

Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тульский государственный университет»

Институт прикладной математики и компьютерных наук
Кафедра информационной безопасности

**Клиент-серверное приложение для работы с базой данных через
MS SQL server**

Отчет по выполнению практического задания
по курсу “Системы автоматизированного управления”

Выполнил: ст. гр. 230761 Кочкин К. Ю.

Проверил: доц. каф. ИБ Баранов А.Н.

Тула 2020

Практическое задание

Цель работы:

Освоить навыки работы с базами данных через «MC SQL Server» с помощью MC SQL Server Management Studio (SSMS). Научиться писать клиентское приложение для работы с базами данных на языке программирования C++\CLI с использованием Windows Form.

Задание на работу:

1. Изучить учебное пособие Работа в MS SQL Server Express.
2. В соответствии с индивидуальной темой разработать базу данных с использованием такой системы (инструмента) управления базами данных для MS SQL Server Express как SQL Server Management Studio (SSMS).
3. Разработать клиентское приложение для управления базой данных, подключенной к серверу. Применение технологий: программная среда Visual Studio, язык программирования C++\CLI, технология соединения с БД – с помощью пространства имен «System::Data::SqlClient»
4. Через клиентское приложение обеспечить доступ к базе данных, подключенной к серверу.

Ход работы:

Клиентское приложение будет обрабатывать информацию, необходимую для работы автоматической заправочной станции (АЗС). Информация будет храниться в 7 связанных таблицах. Инфологическая модель базы данных представлена на рисунке 1.

Приложение будет позволять работать с базой данных, а именно выполнять следующие функции:

- просмотр всех таблиц с помощью разных фильтров;
- детализированный просмотр связанных таблиц;
- добавление новых записей;
- редактирование данных;
- удаление данных;
- разграничение прав доступа;

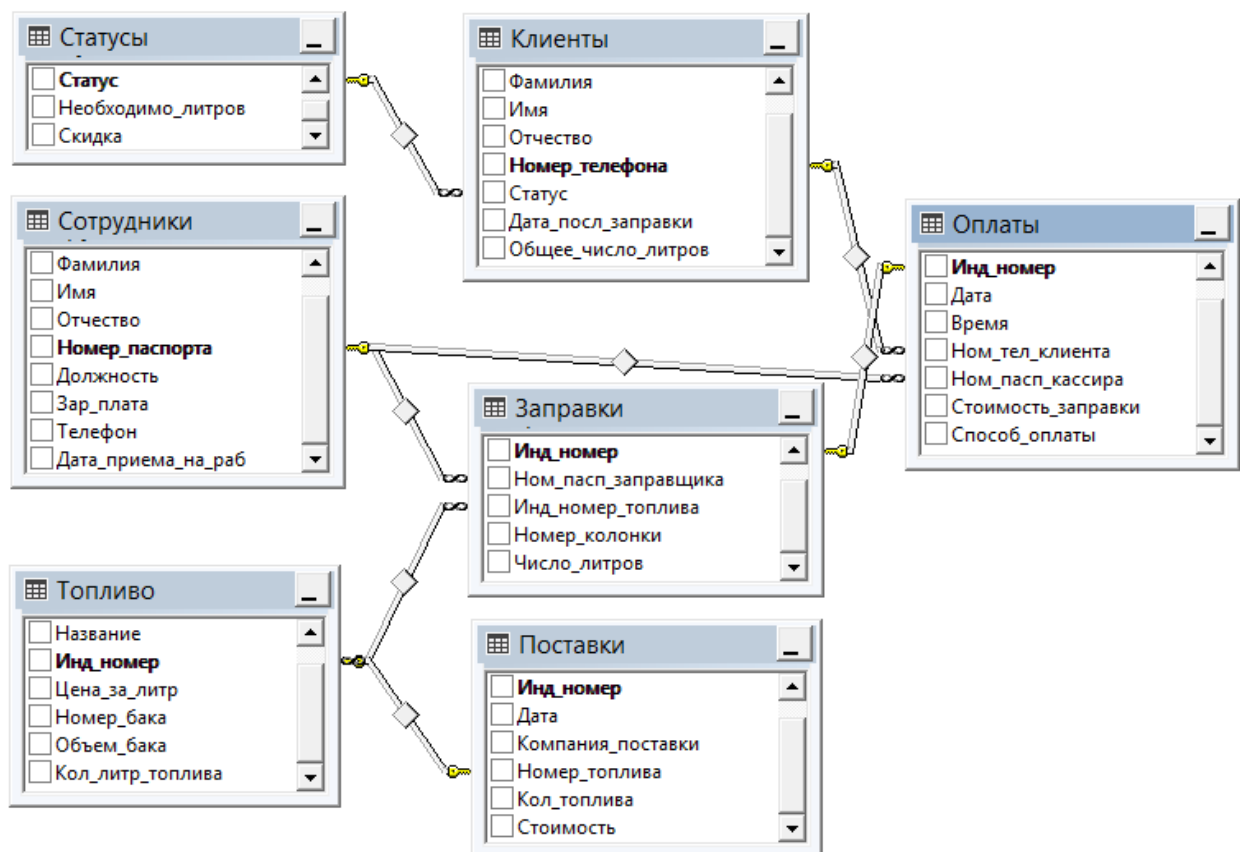


Рисунок 1 - Инфологическая модель

1. Создание базы данных АЗС

1.1. Откроем Microsoft SQL Server Management Studio, затем нажмем на “Базы” → «Создать базу» и в открывшемся окне указываем имя БД, имя владельца базы и конфигурацию файлов с данными и логами. Нажимаем на кнопку «Создать».

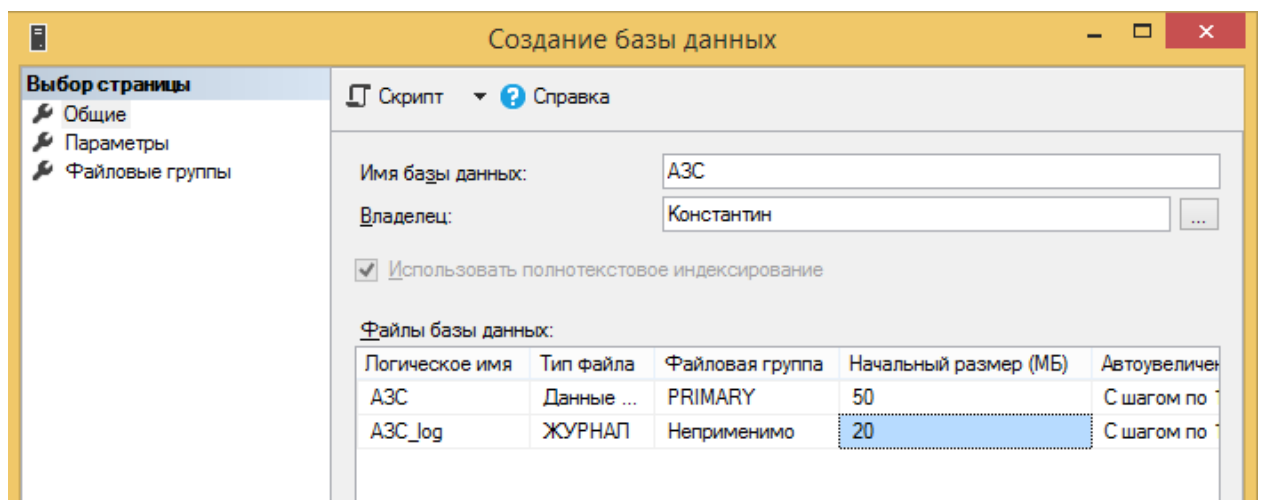
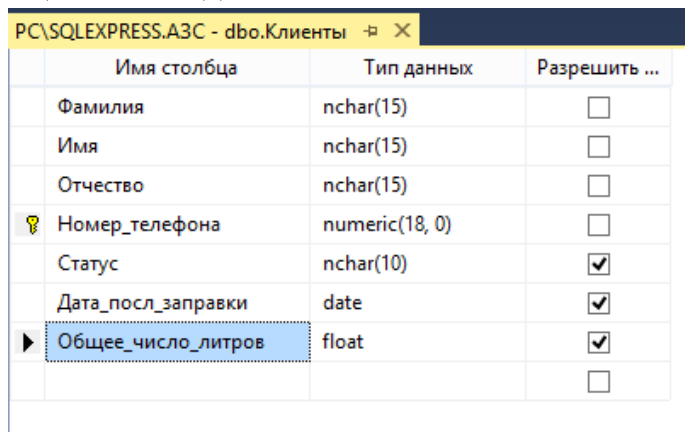


Рисунок 2 - Создание базы данных

1.2. Теперь дважды кликаем на созданную базу левой кнопкой мыши, после нажимаем правой кнопкой мыши на «Таблицы» и

выбираем «Создать» → «Таблицу». В открывшейся вкладке зададим нужные столбцы и типы данных.



Имя столбца	Тип данных	Разрешить ...
Фамилия	nchar(15)	<input type="checkbox"/>
Имя	nchar(15)	<input type="checkbox"/>
Отчество	nchar(15)	<input type="checkbox"/>
Номер_телефона	numeric(18, 0)	<input type="checkbox"/>
Статус	nchar(10)	<input checked="" type="checkbox"/>
Дата_посл_заправки	date	<input checked="" type="checkbox"/>
Общее_число_литров	float	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Рисунок 3 - Создание таблицы

Для установки первичного ключа кликнем правой кнопкой мыши слева от нужного столбца и выберем «Задать первичный ключ». Теперь кликнем на заголовок вкладки правой кнопкой мыши и нажмем «Сохранить», после останется присвоить имя таблице и нажать кнопку «ОК».

1.3. Подобным образом создадим все остальные таблицы.

1.4. Нажмем правой кнопкой мыши на «Диаграммы баз данных» → «Создать». В открывшемся окне добавим все таблицы и разместим их на рабочем поле. Теперь нажмем правой кнопкой мыши на таблицу и выберем «Отношения». Тут мы установим связи между таблицами с помощью внешних ключей.

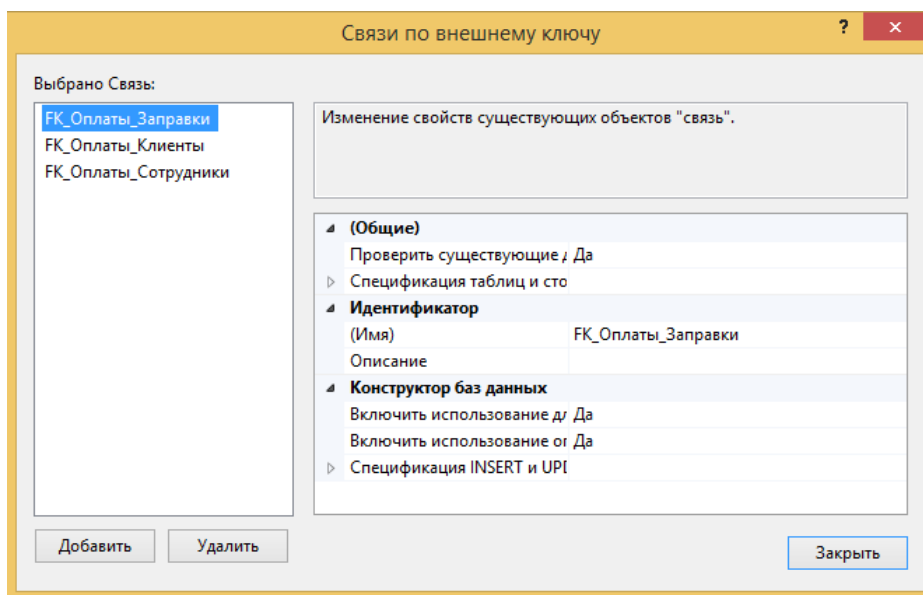


Рисунок 4 - Окно связей по внешнему ключу

В этом же окне зададим спецификацию INSERT и DELETE для каждого ключа: «Каскадно» для изменения и «Нет действий» для удаления. Сохраним изменения.

1.5. Закроем диаграмму и снова откроем «Проект» таблицы «Заправки» и «Поставки». В каждой из них для столбца первичного ключа установим свойство «AutoIncrement». Для этого кликнем на нужный столбец и внизу окна в панели «Свойства» найдем строку «Спецификация идентификатора» и установим нужные значения.

1.6. Заполним все таблицы данными.

1.7. Создадим отдельную таблицу для авторизации всех пользователей, которые смогут работать с приложением. Она будет содержать логин, пароль, уровень доступа, а также фамилию, имя и электронную почту пользователя.

	Логин	Пароль	Уровень_доступа	Фамилия	Имя	Эл_почта
	Masha	123	2	Трубова ...	Мария ...	maha@mail.ru ...
	Misha	000	1	Девятайкин ...	Михаил ...	misha@gmail.com ...
✎	Sasha	777	0	Александр ...	Бубнов ...	Sasha@yandex.com
*	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 5 - Таблица с пользователями

2. Принцип работы клиентского приложения

2.1. Приложение будет состоять из 4 windows form:

- Форма авторизации («AuthorizationForm.h»);
- Форма главного окна для просмотра данных («Tables.h»);
- Форма для добавления новой записи в таблицу («AddRecordForm.h»);
- Форма для изменения данных («ChangeDataForm.h»).

2.2. Для работы с MS SQL Server напомним класс со статическими функциями, которым будем обращаться для реализации запросов к БД. Класс написан в заголовочном файле «Functional.h» и в ресурсном файле «Functional.cpp».

2.3. Форма авторизации предлагает пользователю ввести логин и пароль для входа. При нажатии на кнопку «Вход», программа отправляет SQL запрос для получения таблицы с пользователями (таблица «Users») с помощью функции «FindUser(логин, пароль)».

Получив эту таблицу программа ищет совпадение пары данных: логина и пароля. В случае успеха класс «Functional» запоминает уровень доступа, считанный из совпавшей строки таблицы.

Рисунок 6 - Форма авторизации

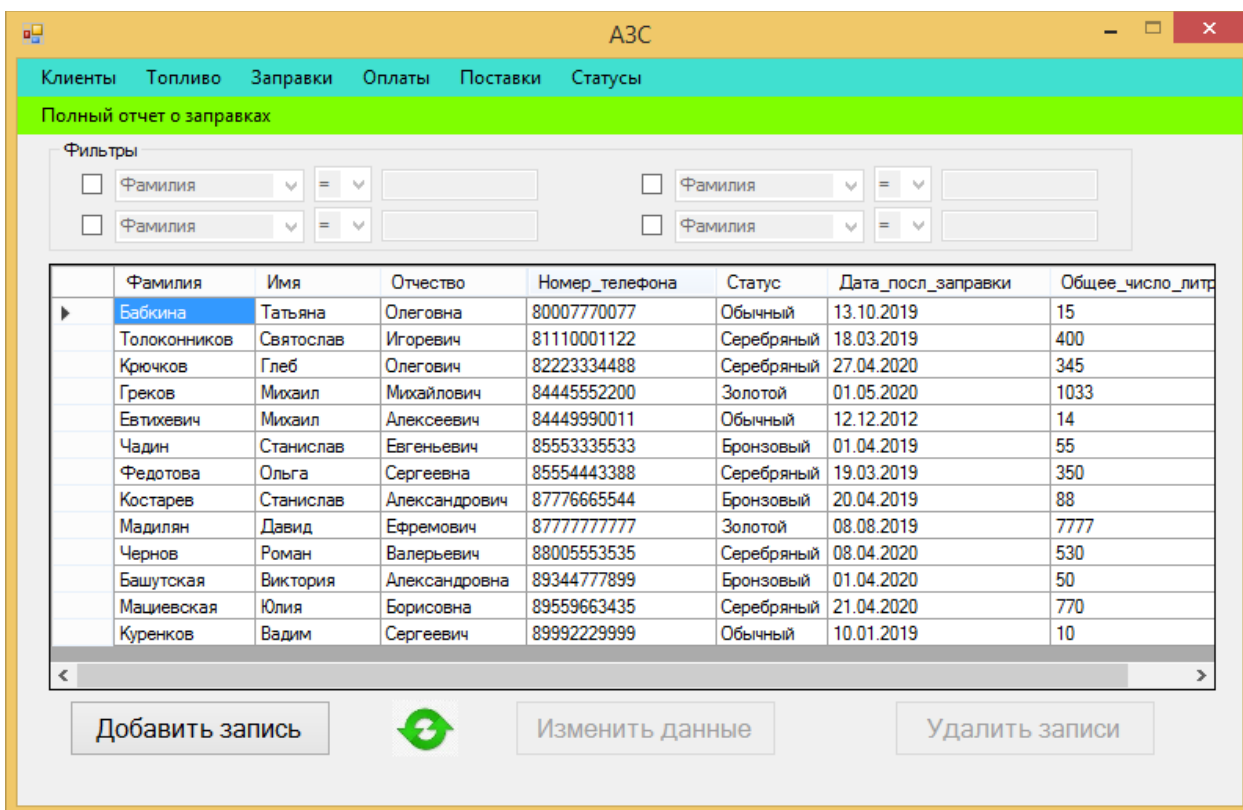
2.4. Программа предусматривает три уровня доступа.

Таблица 1 - Уровни доступа

Название таблицы	Уровень 2 (Кассир)				Уровень 1 (Администратор)				Уровень 0 (Директор)			
Клиенты	+	+	-	-	+	-	+	+	+	-	-	-
Сотрудники	-	-	-	-	+	-	-	-	+	+	+	+
Топливо	+	-	-	-	+	-	+	-	+	+	-	+
Заправки	+	+	-	-	+	-	+	+	+	-	-	-
Оплата	+	+	-	-	+	-	+	+	+	-	-	-
Поставки	+	-	-	-	+	+	+	+	+	-	-	-
Статусы	+	-	-	-	+	+	+	+	+	-	-	-
Users	-	-	-	-	+	+	+	+	+	-	-	-
Полный отчет (Динамическая)	+	-	-	-	+	-	-	-	+	-	-	-
	Просмотр	Добавление	Редактирование	Удаление	Просмотр	Добавление	Редактирование	Удаление	Просмотр	Добавление	Редактирование	Удаление

2.5. После успешной авторизации запускается основная форма просмотра данных. В зависимости от уровня доступа скрываются некоторые пункты меню, имеющие названия таблиц. Форма состоит из:

- меню, в котором можно выбрать для просмотра, одну из таблиц или «полного отчета о заправках», имеющего выборочные данные, сразу из нескольких таблиц;
- 4 фильтров, которые предлагают для выбора столбцы из текущей таблиц и варианты сравнения, и имеют текстовое поля для ввода значения выбранного параметра.
- удобной таблиц, вмещающей всей данные, которые можно пролистать сверху вниз и слева направо;
- кнопки добавления новой записи;
- кнопки обновления, выполненный в форме рисунка, зеленой круглой стрелочки;
- кнопки изменения данных;
- и кнопки удаления записей.



	Фамилия	Имя	Отчество	Номер_телефона	Статус	Дата_посл_заправки	Общее_число_литр
▶	Бабкина	Татьяна	Олеговна	80007770077	Обычный	13.10.2019	15
	Толоконников	Святослав	Игоревич	81110001122	Серебряный	18.03.2019	400
	Крючков	Глеб	Олегович	82223334488	Серебряный	27.04.2020	345
	Греков	Михаил	Михайлович	84445552200	Золотой	01.05.2020	1033
	Евтихевич	Михаил	Алексеевич	84449990011	Обычный	12.12.2012	14
	Чадин	Станислав	Евгеньевич	85553335533	Бронзовый	01.04.2019	55
	Федотова	Ольга	Сергеевна	85554443388	Серебряный	19.03.2019	350
	Костарев	Станислав	Александрович	87776665544	Бронзовый	20.04.2019	88
	Мадиян	Давид	Ефремович	87777777777	Золотой	08.08.2019	7777
	Чернов	Роман	Валерьевич	88005553535	Серебряный	08.04.2020	530
	Башутская	Виктория	Александровна	89344777899	Бронзовый	01.04.2020	50
	Мациевская	Юлия	Борисовна	89559663435	Серебряный	21.04.2020	770
	Куренков	Вадим	Сергеевич	89992229999	Обычный	10.01.2019	10

Рисунок 7 - Форма просмотра данных

2.6. При нажатии на один из пунктов верхнего меню программа с помощью SQL запросов, получает данные соответствующей таблицы, заполняет все фильтры названиями столбцов таблицы, а также включает или выключает кнопки добавления, редактирования и удаления в зависимости от уровня доступа.

2.7. При нажатии на элемент меню «Полный отчет о заправках» программа создает новую таблицу, в которую записан результат запроса к нескольким столбцам разных таблиц. Данный запрос показывает следующие данные:

- Идентификационный номер;
- Дату и время заправки;
- ФИО Клиента;
- ФИО Заправщика;
- Число литров;
- Название топлива;
- Цену за литр бензина;
- Полную стоимость заправки;
- Стоимость со скидкой;
- Скидку в процентах;
- ФИО кассира.

После записи запроса в таблицу, в ней возможен поиск по фильтрам.

После завершения работы программы таблица удаляется. Таблица обновляется каждый раз, как пользователь кликает на соответствующий названию таблицы пункт меню.

АЗС

Клиенты Топливо Заправки Оплаты Поставки Статусы

Полный отчет о заправках

Фильтры

☐ Инд_номер = ☐ Инд_номер =

☐ Инд_номер = ☐ Инд_номер =

	Инд_номер	Дата	Время	ФИО_клиента	ФИО_Заправщика	Число_литров	Топливо	Цена_за_л
	1013	05.04.2020	20:00:00	Куренков В.С.	Ермаков А.М.	10	ДТ	48,8000
	1006	01.04.2020	12:00:00	Чадин С.Е.	Ермаков А.М.	15	92	42,7000
	1007	01.04.2020	13:00:00	Костарев С.А.	Ермаков А.М.	15	92	42,7000
▶	1010	01.04.2020	16:00:00	Башутская В.А.	Ермаков А.М.	15	98	49,8000
	1025	10.04.2020	09:00:00	Костарев С.А.	Пух В.Ш.	15	95	45,9000
	1029	10.04.2020	13:00:00	Мацневская Ю.Б.	Пух В.Ш.	15	98	49,8000
	1038	15.04.2020	11:00:00	Федотова О.С.	Крокодилов Г.Г.	15	92	42,7000
	1042	15.04.2020	15:00:00	Башутская В.А.	Крокодилов Г.Г.	15	92	42,7000
	1003	01.04.2020	09:30:00	Евтихевич М.А.	Ермаков А.М.	17	92	42,7000
	1016	05.04.2020	17:00:00	Толоконников С.И.	Пух В.Ш.	17	92	42,7000
	1052	20.04.2020	20:30:00	Чадин С.Е.	Крокодилов Г.Г.	18	98	49,8000
	1002	01.04.2020	09:30:00	Федотова О.С.	Ермаков А.М.	20	92	42,7000
	1008	01.04.2020	14:00:00	Мадиян Д.Е.	Ермаков А.М.	20	95	45,9000
	1009	01.04.2020	15:00:00	Чадин С.Е.	Ермаков А.М.	20	95	45,9000

Добавить запись Изменить данные Удалить записи

Рисунок 8 - Полный отчет о заправках

2.8. Программа позволяет фильтровать записи с помощью фильтров. изначально все четыре фильтра выключены, что можно увидеть на рисунках 7 и 8. Чтобы включить фильтр необходимо поставить галочку слева от выбранного пользователем фильтра. После этого пользователь сможет выбрать нужный столбец, операцию сравнения и ввести значение. Чтобы отключить фильтр, достаточно просто убрать галочку. Возможно использование как одного, так и всех четырех фильтров и при этом не обязательно включать их последовательно. Например, можно включить левый нижний и правый верхний.

Если фильтр был включен, однако поле ввода не было заполнено, то программа сообщит об этом и произведет выборку данных без учета этого фильтра.

Чтобы произвести поиск записей с помощью фильтров, нужно:

- 1) Включить нужное число фильтров;
- 2) Выбрать нужные столбцы;
- 3) Выбрать операции сравнения;
- 4) Вести значения во все активные поля ввода;
- 5) Обновить данные, нажав на зеленую стрелочку.

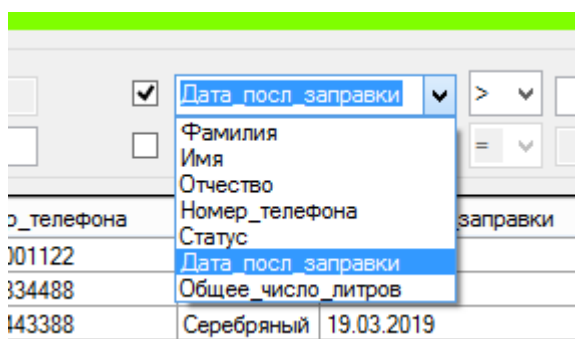


Рисунок 9 - Выбор столбца для фильтра

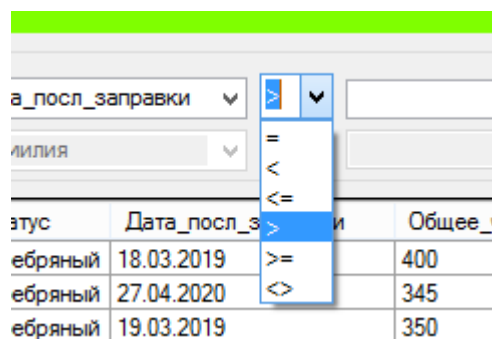


Рисунок 10 - Выбор операции сравнения

АЗС

Клиенты Сотрудники Топливо Заправки Оплаты Поставки Статусы Users

Полный отчет о заправках

Фильтры

☐ Фамилия = ☒ Дата_посл_заправки > 01.01.2020

☒ Статус = Серебряный ☐ Фамилия =

	Фамилия	Имя	Отчество	Номер_телефона	Статус	Дата_посл_заправки	Общее_число_литров
▶	Крючков	Глеб	Олегович	82223334488	Серебряный	27.04.2020	345
	Чернов	Роман	Валерьевич	88005553535	Серебряный	08.04.2020	530
	Мациевская	Юлия	Борисовна	89559663435	Серебряный	21.04.2020	770


Добавить запись  Изменить данные Удалить записи

Рисунок 11 - Результат выборки данных в таблице «Клиенты»

АЗС

Клиенты Сотрудники Топливо Заправки Оплаты Поставки Статусы Users

Полный отчет о заправках

Фильтры

☒ Число_литров >= 30 ☒ Топливо <> ДТ

☒ Число_литров <= 40 ☒ ФИО_Заправщика = Крючков Г.Г.

	ФИО_Заправщика	Число_литров	Топливо	Цена_за_литр	Полная_стоимость	Цена_со_скидкой	Скид
▶	Крючков Г.Г.	30	92	42,7000	1281,00	1191,3300	7
	Крючков Г.Г.	30	92	42,7000	1281,00	1255,3800	2
	Крючков Г.Г.	30	92	42,7000	1281,00	1191,3300	7
	Крючков Г.Г.	30	92	42,7000	1281,00	1229,7600	4
	Крючков Г.Г.	40	95	45,9000	1836,00	1836,0000	0
	Крючков Г.Г.	30	98	49,8000	1494,00	1389,4200	7
	Крючков Г.Г.	30	98	49,8000	1494,00	1494,0000	0


Добавить запись  Изменить данные Удалить записи

Рисунок 12 - Выборка данных в "Полном отчете о заправках"

2.9. Для добавления записей, если у пользователя есть соответствующий уровень доступа, необходимо нажать на кнопку «Добавить запись». Далее откроется форма, в которой нужно ввести или выбрать значения в каждое активное поле. Форма строится динамическим образом и в зависимости от активной таблицы в главной форме, появляются именно столько полей, сколько столбцов содержится в таблице в данный момент. Пока открыта форма добавления записей, основная форма недоступна.

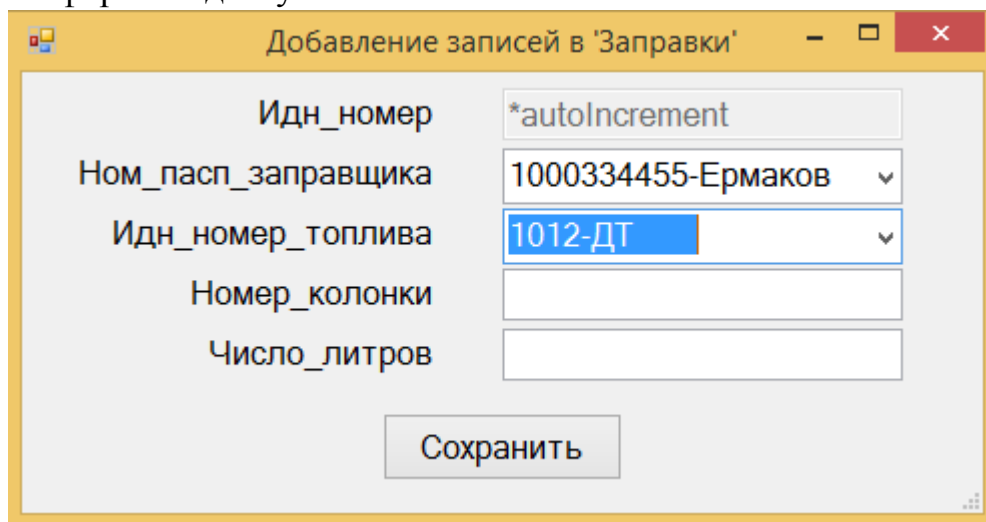


Рисунок 13 – Окно добавления записи в таблицу "Заправки"

В таблицах «Заправки» и «Поставки» на столбец с ключевым значением установлено свойство «AutoIncrement» и поэтому в данное поля программа не даст ввести свое значение. В полях внешних ключей для удобства пользователей программа сама загружает все возможные варианты значений, а также для большего понимания к каждому вариант подписывается другим полем той таблицы, на которую ссылается внешний ключ. Так, например, для номера паспорта сотрудника 1000334455, подписана его фамилия.

После ввода или выбора всех значений, необходимо нажать на кнопку «Сохранить» и тогда программа сформирует запрос и отправит его на сервер. Если не все поля были заполнены, то программа сообщит об этом пользователю.

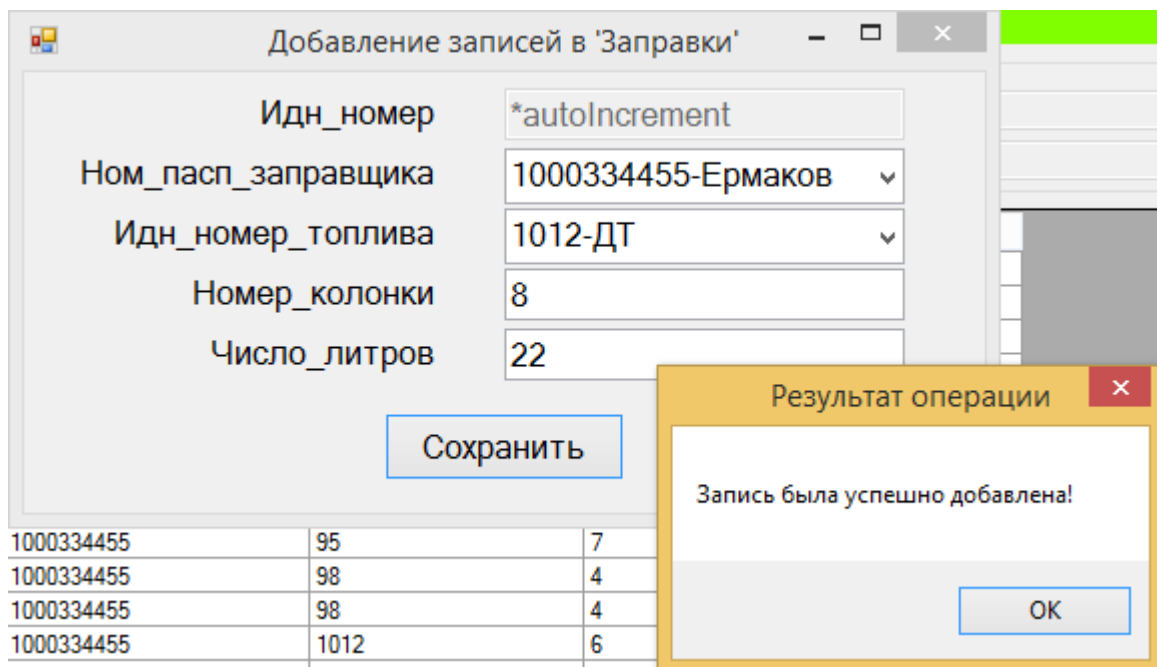
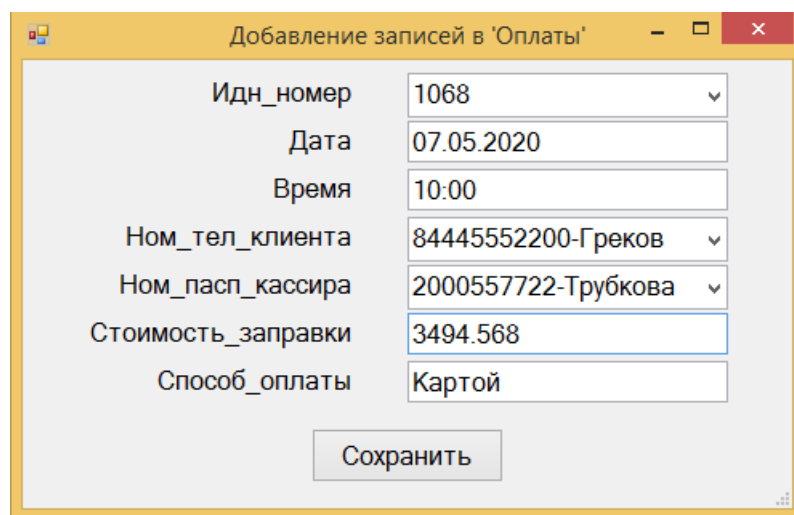


Рисунок 14 - Результат добавление новой записи

Форма добавления записей для таблицы «Оплаты», имеет свою особенность. Так как таблицы «Заправки» и «Оплаты» имеют отношение 1→1, то для если запись из таблицы «Заправки» получила пару из таблиц «Оплаты», то больше другую пару она уже получить не может. Поэтому поле «Идн_номер» таблицы «Оплаты» дает выбрать только те записи из таблицы «Заправки», которые еще не получили пару. Это ситуация означает, что машина уже была заправлена, а клиент еще не расплатился за бензин.

Рисунок 15 - Окно добавления записи для таблицы "Оплаты"

Есть и другая особенность формы для таблицы «Оплаты». После выбора Идентификационного номера и Клиента, поле «Стоимость заправки» автоматически заполнится рассчитанной суммой с учетом скидки для выбранного клиента. При изменении полей выбора, сумма будет пересчитываться заново. Чтобы подсчитать сумму, программа отправляет запросы с включением внешних ключей к БД, получая число литров заправленного топлива из таблицы «Заправки», цену за литр бензина из таблицы «Топливо» и скидку из таблицы «Статусы».

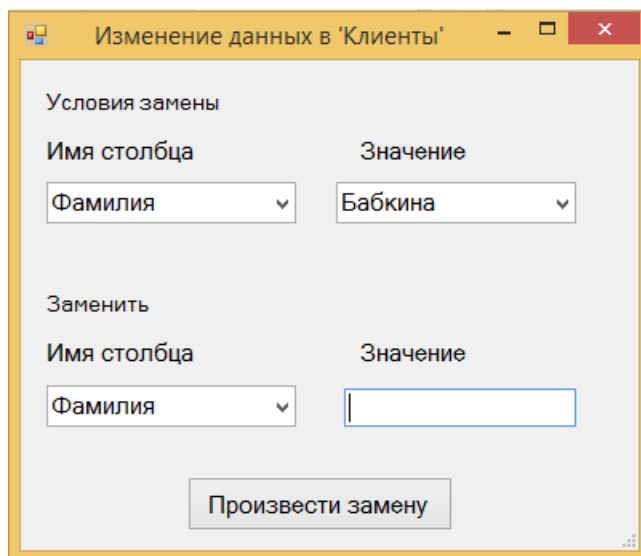


Идн_номер	1068
Дата	07.05.2020
Время	10:00
Ном_тел_клиента	84445552200-Греков
Ном_пасп_кассира	2000557722-Трубкова
Стоимость_заправки	3494.568
Способ_оплаты	Картой

Сохранить

Рисунок 16 - Автоматический подсчет суммы

2.10. Для изменения существующих записей, если у пользователя есть соответствующий уровень доступа, необходимо нажать на кнопку «Изменить данные». Тогда запустится форма изменения данных, а основная форма заблокируется до момента закрытия текущей формы.



Условия замены	
Имя столбца	Значение
Фамилия	Бабкина

Заменить	
Имя столбца	Значение
Фамилия	

Произвести замену

Рисунок 17 -Форма изменения данных

Форма состоит из двух частей, сверху – условия замены, представлены в виде имени столбца и значения, а снизу – в каком и на что заменить текущее значение.

Сразу после открытия форма загружает все названия столбцов в два левых поля. А как только пользователь выбирает имя столбца в условиях замены, программы посылает запрос на SQL Server и заполняет поле «Значение» в условиях замены полученными вариантами значений.

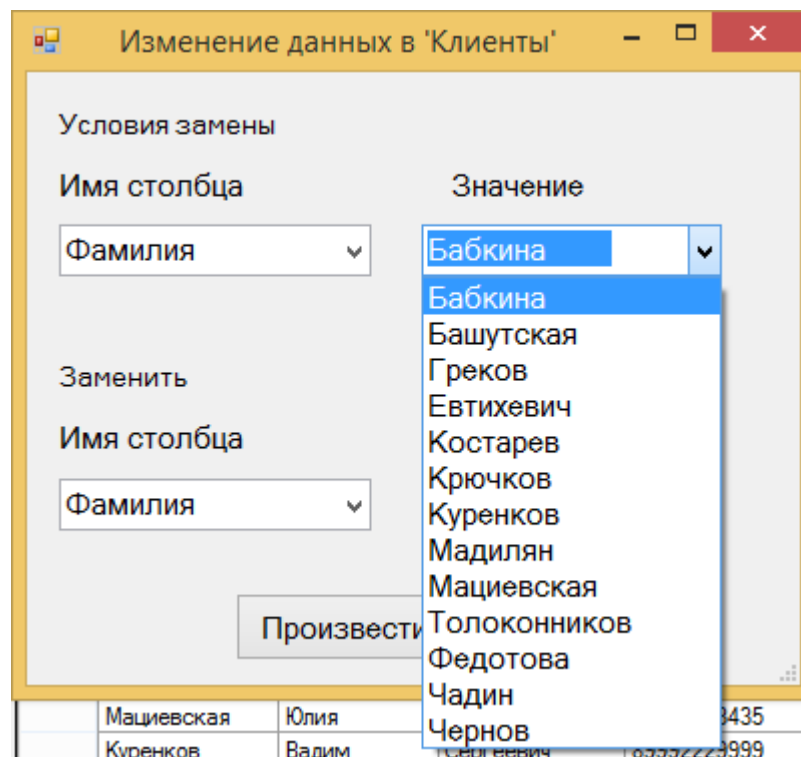


Рисунок 18 - Варианты значений

Заполнив или выбрав все значение нужно нажать на «Произвести замену», и тогда программа сформирует запрос и отошлет его на сервер, после появится окно с сообщением «Данные успешно изменены». В случае, если пользователь забыл ввести значение в поле, программа сообщит ему об этом.

2.11. Чтобы удалить записи, необходимо в основной форме программы отфильтровать те записи, которые мы хотим удалить, а после нажать на кнопку удалить, если уровень доступа позволяет это сделать. После нажатия на кнопку, появится предупреждение, и, если пользователь нажмет на «ОК», записи будут удалены.

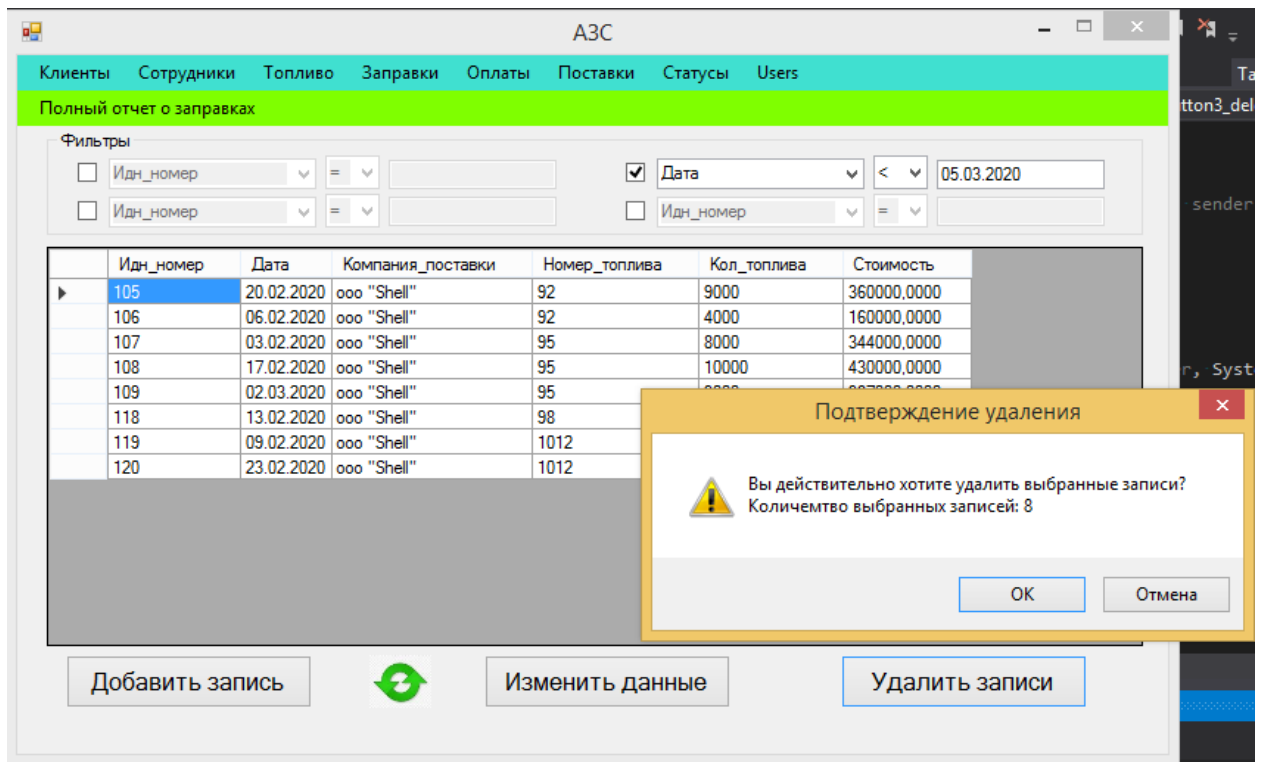


Рисунок 19 - Удаление данных

3. Листинги клиентского приложения, написанного на языке C++\CLI.

3.1. Файл «Functional.h» содержащий объявление класса «Functional» содержащий все нужные функции для выполнения SQL запросов и получения, изменения данных на сервере.

Листинг «Functional.h»:

```
#pragma once

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Data::SqlClient;

enum accesses { director=0, admin=1, cashier=2 }; //возможные уровни доступа

ref struct FilterType //структура хранения значений фильтров
{
    int count; //число фильров (условий)
    array<String^>^ value; //условия поиска

    //конструкторы
    FilterType() :count(0) {};
    FilterType(int n, array<String^>^ val) :count(n), value(val) {}
};

ref class Functional
{
public:
```

```

//матрица доступа (Номер таблицы,уровень доступа, добавление/редактирование/удаление)
static const array<const bool, 3>^ matrixAccess =
{ {{0,0,0},{0,1,1},{1,0,0}},
  {{1,1,1},{0,0,0},{0,0,0}},
  {{1,0,1},{0,1,0},{0,0,0}},
  {{0,0,0},{0,1,1},{1,0,0}},
  {{0,0,0},{0,1,1},{1,0,0}},
  {{0,0,0},{1,1,1},{0,0,0}},
  {{0,0,0},{1,1,1},{0,0,0}},
  {{0,0,0},{1,1,1},{0,0,0}},
  {{0,0,0},{0,0,0},{0,0,0}} };

// массив, содержащий названия всех таблиц
static const array<String^>^ allTableNames = { "dbo.Клиенты","dbo.Сотрудники",
"dbo.Топливо","dbo.Заправки","dbo.Оплаты","dbo.Поставки","dbo.Статусы","dbo.Users",
"dbo.Полный_отчет" };

static String^ user; //логин пользователя
static accesses access; //уровень доступа текущего входа
static SqlConnection^ conn; //подключение к серверу
static SqlConnectionStringBuilder^ connStringBuilder; // строка для подключения
static String^ tableName; //название текущей таблицы
static int tabNum; //номер текущей таблицы

Functional(){ } //пустой конструктор

//функция установки соединения с сервером
static void ConnectToDB();
//поиск пары логин-пароль в таблице "Users"
static bool Functional::FindUser(String^ log, String^ pas);

//static void FillSource(DataGridView^ DataGV);
//получение текущей таблицы с данными с учетом фильтров
static void FillSource(DataGridView^ DataGV, FilterType^ wheres);

//получение таблицы информации о текущей таблице
static DataTable^ GetDataTable();

//получение списка имен столбцов текущей таблицы
static ListBox::ObjectCollection^ FillColumnName();
//получение списка операций сравнения
static ListBox::ObjectCollection^ Filloperation();
//создание таблицы "Полный отчет о заправках"
static void CreateFullReportTable();
//удаление таблицы "Полный отчет о заправках"
static void DeleteFullReportTable();

//получение значений столбца текущей таблицы
static ListBox::ObjectCollection^ FillDataColumn(String^ columnName);
//получение значений столбца выбранной таблицы
static ListBox::ObjectCollection^ FillDataColumn(String^ columnName, String^
theTableName);
//выполнение запроса на изменения данных
static void Updata(String^ colNameWhere, String^ valWhere, String^ colNameSet,
String^ valSet);

//проверка, является ли столбец текущей таблицы внешним ключем
static bool isForgetKey(String^ columName);
//получение значений внешнего ключа для столбца
static ListBox::ObjectCollection^ GetValuesForgetKey(String^ columName);
//расчет полной стоимости заправки для конкретного значения поля "Идн_номер"
static double GetFullPrice(String^ IdRefueling);
//получение скидки в процентах для конкретного значения поля "Номер_телефона" клиента
static int GetPercentageDiscount(String^ phoneNumb);

```



```

//выполнение запроса на добавления, где в массиве передаются все
//значения полей, кроме поля,имеющего свойство AutoIncrement
static void InsertData(array<String^>^);

//Выполнения запроса удаления всех отфильтрованных записей
static void DeleteData(FilterType^ wheres);
};

```

3.2. Единственное событие формы авторизации, нажатие на кнопку «Вход», вызывает функцию, описанную в файле «AuthorizationForm.cpp», она осуществляет проверку логина и пароля и в случае успеха запускает основную форму и скрывает форму авторизации.

Листинг функции «button1_intobd_Click»:

```

System::Void AuthorizationForm::button1_intobd_Click(Object^ sender, EventArgs^ e)
{
    if (Functional::FindUser(this->textBox1_login->Text, this->textBox2_password->Text))
    {
        Tables ^Tables1 = gcnew Tables(this);
        Tables1->Show();
        this->Visible = false;
    }
    else MessageBox::Show("Пароль или логин введен неверно!", "Ошибка входа");
}

```

3.3. Листинг обработчиков событий основной формы «Tables.h»:

```

//Загрузка формы
private: System::Void Tables_Load(System::Object^ sender, System::EventArgs^ e)
{
    //вызвать событие выбора одной из таблиц в меню
    ToolStripMenuItem_Click(menuStrip1->Items[0], e);
    for (int j = FCOUNT*2; j < FCOUNT*3; j++)
        //заполнение списков выбора полей фильтра операциями сравнения
        safe_cast<ComboBox^>(groupBox1_filter->Controls[j])->DataSource =
Functional::Filloperation();

    //включение или отключение таблиц из меню в зависимости от уровня доступа
    if (Functional::access == 0 || Functional::access == 1) {
        this->EmployeesToolStripMenuItem->Visible = true;
        this->usersToolStripMenuItem->Visible = true;
    }
    else {
        this->EmployeesToolStripMenuItem->Visible = false;
        this->usersToolStripMenuItem->Visible = false;
    }
}

//функция, выполняющаяся при выборе таблицы в пункте меню
//аргументы - номер таблицы и верхнее|нижнее меню
private: void FuncForEachMenu(int numTable, System::Windows::Forms::MenuStrip^ ms)
{
    Functional f;
    f.tableName = f.allTableNames[numTable]; //запоминание текущей таблицы
    f.tabNum = numTable; //запоминание номера текущей таблицы
    //создание таблицы, если выбран "Полный отчет о запавках"
    if (numTable == 8) f.CreateFullReportTable();
}

```

```

        f.FillSource(dataGridView1, gcnew FilterType); //получение таблицы данных
        for (int j = FCOUNT; j < FCOUNT*2; j++)
            //заполнение полей фильтров с названиями столбцов
            safe_cast<ComboBox>(groupBox1_filter->Controls[j])->DataSource =
f.FillColumnName();

        //включение или отключение кнопок добавления, редактирования и просмотра
        this->button1_addData->Enabled = f.matrixAccess[numTable, f.access, 0];
        this->button2_changeData->Enabled = f.matrixAccess[numTable, f.access, 1];
        this->button3_deleteData->Enabled = f.matrixAccess[numTable, f.access, 2];
    }

    //Сборка каждой группы полей включенных фильтров в цельное условие
    private: FilterType^ GetWheres()
    {
        //создание пустого массива условий
        array<String>^ wheres = gcnew array<String>(FCOUNT);
        int countFindColumns = 0; //число активных фильтров
        for (int j = 0; j < FCOUNT; j++)
            if (safe_cast<CheckBox>(groupBox1_filter->Controls[j])->Checked)
            {
                if (groupBox1_filter->Controls[j + FCOUNT * 3]->Text == "")
                    MessageBox::Show("Поле фильтра не заполнено!", "Ошибка");
                else {
                    wheres[countFindColumns] = groupBox1_filter->Controls[j + FCOUNT]->Text
                    + " " + groupBox1_filter->Controls[j + FCOUNT * 2]->Text
                    + " '" + groupBox1_filter->Controls[j + FCOUNT * 3]->Text + "'";
                    countFindColumns++;
                }
            }
        //создание и возвращение объекта структуры фильтра
        return gcnew FilterType(countFindColumns, wheres);
    }

    //При клике на один из пунктов меню, вызывает функцию "FuncForEachMenu"
    private: System::Void ToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e)
    {
        //проверка, был ли выбран один из пунктов верхнего меню
        for (int j = 0; j < menuStrip1->Items->Count; j++)
            if (sender->Equals(menuStrip1->Items[j]))
                FuncForEachMenu(j, menuStrip1);

        //проверка, был ли выбран один из пунктов нижнего меню
        for (int j = 0; j < menuStrip2->Items->Count; j++)
            if (sender->Equals(menuStrip2->Items[j]))
                FuncForEachMenu(j + menuStrip1->Items->Count, menuStrip2);
    }

    //Закрытие формы
    private: System::Void Tables_FormClosed(System::Object^ sender,
System::Windows::Forms::FormClosedEventArgs^ e)
    {
        if (parentForm != nullptr) parentForm->Visible = true;
        Functional::DeleteFullReportTable();
    }

    //Нажатие на кнопку "Добавить запись"
    private: System::Void button1_addData_Click(System::Object^ sender,
System::EventArgs^ e)
    {
        AddRecordForm^ AddForm1 = gcnew AddRecordForm();
        AddForm1->ShowDialog();
    }

```

```

        //Нажатие на кнопку "Изменить данные"
        private: System::Void button2_changeData_Click(System::Object^ sender,
System::EventArgs^ e)
        {
            ChangeDataForm^ changeData1 = gcnew ChangeDataForm();
            changeData1->ShowDialog();
        }

        //Нажатие на картинку с зеленой стрелочкой для обновления данных
        private: System::Void pictureBox1_Click(System::Object^ sender,
System::EventArgs^ e)
        {
            Functional::FillSource(dataGridView1, GetWheres());
        }

        //Очистка поля ввода фильтра, при изменении имени столбца
        private: System::Void comboBoxs_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e)
        {
            for (int j = FCOUNT; j < FCOUNT*2; j++)
                if (sender->Equals(groupBox1_filter->Controls[j]))
                    groupBox1_filter->Controls[j+ FCOUNT*2]->Text = "";
        }

        //Нажатие на кнопку "Удалить записи"
        private: System::Void button3_deleteData_Click(System::Object^ sender,
System::EventArgs^ e)
        {
            if (MessageBox::Show("Вы действительно хотите удалить выбранные записи?"
+ "\nКоличество выбранных записей: " + dataGridView1->RowCount,
"Подтверждение удаления", MessageBoxButtons::OKCancel,
MessageBoxIcon::Exclamation) == ::DialogResult::OK)
            {
                Functional::DeleteData(GetWheres());
                MessageBox::Show("Готово!", "Результат удаления");
            }
        }

        //Включение или отключение фильтра, при нажатии на "checkBox"
        private: System::Void checkBoxs_CheckedChanged(System::Object^ sender,
System::EventArgs^ e)
        {
            for (int j = 0; j < FCOUNT; j++)
            {
                if (sender->Equals(groupBox1_filter->Controls[j]))
                {
                    bool TF = safe_cast<CheckBox^>(groupBox1_filter->Controls[j])->Checked;
                    groupBox1_filter->Controls[j + FCOUNT]->Enabled = TF;
                    groupBox1_filter->Controls[j + FCOUNT*2]->Enabled = TF;
                    groupBox1_filter->Controls[j + FCOUNT*3]->Enabled = TF;
                }
            }
        }

```

3.4. Листинг обработчиков событий формы добавления записей «AddRecordForm.h»:

```

//Загрузка формы
private: System::Void AddRecordForm_Load(System::Object^ sender,
System::EventArgs^ e)
{
    //получение таблицы с информацией о текущей таблице
    DataTable^ tableInfo = Functional::GetDataTable();

```

```

columnCount = tableInfo->MinimumCapacity;//число столбцов текщей таблицы
//размер формы
this->ClientSize = System::Drawing::Size(480, 70 + columnCount * 30);
//Изменение положения кнопки "Сохранить"
this->button1_saveRecord->Location = System::Drawing::Point(180, 20 +
columnCount * 30);
//Изменение имени формы
this->Text = L"Добавление записей" + " в '" + Functional::tableName -
>Split('.')[1] + "'";

for (int j = 0; j < columnCount; j++)
{
    //Добавление названия столбца текущей таблицы на форму
    Label^ label1 = gcnew Label;
    label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif",
        12, FontStyle::Regular, GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
    label1->TextAlign = System::Drawing::ContentAlignment::MiddleRight;
    label1->Location = System::Drawing::Point(10, 10+j*30);
    label1->Size = System::Drawing::Size(200, 20);
    this->Controls->Add(label1);

    DataRow^ row = tableInfo->Rows[j];
    label1->Text = row["ColumnName"]->ToString();

    //Если столбец является внешним ключем
    if (Functional::isForeignKey(row["ColumnName"]->ToString()))
    {
        //создадим поле с выбором
        ComboBox^ comboBox1 = gcnew ComboBox;
        //загрузим значения внешнего ключа
        comboBox1->DataSource = Functional::GetValuesForKey(row["ColumnName"]->ToString());
        //присвоим элементу имя
        comboBox1->Name = L"itemBox0" + j.ToString();

        //Для полей "Идн_номер" и "Ном_тел_клиента" добавляем событие, которое
        //позволит рассчитать сумму оплаты при каждом изменении одного из этих полей
        if (Functional::tableName == "dbo.Оплаты" &&
            (row["ColumnName"]->ToString() == "Идн_номер"
            || row["ColumnName"]->ToString() == "Ном_тел_клиента"))
        {
            comboBox1->SelectedIndexChanged += gcnew
System::EventHandler(this, &AddRecordForm::comboBoxs_SelectedIndexChanged);
        }
        //добавим элемент в форму
        this->Controls->Add(comboBox1);
    }
    else
    {
        //создадим обычное поле для ввода
        TextBox^ textBox1 = gcnew TextBox;
        //присвоим элементу имя
        textBox1->Name = L"itemBox0" + j.ToString();

        //запомним имя поля с суммой, для дальнейшего расчета в нее
        if (row["ColumnName"]->ToString() == "Стоимость_заправки")
            nameTextboxSum = textBox1->Name;

        //добавим элемент в форму
        this->Controls->Add(textBox1);
    }
}
//Добавим некоторые параметры элементов

```

```

        Controls["itemBox0" + j.ToString()]->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,static_cast<System::Byte>(204)));
        Controls["itemBox0" + j.ToString()]->Location =
System::Drawing::Point(240, 8+j*30);
        Controls["itemBox0" + j.ToString()]->Size =
System::Drawing::Size(200, 24);
        //если поле имеет свойство AutoIncrement, то выключим поле
        if (row["IsAutoIncrement"]->ToString() == "True")
        {
            Controls["itemBox0" + j.ToString()]->Text = "*autoIncrement";
            Controls["itemBox0" + j.ToString()]->Enabled = false;
        }
    }
}

//Нажатие на кнопку "Сохранить"
private: System::Void button1_saveRecord_Click(System::Object^ sender,
System::EventArgs^ e)
{
    //создадим массив для записи значений всех полей
    array<String^>^ textboxlist = gcnew array<String^>(columnCount);
    for (int j = 0; j < columnCount; j++)
    {
        //запишем значение поля в массив
        textboxlist[j] = Controls["itemBox0" + j.ToString()]->Text->Split('-')[0];
        if (textboxlist[j] == "")
        {
            MessageBox::Show("Заполните все поля!", "Ошибка");
            return;
        }
    }
    //Отправим запрос на добавление
    Functional::InsertData(textboxlist);
    MessageBox::Show("Запись была успешно добавлена!", "Результат операции");
    if (Functional::tableName == "dbo.Оплаты")
        Controls[nameTextboxSum]->Text = "";
}

//Обработчик события изменения полей "Идн_номер" или "Ном_тел_клиента"
private: System::Void comboBoxs_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e)
{
    //расчет суммы
    double price = Functional::GetFullPrice(Controls["itemBox00"]->Text)*0.01*
(100 - Functional::GetPercentageDiscount(Controls["itemBox03"]->Text->Split('-')[0]));
    //замена запятой на точку в поле суммы
    Controls[nameTextboxSum]->Text = Convert::ToString(price)->Replace(",", ".");
}

```

3.5. Листинг обработчиков событий формы изменения данных «ChangeDataForm.h»:

```

//Загрузка формы
private: System::Void ChangeData_Load(System::Object^ sender, System::EventArgs^ e)
{
    this->Text = L"Изменение данных" + " в '" + Functional::tableName->Split('.')[1] + "'";
    //Заполнение вариантов полей названиями столбцов
    comboBox1_columnName->DataSource = Functional::FillColumnName();
    comboBox3_columnName->DataSource = Functional::FillColumnName();
}

```

```

//Выбор значения в поле Условие -> Имя столбца
private: System::Void comboBox1_columnName_SelectedIndexChanged(System::Object^
sender, System::EventArgs^ e)
{
    //Заполнение вариантов поля данными
    String^ columnName = Convert::ToString(comboBox1_columnName->SelectedItem);
    comboBox2_Value->DataSource = Functional::FillDataColumn(columnName);
}

//Нажатие на кнопку "Произвести замену"
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (this->textBox1_changedValue->Text == "")
        MessageBox::Show("Заполните все поля!", "Ошибка");
    else
    {
        //считывание данных из полей
        String^ colNameWhere = this->comboBox1_columnName->Text;
        String^ valWhere = this->comboBox2_Value->Text;
        String^ colNameSet = this->comboBox3_columnName->Text;
        String^ valSet = this->textBox1_changedValue->Text;
        //Отправка запроса на изменения данных
        Functional::Updata(colNameWhere, valWhere, colNameSet, valSet);
        MessageBox::Show("Данные были успешно изменены!", "Результат операции");
        //Обновить варианты значений столбца "Значение" в условии замены
        String^ columnName = Convert::ToString(comboBox1_columnName->SelectedItem);
        comboBox2_Value->DataSource = Functional::FillDataColumn(columnName);
    }
}

```

3.6. Листинг всех функций класса `Functional` «Functional.cpp»:

```

#include "Functional.h"

void Functional::ConnectToDB()
{
    if (connStringBuilder!=nullptr) delete connStringBuilder;
    connStringBuilder = gcnew SqlConnectionStringBuilder();
    connStringBuilder->DataSource = "PC\\SQLEXPRESS";
    connStringBuilder->UserID = "Константин";
    connStringBuilder->Password = "12345";
    connStringBuilder->InitialCatalog = "АЗС";
    connStringBuilder->PersistSecurityInfo = false; //защита конфиденц даанных в
строке подключения вклчена
    connStringBuilder->IntegratedSecurity = false;
    if (conn != nullptr) delete conn;
    conn = gcnew SqlConnection(Convert::ToString(connStringBuilder));
}

bool Functional::FindUser(String^ log,String^ pas)
{
    bool result = false;
    try {
        ConnectToDB();

        String^ cmdText = "SELECT * FROM dbo.Users";
        SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
        conn->Open();
        SqlDataReader^ reader = cmd->ExecuteReader();

        while (reader->Read())
        {
            if (reader["Логин"]->ToString() == log && reader["Пароль"]->
ToString() == pas)

```

```

        {
            user = reader["Логин"]->ToString();
            access = accesses(Convert.ToInt32(reader["Уровень_доступа"])-
>ToString()));
            result = true;
            break;
        }
    }
}
finally{
    if (conn != nullptr)
        conn->Close();
}
return result;
}

DataTable^ Functional::GetDataTable()
{
    DataTable^ dataTabl;
    try {
        ConnectToDB();

        String^ cmdText = "SELECT * FROM " + tableName;
        SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
        conn->Open();
        SqlDataReader^ reader = cmd->ExecuteReader();
        dataTabl = reader->GetSchemaTable();
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
    return dataTabl;
}

bool Functional::isForgetKey(String^ columnName)
{
    if (tableName == "dbo.Клиенты" && columnName == "Статус" ||
        tableName == "dbo.Поставки" && columnName == "Номер_топлива" ||
        tableName == "dbo.Заправки" && columnName == "Идн_номер_топлива" ||
        tableName == "dbo.Заправки" && columnName == "Ном_пасп_заправщика" ||
        tableName == "dbo.Оплаты" && columnName == "Идн_номер" ||
        tableName == "dbo.Оплаты" && columnName == "Ном_тел_клиента" ||
        tableName == "dbo.Оплаты" && columnName == "Ном_пасп_кассира")
        return true;
    return false;
}

ListBox::ObjectCollection^ Functional::GetValuesForgetKey(String^ columnName)
{
    if (tableName == "dbo.Клиенты" && columnName == "Статус")
        return FillDataColumn("Статус", "dbo.Статусы");
    if (tableName == "dbo.Поставки" && columnName == "Номер_топлива")
        return FillDataColumn("Идн_номер", "dbo.Топливо");
    if (tableName == "dbo.Заправки" && columnName == "Идн_номер_топлива")
        return FillDataColumn("Идн_номер", "dbo.Топливо");
    if (tableName == "dbo.Заправки" && columnName == "Ном_пасп_заправщика")
        return FillDataColumn("Номер_паспорта", "dbo.Сотрудники");
    if (tableName == "dbo.Оплаты" && columnName == "Идн_номер") {
        ListBox::ObjectCollection^ RefuelingsList = FillDataColumn("Идн_номер",
"dbo.Заправки");
        ListBox::ObjectCollection^ PaymentsList= FillDataColumn("Идн_номер",
"dbo.Оплаты");
        for (int j=0;j<PaymentsList->Count;j++)

```

```

        RefuelingsList->Remove(PaymentsList[j]);
        return RefuelingsList;
    }
    if (tableName == "dbo.Оплаты" && columnName == "Ном_тел_клиента")
        return FillDataColumn("Номер_телефона", "dbo.Клиенты");
    if (tableName == "dbo.Оплаты" && columnName == "Ном_пасп_кассира")
        return FillDataColumn("Номер_паспорта", "dbo.Сотрудники");
}

ListBox::ObjectCollection^ Functional::FillDataColumn(String^ columnName, String^
theTableName)
{
    try {
        ConnectToDB();
        ListBox^ dataList = gcnew ListBox();

        String^ cmdText = "SELECT * FROM " + theTableName + " ORDER BY " +
columnName;
        SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
        conn->Open();
        SqlDataReader^ reader = cmd->ExecuteReader();

        while (reader->Read())
        {
            if (reader[columnName]->ToString() != reader[0]->ToString())
                dataList->Items->Add(reader[columnName]->ToString() + "-" +
reader[0]->ToString());
            else
                dataList->Items->Add(reader[columnName]->ToString());
        }
        return dataList->Items;
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
}

void Functional::InsertData(array<String^>^ dataList)
{
    try {
        ConnectToDB();
        String^ cmdText = "INSERT INTO " + tableName + "(";
        //Name, Instruct, Unit, Vol, Name_Maker) VALUES(@NV,'a','bb',3,'dddd')";

        DataTable^ tableInfo = GetDataTable();
        int columnCount = tableInfo->MinimumCapacity;
        for (int j = 0; j < columnCount; j++)
        {
            DataRow^ row = tableInfo->Rows[j];
            if (row["IsAutoIncrement"]->ToString() == "False")
            {
                cmdText += row["ColumnName"]->ToString();
                if (j < columnCount - 1) cmdText += ", ";
            }
        }
        cmdText += ") VALUES('";

        for (int j = 0; j < columnCount; j++)
        {
            DataRow^ row = tableInfo->Rows[j];
            if (row["IsAutoIncrement"]->ToString() == "False")
            {
                cmdText += dataList[j];
            }
        }
    }
}

```



```

        if (j < columnCount - 1) cmdText += ", ";
    }
    cmdText += "'";

    SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
    conn->Open();
    cmd->ExecuteNonQuery();
}
finally{
    if (conn != nullptr)
        conn->Close();
}
}

ListBox::ObjectCollection^ Functional::FillColumnName()
{
    try {
        ConnectToDB();
        ListBox^ ListColumnNames = gcnew ListBox();

        DataTable^ tableInfo = GetDataTable();
        int columnCount = tableInfo->MinimumCapacity;
        for (int j = 0; j < columnCount; j++)
        {
            DataRow^ row = tableInfo->Rows[j];
            ListColumnNames->Items->Add(row["ColumnName"]->ToString());
        }
        return ListColumnNames->Items;
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
}

ListBox::ObjectCollection^ Functional::FillDataColumn(String^ columnName)
{
    try {
        ConnectToDB();
        ListBox^ dataList = gcnew ListBox();

        String^ cmdText = "SELECT * FROM " + tableName + " ORDER BY " + columnName;
        SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
        conn->Open();
        SqlDataReader^ reader = cmd->ExecuteReader();

        while (reader->Read())
        {
            dataList->Items->Add(reader[columnName]->ToString());
        }
        return dataList->Items;
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
}

void Functional::Udata(String^ colNameWhere, String^ valWhere, String^ colNameSet,
String^ valSet)
{
    try {

```

```

        ConnectToDB();

        String^ cmdText = "UPDATE " + tableName + " SET " + colNameSet + " = '" +
valSet;
        cmdText += "' WHERE " + colNameWhere + " = '" + valWhere + "'";
        conn->Open();

        SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
        cmd->ExecuteNonQuery();
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
}
//=====

void Functional::FillSource(DataGridView^ DataGV, FilterType^ wheres)
{
    try {
        ConnectToDB();
        String^ SqlText = "SELECT * FROM " +tableName;

        if (wheres->count != 0) SqlText += " WHERE (";
        for (int j = 0; j < wheres->count; j++)
        {
            SqlText += wheres->value[j];
            if (j < wheres->count - 1) SqlText += " AND ";
        }
        if (wheres->count != 0) SqlText += ")";

        conn->Open();
        SqlDataAdapter^ da = gcnew SqlDataAdapter(SqlText, conn);
        DataSet^ ds = gcnew DataSet();
        da->Fill(ds, tableName);
        DataGV->DataSource = ds->Tables[tableName]->DefaultView;
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
}

ListBox::ObjectCollection^ Functional::Filloperation()
{
    ListBox^ ListWithNames = gcnew ListBox();

    ListWithNames->Items->Add("=");
    ListWithNames->Items->Add("<");
    ListWithNames->Items->Add("<=");
    ListWithNames->Items->Add(">");
    ListWithNames->Items->Add(">=");
    ListWithNames->Items->Add("<>");
    return ListWithNames->Items;
}

void Functional::DeleteData(FilterType^ wheres)
{
    try {
        ConnectToDB();
        String^ SqlText = "DELETE FROM " + tableName;

        if (wheres->count != 0) SqlText += " WHERE (";
        for (int j = 0; j < wheres->count; j++)

```

```

        {
            SqlText += wheres->value[j];
            if (j < wheres->count - 1) SqlText += " AND ";
        }
        if (wheres->count != 0) SqlText += ")";

        conn->Open();
        SqlCommand^ cmd = gcnew SqlCommand(SqlText, conn);
        cmd->ExecuteNonQuery();
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
}

double Functional::GetFullPrice(String^ IdRefueling)
{
    try {
        ConnectToDB();
        ListBox^ dataList = gcnew ListBox();

        String^ cmdText =
            "SELECT Заправки.Идн_номер, Заправки.Число_литров,
Топливо.Цена_за_литр " +
            "FROM Заправки INNER JOIN " +
            "Топливо ON Заправки.Идн_номер_топлива = Топливо.Идн_номер " +
            "WHERE(Заправки.Идн_номер = " + IdRefueling + ")";
        SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
        conn->Open();
        SqlDataReader^ reader = cmd->ExecuteReader();

        double price=0;
        while (reader->Read())
        {
            price = Convert::ToDouble(reader["Число_литров"]->ToString())*
                Convert::ToDouble(reader["Цена_за_литр"]->ToString());
        }
        return price;
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
    return 0;
}

int Functional::GetPercentageDiscount(String^ phoneNumb)
{
    try {
        ConnectToDB();
        ListBox^ dataList = gcnew ListBox();

        String^ cmdText = "SELECT Статусы.Скидка " +
            "FROM Клиенты INNER JOIN " +
            "Статусы ON Клиенты.Статус = Статусы.Статус " +
            "WHERE(Клиенты.Номер_телефона = "+phoneNumb+" )";
        SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
        conn->Open();
        SqlDataReader^ reader = cmd->ExecuteReader();

        int percent=0;
        while (reader->Read())
        {

```

```

        percent = Convert.ToInt32(reader["Скидка"]->ToString());
    }
    return percent;
}
finally{
    if (conn != nullptr)
        conn->Close();
}
return 0;
}

void Functional::DeleteFullReportTable()
{
    try {
        ConnectToDB();
        String^ SqlText = "IF OBJECT_ID(N'" + allTableNames[8] + "','U') IS NOT
NULL"
+ " DROP TABLE " + allTableNames[8];
        conn->Open();
        SqlCommand^ cmd = gcnew SqlCommand(SqlText, conn);
        cmd->ExecuteNonQuery();
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
}

void Functional::CreateFullReportTable()
{
    try {
        DeleteFullReportTable();
        ConnectToDB();
        String^ SqlText = "SELECT Оплаты.Идн_номер, Оплаты.Дата, Оплаты.Время,"
+ " Клиенты.Фамилия + LEFT(Клиенты.Имя, 1) + '.' + LEFT(Клиенты.Отчество, 1) + '.' AS
ФИО_клиента,"
+ " E1.Фамилия + LEFT(E1.Имя, 1) + '.' + LEFT(E1.Отчество, 1) + '.' AS ФИО_заправщика,"
+ " Заправки.Число_литров, Топливо.Название AS Топливо, Топливо.Цена_за_литр,"
+ " CAST(Заправки.Число_литров*Топливо.Цена_за_литр AS decimal(8, 2)) AS
Полная_стоимость,"
+ " Оплаты.Стоимость_заправки AS Цена_со_скидкой, Статусы.Скидка,"
+ " E2.Фамилия + LEFT(E1.Имя, 1) + '.' + LEFT(E1.Отчество, 1) + '.' AS ФИО_кассира"
+ " INTO[" + allTableNames[8]->Split('.')[1] + "]"
+ " FROM
        Заправки INNER JOIN"
+ " Оплаты ON Заправки.Идн_номер = Оплаты.Идн_номер INNER JOIN"
+ " Клиенты ON Оплаты.Ном_тел_клиента = Клиенты.Номер_телефона INNER JOIN"
+ " Сотрудники E1 ON Заправки.Ном_пасп_заправщика = E1.Номер_паспорта INNER JOIN"
+ " Сотрудники E2 ON Оплаты.Ном_пасп_кассира = E2.Номер_паспорта INNER JOIN"
+ " Статусы ON Клиенты.Статус = Статусы.Статус INNER JOIN"
+ " Топливо ON Заправки.Идн_номер_топлива = Топливо.Идн_номер";
        conn->Open();
        SqlCommand^ cmd = gcnew SqlCommand(SqlText, conn);
        cmd->ExecuteNonQuery();
    }
    finally{
        if (conn != nullptr)
            conn->Close();
    }
}
}

```

4. Реализованные типы подключения

4.1. Задание

Требуется реализация трех вариантов подключения:

- а) База данных, подключенная к серверу, и клиентское приложение располагаются в пределах одного компьютера, при этом доступ к базе данных осуществляется с одного компьютера, (размещающего и БД, и сервер, и клиентское приложение).
- б) База данных, подключенная к серверу, и клиентское приложение располагаются в пределах одного компьютера, на который установлена сетевая операционная система (например, Microsoft Windows Server), доступ к базе данных осуществляется с удалённых рабочих станций (других компьютеров) с применением технологии RDP (удаленный рабочий стол). Примечание: администрирование БД здесь осуществляется средствами сетевой операционной системы.
- в) База данных, подключенная к серверу, и клиентские приложения установлены на различных компьютерах (доступ к БД, подключенной к серверу, осуществляется с нескольких компьютеров, являющихся рабочими станциями, на которые установлены клиентские приложения), при этом сетевая операционная система на сервере может отсутствовать.

Примечание: для удаленного доступа к серверу с подключённой к нему БД, потребуется обеспечение сетевой связи между основным компьютером с БД, подключенной к серверу, и рабочими станциями; администрирование БД осуществляется через файл конфигурации.

4.2. Локальное подключение

Клиентское приложение установлено на локальном компьютере, на том же где и настроен SQL сервер с необходимой базой данных. Для работы с приложением необходимо запустить «Clientdb.exe».



Рисунок 20 - Ярлык для запуска приложения

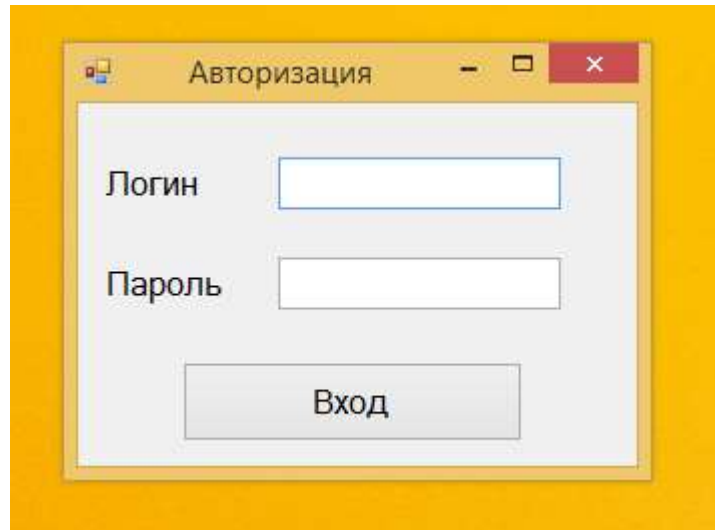


Рисунок 21 - Результат запуска приложения

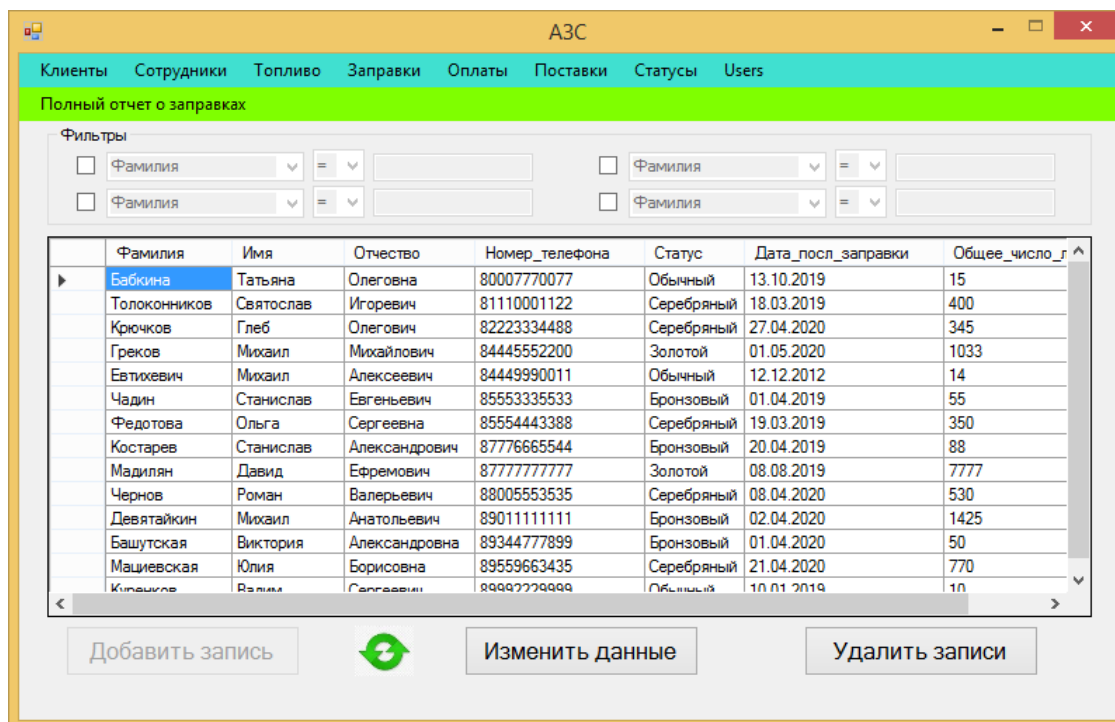


Рисунок 22 - Главное окно программы после успешной авторизации

Как видно из рисунков 21 и 22 приложение запускается и работает, подключаясь к базе данных.

4.3. Подключение с применением технологии RDP (удаленный рабочий стол)

Подключение с помощью удаленного рабочего стола подразумевает то, что пользователь приложения находится за одним персональным компьютером (ПК-клиент), который по сети подключен к другому ПК(ПК-сервер), имеющему наше приложения и сам сервер с базой данных.

Для того, чтобы реализовать данное подключение необходимо настроить сеть, в нашем случае, настроить локальную сеть. Для этого на ПК-сервере и ПК-клиенте заходим «Панель управления\Система и безопасность\Система», в левой части экрана выбираем «Настройка удаленного доступа».

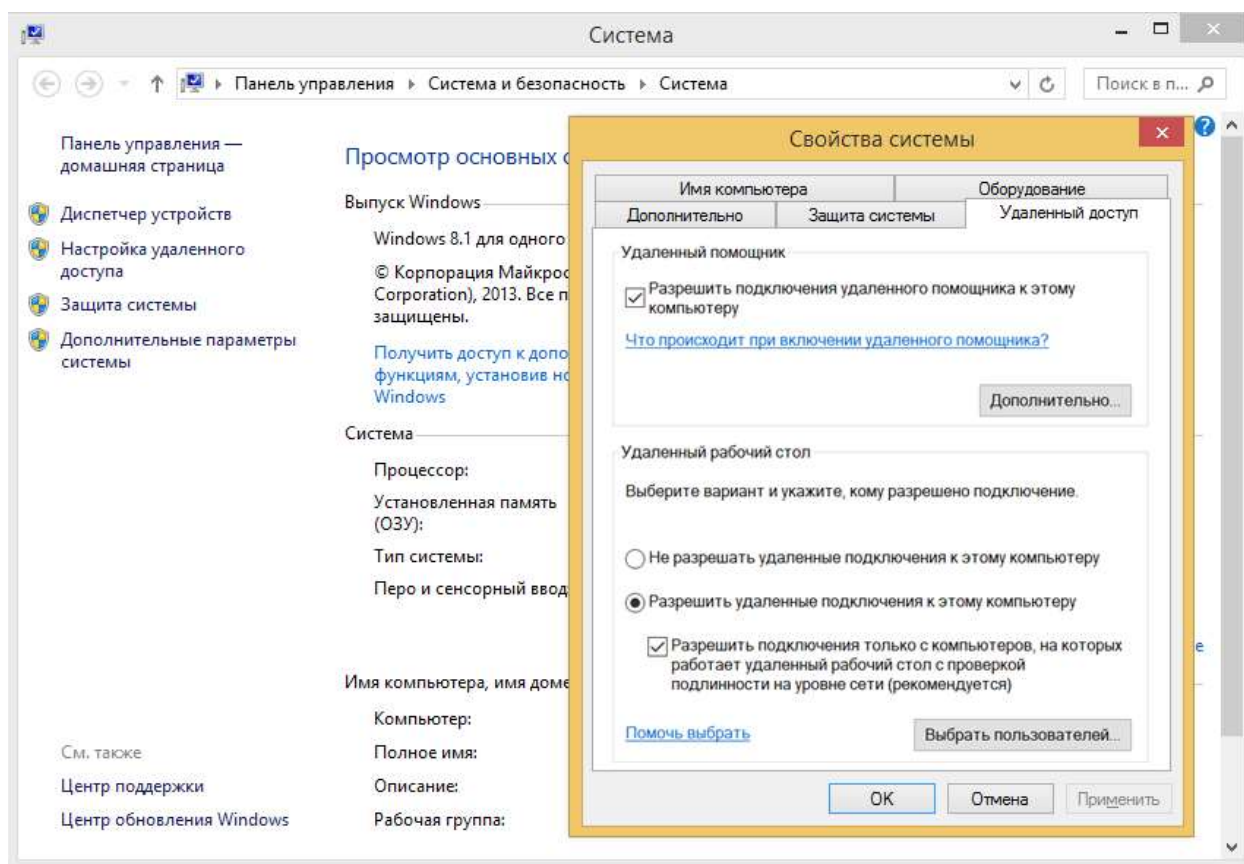


Рисунок 23 - Окно настройки удаленного доступа

В открывшемся окне выбираем разрешить удаленный доступ к этому компьютеру и нажимаем кнопку «ОК».

Далее открываем открываем «Панель управления\Сеть и Интернет\Центр управления сетями и общим доступом» и нажимаем на «Изменить дополнительные параметры общего доступа» в левой части окна.

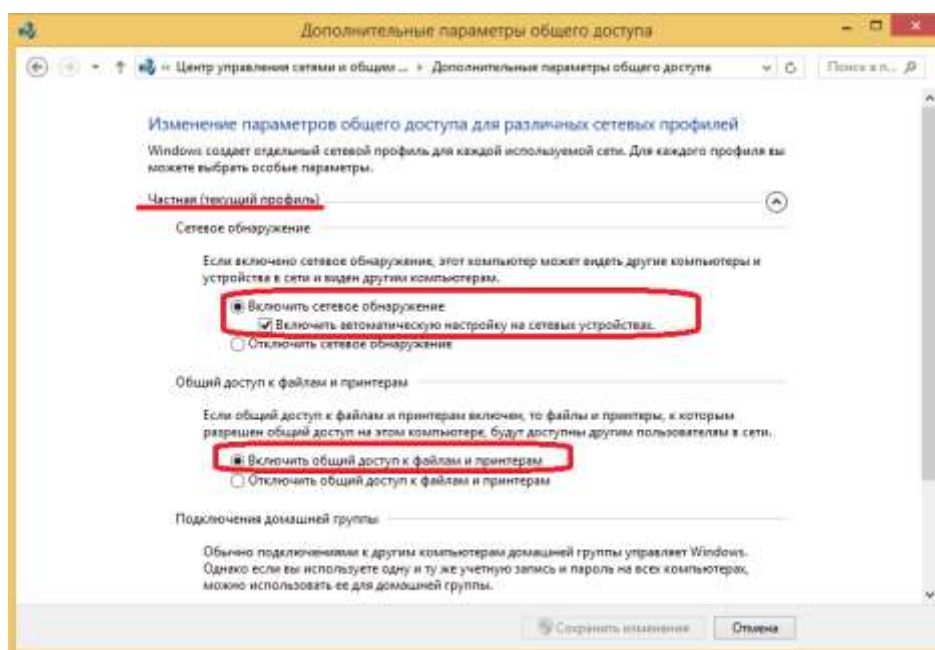


Рисунок 24 – Дополнительные параметры общего доступа

В открывшемся окне раскрываем раздел «Частная(текущий профиль)» и включаем сетевое обнаружение и общий доступ к файлам и принтерам.

Раскрываем в этом же окне «Гостевая или общедоступная» и также включаем сетевое обнаружение и общий доступ к файлам и принтерам.

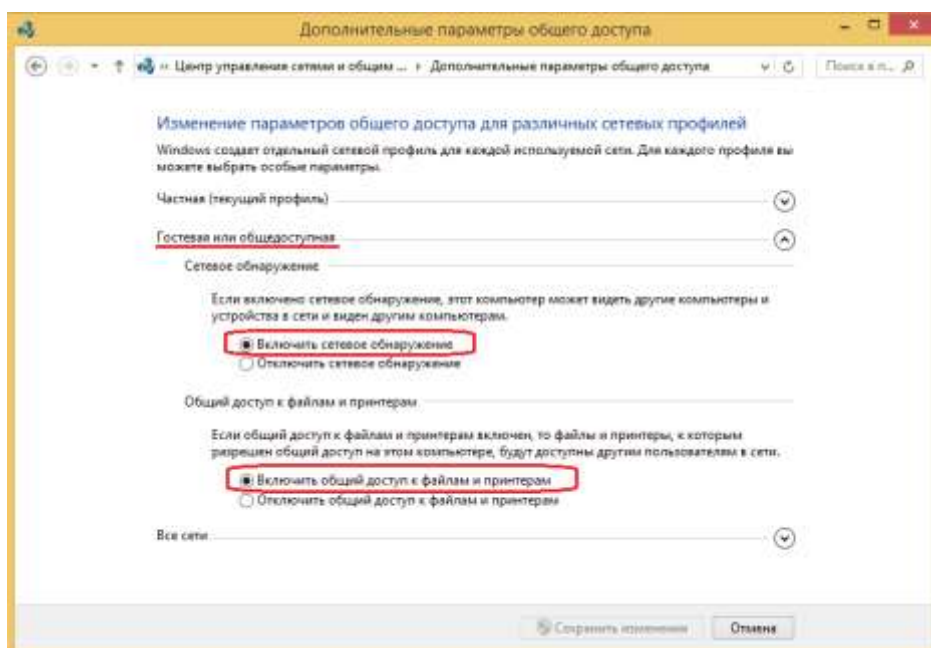


Рисунок 25 - Дополнительные параметры общего доступа

Теперь нажимаем комбинацию клавиш Win+R и в открывшемся окне пишем команду «services.msc», нажмем клавишу «Enter». После этих действий откроется окно «Службы».

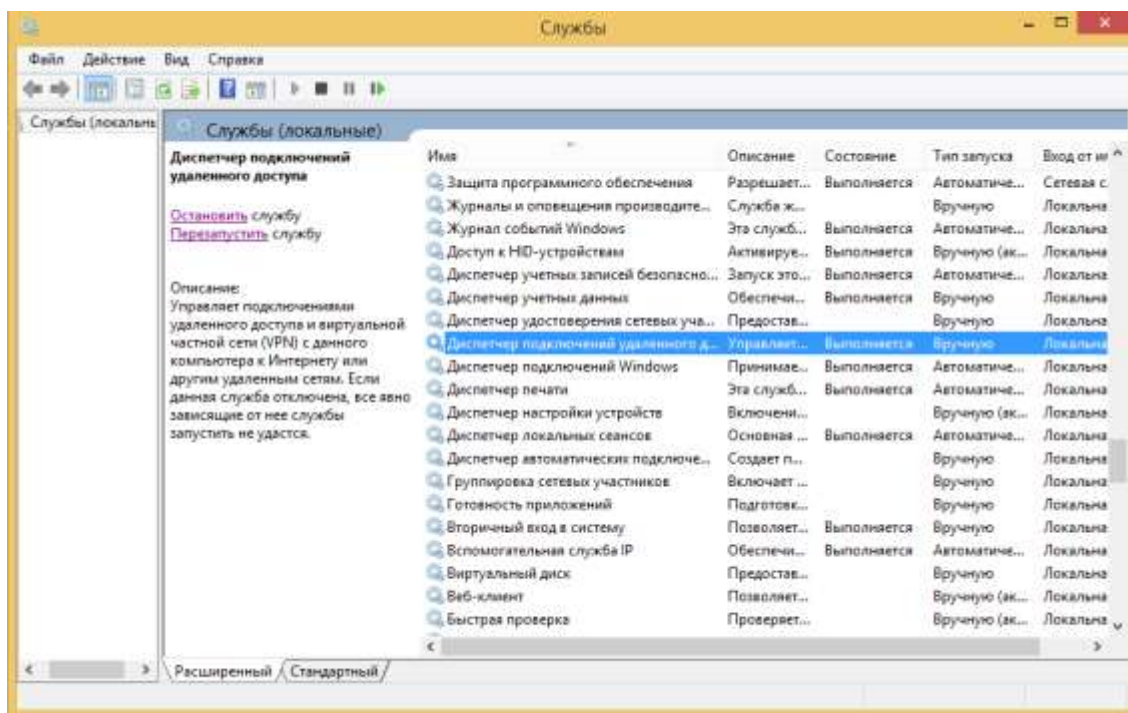


Рисунок 26 - Службы операционной системы

Здесь необходимо включить все службы, связанные с локальной сетью и удаленным доступом, а также запустить их.

Если все сделано правильно на обоих компьютерах, то в «Проводнике», выбрав в списке слева «Сеть», мы увидим оба ПК (ПК-клиент и ПК-сервер) с обоих компьютеров.

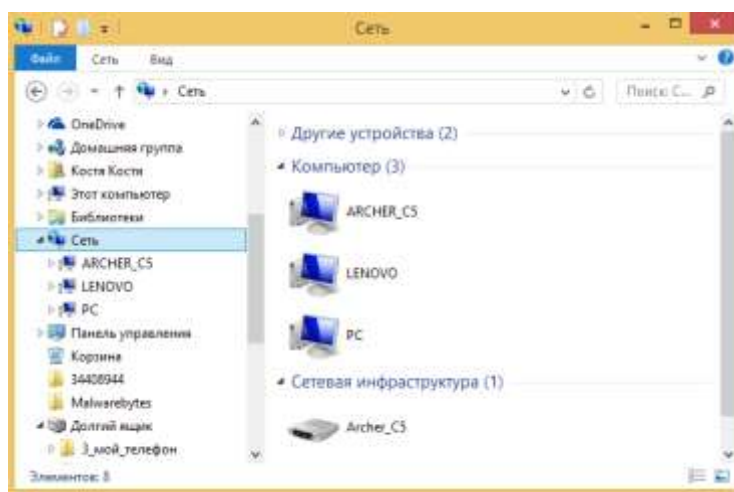


Рисунок 27 - Устройства, подключенные к сети

Как видно из рисунка ПК-клиент (LENOVO) и ПК-сервер (PC) подключены к сети и являются видимыми и доступными. Третий объект сети - это Wi-Fi-роутер, реализующий подключение устройств с помощью сети Wi-Fi.

Теперь в таком окне на ПК-клиенте (LENOVO) нажимаем правой кнопкой мыши на ПК-сервер (PC) и выбираем «Подключиться к удаленному рабочему столу».

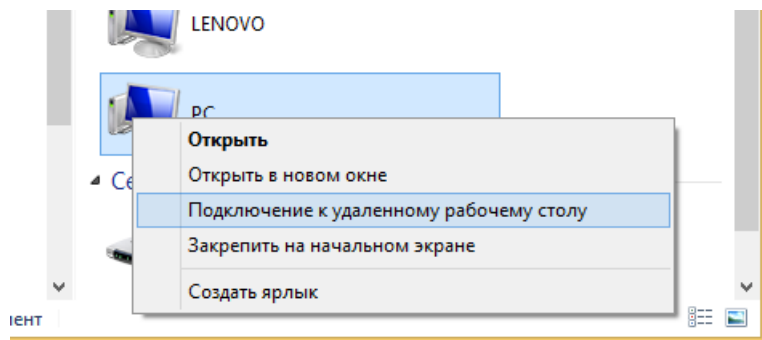


Рисунок 28 - Подключение к удаленному рабочему столу

Далее при первом подключение откроется окно, которое запросит ввести пароль текущей учетной записи Microsoft (ПК-клиента). После ввода данный и нажатия кнопки «Enter» появиться предупреждение.

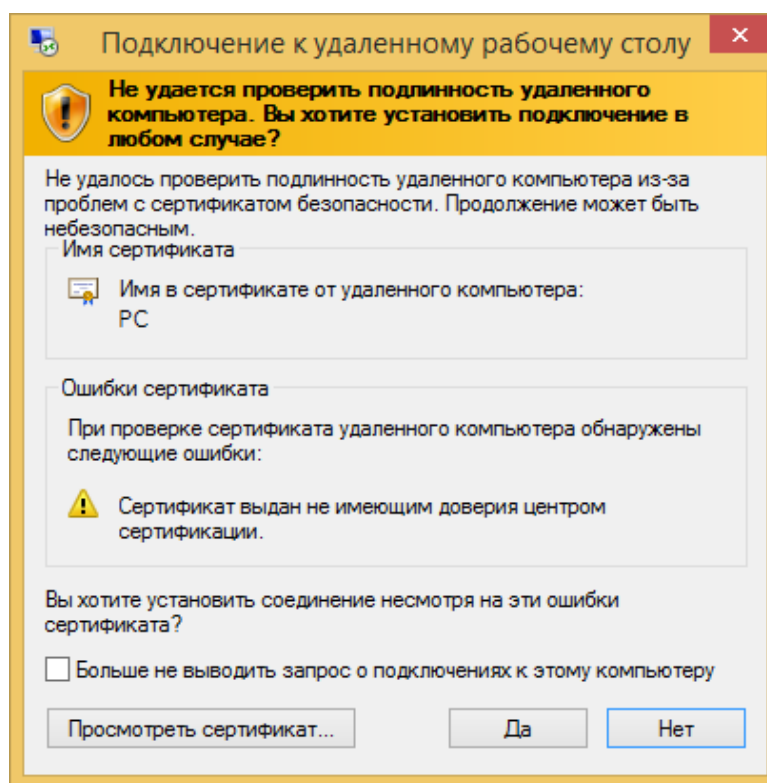


Рисунок 29 - Проверка подлинности сертификата

PC — Подключение к удаленному рабочему столу

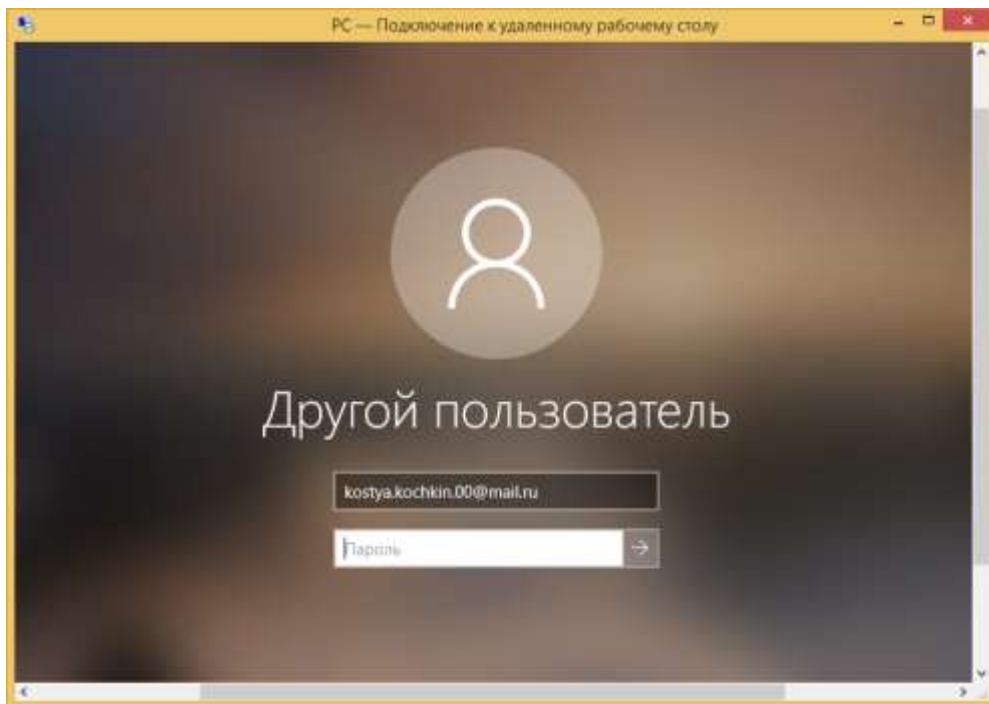


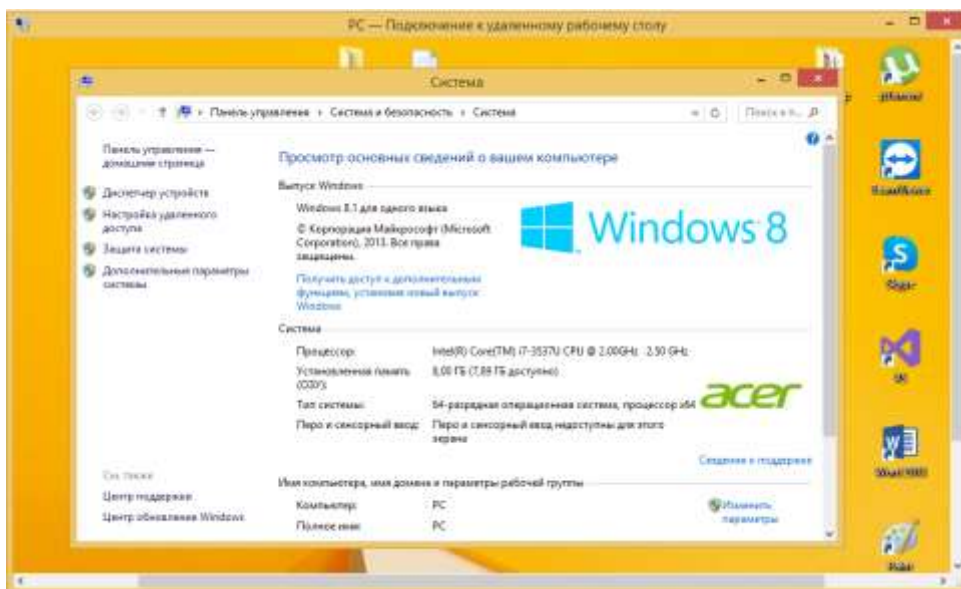
Рисунок 30 - Окно входа в систему ПК-сервера

Указываем данные пользователя, которые используются для работы на ПК-сервере и нажимаем далее. Если данные введены верно, то происходит вход в систему.

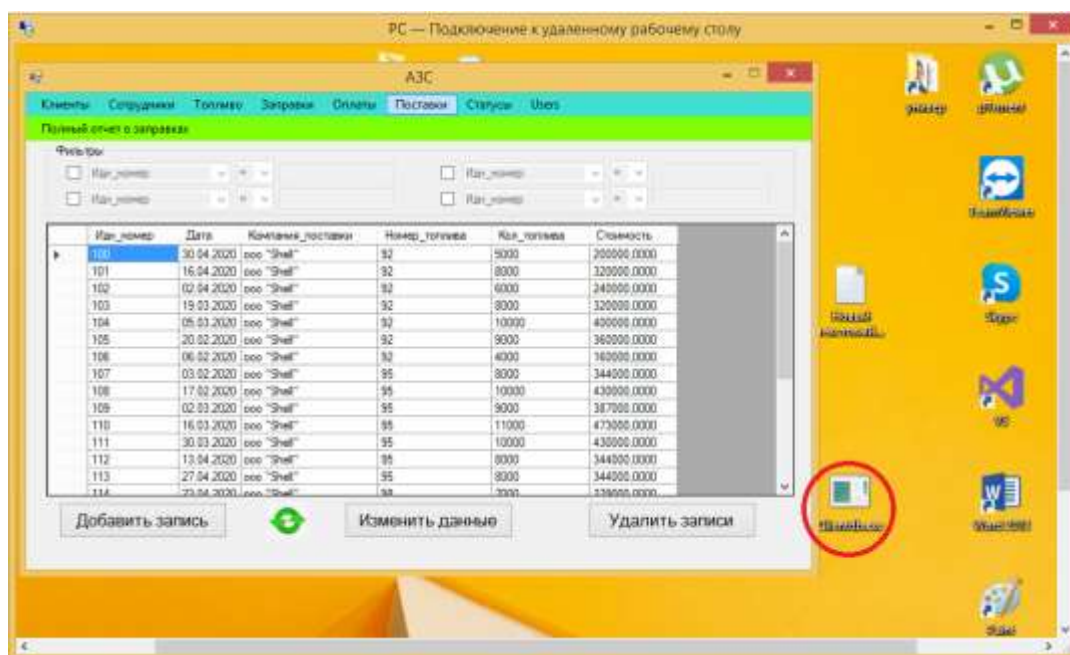


Рисунок 31 - Окно ПК-клиента содержащие систему управления ПК-сервера

Чтобы убедиться, что это действительно система ПК-сервера, можно посмотреть свойства системы.



Для запуска приложения кликаем дважды левой кнопки мыши на ярлык «Clientdb.exe». Далее в окне авторизации вводим логин и пароль и нажимаем «Войти».



36

Теперь мы можем работать с приложением на сервере удаленно со своего ПК, подключенного по локальной сети. Все возможности программы при таком подключении сохраняются, и пользователи могут пользоваться приложением со всеми теми правами, которые предусматривает их уровень доступа.

4.4. Сетевое подключение к SQL серверу через TCP

Данное подключение реализует взаимодействие ПК-клиентов, на каждом из которых установлена копия приложения, с сервером, на котором работает SQL сервер и хранятся базы, который может и не иметь собственной операционной системы с интерфейсом.

Чтобы выполнить данное соединения необходимо, чтобы все пользователи приложения были соединены локальной или глобальной сетью с сервером. Как реализовать сетевое соединения, было описано в пункте 4.3. Теперь настроим SQL сервер.

Для настройки SQL сервера нужно в меню «Пуск» найти «Диспетчер конфигурации SQL server» и запустить его.

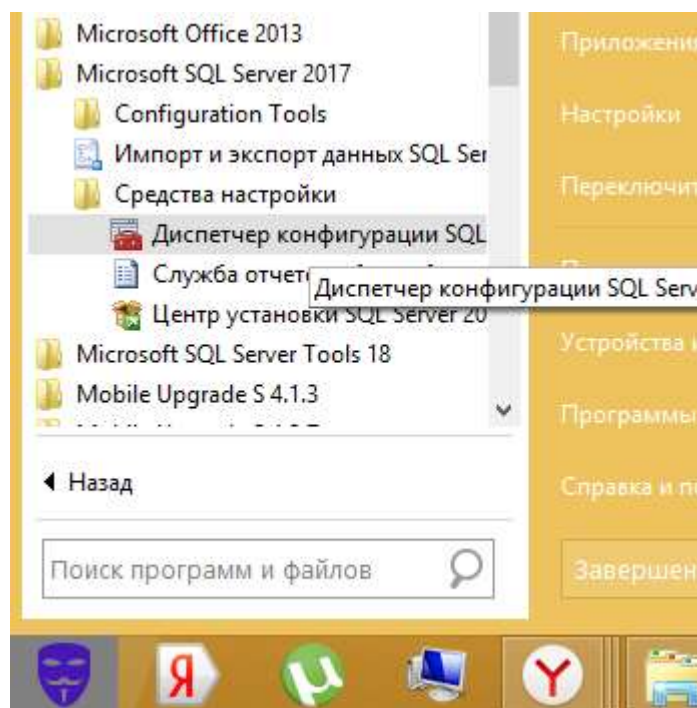


Рисунок 34 - Программы в меню "Пуск"

В левой части открывшегося окна выбираем «Сетевая конфигурация SQL server» и в нем «Протоколы для SQLEXPRESS».

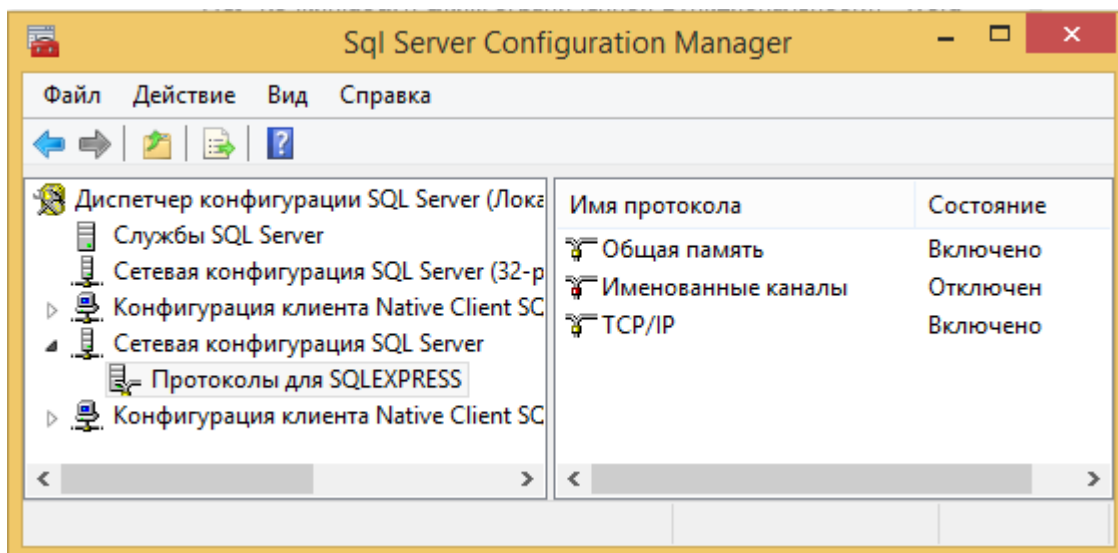


Рисунок 35 - Диспетчер конфигурации SQL server

Нажимаем правой кнопкой мыши на «TCP/IP» и выбираем «Включить». В свойствах мы можем посмотреть какой порт использует SQL сервер. Тогда стоит проверить открыт ли этот порт в данном компьютере. Если он открыт, то приступаем к следующему действию, а если нет, то сначала нужно написать правило для входящих подключений в Брандмауэре Windows.

Теперь в левом окне выбираем «Службы SQL server» и запускаем «обозреватель SQL server», чтобы сервер был доступен для работы по сети.

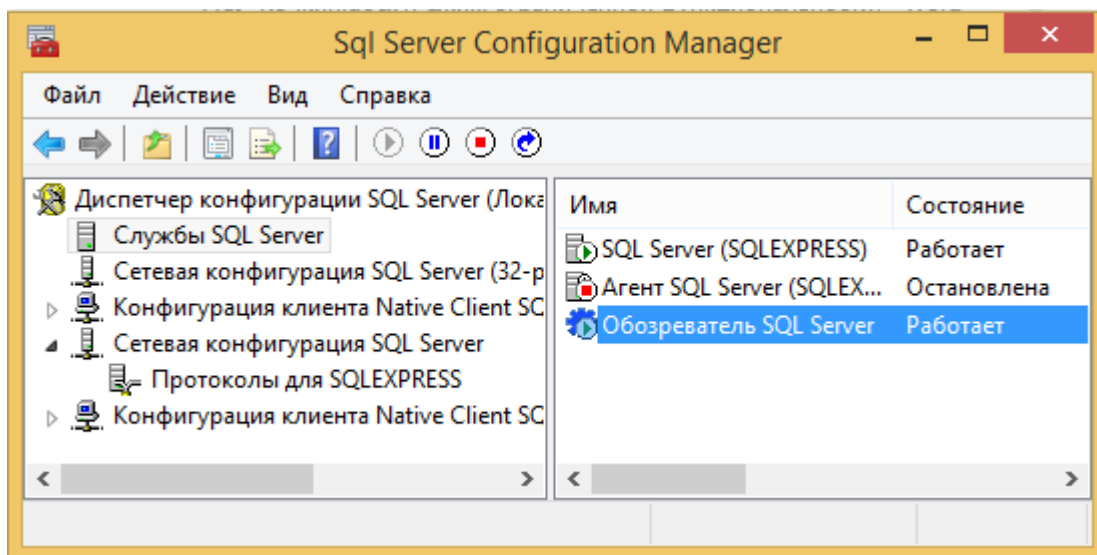


Рисунок 36 - Включение обозревателя SQL server

Осталось лишь запустить приложения на ПК-клиенте. Для это у каждого пользователя должна быть копия приложение «Client.exe».

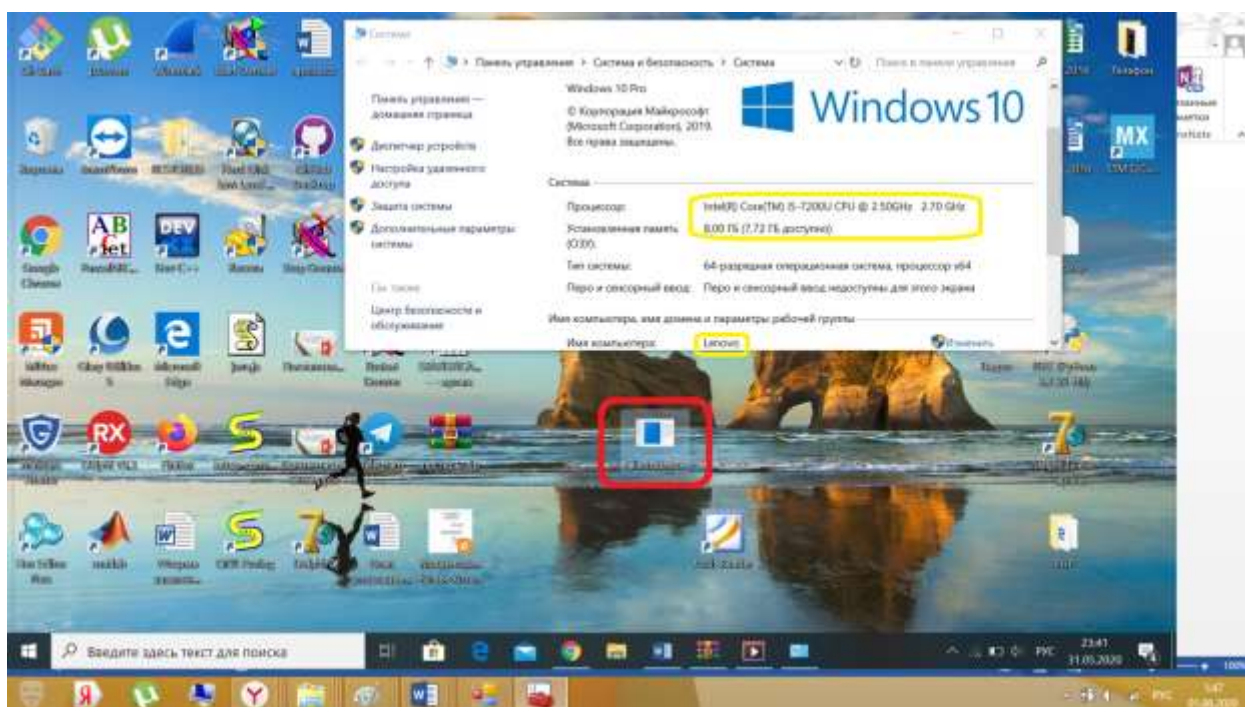


Рисунок 37 - Рабочий стол ПК пользователя с характеристиками системы

На рисунке 37 представлена система пользователя и красным выделено приложение, которое было скопировано на этот компьютер. Запустим его и убедимся, что все работает.



Рисунок 38 - Авторизация в приложении

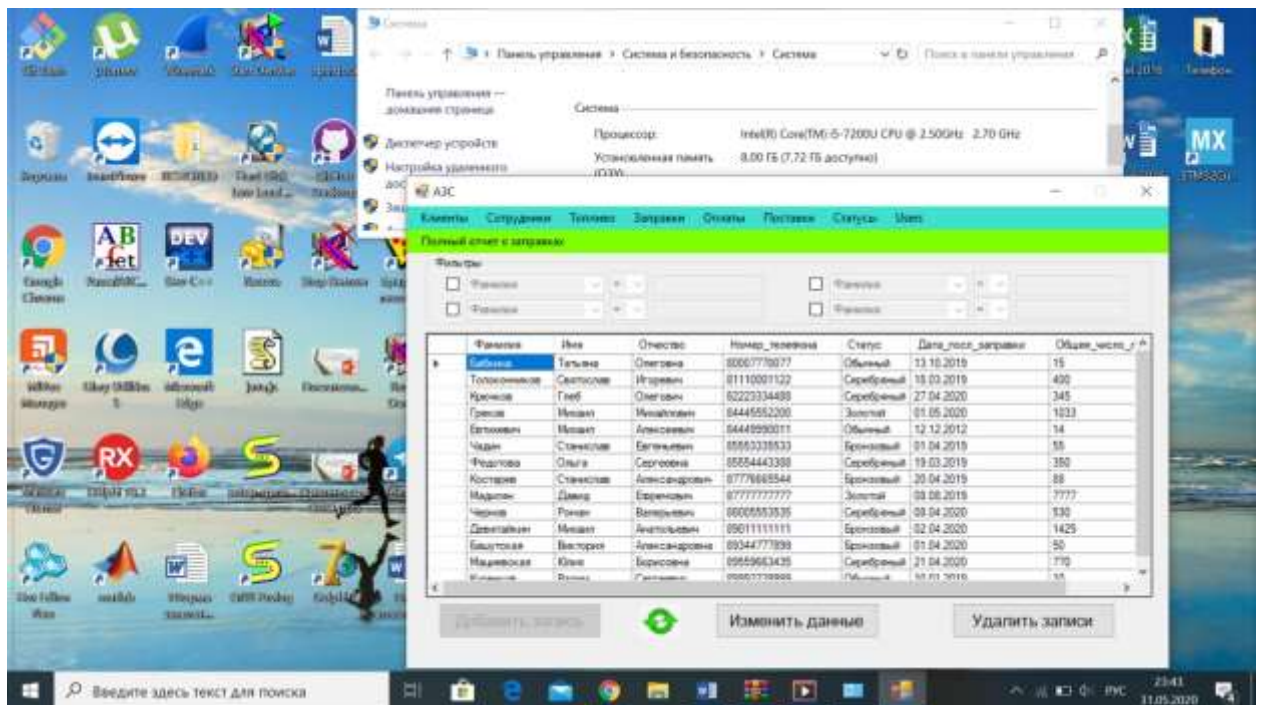


Рисунок 39 - Главное окно приложения

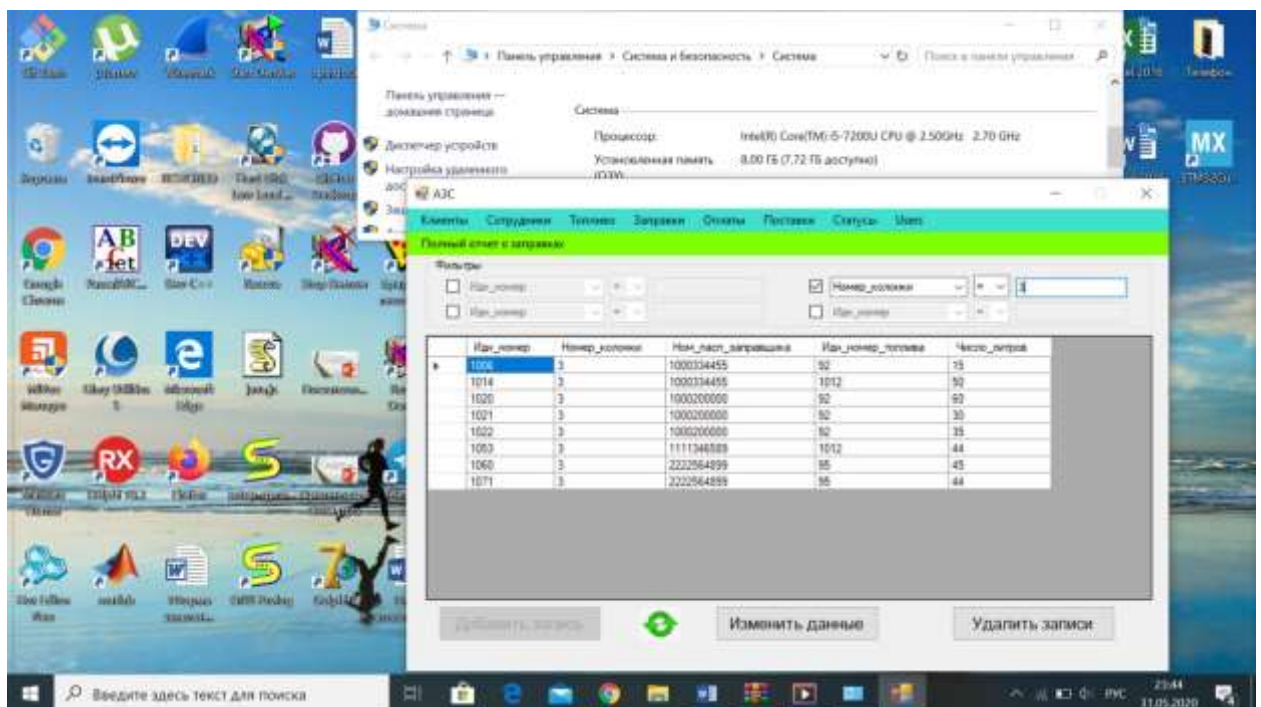


Рисунок 40 - Открытие другой таблицы и применение фильтров

Как видно из рисунков 38,39 и 40 приложение полноценно работает и взаимодействует с сервером, расположенным на другом ПК.

5. Система защиты базы данных средствами MS SQL Server Express

5.1. Задание

Обеспечить систему защиты БД средствами MS SQL Server Express.

Примечание: помимо администрирования БД (описания фиксированных ролей сервера, к которому подключена БД, пользовательских ролей и встроенных учетных записей) целесообразно предусмотреть:

- проверка подлинности в SQL Server;
- авторизаций (разрешения) в SQL Server;
- шифрование данных в SQL Server;
- безопасность исполняющей среды для байт-кода (CLR-среды) в SQL Server и прочее.

5.2. Проверка подлинности SQL server

Для того, чтобы подключаться к серверу используя проверку подлинности SQL Server необходимо создать «новое имя для входа» на сервер. Для этого откроем Microsoft SQL Server Management Studio и подключимся к серверу используя проверку подлинности Windows. В «Обозревателе объектов» нужно развернуть сервер, а в нем развернуть раздел «Безопасность», далее нажать правой кнопкой мыши на «Имена для входа» и выбрать «Создать имя для входа».

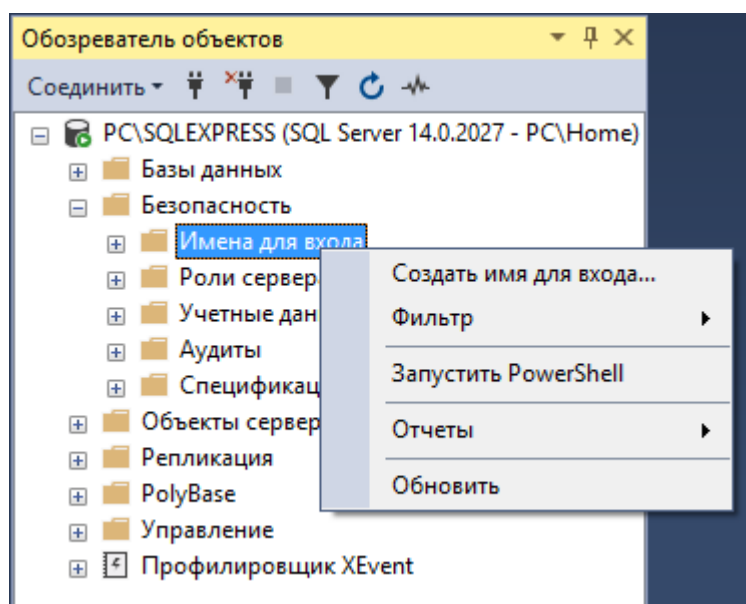


Рисунок 41 - Создание имени для входа

В открывшемся окне выбираем «Проверка подлинности SQL Server», вводим новый логин и пароль дважды, устанавливаем политику паролей, если это требуется. В строке «База данных по умолчанию» выбираем «A3C».

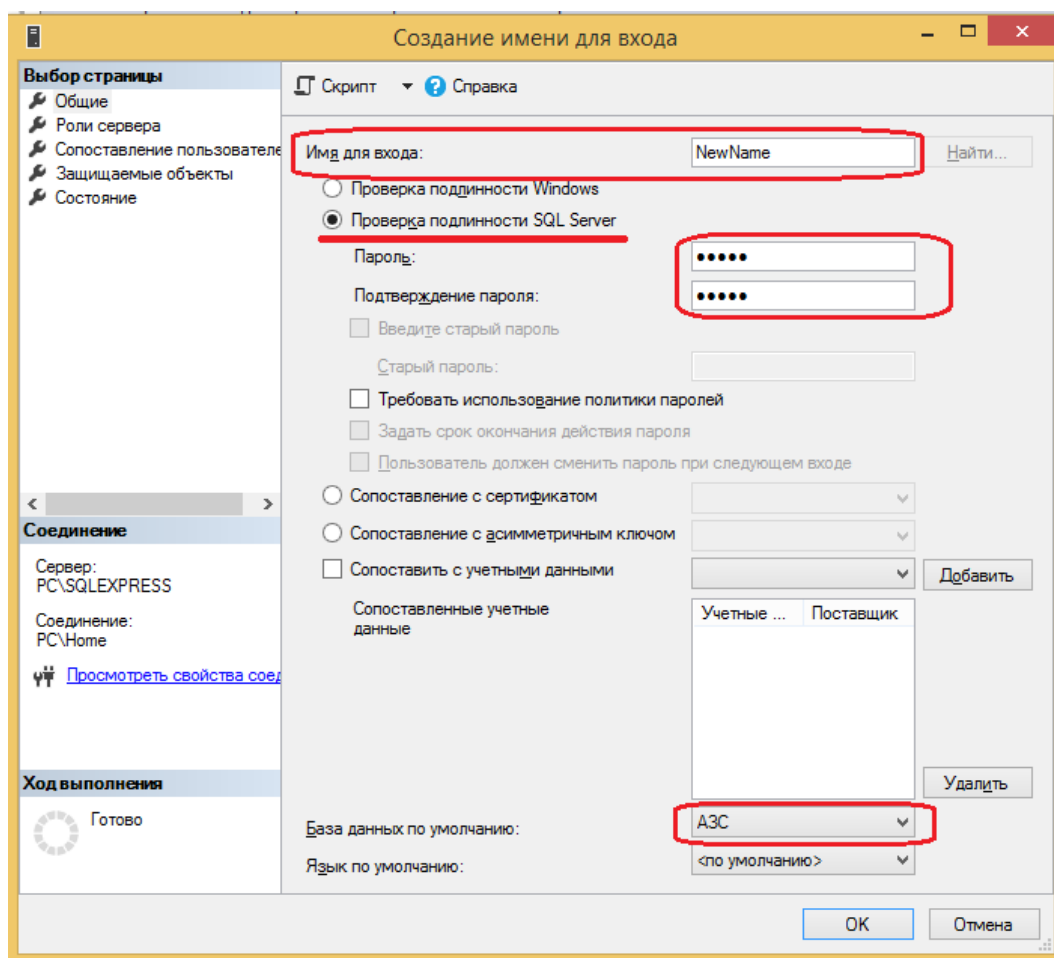


Рисунок 42 - Создание имени для входа

Далее переходим на страницу роли сервера и выбираем «sysadmin».

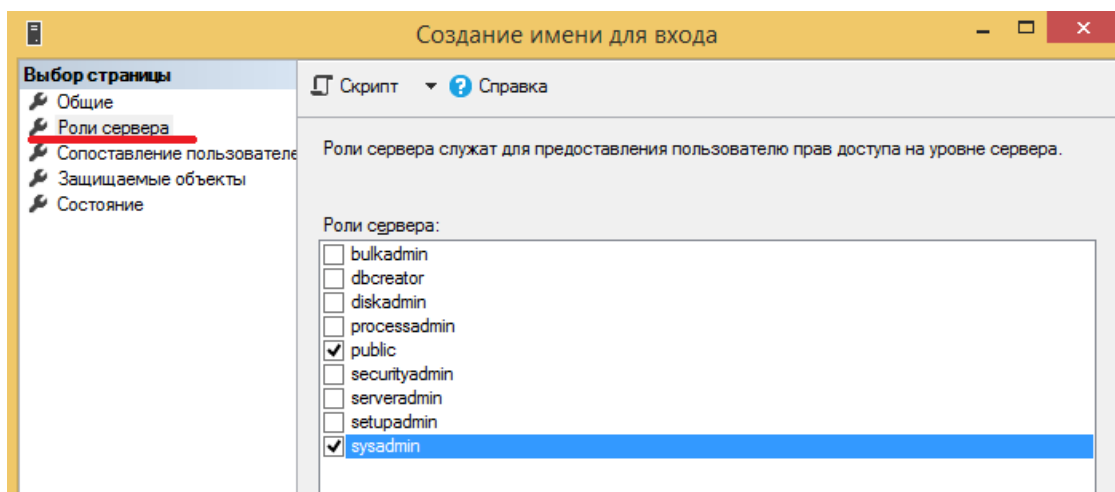


Рисунок 43 - Роли сервера

Теперь переходим на страницу «Сопоставление пользователей», выбираем базу данных «АЗС» и членство в группе «db_owner».

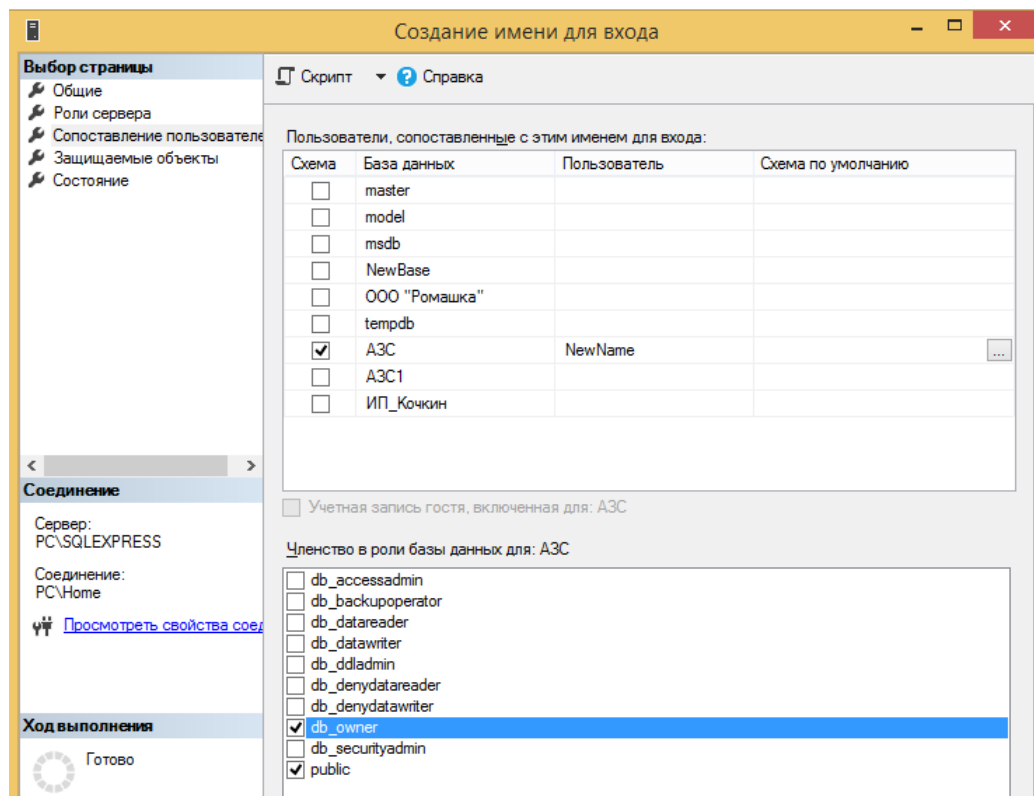


Рисунок 44 - Сопоставление пользователей

Нажимаем кнопку «ОК» и пользователь будет создан.

Теперь раскрываем раздел «Базы данных», находим базу «АЗС» и открываем ее свойства.

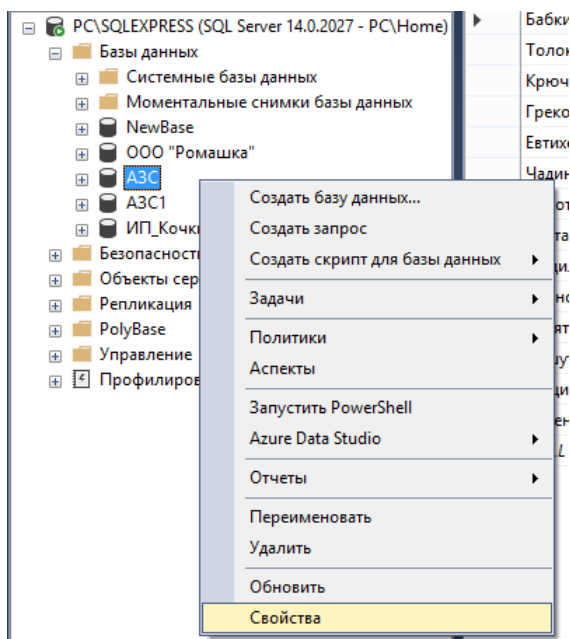


Рисунок 45 - Открытие свойств базы "АЗС"

В свойствах переходим на страницу «Разрешения», выбираем наше имя входа и предоставляем следующие разрешения:

- вставка;
- выборка;
- выполнение;
- изменение;
- обновление;
- соединение;
- удаление;
- управление.

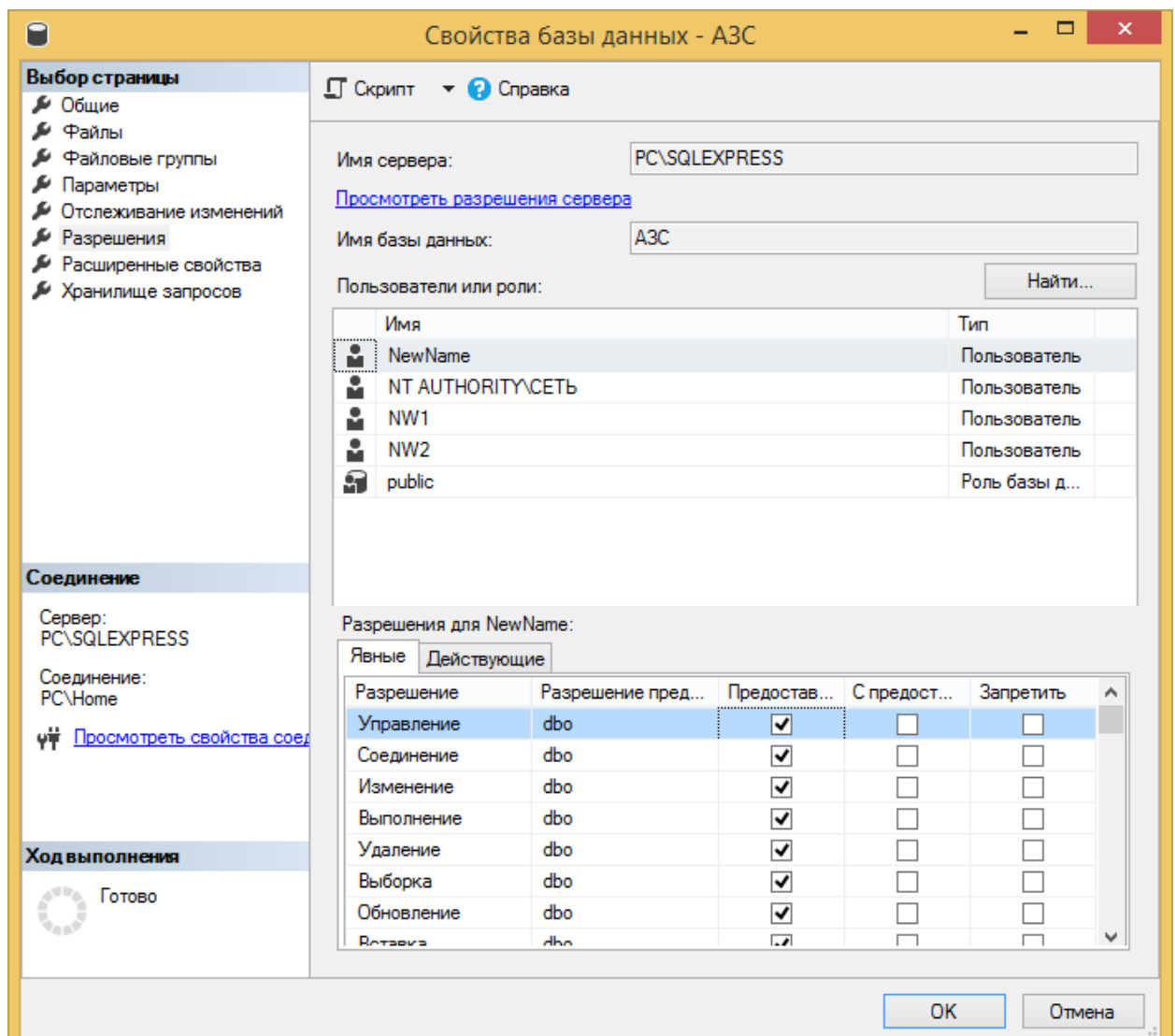


Рисунок 46 - Свойства базы данных "A3C"

Нажимаем кнопку «Ок», отключаемся от сервера и перезагружаем его.

Теперь в окне «Соединение с сервером» выбираем проверку подлинности SQL Server, вводим имя и пароль и нажимаем «Соединить».

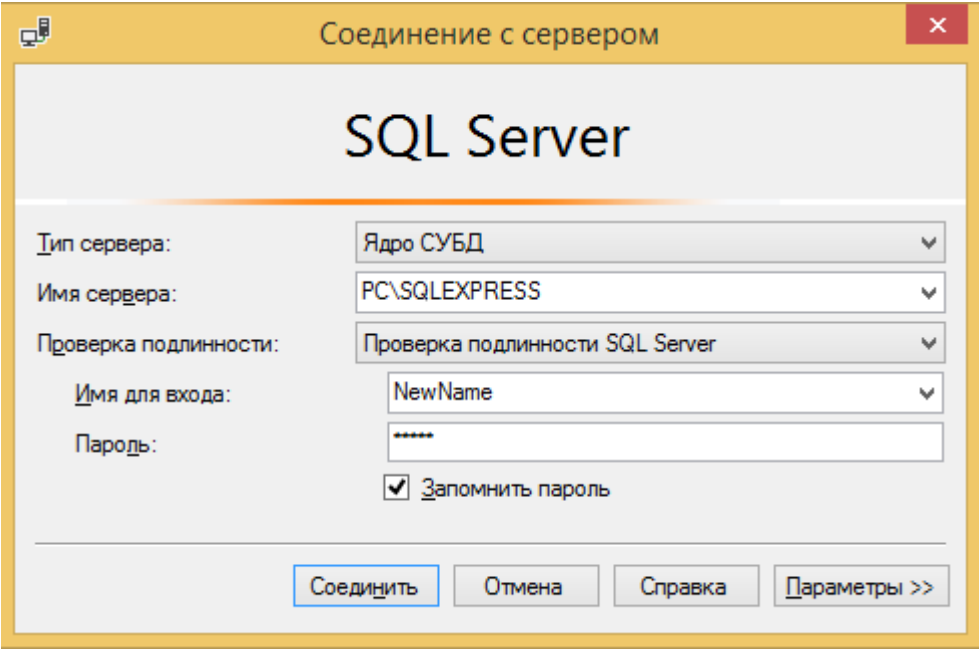


Рисунок 47 - Соединение с сервером

После успешного подключения откроем в обозревателе объектов нашу базу и посмотрим содержимое одной из ее таблиц.

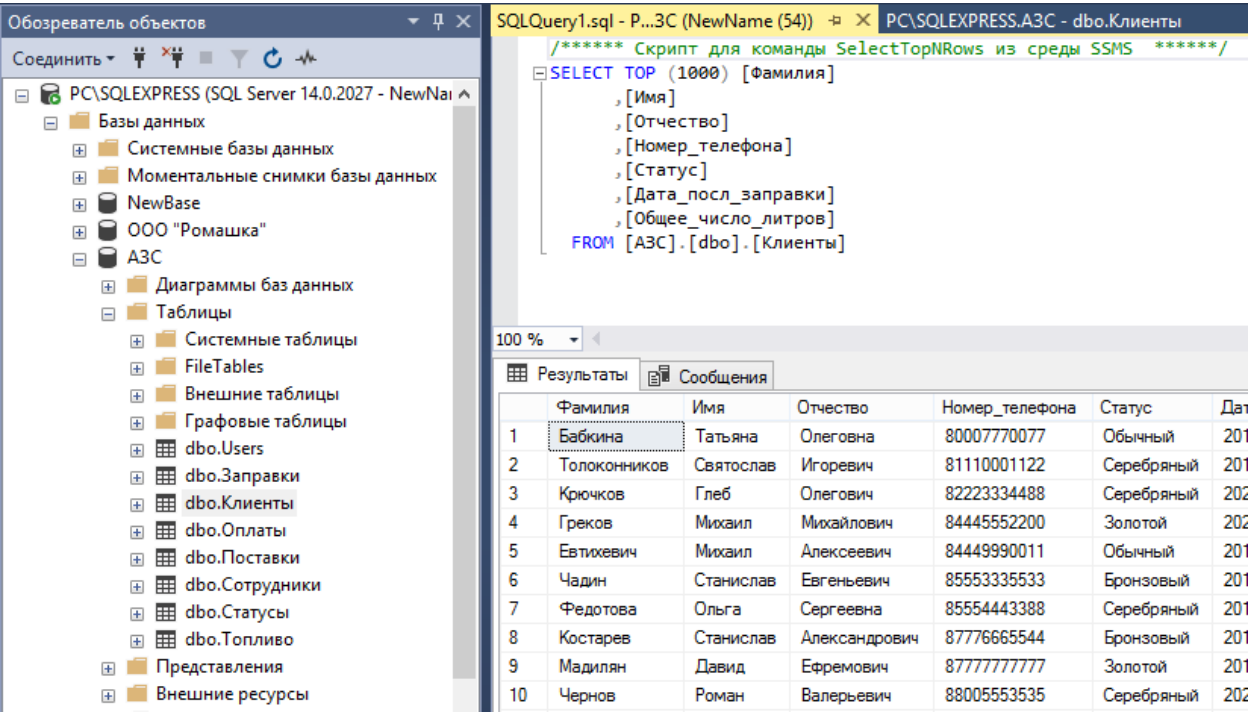


Рисунок 48 - Содержимое таблицы "Клиенты"

Как видно из рисунка 48, данные были получены. Также были проверены функции добавления, изменения и удаления данных. Все они работают, а значит подключение с проверкой подлинности SQL Server удалось.

Для того чтобы подключиться к базе данных, используя проверку подлинности SQL Server программно, используется класс «SqlConnectionStringBuilder», который содержит в себе строку подключения.

```
connStringBuilder = gnew SqlConnectionStringBuilder();
connStringBuilder->DataSource = "PC\\SQLEXPRESS"; //полное имя сервера
connStringBuilder->UserID = "Константин"; //логин
connStringBuilder->Password = "12345"; //пароль
connStringBuilder->InitialCatalog = "АЗС"; //имя базы данных
connStringBuilder->PersistSecurityInfo = false; //защита конфиденц
даанных в строке подклчения включена
connStringBuilder->IntegratedSecurity = false; //отключить
авторизацию по проверки подлинности Windows
```

Таким образом происходит подключение с использованием проверки подлинности SQL Server в приложении «Clientdb».

5.3. Разрешения в SQL Server

Все пользователи приложения взаимодействуют с SQL Server через одно имя входа, которое имеет только те разрешения, которые требуются для выполнения функций всех уровней доступа, остальные разрешения отключены. Разграничение доступа для разных уровней доступа предусмотрено на программном уровне.

Чтобы повысить безопасность нужно сделать по одному имени входа на каждый уровень доступа и наделить их только теми разрешениями, которые нужны для выполнения функций, предусмотренных соответствующим уровнем доступа.

Вывод

В результате написания программы были изучены и усвоены методы работы с MS SQL Server, Microsoft SQL Server Management Studio, были приобретены навыки написания приложения на языке C++/CLI, взаимодействующее с SQL Server, а также изучены и реализованы на практике такие виды сетевого подключения как «Удаленные рабочий стол» и подключение программы с компьютера-клиента к SQL Server компьютера-сервера. Рассмотрены некоторые методы защиты БД средствами MS SQL Server Express.