaderText = "Фамилия";

ClientDataGridView.Columns[2].HeaderText = "Имя";

ClientDataGridView.Columns[3].HeaderText = "Отчество";

ClientDataGridView.Columns[4].HeaderText = "Номер телефона";

ClientDataGridView.Columns[5].HeaderText = "Адрес";

ClientDataGridView.Columns[6].HeaderText = "Дата рождения";

ClientDataGridView.Columns[7].HeaderText = "Дата последнего визита";

}

private void ClientDataGridView\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (UpCheckBoxClient.Checked && DelCheckBoxClient.Checked)

{

MessageBox.Show($"Вы не можете одновременно редактировать и удалять данные");

return;

}

if (DelCheckBoxClient.Checked)

{

var result = MessageBox.Show($"Вы уверены, что хотите удалить клиента с иминем {ClientDataGridView[2, ClientDataGridView.CurrentRow.Index].Value.ToString()}?", "Выход из системы", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

int removeId = Convert.ToInt32(ClientDataGridView[0, ClientDataGridView.CurrentRow.Index].Value.ToString());

SqlCommand CDel1 = new SqlCommand($"DELETE FROM client WHERE id\_client = {removeId}", connection);

CDel1.ExecuteNonQuery();

}

else

{

return;

}

}

if (UpCheckBoxClient.Checked)

{

int Id = Convert.ToInt32(ClientDataGridView[0, ClientDataGridView.CurrentRow.Index].Value.ToString());

string surname\_client = ClientDataGridView[1, ClientDataGridView.CurrentRow.Index].Value.ToString();

string name\_client = ClientDataGridView[2, ClientDataGridView.CurrentRow.Index].Value.ToString();

string patronymic\_client = ClientDataGridView[3, ClientDataGridView.CurrentRow.Index].Value.ToString();

string phoneNumber\_client = ClientDataGridView[4, ClientDataGridView.CurrentRow.Index].Value.ToString();

string address\_client = ClientDataGridView[5, ClientDataGridView.CurrentRow.Index].Value.ToString();

string dataOfBirth\_client = ClientDataGridView[6, ClientDataGridView.CurrentRow.Index].Value.ToString();

string dataOfApplication\_client = ClientDataGridView[7, ClientDataGridView.CurrentRow.Index].Value.ToString();

new Up\_Client(connection, Id,surname\_client, name\_client, patronymic\_client, phoneNumber\_client, address\_client, dataOfBirth\_client, dataOfApplication\_client).Show();

}

}

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

{

SqlDataAdapter poisk = new SqlDataAdapter($"use AdvocateFirm EXECUTE InfoOfAll N'{SutTextBox.Text}'", connection);

DataSet dataPoisk = new DataSet();

poisk.Fill(dataPoisk);

PoiskDataGridView.DataSource = dataPoisk.Tables[0];

PoiskDataGridView.Columns[0].HeaderText = "Фамилия клиента";

PoiskDataGridView.Columns[1].HeaderText = "Имя клиента";

PoiskDataGridView.Columns[2].HeaderText = "Отчество клиента ";

PoiskDataGridView.Columns[3].HeaderText = "Фамилия адвоката";

PoiskDataGridView.Columns[4].HeaderText = "Имя адвоката";

PoiskDataGridView.Columns[5].HeaderText = "Отчество адвоката";

PoiskDataGridView.Columns[6].HeaderText = "Суть дела";

PoiskDataGridView.Columns[7].HeaderText = "Дата начала";

PoiskDataGridView.Columns[8].HeaderText = "Дата окончания";

PoiskDataGridView.Columns[9].HeaderText = "Решение суда";

PoiskDataGridView.Columns[10].HeaderText = "Название услуги";

PoiskDataGridView.Columns[11].HeaderText = "Дата процедуры";

}

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

{

SqlDataAdapter rasschet = new SqlDataAdapter($"use AdvocateFirm SELECT dbo.GSProcedure (N'{ANameTextBox.Text}', N'{ASnameTextBox.Text}', N'{ServiceTextBox.Text}')", connection);

DataSet dataRasschet = new DataSet();

rasschet.Fill(dataRasschet);

RasschetDataGridView.DataSource = dataRasschet.Tables[0];

RasschetDataGridView.Columns[0].HeaderText = "Стоимость процедуры";

}

}

}

Код формы авторизации

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AdvocateFirm

{

public partial class Authorization : Form

{

string connectionString;

public static string GetRemoteConnectionString(string login, string password)

{

SqlConnectionStringBuilder sqlString = new SqlConnectionStringBuilder()

{

DataSource = $"(localdb)\\MSSQLLocalDB", /\*(localdb)\\MSSQLLocalDB\*/

InitialCatalog = "AdvocateFirm",

IntegratedSecurity = false,

MultipleActiveResultSets = true,

ApplicationName = "EntityFramework",

UserID = login,

Password = password

};

return sqlString.ToString();

}

public Authorization()

{

InitializeComponent();

}

private enum Role

{

ADMIN,

USER

}

Role role;

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

{

connectionString = GetRemoteConnectionString(LoginTextBox.Text, PasswordTextBox.Text);

SqlConnection connection = new SqlConnection(connectionString);

try

{

connection.Open();

if (Convert.ToInt32(new SqlCommand("SELECT IS\_MEMBER('ADMIN')", connection).ExecuteScalar()) == 1)

{

role = Role.ADMIN;

}

else if (Convert.ToInt32(new SqlCommand("SELECT IS\_MEMBER('USER')", connection).ExecuteScalar()) == 1)

{

role = Role.USER;

}

MessageBox.Show("Подключение установлено!");

}

catch

{

MessageBox.Show("При авторизации произошла ошибка, проверьте введенные данные", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

switch (role)

{

case Role.ADMIN:

new AdminForm(connection, this).Show();

this.Hide();

break;

case Role.USER:

new UserForm(connection, this).Show();

this.Hide();

break;

}

}

}

}

Код формы добавления клиента

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Net.Sockets;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AdvocateFirm

{

public partial class Dob\_Client : Form

{

private SqlConnection connection;

public Dob\_Client(SqlConnection connection)

{

this.connection = connection;

InitializeComponent();

}

public void InsertClient(string surname\_client, string name\_client, string patronymic\_client, string phoneNumber\_client,

string address\_client, string dataOfBirth\_client, string dataOfApplication\_client)

{

string insert = $"INSERT INTO client VALUES (N'{surname\_client}', N'{name\_client}', N'{patronymic\_client}', N'{phoneNumber\_client}', " +

$"N'{address\_client}', '{dataOfBirth\_client}', '{dataOfApplication\_client}')";

try

{

new SqlCommand(insert, connection).ExecuteNonQuery();

MessageBox.Show($"Запись о {surname\_client} {name\_client} {patronymic\_client} была успешно добавлена!", "Добавление данных", MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

InsertClient(textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text, textBox5.Text, textBox6.Text, textBox7.Text);

this.Close();

}

}

}