

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Разработка клиент-серверных приложений»

на тему «Сетевое приложение B2B на основе клиент-серверной технологии»

Выполнила:

студентка группы 19ВВ3.1

Дерябина А.О.

Принял:

Зинкин С.А.

ХОР
26.12.22

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПЕНЗЕНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

М. А. Митрохин

«15» сентября 2022г

ЗАДАНИЕ

на курсовое проектирование по курсу

Разработка клиент-серверных приложений

Студенту Дерединой А.О. Группа 19ВВ3.1

Вариант № 7

Тема проекта: Сетевое приложение В2В («бизнес-бизнесу») на основе клиент-серверной технологии

Исходные данные (технические требования) на проектирование

Необходимо разработать сетевое приложение в архитектуре В2В, работающее с базой данных.

Программа должна реализовать методы работы в TCP/IP сети (протоколы TCP и UDP). Для реализации этого необходимо определить, по каким портам будет происходить обмен информацией и порядок установления соединения.

Программа должна выполнять функции клиентского приложения и серверного приложения. Для этого необходимо создать графический пользовательский интерфейс пользователя и разделить обмен информацией клиента с другим экземпляром программы и обмен информацией другого клиента с серверной частью программы.

Программа должна обеспечивать работу с базой данных. Для этого нужно реализовать методы получения информации из базы данных, методы добавления строк в таблицы базы данных и удаления их.

Пользовательский интерфейс должен предоставлять пользователю информацию о таблицах базы данных, о данных, содержащихся в таблицах. Он должен предоставлять пользователю возможность выбора таблиц для просмотра, позволять добавлять и удалять данные при работе со своей базой данных и просматривать таблицы базы данных другого экземпляра программы.

Объем работы по курсу

1. Расчетная часть

Для написания данной программы будет использован программирования C++ (может быть выбран другой язык по согласованию с преподавателем). Предлагается использовать нисходящий проектирования: спроектировать классы, описывающие работу в сети и работу с базой данных, а затем реализовать конкретные методы.

Работа в TCP/IP сети была реализована в классе, например, Socket. В классе применить протоколы TCP и UDP для установления соединения между двумя экземплярами программы, передачи данных и определения готовности для установления соединения. Также в этом классе Socket должен присутствовать метод для шифрования и расшифровывания данных, которые передаются и принимаются через протокол TCP.

2. Графическая часть

Основные UML диаграммы.

3. Экспериментальная часть

Отладка приложения на сети.

Срок выполнения проекта по разделам

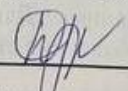
По графику кафедры.

Дата выдачи задания: "15" сентября 2022 г.

Дата защиты проекта: по графику кафедры

Руководитель  /Зинкин С.А./

Задание получил "15" сентября 2022 г.

Студент  /Перевина А.О./

Подпись

Фамилия И.О.

Оглавление

Введение.....	6
Постановка задачи.....	7
Выбор решения.....	8
Описание программы.....	9
Тестирование приложения	12
Заключение	17
Список литературы	18
ПРИЛОЖЕНИЕ А	19
ПРИЛОЖЕНИЕ Б.....	29

Введение

В качестве языка для создания клиент-серверного приложения был выбран C#. Это объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).

SQL(Structured Query Language — «язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. В общем случае SQL (без ряда современных расширений) считается языком программирования неполным по Тьюрингу, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений.

Постановка задачи

Необходимо разработать сетевое приложение в архитектуре В2В, работающее с базой данных.

Программа должна реализовывать методы работы в TCP/IP сети (протоколы TCP и UDP). Для реализации этого необходимо определить по каким портам будет происходить обмен информацией и порядок установления соединения.

Программа должна выполнять функции клиентского приложения и серверного приложения. Для этого необходимо создать графический пользовательский интерфейс пользователя и разделить обмен информацией клиента с другим экземпляром программы и обмен информацией другого клиента с серверной частью программы.

Программа должна обеспечивать работу с базой данных. Для этого нужно реализовать методы получения информации из базы данных, метод добавления строк в таблицы базы данных и удаления их.

Пользовательский интерфейс должен предоставлять пользователю информацию о таблицах базы данных, о данных, содержащихся в таблицах. Он должен предоставлять пользователю возможность выбора таблиц для просмотра, позволять добавлять и удалять данные при работе со своей базой данных и просматривать таблицы базы данных другого экземпляра программы.

Выбор решения

Для написания данной программы использован язык программирования С#. Язык программирования С# - универсальный язык программирования, который завоевал особую популярность у программистов, потому что обеспечивает кроссплатформенность и аппаратную переносимость программ.

Работа в TCP/IP сети была реализована в классе TcpManager. В классе применён протокол TCP для установления соединения между двумя экземплярами программы, передачи данных и определения готовности для установления соединения. Так же в классе присутствуют методы для шифрования и расшифровывания данных, которые передаются и принимаются через протокол TCP.

Работа с базой данных SQL была реализована в классе SqlManager. В классе реализованы методы получения информации из базы данных, метод добавления строк в таблицы базы данных и удаления их.

Описание программы

Для реализации графического пользовательского интерфейса была выбрана среда разработки WinForm. В графическом пользовательском интерфейсе используются элементы для управления TCP соединением, работой с базой данных SQL и установления ключа шифрования (рисунок 1).

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
NewCompany	Penza	Book	100p	63721

Введите ключ для шифрования

Установить ключ

Собственный интернет адрес

127.0.0.1

Порт

1234

Установить адрес

Введите удалённый адрес

127.0.0.1

Порт

1234

Подключиться к удалённому приложению

Введите строку для подключения к базе данных SQL

Server=(localdb)\mssqllocaldb;Database=Bussines;Trusted_Connection=True;

Добавить строку в БД

Подключиться к базе данных

Удалить выбранную строку из БД

Рисунок 1 – Графический интерфейс

Для работы с базой данных используются следующие элементы управления – таблица для вывода информации из базы данных, текстовое поле для подключения к базе данных и кнопка «Подключиться к базе данных», а так же кнопки «Добавить строку в БД» и «Удалить выбранную строку из БД».

Для подключения к базе данных необходимо ввести строку для подключения к базе данных в поле «Введите строку для подключения к базе данных SQL», после чего нажать на кнопку «Подключиться к базе данных». После этого в таблицу слева будет выведена информация из базы данных.

Удаление строки из базы данных происходит следующим образом. Необходимо в таблице выбрать строку, которую вы хотите удалить и нажать на кнопку «Удалить выбранную строку из БД».

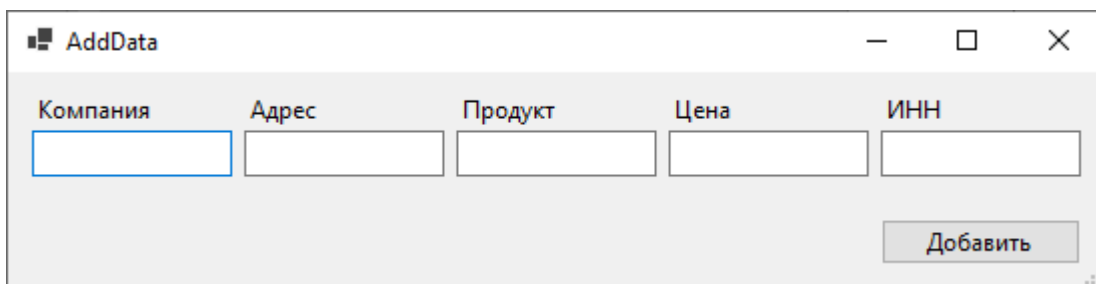


Рисунок 2 - Форма добавления строки

Чтобы добавить строку в базу данных необходимо нажать на кнопку «Добавить строку в БД», после чего появится новое окно (рисунок 2), в котором необходимо заполнить соответствующие поля и нажать кнопку «Добавить», вся введенная информация добавится в базу данных, а таблица обновится.

Для работы по ТСР/IP используются следующие элементы – текстовые поля для ввода ключа, собственного и удалённого IP адреса и порта, кнопки «Установить ключ», «Установить адрес», «Подключиться к удалённому приложению».

Для установления ключа шифрования необходимо ввести ключ в поле «Введите ключ для шифрования», после чего нажать на кнопку «Установить ключ». После этого ключ шифрования сменится, до выполнения этого действия используется стандартный ключ.

Чтобы разрешить удалённым клиентам подключаться к вашей базе данных, нужно в поле «Собственный интернет адрес» ввести желаемый адрес, а в поле «Порт» находящееся под ним свободный порт и нажать на кнопку «Установить адрес». После этого приложение начнёт принимать клиентов, изменить параметры возможно в любой момент.

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
NewCompany	Penza	Book	100p	63721

Добавить строку в БД

Рисунок 3 - Таблица для вывода информации из базы данных

Для подключения к удалённому клиенту, необходимо в поле «Введите удалённый адрес» ввести адрес удалённого приложения, а в поле «Порт» находящееся под ним его порт. После нажатия кнопки «Подключиться к удалённому приложению» появится новое окно с базой данных клиента (рисунок 3). В новом окне есть таблица, из которой можно выбрать любое поле и добавить его в свою базу данных, с помощью кнопку добавить.

Тестирование приложения

По описанному выше алгоритму выполним подключение к базе данных. Результат представлен на рисунка 4 – 5.

The screenshot shows the 'SQL TCP Client' application window. On the left is a table with five columns: 'Компания', 'Адрес', 'Продукт', 'Цена', and 'ИНН'. The table is currently empty. On the right is a form with several input fields and buttons. The form includes a 'Введите ключ для шифрования' field with an 'Установить ключ' button. Below this are fields for 'Собственный интернет адрес' (127.0.0.1) and 'Порт' (1234) with an 'Установить адрес' button. Further down are fields for 'Введите удалённый адрес' (127.0.0.1) and 'Порт' (1234) with a 'Подключиться к удаленному приложению' button. At the bottom, there is a 'Введите строку для подключения к базе данных SQL' field containing the string 'Server=(localdb)\mssqllocaldb;Database=Bussines;Trusted_Connection=True;'. Below this are buttons for 'Добавить строку в БД', 'Удалить выбранную строку из БД', and 'Подключиться к базе данных'.

Рисунок 4 – Исходная форма.

The screenshot shows the 'SQL TCP Client' application window after a successful database connection. The table on the left now contains five rows of data:

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
NewCompany	Penza	Book	100p	63721

The form on the right remains the same as in the previous screenshot, but the 'Подключиться к базе данных' button is now highlighted in blue, indicating a successful connection.

Рисунок 5 – Результат подключения к базе данных.

По описанному выше алгоритму выполним добавление строки в базу данных. Результат представлен на рисунках 6-8.

SQL TCP Client

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
NewCompany	Penza	Book	100p	63721

Введите ключ для шифрования

Установить ключ

Собственный интернет адрес

Порт

Установить адрес

Введите удалённый адрес

Порт

Подключиться к удалённому приложению

Введите строку для подключения к базе данных SQL

Добавить строку в БД
Подключиться к базе данных

Удалить выбранную строку из БД

Рисунок 6 – Исходная база данных

AddData

Компания	Адрес	Продукт	Цена	ИНН
<input type="text" value="Dior"/>	<input type="text" value="Paris"/>	<input type="text" value="Shoes"/>	<input type="text" value="10 000\$"/>	<input type="text" value="932182"/>

Добавить

Рисунок 7 – Форма добавления строки

SQL TCP Client

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
NewCompany	Penza	Book	100p	63721
Dior	Paris	Shoes	10 000\$	932182

Введите ключ для шифрования

Установить ключ

Собственный интернет адрес

Порт

Установить адрес

Введите удалённый адрес

Порт

Подключиться к удалённому приложению

Введите строку для подключения к базе данных SQL

Добавить строку в БД
Подключиться к базе данных

Удалить выбранную строку из БД

Рисунок 8 – Результат добавления строки

Удалим по описанному выше алгоритму строки из базы данных,
рисунок 9 - 10.

SQL TCP Client

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
NewCompany	Penza	Book	100p	63721
Dior	Paris	Shoes	10 000\$	932182

Введите ключ для шифрования

Установить ключ

Собственный интернет адрес

127.0.0.1

Порт

1234

Установить адрес

Введите удалённый адрес

127.0.0.1

Порт

1234

Подключиться к удалённому приложению

Введите строку для подключения к базе данных SQL

Server=(localdb)\mssqllocaldb;Database=Bussines;Trusted_Connection=True;

Добавить строку в БД

Подключиться к базе данных

Удалить выбранную строку из БД

Рисунок 9 – Выбор строки для удаления.

SQL TCP Client

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
Dior	Paris	Shoes	10 000\$	932182

Введите ключ для шифрования

Установить ключ

Собственный интернет адрес

127.0.0.1

Порт

1234

Установить адрес

Введите удалённый адрес

127.0.0.1

Порт

1234

Подключиться к удалённому приложению

Введите строку для подключения к базе данных SQL

Server=(localdb)\mssqllocaldb;Database=Bussines;Trusted_Connection=True;

Добавить строку в БД

Подключиться к базе данных

Удалить выбранную строку из БД

Рисунок 10 – Результат удаления.

Выполним подключение к базе данных по описанному выше
алгоритму, рисунок 11 – 13.

SQL TCP Client

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
Dior	Paris	Shoes	10 000\$	932182

Введите ключ для шифрования

Установить ключ

Собственный интернет адрес

Порт

Установить адрес

Введите удалённый адрес

Порт

Подключиться к удаленному приложению

Введите строку для подключения к базе данных SQL

Добавить строку в БД

Подключиться к базе данных

Удалить выбранную строку из БД

Рисунок 11 – Исходная база данных

FormRemoteData

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
Dior	Paris	Shoes	10 000\$	932182
H&M	USA	Dress	100\$	321432

Добавить строку в БД

Рисунок 12 – Получение удаленной базы данных

SQL TCP Client

Компания	Адрес	Продукт	Цена	ИНН
Microsoft	California	MacBook13	12 999\$	321123
NeMoloko	Moscow	Milk	150p	376982
Samsung	Cambodgia	TV	5000\$	35412
Dior	Paris	Shoes	10 000\$	932182
H&M	USA	Dress	100\$	321432

Введите ключ для шифрования

Установить ключ

Собственный интернет адрес

127.0.0.1

Порт

1234

Установить адрес

Введите удалённый адрес

127.0.0.1

Порт

1233

Подключиться к удалённому приложению

Введите строку для подключения к базе данных SQL

Server=(localdb)\mssqllocaldb;Database=Bussines;Trusted_Connection=True;

Добавить строку в БД

Подключиться к базе данных

Удалить выбранную строку из БД

Рисунок 13 – Результат

Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая обмен информацией между базой данных и пользователями на языке C# в среде разработки Visual Studio 2022 с использованием графического интерфейса.

При выполнении данной курсовой работы были закреплены знания, полученные при прослушании курса и приобретены практические навыки разработки многомодульных приложений на языке C#.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

Список литературы

1. Эйри, Джоунс Функции SQL. Справочник программиста / Джоунс Эйри.
- М.: Диалектика / Вильямс, **2013**. - 0 с.
2. Nick Symmonds, Nicholas Symmonds Data Entry and Validation with C# and
VB. NET Windows Forms; **Гостехиздат** - Москва, **1990**. - **103**с.
3. .NET Сетевое программирование. - М.: ЛОРИ, 2013. - 420 с.

ПРИЛОЖЕНИЕ А

Файл Form1.cs

```
namespace SQL_TCP_Client
{
    public partial class Form1 : Form
    {
        private SqlManager m_sqlManager = new SqlManager();
        private string m_sqlName;

        private TcpManager m_tcpManager = new TcpManager();

        public Form1()
        {
            InitializeComponent();
        }

        private void ConnectToSql()
        {
            m_sqlManager.Connect(m_sqlName);
        }

        private void UpdateSql()
        {
            dataGridViewBDSql.Rows.Clear();

            List<sqlData> data = new List<sqlData>();
            m_sqlManager.GetAllData(data);

            for (int i = 0; i < data.Count; i++)
            {
                data[i].Company.TrimEnd();
                data[i].Address.TrimEnd();
                data[i].Product.TrimEnd();
                data[i].Price.TrimEnd();
                data[i].INN.TrimEnd();

                dataGridViewBDSql.Rows.Add(data[i].Company,
                    data[i].Address, data[i].Product,
                    data[i].Price, data[i].INN);
            }

            m_tcpManager.m_data = data;
        }

        private void AddSql(sqlData data)
        {
            m_sqlManager.AddData(data);

            List<sqlData> listData = new List<sqlData>();

            for (int i = 0; i < dataGridViewBDSql.Rows.Count - 1; i++)
            {
                sqlData ndata = new sqlData();
```

```

        ndata.Company =
dataGridViewBDSql.Rows[i].Cells[0].Value.ToString();
        ndata.Address =
dataGridViewBDSql.Rows[i].Cells[1].Value.ToString();
        ndata.Product =
dataGridViewBDSql.Rows[i].Cells[2].Value.ToString();
        ndata.Price =
dataGridViewBDSql.Rows[i].Cells[3].Value.ToString();
        ndata.INN =
dataGridViewBDSql.Rows[i].Cells[4].Value.ToString();

        listData.Add(ndata);
    }
    m_tcpManager.m_data = listData;
}

private void DeleteFormSql(int Id)
{
    sqlData data = new sqlData();
    data.Company =
dataGridViewBDSql.Rows[Id].Cells[0].Value.ToString();
    data.Address=
dataGridViewBDSql.Rows[Id].Cells[1].Value.ToString();
    data.Product =
dataGridViewBDSql.Rows[Id].Cells[2].Value.ToString();
    data.Price =
dataGridViewBDSql.Rows[Id].Cells[3].Value.ToString();
    data.INN = dataGridViewBDSql.Rows[Id].Cells[4].Value.ToString();
    m_sqlManager.DeleteRow(data);
    dataGridViewBDSql.Rows.RemoveAt(Id);

    List<sqlData> listData = new List<sqlData>();

    for (int i = 0; i < dataGridViewBDSql.Rows.Count - 1; i++)
    {
        sqlData ndata = new sqlData();
        ndata.Company =
dataGridViewBDSql.Rows[i].Cells[0].Value.ToString();
        ndata.Address =
dataGridViewBDSql.Rows[i].Cells[1].Value.ToString();
        ndata.Product =
dataGridViewBDSql.Rows[i].Cells[2].Value.ToString();
        ndata.Price =
dataGridViewBDSql.Rows[i].Cells[3].Value.ToString();
        ndata.INN =
dataGridViewBDSql.Rows[i].Cells[4].Value.ToString();

        listData.Add(ndata);
    }
    m_tcpManager.m_data = listData;
}

private void buttonConnectToSql_Click(object sender, EventArgs e)
{
    m_sqlName = textBoxSqlConnectStr.Text;
    ConnectToSql();
}

```

```

        UpdateSql();
    }

    private void buttonAddRow_Click(object sender, EventArgs e)
    {
        FormAddData formAddData = new FormAddData();
        if (formAddData.ShowDialog() == DialogResult.OK)
        {
            AddSql(formAddData.Data);
            UpdateSql();
        }

        formAddData.Dispose();
    }

    private void buttonDeleteSelectRow_Click(object sender, EventArgs e)
    {
        if (dataGridViewBDSql.SelectedRows.Count > 0 &&
            dataGridViewBDSql.SelectedRows[0].Index !=
            dataGridViewBDSql.Rows.Count - 1)
        {
            DeleteFormSql(dataGridViewBDSql.SelectedRows[0].Index);

            List<sqlData> listData = new List<sqlData>();

            for (int i = 0; i < dataGridViewBDSql.Rows.Count - 1; i++)
            {
                sqlData data = new sqlData();
                data.Company =
dataGridViewBDSql.Rows[i].Cells[0].Value.ToString();
                data.Address =
dataGridViewBDSql.Rows[i].Cells[1].Value.ToString();
                data.Product =
dataGridViewBDSql.Rows[i].Cells[2].Value.ToString();
                data.Price =
dataGridViewBDSql.Rows[i].Cells[3].Value.ToString();
                data.INN =
dataGridViewBDSql.Rows[i].Cells[4].Value.ToString();

                listData.Add(data);
            }
            m_tcpManager.m_data = listData;
        }
    }

    private void buttonSetAddress_Click(object sender, EventArgs e)
    {
        m_tcpManager.SetAcceptAddress(textBoxHost.Text,
Convert.ToInt32(textBoxPort.Text));
    }

    private void buttonConnectToRemoteSql_Click(object sender, EventArgs
e)
    {

```

```

        m_tcpManager.Connect(textBoxRemoteHost.Text,
Convert.ToInt32(textBoxRemotePort.Text));

        List<sqlData> listData = new List<sqlData>();

        m_tcpManager.ReciveData(ref listData);

        FormRemoteData formRemoteData = new FormRemoteData(listData);
        formRemoteData.ShowDialog();

        listData = formRemoteData.Data;

        formRemoteData.Dispose();

        foreach (sqlData data in listData)
            AddSql(data);

        UpdateSql();
    }

    private void buttonSetKey_Click(object sender, EventArgs e)
    {
        string strKey = textBoxKey.Text;
        m_tcpManager.m_key = 0;
        while (strKey.Length > 4)
        {
            m_tcpManager.m_key +=
Convert.ToUInt16(strKey.Substring(0,3));
            strKey = strKey.Substring(3);
        }
        m_tcpManager.m_key += Convert.ToUInt16(strKey);
    }
}
}

```

Файл FormAddData.cs

```

namespace SQL_TCP_Client
{
    public partial class FormAddData : Form
    {
        public sqlData Data = new sqlData();

        public FormAddData()
        {
            InitializeComponent();
        }

        private void buttonAdd_Click(object sender, EventArgs e)
        {
            Data.Company = textBoxCompany.Text;
            Data.Address = textBoxAddress.Text;
        }
    }
}

```

```

        Data.Product = textBoxProduct.Text;
        Data.Price = textBoxPrice.Text;
        Data.INN = textBoxINN.Text;

        this.DialogResult = DialogResult.OK;
        this.Close();
    }
}
}

```

Файл FromRemoteData.cs

```

namespace SQL_TCP_Client
{
    public partial class FormRemoteData : Form
    {
        public List<sqlData> Data = new List<sqlData>();

        public FormRemoteData()
        {
            InitializeComponent();
        }

        public FormRemoteData(List<sqlData> data)
        {
            InitializeComponent();

            for (int i = 0; i < data.Count; i++)
            {
                dataGridViewData.Rows.Add(data[i].Company.TrimEnd(),
                    data[i].Address.TrimEnd(), data[i].Product.TrimEnd(),
                    data[i].Price.TrimEnd(), data[i].INN.TrimEnd());
            }
        }

        private void buttonAddRow_Click(object sender, EventArgs e)
        {
            if (dataGridViewData.SelectedRows.Count > 0 &&
                dataGridViewData.SelectedRows[0].Index !=
                dataGridViewData.Rows.Count - 1)
            {
                int Id = dataGridViewData.SelectedRows[0].Index;
                sqlData data = new sqlData();
                data.Company =
                    dataGridViewData.Rows[Id].Cells[0].Value.ToString();
                data.Address =
                    dataGridViewData.Rows[Id].Cells[1].Value.ToString();
                data.Product =
                    dataGridViewData.Rows[Id].Cells[2].Value.ToString();
                data.Price =
                    dataGridViewData.Rows[Id].Cells[3].Value.ToString();
                data.INN =
                    dataGridViewData.Rows[Id].Cells[4].Value.ToString();

                Data.Add(data);
            }
        }
    }
}

```

```

    }
}
}

```

Файл TcpManager.cs

```

using System.Text;
using System.Net.Sockets;
using System.Net;

namespace SQL_TCP_Client
{
    internal class TcpManager : IDisposable
    {
        private Socket m_socketRecive = new
Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        private Socket m_socketAccept = new
Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        public List<sqlData> m_data = new List<sqlData>();

        public ushort m_key = 123;
        public static string EncodeDecrypt(string str, ushort secretKey)
        {
            var ch = str.ToArray();
            string newStr = "";
            foreach (var c in ch)
                newStr += TopSecret(c, secretKey);
            return newStr;
        }

        public static char TopSecret(char character, ushort secretKey)
        {
            character = (char)(character ^ secretKey);
            return character;
        }

        private bool m_isAccept;
        public void Dispose()
        {
            m_data.Clear();

            m_socketAccept.Close();
            m_socketAccept.Dispose();
            m_socketRecive.Close();
            m_socketRecive.Dispose();
        }

        public void SetAcceptAddress(string host, int port)
        {
            if (m_isAccept)
                return;

            m_isAccept = false;

```

```

        IPAddress address = IPAddress.Parse(host);
        IPEndPoint ipPoint = new IPEndPoint(address, port);
        m_socketAccept.Bind(ipPoint);
        m_socketAccept.Listen();

        new Thread(() => SendData()).Start();
    }

    public void Connect(string host, int port)
    {
        m_socketRecive = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
        m_socketRecive.Connect(host, port);
        Thread.Sleep(1000);
    }

    public void SendData()
    {
        m_isAccept = true;
        while(m_isAccept)
        {
            Socket socketSend = m_socketAccept.Accept();
            try
            {
                lock(m_data)
                {
                    if (m_data.Count > 0)
                    {
                        byte[] buffer = new byte[1024];
                        buffer =
Encoding.Unicode.GetBytes(m_data.Count.ToString());
                        string strResult = "";
                        foreach (sqlData sqldata in m_data)
                        {
                            strResult += sqldata.Company.TrimEnd() + "\r"
+ sqldata.Address.TrimEnd() + "\r" +
                                sqldata.Product.TrimEnd() + "\r" +
                                sqldata.Price.TrimEnd() + "\r"
                                + sqldata.INN.TrimEnd() + "\r\t";
                        }
                        strResult = strResult.Substring(0,
strResult.Length - 1);

                        strResult = EncodeDecrypt(strResult, m_key);

                        socketSend.Send(Encoding.Unicode.GetBytes(strResult));
                        socketSend.Receive(buffer);
                    }
                }
            }
            finally
            {
                socketSend.Close();
            }
        }
    }

```

```

    }

}

public void ReciveData(ref List<sqlData> data)
{
    try
    {
        byte[] reciveBuffer = new byte[1024];
        m_socketRecive.Receive(reciveBuffer);

        sqlData dataBuffer = new sqlData();
        string strResult =
EncodeDecrypt(Encoding.Unicode.GetString(reciveBuffer), m_key);

        var strData = strResult.Split(new char[] { '\t' });
        foreach(var str in strData)
        {
            var subStr = str.Split(new char[] { '\r' });
            dataBuffer.Company = subStr[0];
            dataBuffer.Address = subStr[1];
            dataBuffer.Product = subStr[2];
            dataBuffer.Price = subStr[3];
            dataBuffer.INN = subStr[4];

            data.Add(dataBuffer);
        }
        m_socketRecive.Send(reciveBuffer);
    }
    finally
    {
    }
}
}
}

```

Файл SqlManager.cs

```

using System.Data.SqlClient;

namespace SQL_TCP_Client
{
    public struct sqlData
    {
        public int Id { get; set; }
        public string Company { get; set; }
        public string Address { get; set; }
        public string Product { get; set; }
        public string Price { get; set; }
        public string INN { get; set; }
    }
}

```

```

internal class SqlManager
{
    private SqlConnection m_sqlConnection = new SqlConnection();
    private SqlCommand m_sqlCommand = new SqlCommand();
    public void Connect(string connectionString)
    {
        m_sqlConnection.ConnectionString = connectionString;
        m_sqlCommand.Connection = m_sqlConnection;
    }

    public void AddData(sqlData data)
    {
        m_sqlConnection.Open();

        string command = $"INSERT INTO Users (Company, Address, Product,
Price, INN) VALUES ('{data.Company}', '{data.Address}', '{data.Product}',
'{data.Price}', '{data.INN}')";
        m_sqlCommand.CommandText = command;
        int number = m_sqlCommand.ExecuteNonQuery();
        m_sqlConnection.Close();
    }

    public void DeleteRow(sqlData data)
    {
        m_sqlConnection.Open();

        string command = $"DELETE FROM Users WHERE
Company='{data.Company}' AND " +
            $"Address='{data.Address}' AND Product='{data.Product}' AND "
+
            $"Price='{data.Price}' AND INN='{data.INN}'";
        m_sqlCommand.CommandText = command;
        int number = m_sqlCommand.ExecuteNonQuery();
        m_sqlConnection.Close();
    }

    public void GetData(int Id, sqlData data)
    {
        m_sqlConnection.Open();

        string command = "SELECT * FROM Users";

        m_sqlCommand.CommandText = command;
        SqlDataReader reader = m_sqlCommand.ExecuteReader();

        for(int i = 0; i < Id; ++i)
            reader.ReadAsync();

        data.Company = reader.GetString(0);
        data.Address = reader.GetString(1);
        data.Product = reader.GetString(2);
        data.Price = reader.GetString(3);
        data.INN = reader.GetString(4);

        reader.Close();
        m_sqlConnection.Close();
    }
}

```

```

    }

    public void GetAllData(List<sqlData> data)
    {
        m_sqlConnection.Open();

        data.Clear();
        sqlData bufferData = new sqlData();

        string command = "SELECT * FROM Users";

        m_sqlCommand.CommandText = command;
        SqlDataReader reader = m_sqlCommand.ExecuteReader();

        while(reader.Read())
        {
            bufferData.Id = reader.GetInt32(0);
            bufferData.Company = reader.GetString(1);
            bufferData.Address = reader.GetString(2);
            bufferData.Product = reader.GetString(3);
            bufferData.Price = reader.GetString(4);
            bufferData.INN = reader.GetString(5);

            data.Add(bufferData);
        }

        reader.Close();
        m_sqlConnection.Close();
    }
}

```

ПРИЛОЖЕНИЕ Б

UML-диаграммы

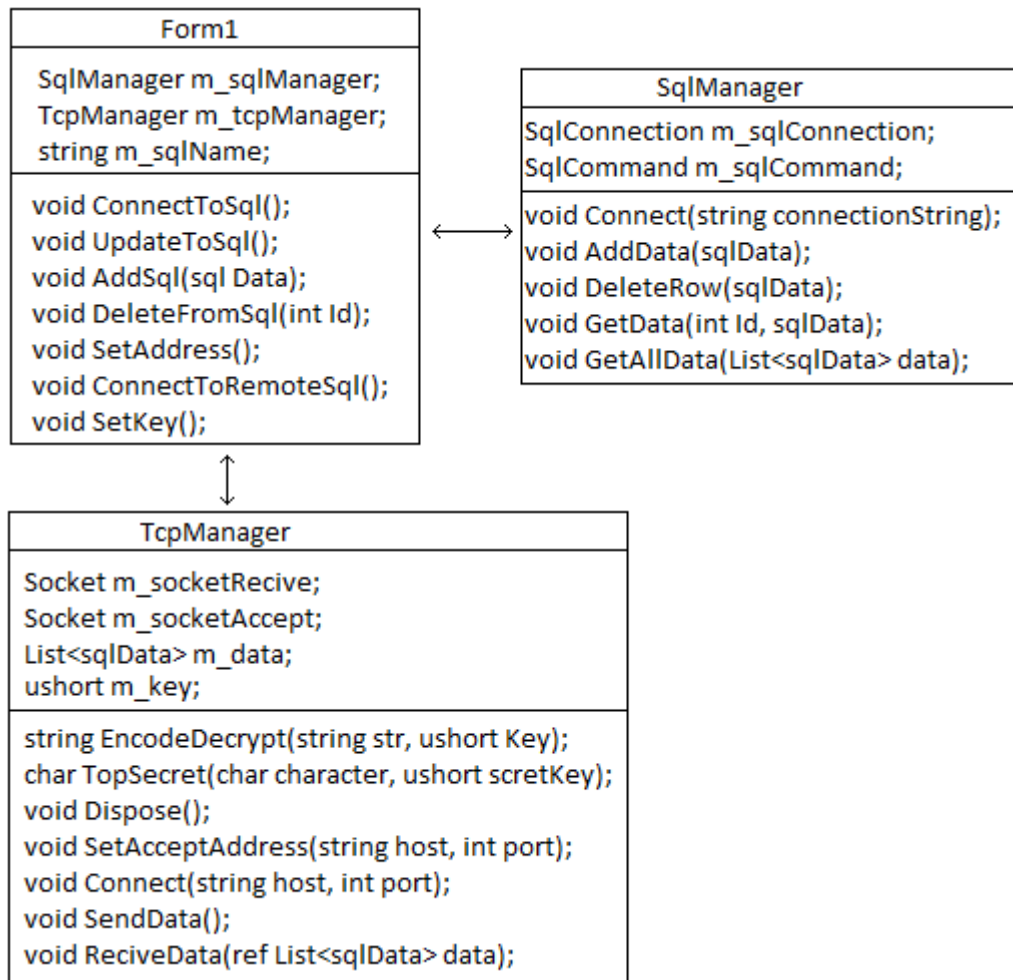


Рисунок 14 – диаграмма класса.

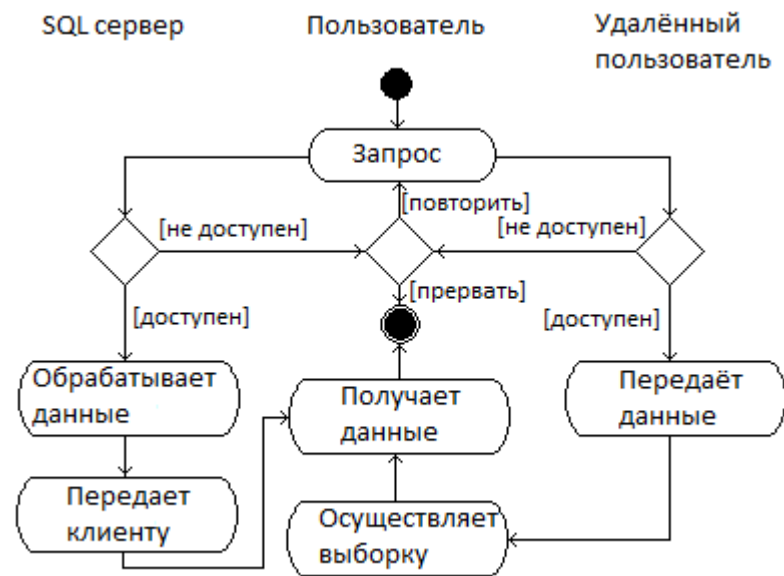


Рисунок 15 – диаграмма деятельности.

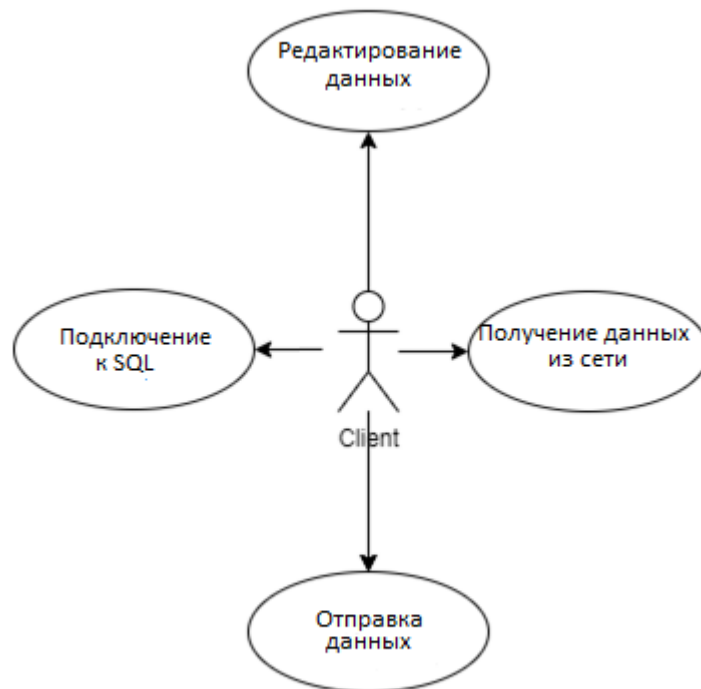


Рисунок 16 – диаграмма вариантов использования.

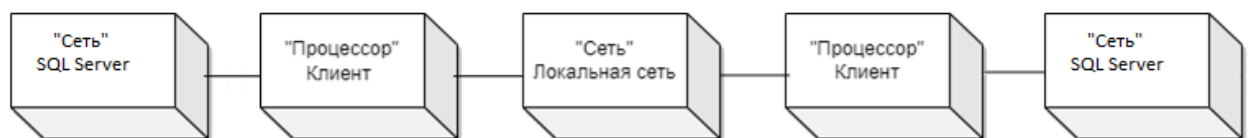


Рисунок 17 – диаграмма развертывания.

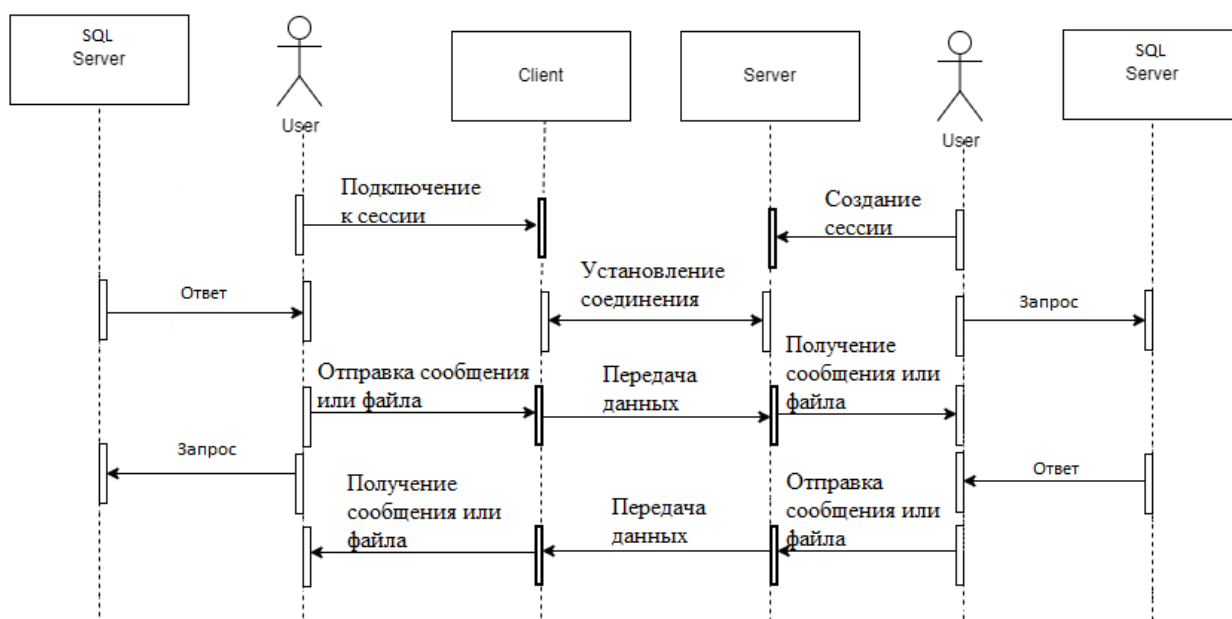


Рисунок 18 – диаграмма последовательности.