МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ**

**БАЗАМИ ДАННЫХ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №2

Выполнил Учащаяся группы ПО-455 О.Е.Листопадова

Шифр 21

2023

# 26 Опишите этапы создания приложения

Если описать процесс разработки кратко, получится такой список:

1. Сбор требований к приложению.
2. Предварительная оценка стоимости разработки.
3. Проектирование прототипа.
4. Составление технического задания.
5. Разработка первого релиза продукта.
6. Тестирование.
7. Публикация.
8. Техническая поддержка.

Остановимся подробнее на каждом этапе.

**Сбор требований к приложению.**

Клиент звоните или оставляете заявку на сайте, или любым удобным образом. Менеджер по продажам в формате телефонного интервью собирает информацию: какое приложение необходимо сделать, для чего оно предназначено, кто будет им пользоваться, какие функции в нем будут, есть ли сайт, нужен ли сервер, будут ли интеграции с программными продуктами, например, с 1С. После сбора информации менеджер отдает заполненный бриф на оценку в отдел разработки.

Предварительная оценка стоимости разработки.

Отдел разработки оценивает, сколько времени займет создание приложения, какие специалисты будут задействованы.

Предварительная стоимость мобильного приложения может отличаться от итоговой стоимости, как в большую, так и в меньшую сторону. Точную сумму затрат на разработку приложения можно подсчитать только после составления технического задания.

**Проектирование прототипа.**

Чтобы показать, как будет работать приложение, нужен прототип - макет будущего приложения. Прототип может быть статичный или интерактивный - с кнопками действий и переходами. За дело берется аналитик. Он продумывает логику работы приложения и путь пользователя в нем - user story.  На этом этапе отдел маркетинга может предложить скорректировать идею, ориентируясь на целевую аудиторию и задачи.

Когда все готово, аналитик согласовывает с прототипы, если нужно вносит правки и передает дизайнеру. Дизайнер определяет стиль оформления приложения и основное направление, используя концепцию Material design guidelines и iOS Human Interface Guidelines с рекомендациями размеров, отступов, визуальных эффектов, анимации. Программисты также ориентируются на руководства для разработки мобильных приложений от Apple и Google.

**Составление технического задания.**

Техническое задание составляет аналитик на основе утвержденных прототипов. В техническом задании подробно описываются: функционал мобильного приложения, переходы, бизнес-процессы, основные сценарии, способы получения и передачи данных. Когда ТЗ готово, можно точно оценить стоимость проекта.

**Разработка первого релиза продукта.**

На основе ТЗ программисты выпускают первый релиз приложения. Для сложных проектов мы советуем создавать MVP - минимальную жизнеспособную версию продукта, чтобы проверить реакцию пользователей на приложение. Если необходимо, разрабатывается сервер для управления мобильным приложением и API - сервис для передачи данных между сервером и приложением.

**Тестирование.**

Приложения проверяются на ошибки с помощью Smoke Test, ручного функционального тестирования, автоматического функционального тестирования и программы Monkey для Android. Если есть ошибки, они исправляются и передается готовое приложение. Для статистики скачиваний и анализа поведения пользователей в приложении можно устанавливить Firebase Analytics.

**Публикация.**

Публикуется приложение в Google Play Market и App Store, если это мобильное приложение, либо передается исполняемый фал и при необходимости прочие файлы.

**Техническая поддержка.**

Основные задачи технической поддержки мобильного приложения:

* получение обратной связи от пользователей, решение возникающих у них затруднений;
* проверка работоспособности приложения;
* обновление для новых устройств и версий операционных систем.

# 53 Опишите функции РНР для работы с СУБД MySQL

Типичный процесс работы с СУБД в PHP-сценарии состоит из нескольких шагов:

1. Установить подключение к серверу СУБД, передав необходимые параметры: адрес, логин, пароль.
2. Убедиться, что подключение прошло успешно: сервер СУБД доступен, логин и пароль верные и так далее.
3. Сформировать правильный SQL запрос (например, на чтение данных из таблицы).
4. Убедиться, что запрос был выполнен успешно.
5. Получить результат от СУБД в виде массива из записей.
6. Использовать полученные записи в своём сценарии (например, показать их в виде таблицы).

## **Функция mysqli connect: соединение с MySQL**

Перед началом работы с данными внутри MySQL, нужно открыть соединение с сервером СУБД. В PHP это делается с помощью стандартной функции mysqli\_connect(). Функция возвращает результат — ресурс соединения. Данный ресурс используется для всех следующих операций с MySQL.

Но чтобы выполнить соединение с сервером, необходимо знать как минимум три параметра:

* Адрес сервера СУБД;
* Логин;
* Пароль.

Базовый синтаксис функции mysqli\_connect():

mysqli\_connect(<адрес сервера>, <имя пользователя>, <пароль>, <имя базы данных>);

## **Проверка соединения**

Первое, что нужно сделать после соединения с СУБД — это выполнить проверку, что оно было успешным. Эта проверка нужна, чтобы исключить ошибку при подключении к БД. Неверные параметры подключения, неправильная настройка или высокая нагрузка заставит MySQL отвергать новые подключения. Все эти ситуации приведут к невозможности соединения, поэтому программист должен проверить успешность подключения к серверу, прежде чем выполнять следующие действия.

Соединение с MySQL устанавливается один раз в сценарии, а затем используется при всех запросах к БД.

Результатом выполнения функции mysqli\_connect() будет значение специального типа — ресурс. Если подключение к MySQL не удалось, то функция mysqli\_connect() вместо ресурса вернёт логическое значение типа «ложь» — false. Хорошей практикой будет всегда проверять результат выполнения этой функции и сравнивать его с ложью.

Соединение с MySQL и проверка на ошибки:

<?php

$link = mysqli\_connect("localhost", "root", "");

if ($link == false){

print("Ошибка: Невозможно подключиться к MySQL " . mysqli\_connect\_error());

}

else {

print("Соединение установлено успешно");

}

?>

Функция mysqli\_connect\_error() просто возвращает текстовое описание последней ошибки MySQL.

## **Установка кодировки**

Первым делом после установки соединения крайне желательно явно задать кодировку, которая будет использоваться при обмене данными с MySQL. Если этого не сделать, то вместо записей со значениями, написанными кириллицей, можно получить последовательность из знаков вопроса: ?????????????????. Вызовите эту функцию сразу после успешной установки соединения: mysqli\_set\_charset($con, "utf8");

## Выполнение запросов

Следует разделять все SQL-запросы на две группы:

1. Чтение информации (SELECT).
2. Модификация (UPDATE, INSERT, DELETE).

При выполнении запросов из среды PHP, запросы из второй группы возвращают только результат их исполнения: успех или ошибку.

Запросы первой группы при успешном выполнении возвращают специальный ресурс результата. Его, в свою очередь, можно преобразовать в ассоциативный массив (если нужна одна запись) или в двумерный массив (если требуется список записей).

## **Добавление записи**

Выражение INSERT INTO используется для добавления новых записей в таблицу базы данных.

Составим корректный SQL-запрос на вставку записи с именем города, а затем выполним его путём передачи этого запроса в функцию mysqli\_query(), чтобы добавить новые данные в таблицу.

<?php

$link = mysqli\_connect("localhost", "root", "");

$sql = 'INSERT INTO cities SET name = "Санкт-Петербург"';

$result = mysqli\_query($link, $sql);

if ($result == false) {

print("Произошла ошибка при выполнении запроса");

}

Обратите внимание, что первым параметром для функции mysqli\_query() передаётся ресурс подключения, полученный от функции mysqli\_connect(), вторым параметром следует строка с SQL-запросом.

При запросах на изменение данных (не SELECT) результатом выполнения будет логическое значение — true или false, которое будет означать, что запрос выполнить не удалось. Для получения строки с описанием ошибки существует функция mysqli\_error($link).

## **Чтение записей**

Другая частая операция при работе с базами данных в PHP — это получение записей из таблиц (запросы типа SELECT). Составим SQL-запрос, который будет использовать SELECT выражение. Затем выполним этот запрос с помощью функции mysqli\_query(), чтобы получить данные из таблицы.

В этом примере показано, как вывести все существующие города из таблицы cities:

<?php

$sql = 'SELECT id, name FROM cities';

$result = mysqli\_query($link, $sql);

while ($row = mysqli\_fetch\_array($result)) {

print("Город: " . $row['name'] . "; Идентификатор: . " . $row['id'] . "<br>");

}

В примере выше результат выполнения функции mysqli\_query() сохранён в переменной $result. В этой переменной находятся не данные из таблицы, а специальный тип данных — так называемая ссылка на результаты запроса.

Чтобы получить действительные данные, то есть записи из таблицы, следует использовать другую функцию — mysqli\_fetch\_array() — и передать ей единственным параметром эту самую ссылку. Теперь каждый вызов функции mysqli\_fetch\_array() будет возвращать следующую запись из всего результирующего набора записей в виде ассоциативного массива.

Цикл while здесь используется для «прохода» по всем записям из полученного набора записей. Значение поля каждой записи можно узнать, просто обратившись по ключу этого ассоциативного массива.

## **Как получить сразу все записи в виде двумерного массива**

Иногда бывает удобно после запроса на чтение не вызывать в цикле mysqli\_fetch\_array для извлечения очередной записи по порядку, а получить их сразу все одним вызовом. PHP так тоже умеет.

Функция mysqli\_fetch\_all($res, MYSQLI\_ASSOC) вернёт двумерный массив со всеми записями из результата последнего запроса. Перепишем пример с показом существующих городов с её использованием:

<?php

$sql = 'SELECT id, name FROM cities';

$result = mysqli\_query($link, $sql);

$rows = mysqli\_fetch\_all($result, MYSQLI\_ASSOC)

foreach ($rows as $row) {

print("Город: " . $row['name'] . "; Идентификатор: . " . $row['id'] . "<br>");

}

## **Как узнать количество записей**

Часто бывает необходимо узнать, сколько всего записей вернёт выполненный SQL-запрос. Это может помочь при организации постраничной навигации или просто в качестве информации. Узнать число записей поможет функция mysqli\_num\_rows(), которой следует передать ссылку на результат запроса.

**88 «Учебное заведение»**

1) Для создания информационной системы спроектируйте базу данных в Microsoft SQL Server, содержащую не менее трех таблиц. Установите отношения между таблицами.

2) Заполните таблицы данными (не менее 10 записей).

3) На языке SQL напишите запросы:

- на вывод некоторых полей из двух таблиц;

- на вывод данных по условию, представляющему выражение:

* 1. типа сравнения;
  2. с логическим оператором «И»;
  3. с логическим оператором «ИЛИ»;

- с вычислениями над полями БД;

- параметрический.

4) Разработать форму на языке C#, содержащую все созданные объекты БД.

5) Реализовать добавление, удаление изменение данных в таблице, а также поиск и фильтрацию данных.

**Решение**

Создадим форму и разместим на ней соответствующие элементы управления. Внешний вид формы представлен на рисунке 1.

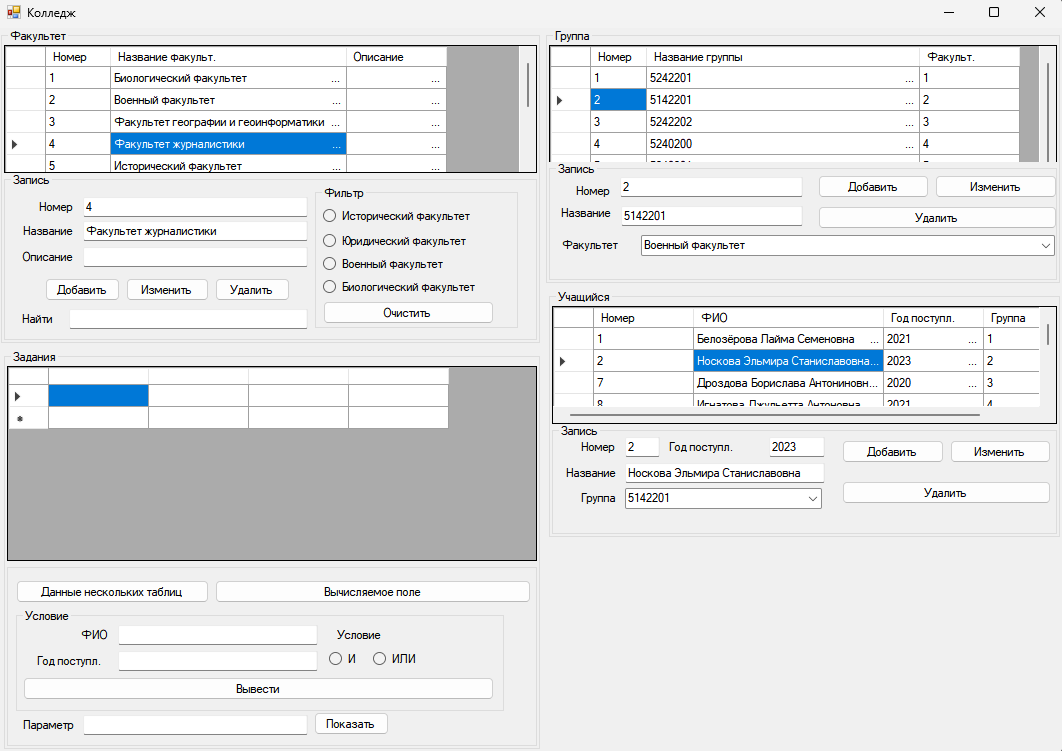


Рисунок 1 Внешний вид главной формы

Заполним таблицы данными, это таблицы: Факультет, Учащийся и Группа. Настроим соединение с базой данных и основные действия, выполняемые по нажатию кнопок добавления, изменения и удаления полей таблиц.

На рисунке 2 представлен внешний вид приложения, в котором выполнено задание вывода сводных данных из таблиц.

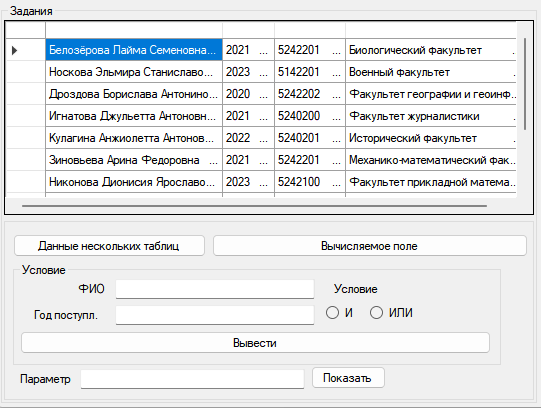


Рисунок 2 Приложение с выполненным заданием

На рисунке 3 представлен результат выполнения действий над полями таблицы Учащийся, выведено ФИ, год поступления, год поступления минус 1000 и поясняющая надпись.

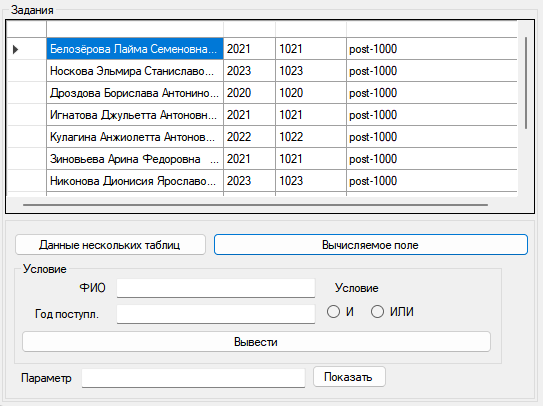


Рисунок 3 Результат выполнения задания вычисляемого поля

На рисунке 4 представлен вид приложения с выполненным заданием выбора учащегося по ФИО, либо по году его поступления.

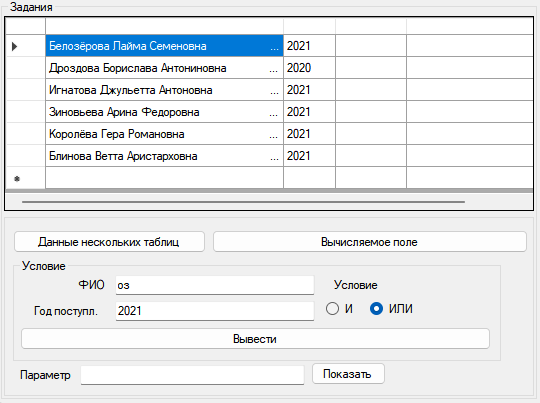


Рисунок 4 Результат выполнения задания Условие

На рисунке 5 представлен вывод задания с параметром, в качестве параметра часть ФИО учащегося.

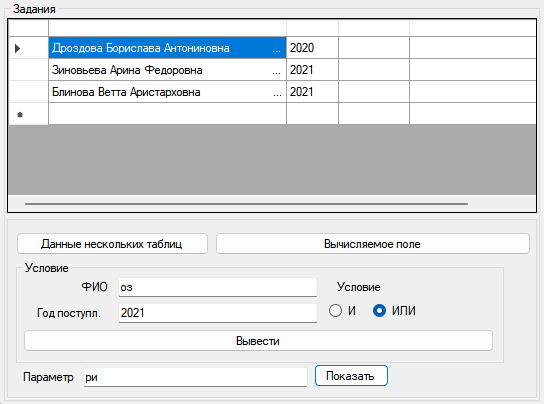
****

Рисунок 5 Результат выполнения задания Параметр

На рисунке 6 представлен результат выполнения задания Фильтр по факультету.

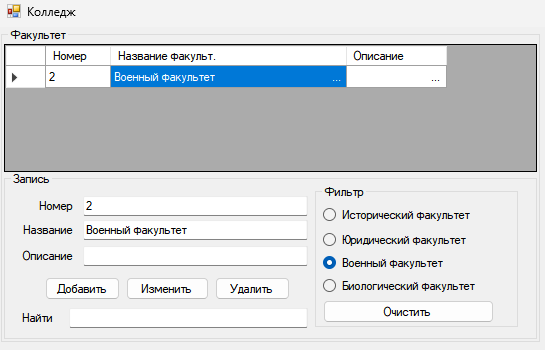
****

Рисунок 6результат выполнения задания Фильтр

На рисунке 7 представлен результат выполнения задания Поиск

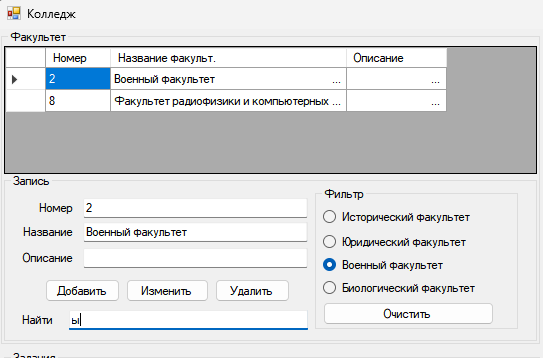


Рисунок 7результат выполнения задания Поиск

**Код программы**

**Главная Форма**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

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

namespace kolledg

{

public partial class MainForm : Form

{

DataBase database = new DataBase();

int selectedRow;

public MainForm()

{

InitializeComponent();

CreateColumns();

refreshDG1(dataGridView1);

refreshDG2(dataGridView2);

refreshDG3(dataGridView3);

refreshDG4(dataGridView4,$"select TOP (1) ' ' as Col1, ' ' as col2, ' ' as col3, ' ' as col4 from [fak]");

}

private void CreateColumns()

{

dataGridView1.Columns.Add("ID", "Номер");

dataGridView1.Columns.Add("Name", "Название факульт.");

dataGridView1.Columns.Add("Descr", "Описание");

dataGridView2.Columns.Add("ID", "Номер");

dataGridView2.Columns.Add("Name", "ФИО");

dataGridView2.Columns.Add("post", "Год поступл.");

dataGridView2.Columns.Add("IDgrp", "Группа");

dataGridView3.Columns.Add("ID", "Номер");

dataGridView3.Columns.Add("Name", "Название группы");

dataGridView3.Columns.Add("IDfak", "Факульт.");

dataGridView4.Columns.Add("C1", String.Empty);

dataGridView4.Columns.Add("C2", String.Empty);

dataGridView4.Columns.Add("C3", String.Empty);

dataGridView4.Columns.Add("C4", String.Empty);

comboBox1.Items.Clear();

string querystr = $" select name from [grp] order by id";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRowc1(reader);

}

reader.Close();

comboBox2.Items.Clear();

querystr = $" select name from [fak] order by id";

com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRowc2(reader);

}

reader.Close();

}

private void ReadSinglRowc1(IDataRecord record)

{

comboBox1.Items.Add(record.GetString(0));

}

private void ReadSinglRowc2(IDataRecord record)

{

comboBox2.Items.Add(record.GetString(0));

}

private void ReadSinglRow(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2));

}

private void ReadSinglRow2(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetString(2), record.GetInt32(3));

}

private void ReadSinglRow3(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetInt32(0), record.GetString(1), record.GetInt32(2));

}

private void ReadSinglRow4(DataGridView dgw, IDataRecord record)

{

dgw.Rows.Add(record.GetString(0), record.GetString(1), record.GetString(2), record.GetString(3));

}

private void refreshDG1(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [fak]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dgw, reader);

}

reader.Close();

}

private void refreshDG2(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [uch]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow2(dgw, reader);

}

reader.Close();

}

private void refreshDG3(DataGridView dgw)

{

dgw.Rows.Clear();

string querystr = $"Select \* from [grp]";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow3(dgw, reader);

}

reader.Close();

}

private void refreshDG4(DataGridView dgw, string s)

{

dgw.Rows.Clear();

string querystr = s;

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dgw, reader);

}

reader.Close();

}

private void button3\_Click(object sender, EventArgs e)

{

database.openConnection();

var addQuery = $"insert into [fak] (Name,Descr) values ('{textBox1.Text}' , '{textBox2.Text}')";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

database.closeConnection();

this.Close();

refreshDG1(dataGridView1);

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [fak] Where concat(Name,Descr) like '%" + textBox4.Text + "%'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton1\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [fak] Where Name = 'Исторический факультет'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void dataGridView1\_CellClick\_1(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox1.Text = row.Cells[0].Value.ToString();

textBox2.Text = row.Cells[1].Value.ToString();

textBox3.Text = row.Cells[2].Value.ToString();

}

}

private void radioButton2\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [fak] Where Name = 'Юридический факультет'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton4\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [fak] Where Name = 'Биологический факультет'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void radioButton3\_Click(object sender, EventArgs e)

{

DataGridView dwg = dataGridView1;

dwg.Rows.Clear();

string querystr = $" select \* from [fak] Where Name = 'Военный факультет'";

SqlCommand com = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

ReadSinglRow(dwg, reader);

}

reader.Close();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

var addQuery = $"delete from [fak] where ID={textBox1.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG1(dataGridView1);

}

private void button1\_Click\_1(object sender, EventArgs e)

{

var addQuery = $"update [fak] set Name='{textBox2.Text}' , Descr = '{textBox3.Text}' where ID = {textBox1.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG1(dataGridView1);

}

private void Form1\_Load\_1(object sender, EventArgs e)

{

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView2.Rows[selectedRow];

textBox5.Text = row.Cells[0].Value.ToString();

textBox6.Text = row.Cells[1].Value.ToString();

textBox7.Text = row.Cells[2].Value.ToString();

comboBox1.SelectedIndex = Int32.Parse(row.Cells[3].Value.ToString())-1;

}

}

private void button5\_Click(object sender, EventArgs e)

{

var addQuery = $"insert into [uch] (Name,post, Idgrp) values ('{textBox6.Text}' , '{textBox7.Text}',{comboBox1.SelectedIndex+1})";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void button7\_Click(object sender, EventArgs e)

{

var addQuery = $"delete from [uch] where ID={textBox5.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void dataGridView3\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView3.Rows[selectedRow];

textBox8.Text = row.Cells[0].Value.ToString();

textBox9.Text = row.Cells[1].Value.ToString();

comboBox2.SelectedIndex = Int32.Parse(row.Cells[2].Value.ToString())-1;

}

}

private void button8\_Click(object sender, EventArgs e)

{

var addQuery = $"insert into [Gruop] (Name,Idfak) values ('{textBox9.Text}' ,{comboBox2.SelectedIndex + 1})";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button6\_Click(object sender, EventArgs e)

{

var addQuery = $"update [uch] set Name='{textBox6.Text}' , post = '{textBox7.Text}', Idgrp = '{comboBox1.SelectedIndex+1}' where ID = {textBox5.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG2(dataGridView2);

}

private void button9\_Click(object sender, EventArgs e)

{

var addQuery = $"update [grp] set Name='{textBox9.Text}' , Idfak = '{comboBox2.SelectedIndex+1}' where ID = {textBox8.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG3(dataGridView3);

}

private void button11\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string querystr = $"select s.fio as c1, s.post as c2, g.name as c3, d.name as c4 from [fak] d, [uch] s, [grp] g where g.idfak=d.id and s.idgrp=g.id";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button12\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string querystr = $"select fio as col1, CAST(post AS varchar(5)) as col2, CAST(post-1000 AS varchar(5)) as col3, 'post-1000' as col4 from [uch] s";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button13\_Click(object sender, EventArgs e)

{// условие

dataGridView4.Rows.Clear();

string s = "OR";

if (radioButton5.Checked) {

s = "AND"; }

string n = '%'+textBox10.Text+'%';

string l = textBox11.Text;

string querystr = $"select fio as c1, CAST(post AS varchar(5)) as c2, '' as c3, '' as c4 from [uch] s where fio like '{n}' {s} post='{l}'";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button14\_Click(object sender, EventArgs e)

{

dataGridView4.Rows.Clear();

string n = '%'+textBox12.Text+'%';

string querystr = $"select fio as c1, CAST(post AS varchar(5)) as c2, '' as c3, '' as c4 from [uch] s where fio like'{n}'";

SqlCommand command = new SqlCommand(querystr, database.getConnection());

database.openConnection();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

ReadSinglRow4(dataGridView4, reader);

}

reader.Close();

}

private void button10\_Click(object sender, EventArgs e)

{

var addQuery = $"delete from [grp] where ID={textBox1.Text}";

var command = new SqlCommand(addQuery, database.getConnection());

command.ExecuteNonQuery();

refreshDG1(dataGridView1);

}

private void button4\_Click(object sender, EventArgs e)

{

refreshDG1(dataGridView1);

}

}

class DataBase

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source=localhost\SQLEXPRESS;Initial Catalog=kolledg;Integrated Security=True");

public void openConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Closed)

{

sqlConnection.Open();

}

}

public void closeConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Open)

{

sqlConnection.Close();

}

}

public SqlConnection getConnection()

{

return sqlConnection;

}

}

}

**Список использованных источников**

1 Бен-Ган, И. Microsoft SQL Server 2012. Основы T-SQL / И. Бен-Ган. М. : Эксмо, 2015. 400 с.

2 Дейт, К.Дж. Введение в системы баз данных / К.Дж. Дейт. М. : Вильямс, 2018. 1382 с.

3 Лазицкас, Е.А. Базы данных и системы управления базами данных / Е.А. Лазицкас, И.Н. Загуменникова, П.Г. Гилевский. Минск : РИПО, 2016. 268 с.

4 Федорова, Г. Разработка и администрирование баз данных / Г. Федорова. М. : Академия, 2015. 313 с.

5 SQL справочник / К. Кляйн [и др.]. СПб. : Символ-плюс, 2016. 56 с.