Комитет по образованию Правительства Санкт-Петербурга  
САНКТ-ПЕТЕРБУРГСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Управление и автоматизация баз данных»

Специальность 09.02.07

Информационные системы и программирование (администратор баз данных)

Выполнил студен группы 225

Седнев Матвей Дмитриевич

Сдал \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург 2024 г.

**ОГЛАВЛЕНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc166570116)

[**1.** **СТРУКТУРА БАЗЫ ДАННЫХ** 4](#_Toc166570117)

[**1.1** **Схема базы данных** 4](#_Toc166570118)

[**1.2** **Структура таблиц** 5](#_Toc166570119)

[2. **СТРУКТУРА ПРИЛОЖЕНИЯ** 13](#_Toc166570120)

[**ЗАКЛЮЧЕНИЕ** 19](#_Toc166570121)

[**ЛИТЕРАТУРА** 20](#_Toc166570122)

[**ПРИЛОЖЕНИЕ** 21](#_Toc166570123)

[**A** **Приложение ERD схема в расширениях pgerd и png** 21](#_Toc166570124)

[**B** **Приложение Скрипт создание схемы в БД** 21](#_Toc166570125)

[**C** **Приложение Разработка приложения C# WinForms** 27](#_Toc166570126)

# **ВВЕДЕНИЕ**

Введение является важной составной частью данного проекта, посвященного разработке базы данных и созданию соответствующего приложения. В данном разделе обсуждаются ключевые аспекты проекта:

Актуальность темы: в свете постоянного развития информационных технологий и их все более широкого использования создание эффективных баз данных становится насущной задачей. В контексте учета и мониторинга данных о потреблении ресурсов, таких как энергия, важность оптимизации и управления данными становится особенно яркой.

Сущность работы: Основной целью данного проекта является разработка базы данных, предназначенной для хранения информации о счетчиках и их показаниях. Это предполагает создание структуры данных, которая будет обеспечивать эффективное и удобное хранение информации, необходимой для учета и анализа энергопотребления.

Замысел работы: Проект предполагает создание надежной и функциональной базы данных, а также разработку приложения для взаимодействия с этой базой данных. Основной задачей будет определение требований к базе данных, разработка ее структуры, а также создание пользовательского интерфейса для удобного доступа к данным.

Цель и задачи: Целью проекта является создание эффективной системы учета и мониторинга данных о счетчиках, что позволит оптимизировать управление энергоресурсами. Для достижения этой цели необходимо определить требования к базе данных, разработать ее структуру, а также создать приложение, обеспечивающее удобное взаимодействие с этой базой данных.

Предполагаемые методы и способы решения: для разработки базы данных и приложения будут использованы современные методы и инструменты программирования, такие как языки SQL и C#, а также соответствующие фреймворки и библиотеки. Это позволит обеспечить эффективное и надежное функционирование создаваемой системы.

# **СТРУКТУРА БАЗЫ ДАННЫХ**

## Схема базы данных

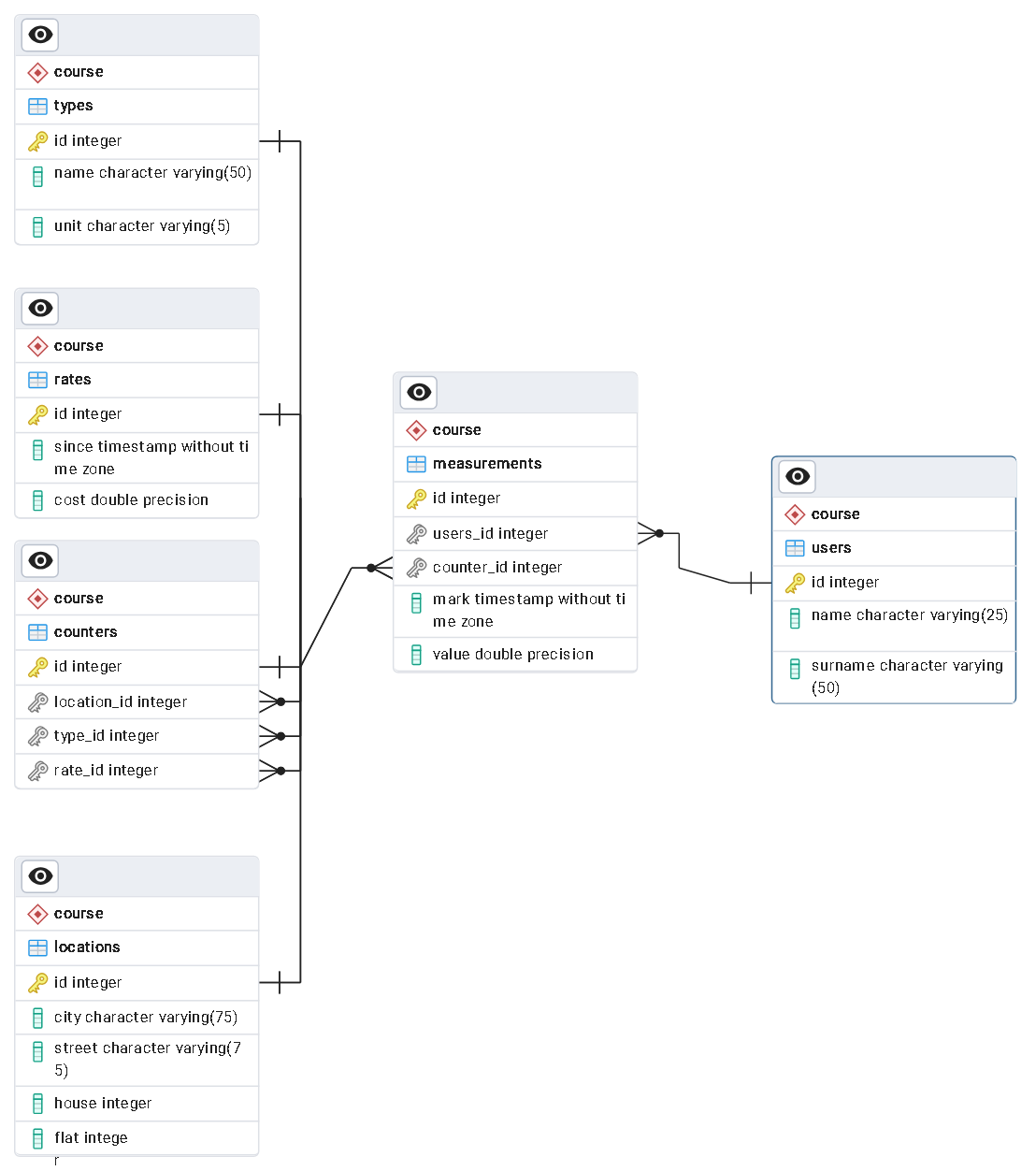


Рисунок 1 – ER диаграмма базы данных

База данных состоит из 6 таблиц, размещённых в схеме course:

1. Locations – содержит полные адреса город, улица, дом, квартира
2. Users – содержит личную информацию о пользователях
3. Rates – содержит историю тарифов по счётчикам
4. Types – содержит типы счётчиков
5. Counters – содержит полную информацию о счётчике(-ах)
6. Measurments – содержит информацию последнего измерения счётчика

## Структура таблиц

Рассмотрим таблицу types (Рис. 2)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя** | **Тип** | **Размер** | **Значение по умолчаию** | **Первичный ключ** | **Уникально** | **Описание** |
| 1 | id | integer | - | Nextval(‘sq\_types\_id’) | Yes | Yes | Номер типа счётчика |
| 2 | name | varchar | 50 | - | No | No | Названия типа счётчика |
| 3 | unit | varchar | 5 | - | No | No | Единицы измерения |

Рисунок 2 – Описание полей таблицы types

Таблица содержит 3 столбца:

1. Id – содержит уникальные номера каждого типа счётчика
2. name – содержит названия типа счётчика, данный столбец обязан содержать название типов больше 5 символов
3. unit – содержит единицы измерения типа счётчика, данный столбец обязан содержать единицы измерения больше или равно 2 символам

На рисунке 3 приведён пример данных, находящихся в таблице types

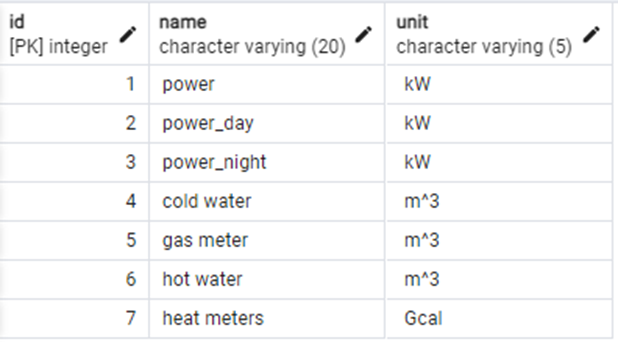


Рисунок 3 – Пример данных в таблице types

На рисунке 4 изображён код для данной таблицы types

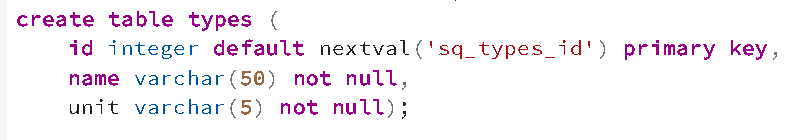


Рисунок 4 – код таблицы types

Рассмотрим таблицу locations (Рис. 5)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Значения**  **по умолчанию** | **Первичный ключ** | **Внешний ключ** | **Уникальность** | **Описание** |
| 1 | id | integer | Nextval  ('sq\_locations\_id') | Yes | - | yes | Уникальный  номер |
| 2 | city | Char | - | No | - | no | Город |
| 3 | street | Char | - | no | - | no | Улица |
| 4 | house | Integer | - | no | - | no | Номер дома |
| 5 | flat | Integer | - | no | - | no | Номер квартиры |

Рисунок 5 - Описание полей таблицы locations

Таблица содержит 5 столбцов:

1. id – уникальный номер каждой локации
2. city – город
3. street – улица
4. house – номер дома
5. flat – номер квартиры

На рисунке 6 приведён пример данных, находящихся в таблице locations



Рисунок 6 – Пример данных в таблице locations

На рисунке 7 изображён код для данной таблицы locations

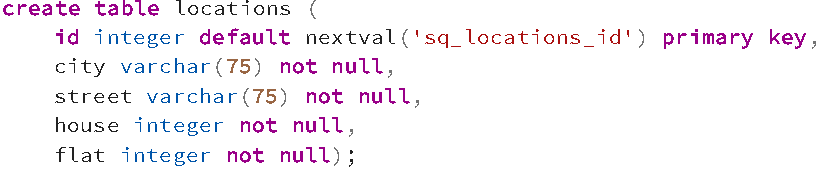


Рисунок 7 - код таблицы locations

Рассмотрим таблицу users (Рис. 8)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Значения**  **по умолчанию** | **Первичный ключ** | **Внешний ключ** | **Уникальность** | **Описание** |
| 1 | Id | Integer | Nextval(‘sq\_users\_id) | yes | - | yes | Уникальный номер |
| 2 | Name | char | 25 | no | - | no | Имя пользователя |
| 3 | Surname | char | 50 | no | - | no | Фамилия пользователя |

Рисунок 8 - Описание полей таблицы users

Таблица содержит 3 столбца:

1. id -Уникальный номер каждого владельца счётчика
2. name - Имя владельца
3. surname - Фамилия владельца

На рисунке 9 приведён пример данных, находящихся в таблице users

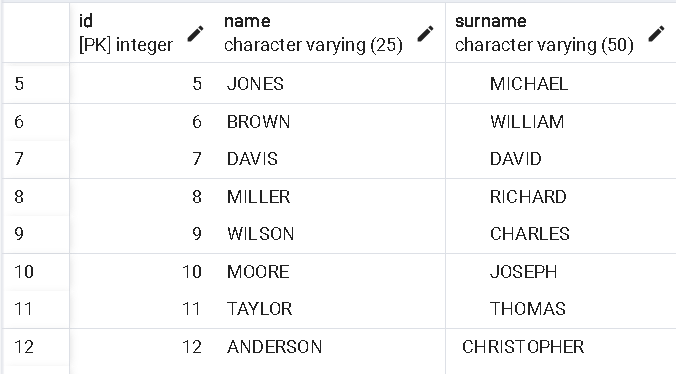


Рисунок 9 – Пример данных в таблице users

На рисунке 10 изображён код для данной таблицы users

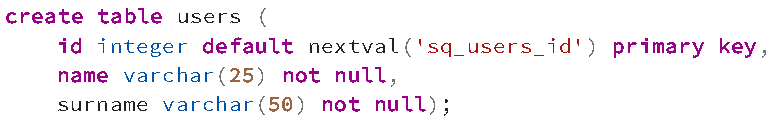


Рисунок 10 – код таблицы owners

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Значения по умолчанию** | **Первичный ключ** | **Внешний ключ** | **Уникальность** | **Описание** |
| 1 | id | integer | nextval ('sq\_couners\_id') | Yes | no | Yes | Номер счётчика |
| 2 | location\_id | integer | - | No | location.id | Yes | Адрес счётчика |
| 3 | type\_id | integer | - | No | types.id | No | Тип счётчика |
| 4 | rate\_id | integer | - | No | rates.id | No | Тип счётчика |

Рисунок 11 – Описание полей таблицы counters

Таблица имеет 3 столбца:

1. id – уникальный номер счётчика
2. location\_id – уникальный номер локации расположения счётчика
3. type\_id –уникальный номер типа счётчика
4. rate\_id -уникальный номер тарифа счётчика

На рисунке 12 приведён пример данных, находящихся в таблице

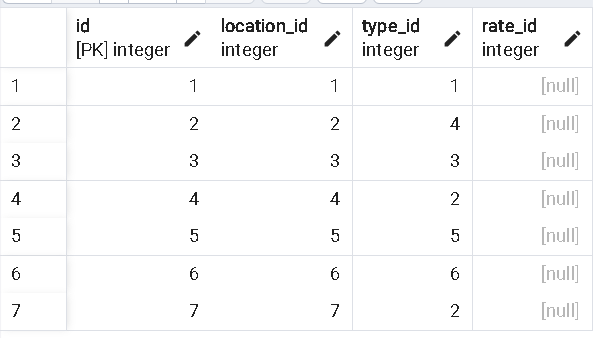


Рисунок 12 – Пример данных в таблице counters

На рисунке 13 изображён код для данной таблицы counters

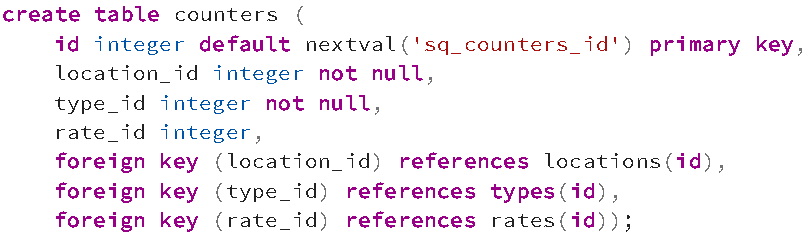


Рисунок 13 – код таблицы counters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Значения по умолчанию** | **Первичный ключ** | **Уникальность** | **Описание** |
| 1 | id | Integer | nextval('sq\_rates\_id') | yes | yes | Номер тарифа |
| 2 | since | timestamp | - | no | no | Время и дата вступления тарифа |
| 3 | cost | Float | - | no | no | Стоимость тарифа |

Рисунок 14 – описание столбцов в таблице rates

Таблица имеет 3 столбца:

1. id – уникальный номер каждого нового тарифа
2. since – дата вступления нового тарифа
3. cost – стоимость тарифов

На рисунке 15 приведён пример данных, находящихся в таблице rates

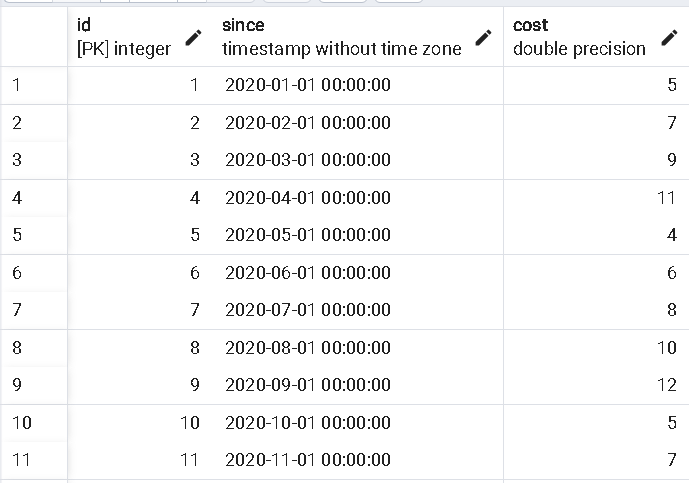


Рисунок 15 – Пример данных в таблице rates

На рисунке 16 изображён код для данной таблицы rates

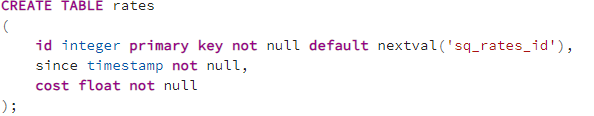


Рисунок 16 – код таблицы rates

Рассмотрим таблицу measurments (Рис. 17)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Размер** | **Значения по умолчанию** | **Первичный ключ** | **Внешний ключ** | **Уникальность** | **Описание** |  |  |  |  |  |
| 1 | Id | integer | - | Nextval  ('sq\_measurments\_id') | yes | no | yes | Номер показателя |  |  |  |  |  |
| 2 | users\_id | integer | - | - | no | users.id | no | Владелец счётчика |  |  |  |  |  |
| 3 | counter\_id | integer | - | - | no | counters.id | no | Счётчик |  |  |  |  |  |
| 4 | mark | timestamp | - | - | no | no | no | Время замера счётчика |  |  |  |  |  |
| 5 | value | float | - | - | no | no | no | Значение счётчика |  |  |  |  |  |

Рисунок 17 – описание столбцов таблицы measurments

Таблица содержит 5 столбцов:

1. Id – уникальный номер показателя счетчиков
2. users\_id - номер владельца счётчика
3. counter\_id – номер счётчика
4. mark – временная отметка измерений счётчиков
5. value – значение счётчиков

На рисунке 18 приведён пример данных, находящихся в таблице measurments

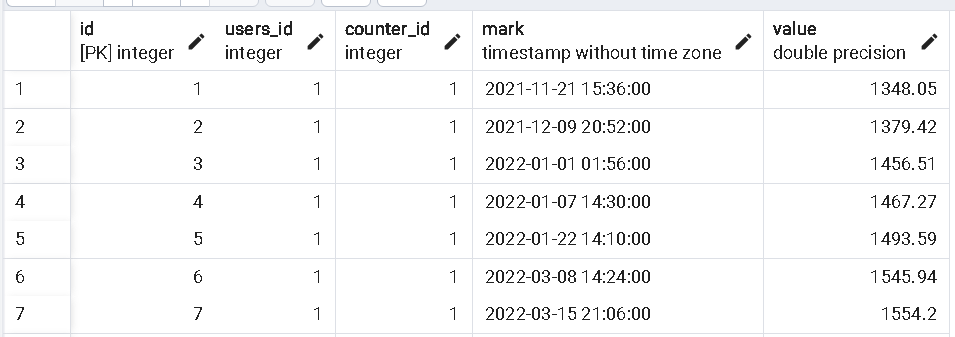


Рисунок 18 – Пример данных в таблице measurments

На рисунке 19 изображён код для данной таблицы measurments

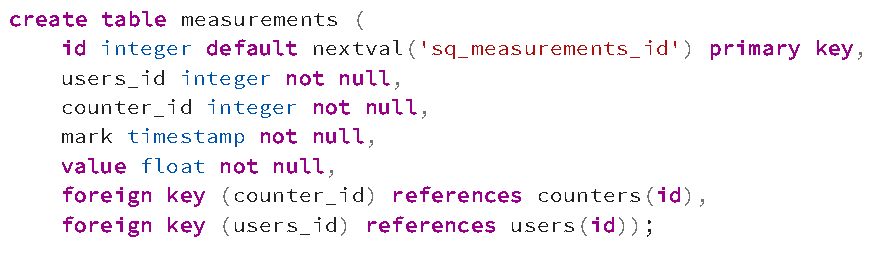


Рисунок 19 – Код таблицы measurments

1. **СТРУКТУРА ПРИЛОЖЕНИЯ**

Приложение написано на C# WinForms. Для этого внутри самого приложения надо указать данные для подключения к пустой базе данных.

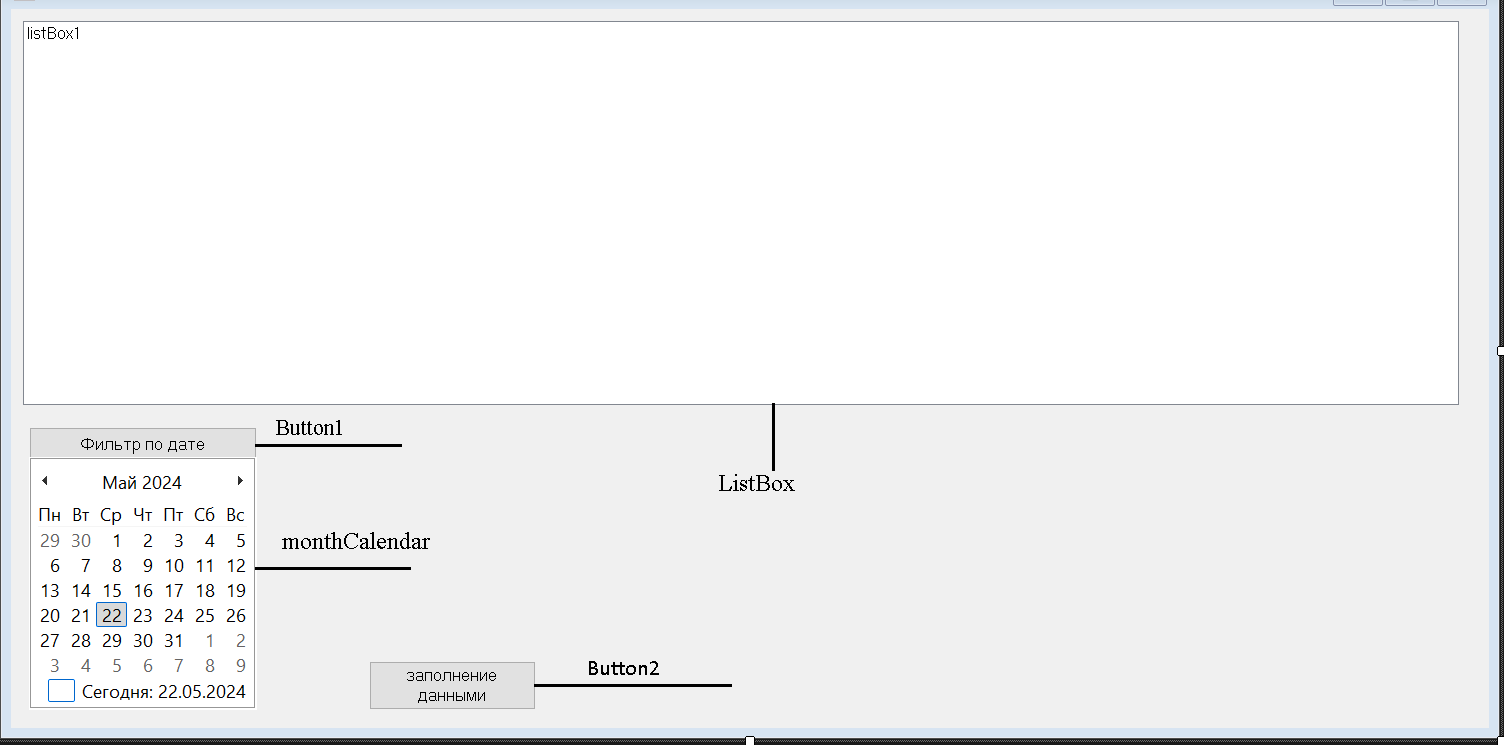
Рисунок 20 представляет из себя наглядный пример конструктора приложения. В данном конструкторе можно заметить 2 Button, 1 ListBox, 1 monthCalendar.

Рисунок 20 – Изображение приложения в конструкторе

В приложении всего 6 элементов:

1. ListBox – Данный элемент выводит данные из БД. (Рис.21)

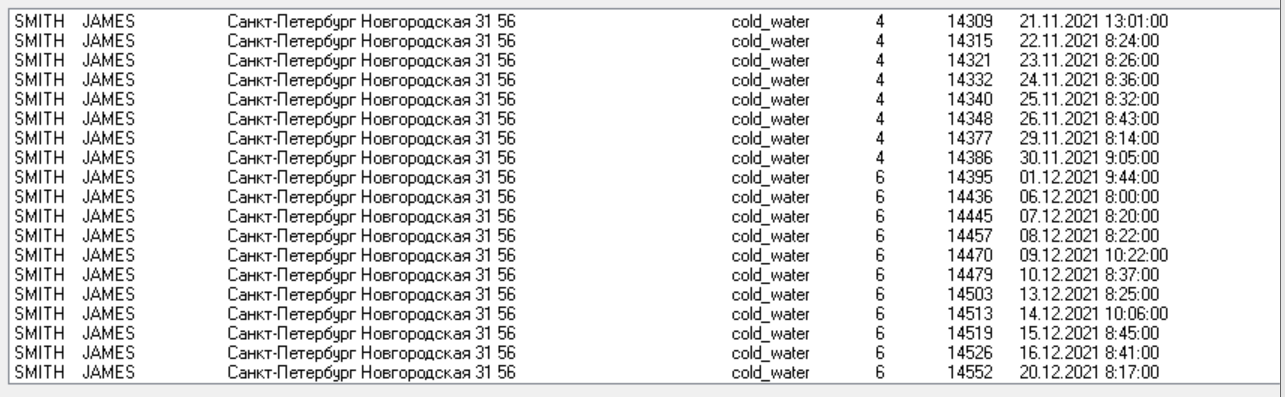


Рисунок 21 – Пример работы ListBox

1. Button1 – Фильтр по дате
2. monthCalendar - Данный элемент позволяет выбрать дату до какого числа нужно вывести данные в ListBox
3. Button2 – Фильтрует данные до даты выбранной в monthCalendar.

Коды каждого элемента:

1. ListBox – код принимает входящие данные из бд (Рис. 22)

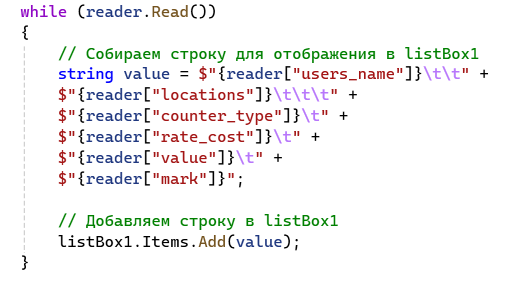
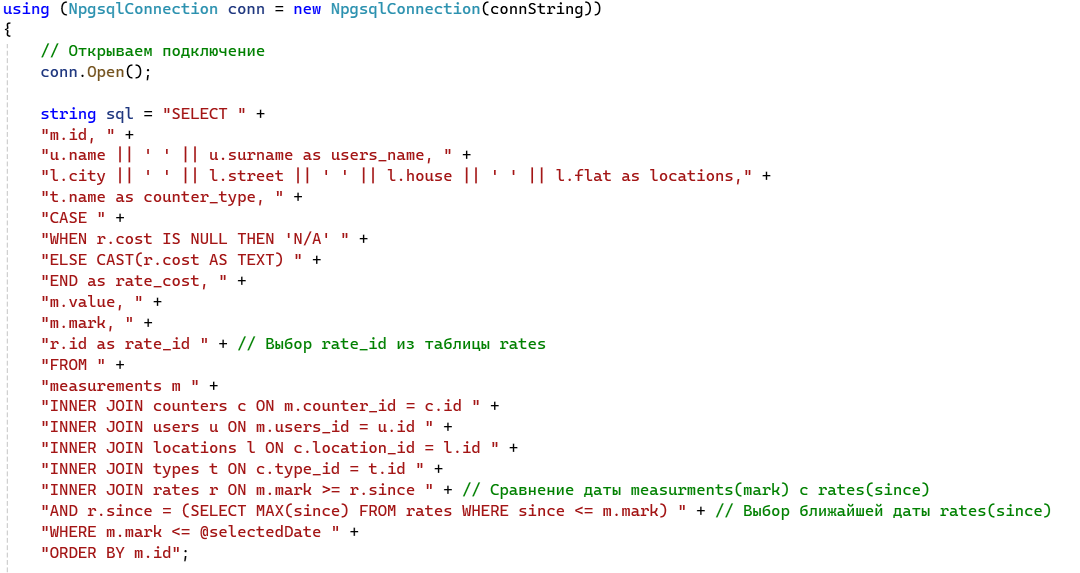


Рисунок 22 – Код заполнения ListBox

1. Button1 – Фильтрация данных по дате. Этот запрос SQL выбирает данные из нескольких таблиц базы данных PostgreSQL на основе определенных условий. Он соединяет таблицы "measurements", "counters", "users", "locations", "types" и "rates" с помощью операторов INNER JOIN. Затем он выбирает определенные столбцы из этих таблиц и выполняет несколько операций над данными, таких как объединение и преобразование типов. Наконец, запрос фильтрует данные по имени и фамилии владельца и сортирует результаты по идентификатору измерения "m.id".

  
Рисунок 23 – Фильтрация по дате

1. MonthCalendar – Выбор даты до которой надо отфильтровать (Рис. 24)



Рисунок 24 – Вывод даты из элемента

# **ЗАКЛЮЧЕНИЕ**

Данный курсовой проект посвящен разработке базы данных и созданию соответствующего приложения для управления и автоматизации баз данных, в контексте учета и мониторинга данных о потреблении ресурсов, таких как энергия.

В ходе работы были рассмотрены ключевые аспекты проекта, начиная с актуальности темы в свете постоянного развития информационных технологий. Разработка эффективных баз данных становится неотъемлемой частью современного мира, где эффективное управление данными играет ключевую роль в оптимизации управления энергоресурсами.

Основной целью проекта было создание надежной и функциональной базы данных, а также разработка приложения для взаимодействия с этой базой данных. Для достижения этой цели были определены требования к базе данных, разработана ее структура, а также создан пользовательский интерфейс для удобного доступа к данным.

Структура базы данных включает в себя восемь таблиц, каждая из которых имеет четко определенные поля и отвечает за хранение определенных категорий информации, начиная от данных об улицах и районах, заканчивая данными о тарифах и измерениях счетчиков.

Приложение, разработанное на языке программирования C# с использованием WinForms, предоставляет удобный интерфейс для работы с базой данных. Возможности фильтрации данных по различным параметрам, а также просмотр данных в удобном формате обеспечивают удобство использования приложения как для администраторов, так и для обычных пользователей.

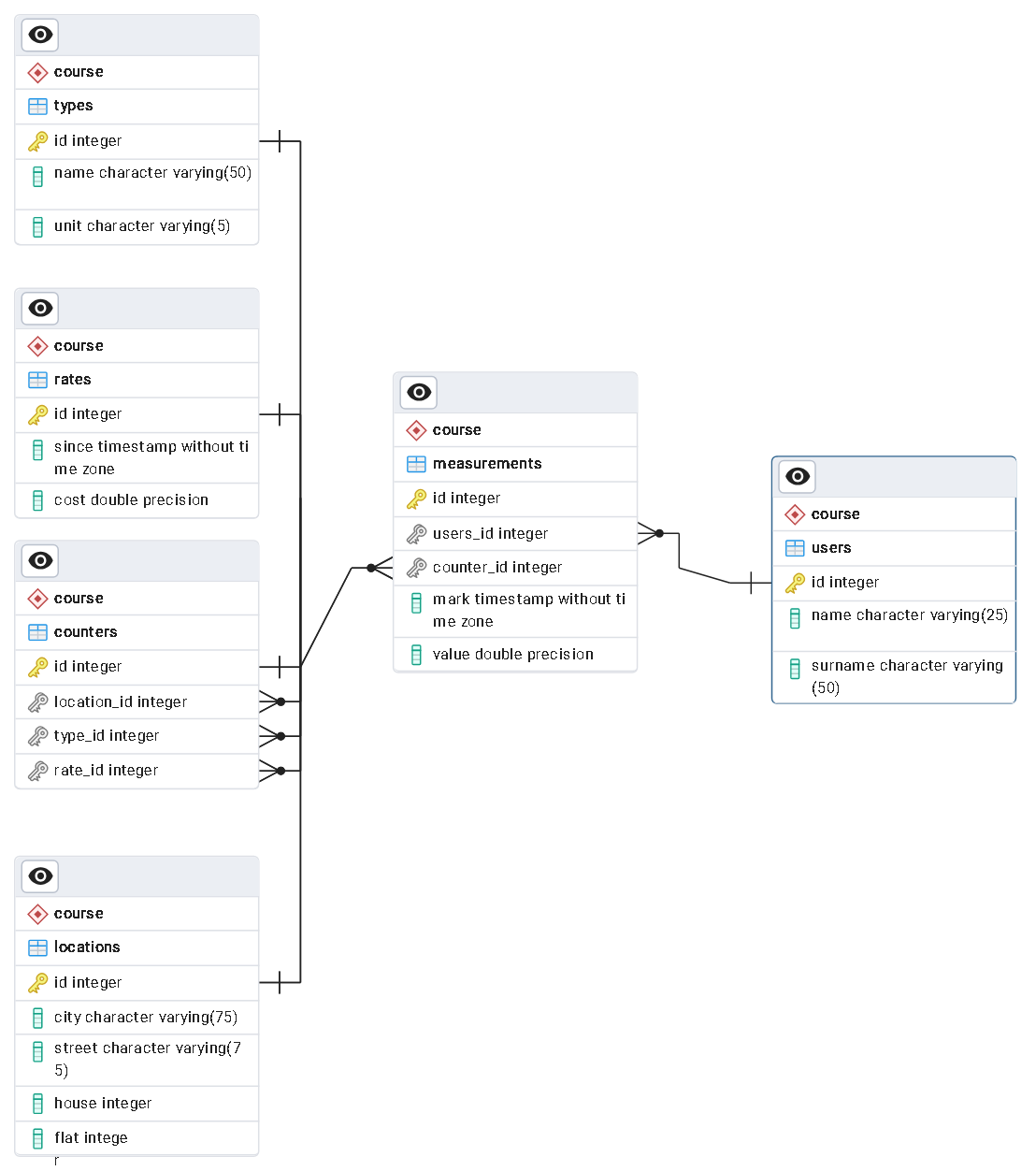
Таким образом, данный проект не только соответствует поставленным целям и задачам, но и представляет собой значимый вклад в область управления и автоматизации баз данных, способствуя оптимизации управления энергоресурсами и повышению эффективности процессов учета и мониторинга данных.

# **ЛИТЕРАТУРА**

1. "Официальная документация PostgreSQL" (<https://www.postgresql.org/docs/>)
2. "Apress. PostgreSQL для профессионалов" (Айман Гиллеспи, 2020)
3. "Packt Publishing. C# 8.0 и платформа .NET Core 3.0" (Пол Деккер и др., 2019)
4. "Технология разработки программного обеспечения: учебное пособие" (Г.Н Федорова, 2020)
5. "C# Programming for Beginners" (Тим Академия, 2019)
6. "Основы разработки приложений на платформе Microsoft .NET Framework с использованием Windows Forms" (Microsoft)
7. "WinForms Programming in C#" (Различные авторы)
8. "Документация по среде разработки Microsoft Visual Studio" (<https://docs.microsoft.com/>)
9. "Документация по системе управления базами данных PostgreSQL" (<https://www.postgresql.org/docs/>)
10. "Metanit. Документация по фреймворку базы данных" (<https://metanit.com/>)
11. "Microsoft. Документация по Windows Forms" (<https://docs.microsoft.com/>)
12. "Microsoft. Документация по .NET Framework" (<https://docs.microsoft.com/>)
13. "Microsoft. Документация по C# programming language" (<https://docs.microsoft.com/>)
14. "C# Station. WinForms Tutorial" (https://www.csharp-station.com/tutorial.aspx)
15. "Codecademy. Learn C#" (https://www.codecademy.com/learn/learn-c-sharp)

# **ПРИЛОЖЕНИЕ**

# **Приложение ERD схема в расширениях pgerd и png**



# **Приложение Скрипт создание схемы в БД**

drop schema if exists course cascade;

create schema course;

set search\_path = 'course';

create sequence sq\_users\_id;

create sequence sq\_types\_id;

create sequence sq\_locations\_id;

create sequence sq\_rates\_id;

create sequence sq\_counters\_id;

create sequence sq\_measurements\_id;

create table users (

id integer default nextval('sq\_users\_id') primary key,

name varchar(25) not null,

surname varchar(50) not null);

create table types (

id integer default nextval('sq\_types\_id') primary key,

name varchar(50) not null,

unit varchar(5) not null);

create table locations (

id integer default nextval('sq\_locations\_id') primary key,

city varchar(75) not null,

street varchar(75) not null,

house integer not null,

flat integer not null);

create table rates (

id integer default nextval('sq\_rates\_id') primary key,

since timestamp,

cost float);

create table counters (

id integer default nextval('sq\_counters\_id') primary key,

location\_id integer not null,

type\_id integer not null,

rate\_id integer,

foreign key (location\_id) references locations(id),

foreign key (type\_id) references types(id),

foreign key (rate\_id) references rates(id));

create table measurements (

id integer default nextval('sq\_measurements\_id') primary key,

users\_id integer not null,

counter\_id integer not null,

mark timestamp not null,

value float not null,

foreign key (counter\_id) references counters(id),

foreign key (users\_id) references users(id));

alter table counters add constraint fk\_counters\_locations foreign key (location\_id) references locations (id) on delete cascade;

alter table counters add constraint fk\_counters\_types foreign key (type\_id) references types (id) on delete cascade;

alter table counters add constraint fk\_counters\_rates foreign key (rate\_id) references rates (id) on delete cascade;

alter table measurements add constraint fk\_measurements\_counters foreign key (counter\_id) references counters (id) on delete cascade;

--check

alter table locations add constraint chk\_locations\_city\_length check (length (city) >= 2);

alter table locations add constraint chk\_locations\_street\_length check (length (street) >= 2);

alter table locations add constraint chk\_locations\_house\_positive check (house > 0);

alter table locations add constraint chk\_locations\_flat\_positive check (flat > 0);

alter table rates add constraint chk\_rates\_since\_after\_date check (since >= '2020-01-01'::timestamp);

alter table rates add constraint chk\_rates\_since\_cost\_positive\_or\_zero check (cost > 0.0);

create function fn\_last\_counter\_value(counter integer) returns float language sql as $$

select value from measurements where counter\_id = counter order by mark desc limit 1;

$$;

alter table measurements add constraint chk\_measurements\_mark\_after\_date check (mark >= '2020-01-01'::timestamp);

alter table measurements add constraint chk\_measurements\_mark\_not\_in\_future check (mark < now());

alter table measurements add constraint chk\_measurements\_value\_positive\_or\_zero check (value >= 0.0);

# **Приложение Разработка приложения C# WinForms**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System;

using System.Data;

using System.Data.SqlClient;

using Npgsql;

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

using System.Xml.Linq;

namespace cource\_db

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

string connString = "Host=localhost;Port=5432;Username=postgres;Password=1;Database=postgres";

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

{

using (NpgsqlConnection conn = new NpgsqlConnection(connString))

{

conn.Open();

string copyTypesCountersSql = "COPY types(name, unit) FROM 'C:/hhh/db/type\_counters.csv' DELIMITER ',' CSV;";

using (NpgsqlCommand copyTypesCountersCmd = new NpgsqlCommand(copyTypesCountersSql, conn))

{

copyTypesCountersCmd.ExecuteNonQuery();

}

}

using (NpgsqlConnection conn = new NpgsqlConnection(connString))

{

conn.Open();

string CopyLocationsSql = "COPY locations(city, street, house, flat) FROM 'C:/hhh/db/course\_work/Locations.csv' DELIMITER ',' CSV; ";

using (NpgsqlCommand generateAreasStreetsCmd = new NpgsqlCommand(CopyLocationsSql, conn))

{

generateAreasStreetsCmd.ExecuteNonQuery();

}

}

using (NpgsqlConnection conn = new NpgsqlConnection(connString))

{

conn.Open();

string copyRatesSql = "COPY rates(since, cost) FROM 'C:/hhh/db/course\_work/Rates.csv' DELIMITER ',' CSV;";

using (NpgsqlCommand copyRatesCmd = new NpgsqlCommand(copyRatesSql, conn))

{

copyRatesCmd.ExecuteNonQuery();

}

}

using (NpgsqlConnection conn = new NpgsqlConnection(connString))

{

conn.Open();

string copyNameSql = "COPY users(name, surname) FROM 'C:/hhh/db/Code\_data\_all\_name.csv' DELIMITER ',' CSV;";

using (NpgsqlCommand copyNameCmd = new NpgsqlCommand(copyNameSql, conn))

{

copyNameCmd.ExecuteNonQuery();

}

}

LoadMeasurements();

}

private void LoadMeasurements()

{

using (NpgsqlConnection conn = new NpgsqlConnection(connString))

{

conn.Open();

string sql = "SELECT " +

"m.id, " +

"u.name || ' ' || u.surname as users\_name, " +

"l.city || ' ' || l.street || ' ' || l.house || ' ' || l.flat as locations," +

"t.name as counter\_type, " +

"CASE " +

"WHEN r.cost IS NULL THEN 'N/A' " +

"ELSE CAST(r.cost AS TEXT) " +

"END as rate\_cost, " +

"m.value, " +

"m.mark, " +

"r.id as rate\_id " +

"FROM " +

"measurements m " +

"INNER JOIN counters c ON m.counter\_id = c.id " +

"INNER JOIN users u ON m.users\_id = u.id " +

"INNER JOIN locations l ON c.location\_id = l.id " +

"INNER JOIN types t ON c.type\_id = t.id " +

"INNER JOIN rates r ON m.mark >= r.since " +

"AND r.since = (SELECT MAX(since) FROM rates WHERE since <= m.mark) " +

"ORDER BY m.id";

using (NpgsqlCommand cmd = new NpgsqlCommand(sql, conn))

{

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

listBox1.Items.Clear();

while (reader.Read())

{

string value = $"{reader["users\_name"]}\t\t" +

$"{reader["locations"]}\t\t\t" +

$"{reader["counter\_type"]}\t" +

$"{reader["rate\_cost"]}\t" +

$"{reader["value"]}\t" +

$"{reader["mark"]}";

listBox1.Items.Add(value);

}

}

}

}

}

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

{

DateTime selectedDate = monthCalendar1.SelectionStart;

using (NpgsqlConnection conn = new NpgsqlConnection(connString))

{

conn.Open();

string sql = "SELECT " +

"m.id, " +

"u.name || ' ' || u.surname as users\_name, " +

"l.city || ' ' || l.street || ' ' || l.house || ' ' || l.flat as locations," +

"t.name as counter\_type, " +

"CASE " +

"WHEN r.cost IS NULL THEN 'N/A' " +

"ELSE CAST(r.cost AS TEXT) " +

"END as rate\_cost, " +

"m.value, " +

"m.mark, " +

"r.id as rate\_id " +

"FROM " +

"measurements m " +

"INNER JOIN counters c ON m.counter\_id = c.id " +

"INNER JOIN users u ON m.users\_id = u.id " +

"INNER JOIN locations l ON c.location\_id = l.id " +

"INNER JOIN types t ON c.type\_id = t.id " +

"INNER JOIN rates r ON m.mark >= r.since " +

"AND r.since = (SELECT MAX(since) FROM rates WHERE since <= m.mark) " +

"WHERE m.mark <= @selectedDate " +

"ORDER BY m.id";

using (NpgsqlCommand cmd = new NpgsqlCommand(sql, conn))

{

cmd.Parameters.AddWithValue("@selectedDate", selectedDate);

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

listBox1.Items.Clear();

while (reader.Read())

{

string value = $"{reader["users\_name"]}\t" +

$"{reader["locations"]}\t" +

$"{reader["counter\_type"]}\t" +

$"{reader["rate\_cost"]}\t" +

$"{reader["value"]}\t" +

$"{reader["mark"]}";

listBox1.Items.Add(value);

}

}

}

}

}

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

{

LoadMeasurements();

}

}

}