МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ “ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ П.М. МАШЕРОВА”

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ГРАЖДАНСКОГО ПРАВА И ГРАЖДАНСКОГО ПРОЦЕССА

**КУРСОВОЙ ПРОЕКТ**

**по дисциплине «Современное построение  
реляционных баз данных»**

Автоматизация слияния ADO.NET SQL-таблиц с использованием XML-сериализации

Белькевич Глеб   
Александрович  
3 курс  
31 группа

Проверил:  
Ермоченко Сергей   
Александрович

Витебск, 2021

Оглавление

[Реферат 4](#_Toc60686260)

[Перечень ключевых слов 4](#_Toc60686261)

[Обзор используемой аппаратуры, инструментов 4](#_Toc60686262)

[MS SQL Server 4](#_Toc60686263)

[C# 4](#_Toc60686264)

[Winforms 5](#_Toc60686265)

[Теоретический блок. Термины. Сокращения 5](#_Toc60686266)

[C# 5](#_Toc60686267)

[MS SQL 5](#_Toc60686268)

[WinForms 8](#_Toc60686269)

[ADO.NET 8](#_Toc60686270)

[Введение 9](#_Toc60686271)

[Цели и задачи 9](#_Toc60686272)

[Объект и предмет исследования 9](#_Toc60686273)

[Акутальность и область применения 9](#_Toc60686274)

[Обзор литературных источников 9](#_Toc60686275)

[Ход работы 11](#_Toc60686276)

[Процесс разработки 11](#_Toc60686277)

[Каркас приложения 11](#_Toc60686278)

[Создание подключения 13](#_Toc60686279)

[Создание запросов 14](#_Toc60686280)

[Создание таблиц в пустой БД 15](#_Toc60686281)

[Отображение данных из выбранной таблицы 15](#_Toc60686282)

[Удаление данных из БД 16](#_Toc60686283)

[Изменение данных после их редактирования в DataGridView 16](#_Toc60686284)

[Завершение работы 18](#_Toc60686285)

[Демонтрация работы 18](#_Toc60686286)

[Запуск программы 18](#_Toc60686287)

[Главное окно 19](#_Toc60686288)

[Выбор таблицы в списке 19](#_Toc60686289)

[Отправка запроса 20](#_Toc60686290)

[Добавление новых записей 21](#_Toc60686291)

[Удаление имеющихся записей 21](#_Toc60686292)

[Изменение данных через DataGridView 22](#_Toc60686293)

[Заключение 22](#_Toc60686294)

[Список использованных источников 24](#_Toc60686295)

[Книги 24](#_Toc60686296)

[Интернет-ресурсы 24](#_Toc60686297)

[Интернет-ресурсы (образовательные YouTube-каналы от .NET Senior-разработчиков) 24](#_Toc60686298)

[Приложения 25](#_Toc60686299)

[Схема БД ITBeacon 25](#_Toc60686300)

[MainForm.cs 25](#_Toc60686301)

[SendRequestForm.cs 32](#_Toc60686302)

[ConnectionForm.cs 33](#_Toc60686303)

[Sql-запрос для создания таблиц в пустой БД 34](#_Toc60686304)

# Введение

## Цели и задачи

Изучить принципы построения панели управления баз данных, освоить взаимодействие баз данных с ранее освоенными контейнерными классами, способы применения данных из баз в пользовательских приложениях, углубить знания работы с бд и языками программирования.

## Объект и предмет исследования

Предметом исследования, в данном случае, является проблема неудобства управления информацией, хранящейся в базе данных.

Объектом исследования являются неудобства при работе с базами данных.

## Акутальность и область применения

Прежде, проблемой при работе с базами данных было неудобство редактирования информации из таблиц.

Необходимо было писать sql-запросы с правильным синтаксисом, больше концентрируясь на том, "как" нужно писать, нежели о том, "что".

Данный проект позволяет экономить время, требуемое для управления информацией, хранящейся в базе данных.

Проект представляет удобный интуитивный интерфейс для решения данной проблемы, построенный с использованием всех современных принципов построения программных продуктов.

Поскольку проект смоделирован таким образом, что может быть использован для работы с любой sql-бд, любви количеством таблиц и полями, сфера применения данного проекта по истине огромна: от банального списка дел или маленького интернет-ресурса, до крупных онлайн-проектов как в сфере продаж, так и в игровой индустрии. В связи с данными фай ю фактами, проект был нужен раньше, актуален сейчас и будет оставаться таковым ещё неопределенно долгий срок.

Взаимодействие с базой данных, редактирование хранящейся информации, ее добавление и удаление, ровно как и просмотр с выборкой были нужны всегда и у этой потребности нет предпосылок к утрате значимости

## Обзор литературных источников

Для достижения поставленных целей и задач было принято решение использовать следующие источники:

* Официальная документация Microsoft для освоения отдельных команд и технологий, изучения всех параметров, нюансов применения, исключительных ситуаций и т.п. т.к. это надёжный источник от разработчиков использованных технологий (языка программирования C# и использованного в примере MS SQL Server'а)
* Информационный портал metanit использовался для получения основных сведений о часто используемых командах в проекте.
* Книга Албахари Джозефа служила основным источником знаний о языке программирования C#.
* Информацию об sql как о языке, черпал из официальной документации PostgreSQL т.к. она написана доступнее остальных.
* YouTube-каналы XpucT, Byte++ и другие, служили дополнительным источником информации по реализации взаимодействия двух основных средств написания проекта.
* YouTube-канал SimpleCode служил основным помощником в решении проблем связанных с пониманием основных аспектов написания программной части проекта.
* YouTube-канал Роман Сакутин использовался для написания чистого кода, увеличения показателей быстродействия кода.

## Обзор используемой аппаратуры, инструментов

### MS SQL Server

MS SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

### C#

Преимущества C#:

* C# – это объектно-ориентированный, простой и в то же время мощный язык программирования, который позволяет разработчикам создавать многофункциональные приложения.
* C# относиться к языкам компилируемого типа, поэтому он обладает всеми преимуществами таких языков.
* C# объединяет лучшие идеи современных языков программирования Java, C++, Visual Basic и т.д.
* Из-за большого разнообразия синтаксических конструкций и возможности работать с платформой .Net, C# позволяет быстрее, чем любой другой язык, разрабатывать программные решения.
* C# отличается надежностью и элегантностью

Примечание:

* Элегантностью C# отличается из-за большого разнообразия синтаксических конструкций.
* Большая надёжность была достигнута из-за работы CLR машины, ведь в отличии от других компилятор CLR запускает разработанное приложение на виртуальном процессоре. Поэтому в случаях возникновения каких-либо ошибок, это никак не повлияет на работу других программ в системе, но это также обозначает, что для запуска программы необходимо дополнительное время. Соответственно программы написанные на языке программирования C# более надёжные, но менее быстрые (нежели те же программы написанные на C++).

### Winforms

В отличии от своего прямого конкурента, WPF, выбор пал на WinForms по ряду причин:

Преимущества WinForms

* Технология более старая и, соответственно, лучше испытанная и протестированная
* На данный момент существует огромное множество готовых элементов управления, которые можно купить либо использовать бесплатно
* С точки зрения написания, дизайнер Visual Studio лучше приспособлен к WinForms, так как в WPF больше необходимо делать самому
* Прост в изучении
* Лично мной она изучена лучше, и, соответственно, качество, в таком случае, будет выше.

# Ход работы

## Теоретический блок. Термины. Сокращения

### C#

C# (произносится си шарп) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота[7] как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

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 — С#, 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).

### MS SQL

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

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 16, которая вышла в 2016 году и которая будет использоваться в текущем руководстве.

SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

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

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Изначально язык SQL был разработан в компании IBM для системы баз данных, которая называлась System/R. При этом сам язык назывался SEQUEL (Structured English Query Language). Хотя в итоге ни база данных, ни сам язык не были впоследствии официально опубликованы, по традиции сам термин SQL нередко произносят как "сиквел".

В 1979 году компания Relational Software Inc. разработала первую систему управления баз данных, которая называлась Oracle и которая использовала язык SQL. В связи с успехом данного продукта компания была переименована в Oracle.

Впоследствии стали появляться другие системы баз данных, которые использовали SQL. В итоге в 1989 году Американский Национальный Институт Стандартов (ANSI) кодифицировал язык и опубликовал его первый стандарт. После этого стандарт периодически обновлялся и дополнялся. Последнее его обновление состоялось в 2011 году. Но несмотря на наличие стандарта нередко производители СУБД используют свои собственные реализации языка SQL, которые немного отличаются друг от друга.

Выделяются две разновидности языка SQL: PL-SQL и T-SQL. PL-SQL используется в таких СУБД как Oracle и MySQL. T-SQL (Transact-SQL) применяется в SQL Server. Собственно поэтому в рамках текущего руководства будет рассматриваться именно T-SQL.

В зависимости от задачи, которую выполняет команда T-SQL, он может принадлежать к одному из следующих типов:

* **DDL** (Data Definition Language / Язык определения данных). К этому типу относятся различные команды, которые создают базу данных, таблицы, индексы, хранимые процедуры и т.д. В общем определяют данные.

В частности, к этому типу мы можем отнести следующие команды:

* + **CREATE**: создает объекты базы данных (саму базу даных, таблицы, индексы и т.д.)
  + **ALTER**: изменяет объекты базы данных
  + **DROP**: удаляет объекты базы данных
  + **TRUNCATE**: удаляет все данные из таблиц
* **DML** (Data Manipulation Language / Язык манипуляции данными). К этому типу относят команды на выбору данных, их обновление, добавление, удаление - в общем все те команды, с помощью которыми мы можем управлять данными.

К этому типу относятся следующие команды:

* + **SELECT**: извлекает данные из БД
  + **UPDATE**: обновляет данные
  + **INSERT**: добавляет новые данные
  + **DELETE**: удаляет данные
* **DCL** (Data Control Language / Язык управления доступа к данным). К этому типу относят команды, которые управляют правами по доступу к данным. В частности, это следующие команды:
  + **GRANT**: предоставляет права для доступа к данным
  + **REVOKE**: отзывает права на доступ к данным

### WinForms

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

### ADO.NET

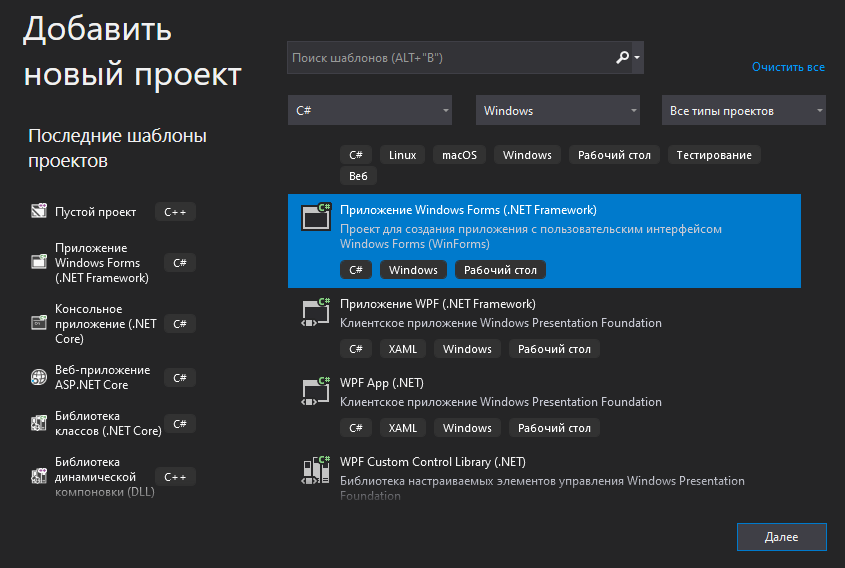
ADO.NET (ActiveX Data Object для .NET) — технология, предоставляющая доступ и управление данными, хранящимся в базе данных или других источниках (Microsoft SQL Server, Microsoft Access, Microsoft Excel, Microsoft Outlook, Microsoft Exchange, Oracle, OLE DB, ODBC, XML, текстовые файлы)[1], основанных на платформе .NET Framework и входящая в состав .NET Framework 2.0, представляет собой набор библиотек[2]. В отличие от технологии ADO, которая была в основном предназначена для тесно связанных клиент-серверных систем, ADO.NET больше нацелена на автономную работу с помощью объектов DataSet. Объекты DataSet представляют локальные копии взаимосвязанных таблиц данных, каждая из которых содержит набор строк и столбцов. Объекты DataSet позволяют вызывающей сборке (наподобие веб-страницы или программы, выполняющейся на настольном компьютере) работать с содержимым DataSet, изменять его, не требуя подключения к источнику данных, и отправлять обратно блоки измененных данных для обработки с помощью соответствующего адаптера данных. Но, пожалуй, самое фундаментальное различие между классической ADO и ADO.NET состоит в том, что ADO.NET является управляемой кодовой библиотекой, и, значит, подчиняется тем же правилам, что и любая управляемая библиотека. Типы, составляющие ADO.NET, используют протокол управления памятью CLR, принадлежат к той же системе типов (классы, интерфейсы, перечисления, структуры и делегаты), и доступ к ним возможен с помощью любого языка .NET. Классы ADO.NET находятся в сборке System.Data.dll.

## Процесс разработки

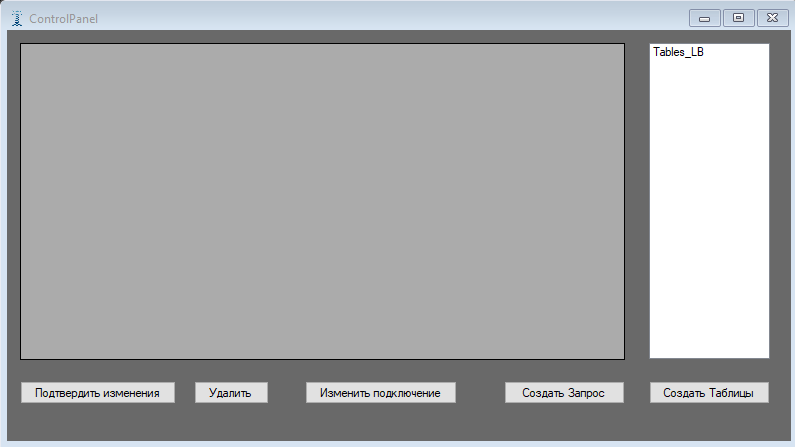
Для тестирования приложения будет использоваться MS SQL Server-ная БД для сайта школы программирования ITBeacon, которая хранит информацию о зарегистрированных пользователях, чатах, курсах и их расписании. (схему БД можно найти [тут](#_Схема_БД_ITBeacon)).

### Каркас приложения

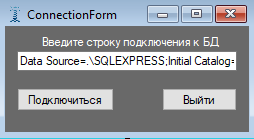
Для начала, создадим проект WinForms (.NET Framework) в программе Visual Studio 2019.



Разместим необходимые элементы на форме и переименуем их в свойствах.

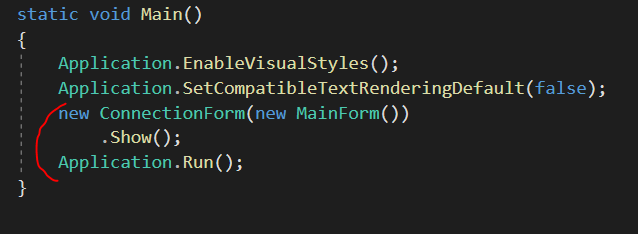


Создадим ещё одну форму для подключения к БД и, аналогично, разместим на ней элементы.

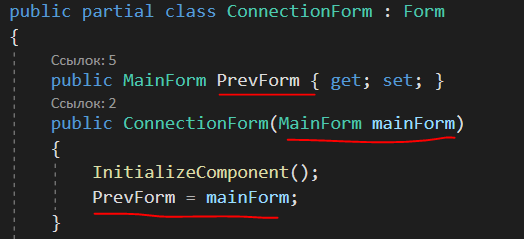


Изменим главную запускаемую форму и, соответственно, изменим обработку открытия/закрытия форм

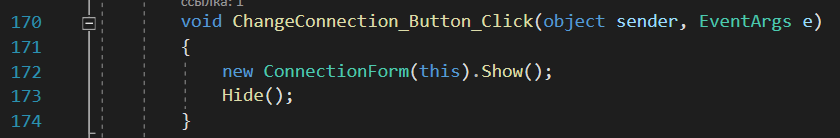
Program.cs:

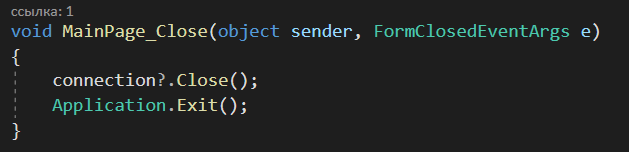


ConnectionForm.cs:



MainForm.cs:

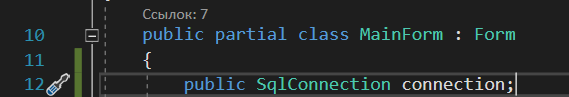




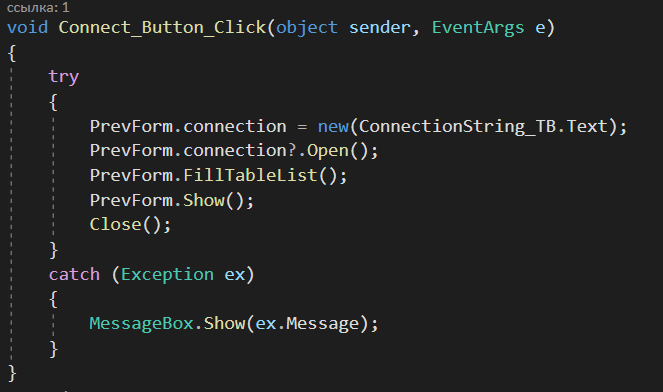
### Создание подключения

Обработаем подключение к БД, создав SqlConnection из ADO.NET в главной форме и обращаясь к ней из формы подключения:

MainForm.cs:

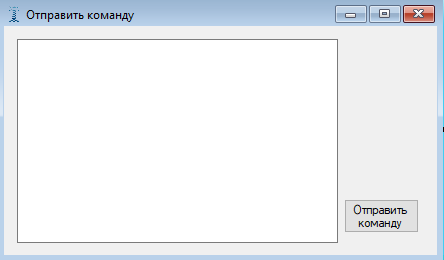


ConnectionForm.cs:

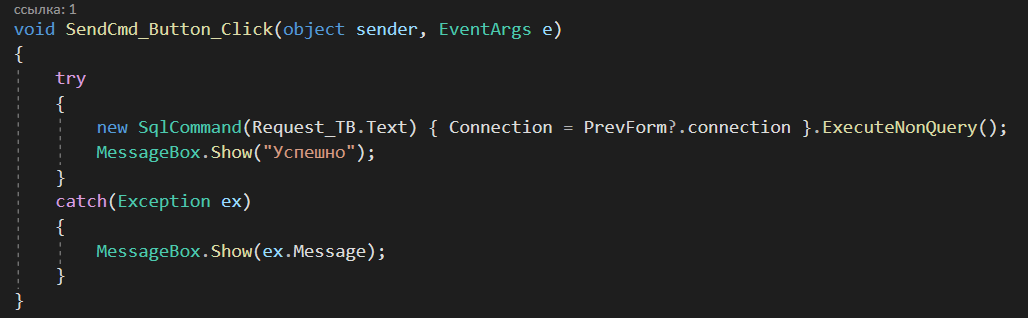


### Создание запросов

Добавим первый функционал к программе: отправка NonQuery-запроса и корректно обработаем возникающие в процессе ошибки. Для этого создадим ещё одну форму и разместим на ней элементы управления:

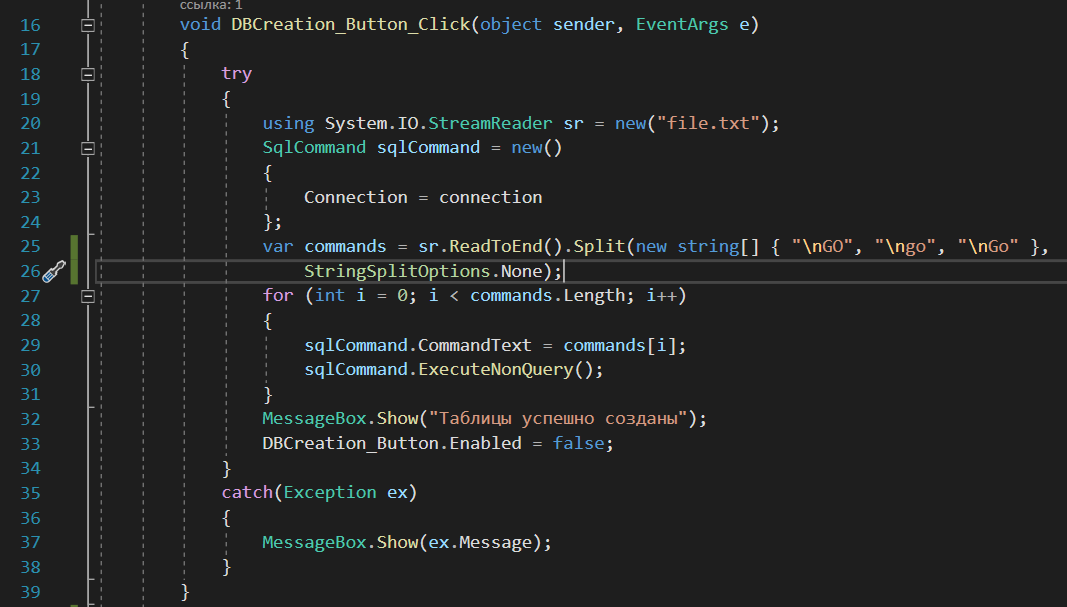


Используя SqlCommand отправляем запрос в БД:



### Создание таблиц в пустой БД

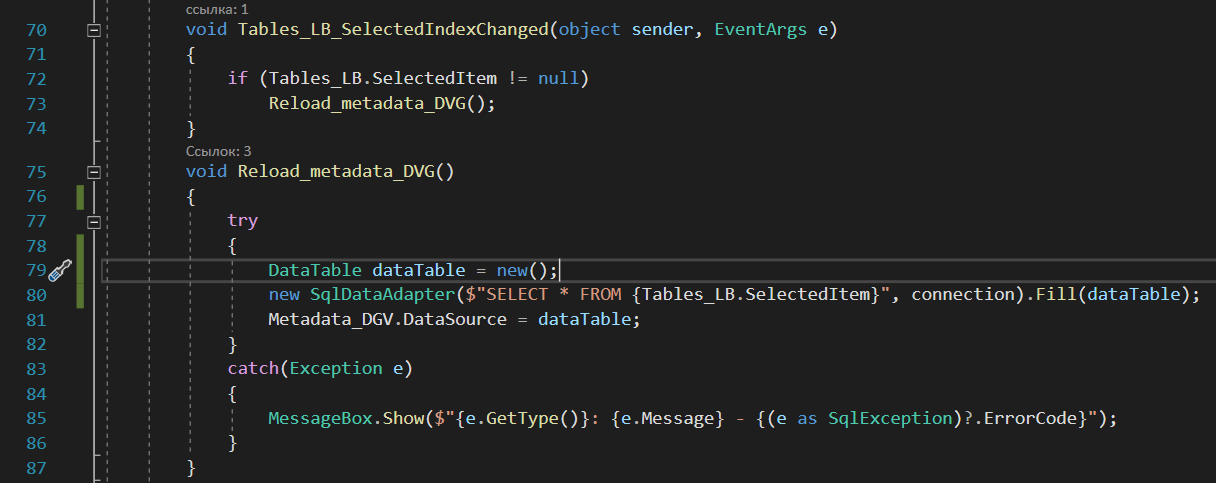
Для этого я заполнил текстовый файл полным текстом запроса и, считывая их в помощью StreamReader’а поочерёдно отправлял их в БД.



(Текст создания БД можно найти [тут](#_Sql-запрос_для_создания))

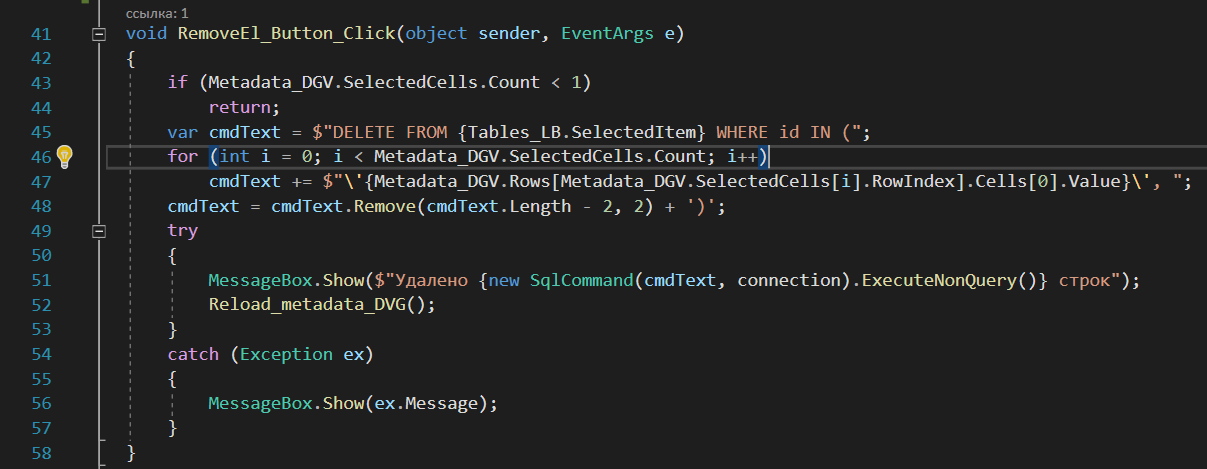
### Отображение данных из выбранной таблицы

Для этого используем SqlAdapter из вышеупомянутой библиотеки.



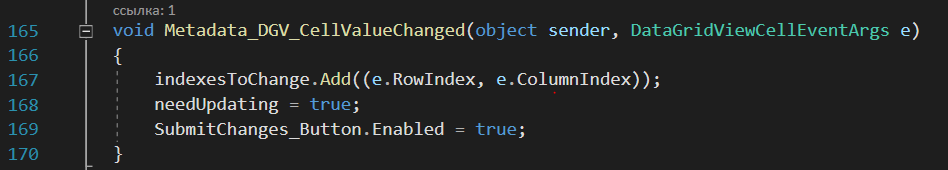
### Удаление данных из БД

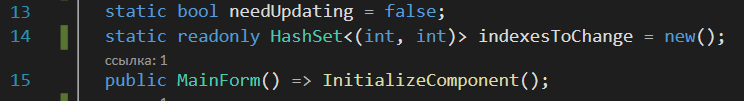
Аналогичным способом производим и удаление



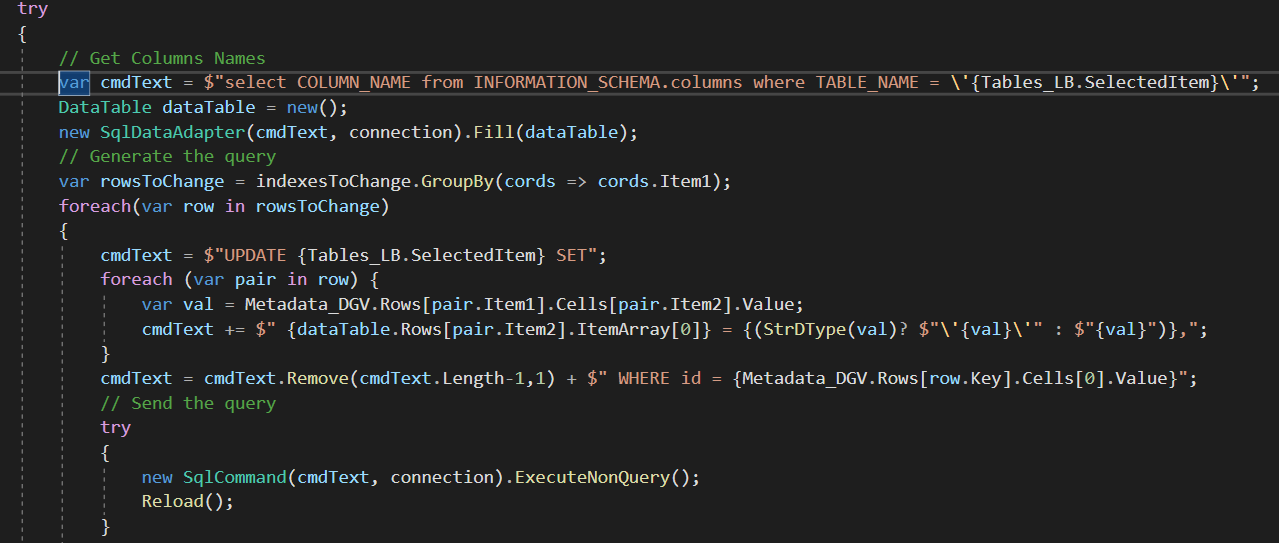
### Изменение данных после их редактирования в DataGridView

Сперва запоминаем индексы ячеек, в которых необходимо произвести изменения

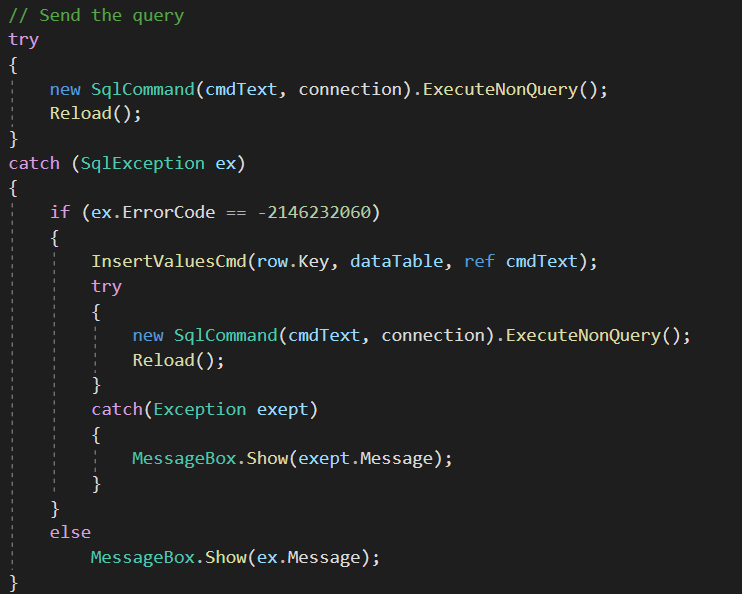


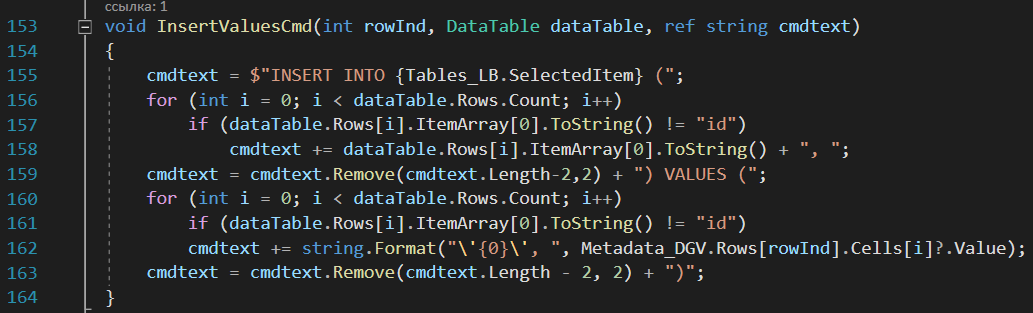


Далее пытаемся обновить данные в БД через команду UPDATE:

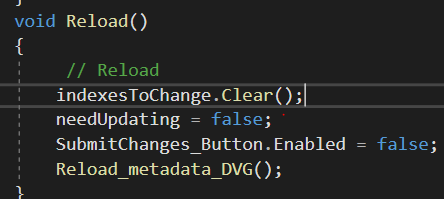


Если данных с id, соттветствующий строке с изменениями нет (определяем по константному литерал-коду значения ошибки), значит нужно добавить такую запись:





При изменении данных, соответственно, производим обновление DataGridView по данным из БД:

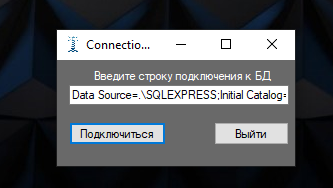


### Завершение работы

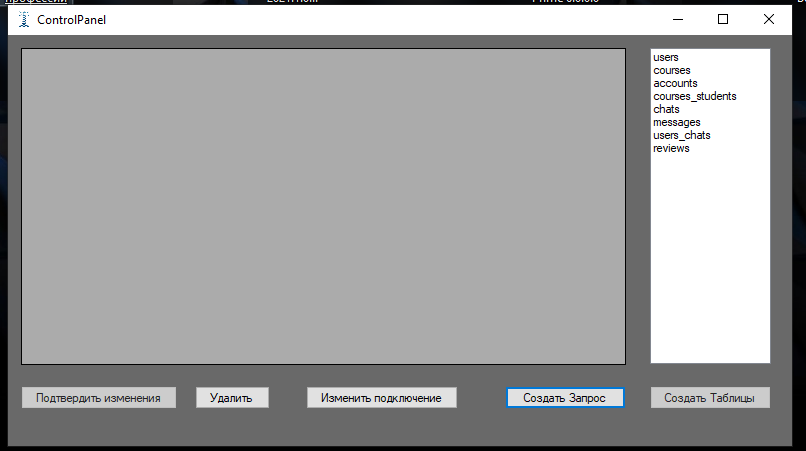
Производим сборку проекта под Release-конфигурацию, очищаем отладочные файлы и используем получившийся exe-файл по назначению

## Демонтрация работы

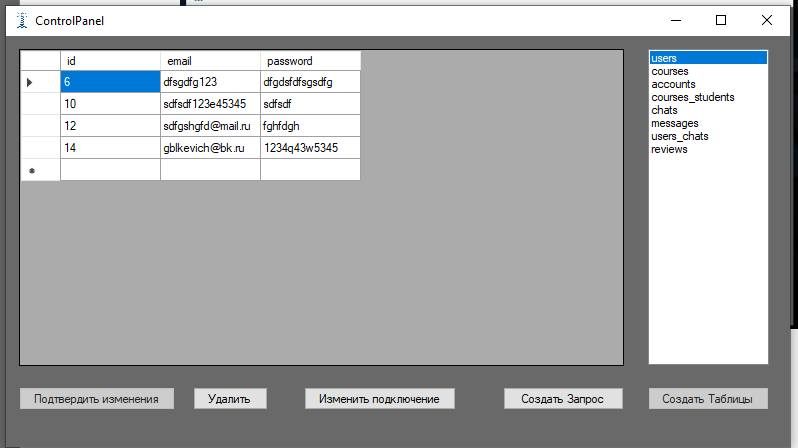
### Запуск программы



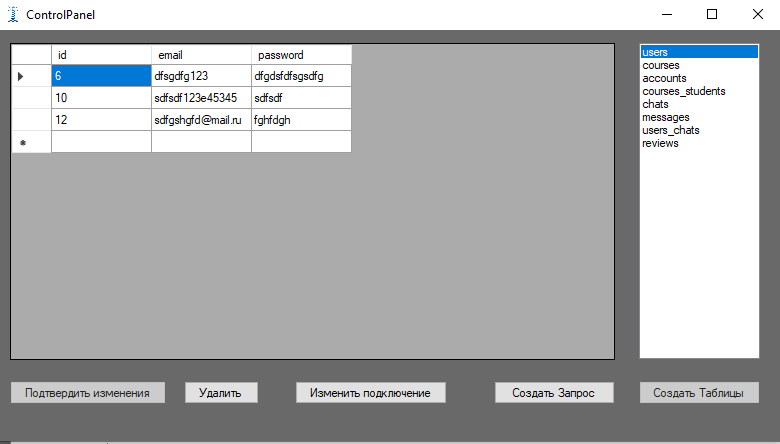
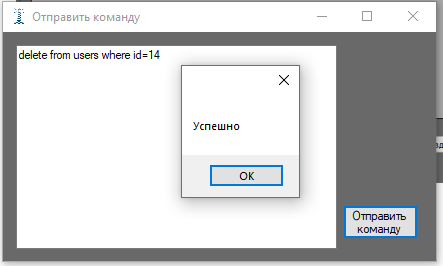
### Главное окно



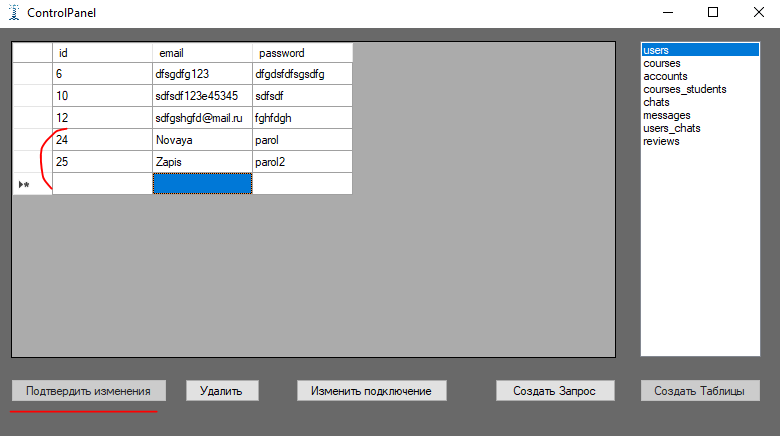
### Выбор таблицы в списке



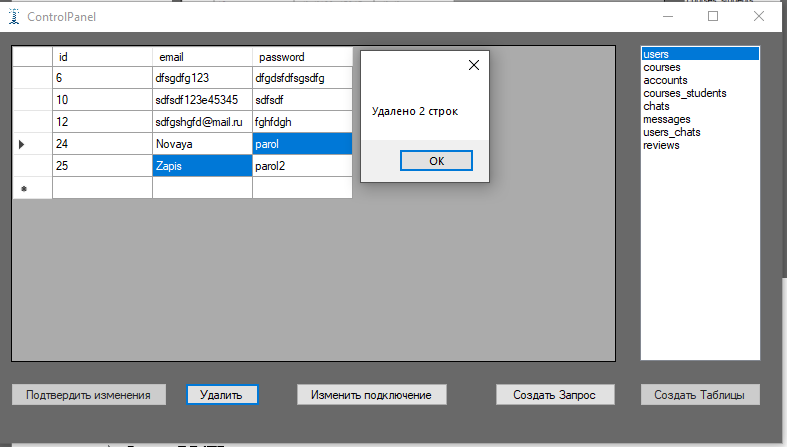
### Отправка запроса

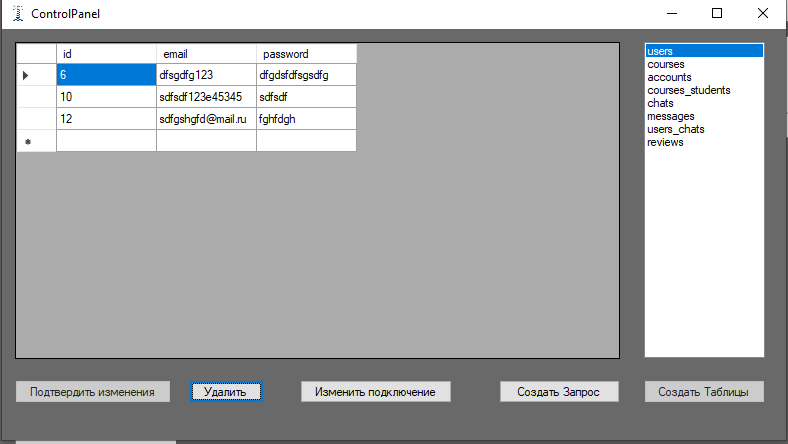


### Добавление новых записей

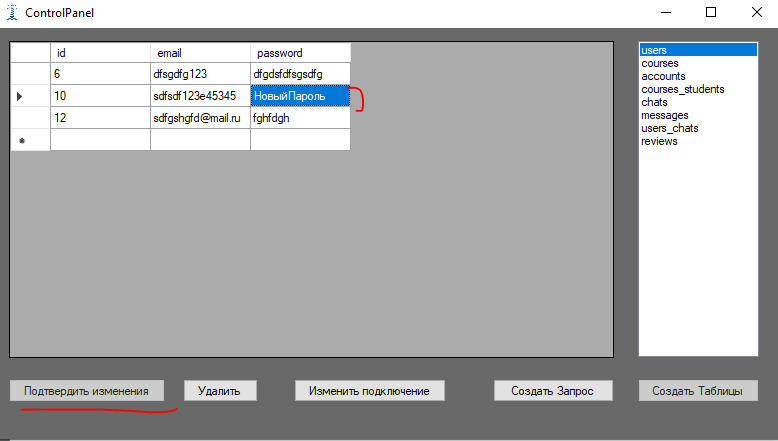


### Удаление имеющихся записей





### Изменение данных через DataGridView



# Заключение

Таким образом, все поставленые цели и задачи были достигнуты:

Были изучить принципы построения панели управления баз данных, освоены взаимодействие баз данных с ранее освоенными контейнерными классами и способы применения данных из баз в пользовательских приложениях, углублены знания работы с бд и языками программирования.

Приложение полностью соответствует ожидаемому функционалу, однако может быть дополнена в дальнейшем.

К примеру, в приложение планируется в дальнейшем внедрить следующее:

1. Блокировать возможность создавать записи с пустыми значениями в NOT-NULL-полях.
2. Блокировать доступ к редактированию автоикрементирущихся полей.
3. Дополнить GUI панелями инструментов, меню, состояния.
4. Вынести все элементы управления в отдельные блоки, к примеру, по glid-layout-канону.
5. Добавить элементы управления для выборки записей по критериям

# Список использованных источников

## Книги

1. Албахари, Джозеф, Албахари, Бен. «C# 8.0. Карманный справочник» Диалектика, 2020 год, 240 стр., ISBN 978-5-907203-14-3.
2. C# 7.0. Справочник. Полное описание языка, 7-е изд. Автор: Джозеф Албахари, Бен Албахари Издательство: Вильямс Год: 2018 Страниц: 1026

## Интернет-ресурсы

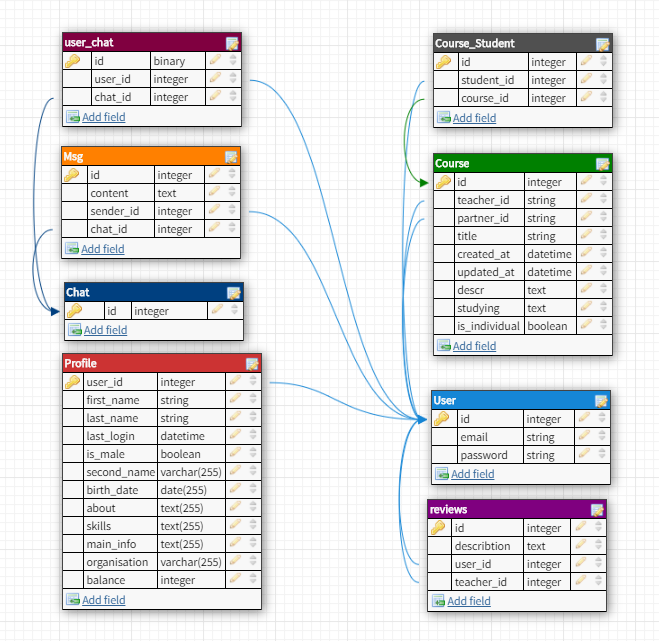
1. <https://docs.microsoft.com/en-us/dotnet/csharp/>
2. <https://metanit.com/sharp/>
3. <https://ru.wikipedia.org/wiki/C_Sharp>
4. <https://postgrespro.ru/docs/postgresql/>

## Интернет-ресурсы (образовательные YouTube-каналы от .NET Senior-разработчиков)

1. <https://www.youtube.com/channel/UC2CiWFIOjQix4E6WrARzDZg>
2. <https://www.youtube.com/channel/UCG7GW-X1cczyzLswoYTTnjQ>
3. <https://www.youtube.com/channel/UCtLKO1Cb2GVNrbU7Fi0pM0w>
4. <https://www.youtube.com/user/OfficialChannelGD>

# Приложения

## Схема БД ITBeacon



## MainForm.cs

using System;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

using System.Windows.Forms;

using System.Collections.Generic;

using System.Linq;

namespace ITBeacon

{

public partial class MainForm : Form

{

static bool needUpdating = false;

static readonly HashSet<(int, int)> indexesToChange = new();

public SqlConnection connection;

public MainForm() => InitializeComponent();

void DBCreation\_Button\_Click(object sender, EventArgs e)

{

try

{

using System.IO.StreamReader sr = new("file.txt");

SqlCommand sqlCommand = new()

{

Connection = connection

};

var commands = sr.ReadToEnd().Split(new string[] { "\nGO", "\ngo", "\nGo" }, StringSplitOptions.None);

for (int i = 0; i < commands.Length; i++)

{

sqlCommand.CommandText = commands[i];

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Таблицы успешно созданы");

DBCreation\_Button.Enabled = false;

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

void SendRequest\_Button\_Click(object sender, EventArgs e)

=> new SendCmdForm(this).Show();

void RemoveEl\_Button\_Click(object sender, EventArgs e)

{

if (Metadata\_DGV.SelectedCells.Count < 1)

return;

var cmdText = $"DELETE FROM {Tables\_LB.SelectedItem} WHERE id IN (";

for (int i = 0; i < Metadata\_DGV.SelectedCells.Count; i++)

cmdText += $"\'{Metadata\_DGV.Rows[Metadata\_DGV.SelectedCells[i].RowIndex].Cells[0].Value}\', ";

cmdText = cmdText.Remove(cmdText.Length - 2, 2) + ')';

try

{

MessageBox.Show($"Удалено {new SqlCommand(cmdText, connection).ExecuteNonQuery()} строк");

Reload\_metadata\_DVG();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

public void FillTableList()

{

Tables\_LB.Items.Clear();

DataTable dataTable = new();

new SqlDataAdapter("SELECT \* FROM sys.objects WHERE type in (N'U')", connection).Fill(dataTable);

var ind = dataTable.Columns.IndexOf("name");

foreach (var row in dataTable.Rows)

Tables\_LB.Items.Add((row as DataRow)?.ItemArray[ind].ToString());

if (Tables\_LB.Items.Count == 0)

DBCreation\_Button.Enabled = true;

}

void Tables\_LB\_SelectedIndexChanged(object sender, EventArgs e)

{

if (Tables\_LB.SelectedItem != null)

Reload\_metadata\_DVG();

}

void Reload\_metadata\_DVG()

{

DataTable dataTable = new();

try

{

new SqlDataAdapter($"SELECT \* FROM {Tables\_LB.SelectedItem}", connection)

.Fill(dataTable);

Metadata\_DGV.DataSource = dataTable;

}

catch(Exception e)

{

MessageBox.Show($"{e.GetType()}: {e.Message} - {(e as SqlException)?.ErrorCode}");

}

}

void SubmitChanges\_Button\_Click(object sender, EventArgs e)

{

static bool StrDType(object val) => !typeof(int).Equals(val.GetType());

if (!needUpdating)

return;

try

{

// Get Columns Names

var cmdText = $"select COLUMN\_NAME from INFORMATION\_SCHEMA.columns where TABLE\_NAME = \'{Tables\_LB.SelectedItem}\'";

DataTable dataTable = new();

new SqlDataAdapter(cmdText, connection).Fill(dataTable);

// Generate the query

var rowsToChange = indexesToChange.GroupBy(cords => cords.Item1);

foreach(var row in rowsToChange)

{

cmdText = $"UPDATE {Tables\_LB.SelectedItem} SET";

foreach (var pair in row) {

var val = Metadata\_DGV.Rows[pair.Item1].Cells[pair.Item2].Value;

cmdText += $" {dataTable.Rows[pair.Item2].ItemArray[0]} = {(StrDType(val)? $"\'{val}\'" : $"{val}")},";

}

cmdText = cmdText.Remove(cmdText.Length-1,1) + $" WHERE id = {Metadata\_DGV.Rows[row.Key].Cells[0].Value}";

// Send the query

try

{

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

Reload();

}

catch (SqlException ex)

{

if (ex.ErrorCode == -2146232060)

{

InsertValuesCmd(row.Key, dataTable, ref cmdText);

try

{

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

Reload();

}

catch(Exception exept)

{

MessageBox.Show(exept.Message);

}

}

else

MessageBox.Show(ex.Message);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

void Reload()

{

// Reload

indexesToChange.Clear();

needUpdating = false;

SubmitChanges\_Button.Enabled = false;

Reload\_metadata\_DVG();

}

}

void InsertValuesCmd(int rowInd, DataTable dataTable, ref string cmdtext)

{

cmdtext = $"INSERT INTO {Tables\_LB.SelectedItem} (";

for (int i = 0; i < dataTable.Rows.Count; i++)

if (dataTable.Rows[i].ItemArray[0].ToString() != "id")

cmdtext += dataTable.Rows[i].ItemArray[0].ToString() + ", ";

cmdtext = cmdtext.Remove(cmdtext.Length-2,2) + ") VALUES (";

for (int i = 0; i < dataTable.Rows.Count; i++)

if (dataTable.Rows[i].ItemArray[0].ToString() != "id")

cmdtext += string.Format("\'{0}\', ", Metadata\_DGV.Rows[rowInd].Cells[i]?.Value);

cmdtext = cmdtext.Remove(cmdtext.Length - 2, 2) + ")";

}

void Metadata\_DGV\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

indexesToChange.Add((e.RowIndex, e.ColumnIndex));

needUpdating = true;

SubmitChanges\_Button.Enabled = true;

}

void ChangeConnection\_Button\_Click(object sender, EventArgs e)

{

new ConnectionForm(this).Show();

Hide();

}

void MainPage\_Close(object sender, FormClosedEventArgs e)

{

connection?.Close();

Application.Exit();

}

}

}

## SendRequestForm.cs

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace ITBeacon

{

public partial class SendCmdForm : Form

{

public MainForm PrevForm { get; set; }

public SendCmdForm(MainForm mainForm)

{

InitializeComponent();

PrevForm = mainForm;

}

void SendCmd\_Button\_Click(object sender, EventArgs e)

{

try

{

SqlCommand sqlCommand = new(Request\_TB.Text) { Connection = PrevForm?.connection };

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Успешно");

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

void SendCmdForm\_Load(object sender, EventArgs e)

{

BackColor = PrevForm.BackColor;

}

void SendCmdForm\_FormClosed(object sender, FormClosedEventArgs e)

=> PrevForm.FillTableList();

}

}

## ConnectionForm.cs

using System;

using System.Windows.Forms;

namespace ITBeacon

{

public partial class ConnectionForm : Form

{

public MainForm PrevForm { get; set; }

public ConnectionForm(MainForm mainForm)

{

InitializeComponent();

PrevForm = mainForm;

}

void Connect\_Button\_Click(object sender, EventArgs e)

{

try

{

//connection = new(ConfigurationManager.ConnectionStrings["ITBeaconDB"].ConnectionString);

PrevForm.connection = new(ConnectionString\_TB.Text);

PrevForm.connection?.Open();

PrevForm.FillTableList();

PrevForm.Show();

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

void Exit\_Button\_Click(object sender, EventArgs e)

=> Close();

}

}

## Sql-запрос для создания таблиц в пустой БД

CREATE TABLE [users] (

id integer IDENTITY(1,1) NOT NULL, PRIMARY KEY

(

[id] ASC

) WITH (IGNORE\_DUP\_KEY = OFF),

email varchar(64) NOT NULL UNIQUE,

password varchar(20) NOT NULL

)

GO

CREATE TABLE [courses] (

id integer IDENTITY(1,1) NOT NULL PRIMARY KEY,

teacher\_id integer NOT NULL,

partner\_id integer,

title varchar(20) NOT NULL,

created\_at datetime NOT NULL,

updated\_at datetime NOT NULL,

descr text NOT NULL,

studying text NOT NULL,

is\_individual bit NOT NULL

)

GO

CREATE TABLE [accounts] (

account\_id integer NOT NULL PRIMARY KEY,

first\_name varchar(20) NOT NULL,

last\_name varchar(20) NOT NULL,

last\_login datetime NOT NULL,

is\_male bit NOT NULL,

second\_name varchar(20),

birth\_date date,

about text,

skills text,

main\_info text,

organisation varchar(20),

balance money NOT NULL DEFAULT '0'

)

GO

CREATE TABLE [courses\_students] (

id integer IDENTITY(1,1) NOT NULL PRIMARY KEY

(

[id] ASC

) WITH (IGNORE\_DUP\_KEY = OFF),

student\_id integer NOT NULL,

course\_id integer NOT NULL

)

GO

CREATE TABLE [chats] (

id integer IDENTITY(1,1) NOT NULL PRIMARY KEY

(

[id] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [messages] (

id integer IDENTITY(1,1) NOT NULL PRIMARY KEY

(

[id] ASC

) WITH (IGNORE\_DUP\_KEY = OFF),

content text NOT NULL,

sender\_id integer NOT NULL,

chat\_id integer NOT NULL

)

GO

CREATE TABLE [users\_chats] (

id integer IDENTITY(1,1) NOT NULL PRIMARY KEY

(

[id] ASC

) WITH (IGNORE\_DUP\_KEY = OFF),

account\_id integer NOT NULL,

chat\_id integer NOT NULL

)

GO

CREATE TABLE [reviews] (

id integer IDENTITY(1,1) NOT NULL PRIMARY KEY

(

[id] ASC

) WITH (IGNORE\_DUP\_KEY = OFF),

describtion text,

account\_id integer NOT NULL,

teacher\_id integer NOT NULL

)

GO

ALTER TABLE courses ADD CONSTRAINT [FK\_teachers\_users] FOREIGN KEY ([teacher\_id]) REFERENCES [users]([id])

GO

ALTER TABLE courses ADD CONSTRAINT [FK\_partner\_users] FOREIGN KEY ([partner\_id]) REFERENCES [users]([id])

GO

ALTER TABLE accounts ADD CONSTRAINT [FK\_acc\_users] FOREIGN KEY ([account\_id]) REFERENCES [users]([id])

GO

ALTER TABLE courses\_students ADD CONSTRAINT [FK\_course\_stud\_cid] FOREIGN KEY ([course\_id]) REFERENCES [courses]([id])

GO

ALTER TABLE courses\_students ADD CONSTRAINT [FK\_course\_stud\_sid] FOREIGN KEY ([student\_id]) REFERENCES [users]([id])

GO

ALTER TABLE messages ADD CONSTRAINT [FK\_sender\_user] FOREIGN KEY ([sender\_id]) REFERENCES [users]([id])

GO

ALTER TABLE messages ADD CONSTRAINT [FK\_msg\_chat] FOREIGN KEY ([chat\_id]) REFERENCES [chats]([id])

GO

ALTER TABLE users\_chats ADD CONSTRAINT [FK\_user\_chat\_uid] FOREIGN KEY ([account\_id]) REFERENCES [users]([id])

GO

ALTER TABLE users\_chats ADD CONSTRAINT [FK\_user\_chat\_cid] FOREIGN KEY ([chat\_id]) REFERENCES [chats]([id])

GO

ALTER TABLE reviews ADD CONSTRAINT [FK\_review\_user] FOREIGN KEY ([account\_id]) REFERENCES [users]([id])

GO

ALTER TABLE reviews ADD CONSTRAINT [FK\_review\_teacher] FOREIGN KEY ([teacher\_id]) REFERENCES [users]([id])