using System.Data.SQLite;

using System;

using System.Windows.Forms;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using System.Collections.Generic;

namespace meteo

{

public class db

{

public static void WriteDB(string data)

{

string dbPath = "meteo.db";

SQLiteConnection sqlite\_conn = new SQLiteConnection($"Data Source={dbPath};Version=3;New=True;Compress=True;");

sqlite\_conn.Open();

try

{

// Парсинг JSON-строки

JObject json = JObject.Parse(data);

JArray sensors = (JArray)json["sensors"];

// Подготовка параметризованного запроса

string query = @"INSERT INTO data1 (value, time) VALUES (@value1, @unix\_time);

INSERT INTO data2 (value, time) VALUES (@value2, @unix\_time);

INSERT INTO data3 (value, time) VALUES (@value3, @unix\_time);

INSERT INTO data4 (value, time) VALUES (@value4, @unix\_time);

INSERT INTO data5 (value, time) VALUES (@value5, @unix\_time);";

SQLiteCommand command = new SQLiteCommand(query, sqlite\_conn);

// Добавление параметров

command.Parameters.AddWithValue("@value1", (double)sensors[0]["value"]);

command.Parameters.AddWithValue("@value2", (double)sensors[1]["value"]);

command.Parameters.AddWithValue("@value3", (double)sensors[2]["value"]);

command.Parameters.AddWithValue("@value4", (double)sensors[3]["value"]);

command.Parameters.AddWithValue("@value5", (double)sensors[4]["value"]);

command.Parameters.AddWithValue("@unix\_time", (int)(DateTime.UtcNow.Subtract(new DateTime(1970, 1, 1))).TotalSeconds);

// Выполнение запроса

command.ExecuteNonQuery();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

// Закрытие соединения с базой данных

sqlite\_conn.Close();

}

}

public List<Tuple<string, string, double>> ReadDB(int device)

{

string dbPath = "meteo.db";

SQLiteConnection sqlite\_conn = new SQLiteConnection($"Data Source={dbPath};Version=3;New=True;Compress=True;");

sqlite\_conn.Open();

using (SQLiteCommand command = new SQLiteCommand(

$"CREATE TABLE IF NOT EXISTS dataTemp1" +

$" (id INTEGER PRIMARY KEY AUTOINCREMENT, unix INTEGER, date TEXT, time TEXT, value REAL);",

sqlite\_conn))

{

command.ExecuteNonQuery();

}

using (SQLiteCommand command = new SQLiteCommand(

$"CREATE TABLE IF NOT EXISTS dataTemp2" +

$" (id INTEGER PRIMARY KEY AUTOINCREMENT, unix INTEGER, date TEXT, time TEXT, value REAL);",

sqlite\_conn))

{

command.ExecuteNonQuery();

}

using (SQLiteCommand command = new SQLiteCommand(

$"CREATE TABLE IF NOT EXISTS dataTemp3" +

$" (id INTEGER PRIMARY KEY AUTOINCREMENT, unix INTEGER, date TEXT, time TEXT, value REAL);",

sqlite\_conn))

{

command.ExecuteNonQuery();

}

using (SQLiteCommand command = new SQLiteCommand(

$"CREATE TABLE IF NOT EXISTS dataPressure" +

$" (id INTEGER PRIMARY KEY AUTOINCREMENT, unix INTEGER, date TEXT, time TEXT, value REAL);",

sqlite\_conn))

{

command.ExecuteNonQuery();

}

using (SQLiteCommand command = new SQLiteCommand(

$"CREATE TABLE IF NOT EXISTS dataHumadity" +

$" (id INTEGER PRIMARY KEY AUTOINCREMENT, unix INTEGER, date TEXT, time TEXT, value REAL);",

sqlite\_conn))

{

command.ExecuteNonQuery();

}

string sql = "";

if( device == 1)

{

sql = "SELECT date, time, value FROM dataTemp1";

}

else if( device == 2) {

sql = "SELECT date, time, value FROM dataTemp2";

}

else if (device == 3)

{

sql = "SELECT date, time, value FROM dataTemp3";

}

else if (device == 4)

{

sql = "SELECT date, time, value FROM dataPressure";

}

else if (device == 5)

{

sql = "SELECT date, time, value FROM dataHumadity";

}

SQLiteCommand sqlite\_cmd = new SQLiteCommand(sql, sqlite\_conn);

SQLiteDataReader sqlite\_datareader = sqlite\_cmd.ExecuteReader();

List<Tuple<string, string, double>> result = new List<Tuple<string, string, double>>();

while (sqlite\_datareader.Read())

{

string date = sqlite\_datareader.GetString(0);

string time = sqlite\_datareader.GetString(1);

double value = sqlite\_datareader.GetDouble(2);

Tuple<string, string, double> row = new Tuple<string, string, double>(date, time, value);

result.Add(row);

}

sqlite\_datareader.Close();

sqlite\_cmd.Dispose();

sqlite\_conn.Close();

return result;

}

}

}