Федеральное казенное профессиональное образовательное учреждение «Новокузнецкий государственный гуманитарно-технический колледж-интернат»

Министерства Труда и социальной защиты Российской Федерации

Специальность: 09.02.04 «Информационные системы (по отраслям)»

КУРСОВОЙ ПРОЕКТ

по дисциплине: МДК 02.02 Управление проектами

на тему: "Разработка программного продукта "Игра АРКАНОИД" с применением СУБД"

РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Студент Михайлов В. В.

дата личная подпись расшифровка подписи

Руководитель проекта Грисман С.С.

дата личная подпись расшифровка подписи

Нормоконтроль Михайлов Н.Н,

дата личная подпись расшифровка подписи

Оценка

Новокузнецк, 2019 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc27055485)

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ПРОГРАММНОГО ПРОДУКТА 5](#_Toc27055486)

[2. ОБЩАЯ ЧАСТЬ 7](#_Toc27055487)

[2.1 Выбор и обоснование средств реализации БД 7](#_Toc27055488)

[2.2. Выбор и обоснование средств разработки программного продукта 9](#_Toc27055489)

[2.3 Анализ прототипов, существующих ИС 9](#_Toc27055490)

[3 СПЕЦИАЛЬНАЯ ЧАСТЬ 12](#_Toc27055491)

[3.1 Разработка алгоритма и проектирование структуры базы данных 12](#_Toc27055492)

[3.2 Разработка программного продукта 13](#_Toc27055493)

[3.3 Руководство пользователя 21](#_Toc27055494)

[ЗАКЛЮЧЕНИЕ 30](#_Toc27055495)

[СПИСОК ЛИТЕРАТУРЫ 31](#_Toc27055496)

[ПРИЛОЖЕНИЯ 32](#_Toc27055497)

# **ВВЕДЕНИЕ**

Игры – это неотъемлемая часть нашего существования. В наш, насыщенный информацией век игры двигают вперед развитие компьютерной сферы.

Основой любой игры является интерес к происходящему, личное участие и самоутверждение по завершении игры.

Для хранения информации о пользователе применяется база данных. База данных – это информационная структура, хранимая и предоставляющая информацию к игре.

Целью данного курсового проекта, является разработка игры «Арканоид» с использованием СУБД MSSQL-server.

Актуальность темы обуславливается необходимостью развития координации и скорости реакции.

Объектом в данной области является игра «Арканоид»

Предметом является возможность оптимизации процессов, связанных с обработкой информации и её обменом с пользователями, регистрация незарегистрированных, и авторизация в системе уже зарегистрированных клиентов.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* Разработать техническое задание;
* Выбрать и обосновать средства реализации БД;
* Проанализировать прототипы;
* Произвести анализ и проектирование структуры базы данных;
* Выбрать и обосновать программные средства реализации;
* Разработать программный продукт;
* Провести тестирование и отладить программный продукт.
* Разработать руководство пользователя

Разрабатываемое приложение сможет применяться как средство для развлечения пользователей, развития внимания и реакции.

Создание данного приложения поможет освоить технологии разработки на языке программирования C#, базы данных в MS SQL, проработать связи программы с базой данных.

# **ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ПРОГРАММНОГО ПРОДУКТА**

**Разработка программного продукта "Игра АРКАНОИД" с применением СУБД**

Прототипом данной системы можно считать интернет [ресурс](http://www.bettergames.ru/game.php?id=3773) (http://www.bettergames.ru/game.php?id=3773), имеющий функциональную часть работы данного приложения.

**Требования к функциональным характеристикам следующие:**

1. Необходимо предусмотреть два режима работы – режим администратора и режим пользователя.
2. В режиме пользователя должна быть возможность зарегистрироваться, заполнить свою анкету (дата рождения, фото). Реализовывать возможность смены пароля и редактирования данных.
3. В режиме пользователя можно начать игру, а также посмотреть свой уровень игры (счет и пр.).
4. В режиме администратора должна быть возможность просматривать и редактировать пользователей, редактировать счет игрока.
5. Игрок характеризуется именем, счетом, фотографией.

В режиме администратора должна быть реализована возможность заполнения игровых данных об игроке.

1. Также у администратора и пользователя имеется возможность формировать рейтинг игроков (по счету, набранным в игре или дате игры и по ткущему пользователю)

**Требования к информационному обеспечению следующие:**

К приложению должна быть предоставлена документация, содержащая техническое описание продукта и руководство пользователя.

**Требования к лингвистическому обеспечению следующие:**

Приложение должно иметь интерфейс и документацию на русском языке.

**Требования к программному обеспечению следующие:**

Разработка программного продукта предусматривается под OS Windows не ниже версии 7.

**Требования к техническому обеспечению следующие:**

Для работы приложения требуется MS SQL Server, Visual Studio С#. Также, необходимо не менее 512Мб оперативной памяти и 100Мб свободного места на жестком диске.

# **ОБЩАЯ ЧАСТЬ**

## **2.1 Выбор и обоснование средств реализации БД**

Основным требованием работы является наличие интерфейса приложения, разрабатываемого в СУБД MS SQL.

**Microsoft SQL Server** — система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

**База данных MS SQL** – это система, предназначенная для хранения и обработки информации. Комплекс таблиц, взаимосвязанных между собой, для доступа к которым применяется система управления базами данных (СУБД) MS SQL. MS SQL – это специальная программа с открытым кодом, которая используется на сервере SQL. Данная программа не способна обрабатывать большое количество информации, однако она идеальна для небольших и крупных веб-ресурсов.

В настоящее время, СУБД MS SQL является одной из самых известных, надежных и быстрых из всего семейства существующих СУБД.

Одной из уникальных особенностей версии 2019 года является temporal data support (временная поддержка данных), которая позволяет отслеживать изменения данных с течением времени. Последняя версия Microsoft SQL–сервер поддерживает dynamic data masking (динамическую маскировку данных), которая гарантирует, что только авторизованные пользователи будут видеть конфиденциальные данные.

Достоинства Microsoft SQL Server:

1. Поддержка постоянной памяти (РМЕМ)

PMEM — это быстрая память для хранения данных после отключения питания. Она позволяет обрабатывать данные in-memory, избавляя от необходимости передавать данные по каналам передачи и ускоряя обработку запросов на 30% для интенсивных рабочих нагрузок ввода-вывода.

1. Кластеры больших данных

В SQL Server 2019 управлять средой для больших данных стало гораздо проще. Этот продукт содержит основные элементы озера данных – распределенную файловую систему Hadoop (HDFS), Spark и инструменты анализа. Все они тесно интегрированы с SQL Server и поддерживаются Microsoft.

1. Scalar UDF Inlining

В SQL Server 2019 появилась функция Scalar UDF Inlining – встраивание скалярных пользовательских функций. Вместо отдельной обработки функции, она встраивается в общий план что приводит к значительному повышению производительности.

1. Интеллектуальная обработка запросов

Параллелизация запросов и улучшенное масштабирование частых запросов благодаря механизмам интеллектуальной обработки запросов делают производительность значительно выше. Отложенная компиляция табличных переменных более чем на 50% ускоряет обработку запросов.

1. Интеграция структурированных и неструктурированных данных

SQL Server 2019 позволяет компаниям получать ценную информацию из структурированных и неструктурированных данных, объединяя мощь новых кластеров больших данных с улучшенной виртуализацией данных, новые функции в SQL Server позволяют не только хранить и запрашивать большие данные в масштабе предприятия, но и объединять их со структурированными данными в любой базе данных, в которой они могут находиться, например, в SQL Server, Oracle, MongoDB и PostgreSQL.

## **Выбор и обоснование средств разработки программного продукта**

При написании приложения был выбран язык программирования C# (Си шарп).

Язык C#, разработанный компанией Майкрософт, один из самых популярных современных языков программирования. Он востребован на рынке разработки, C# применяют при работе с программами для ПК, при создании сложных веб-сервисов или мобильных приложений.

Приложение разрабатывалось в программе Microsoft Visual Studio 2017, она имеет графический интерфейс для построения приложения и программную оболочку для работы непосредственно с кодом. Работать в программе удобно и просто.

В Visual Studio можно быстро подключить внешнюю базу данных используя мастера подключений, связав тем самым базу и разрабатываемую вами программу. Язык C# достаточно прост и популярен у начинающих программистов. Разработка приложения в Visual Studio путем графического добавления элементов на создаваемую форму и с последующим добавлением кода в программной оболочке на нужный вам элемент это довольно удобно, так как все находится в одной программе и не нужно переключаться, что исключает снижение концентрации на выполняемой работе.

## **2.3 Анализ прототипов, существующих ИС**

При реализации программного продукта, разработчик анализирует и систематизирует данные, просматривая аналогичные приложения. На данном этапе происходит весь сбор информации, выявление достоинств и недостатков, которые имеются у прототипов. После проведённой работы, он продумывает план создания приложения, куда будут включены плюсы, а также исключены ошибки прототипов.

Схема анализа прототипов включает в себя:

* дизайн приложения и общее впечатление от него;
* доступность информации для пользователей;
* структура программного продукта и удобство использования;
* навигация в приложении;
* объем и актуальность представленной информации в данном программном продукте;
* удобство использования панели администратора данного приложения.

При проектировании приложения «АРКАНОИД» прототипами стали Web-приложения «3DNOID». Перейти на сайт можно по адресу http://www.bettergames.ru/arcanoid.php. Представлено на рисунке 1.

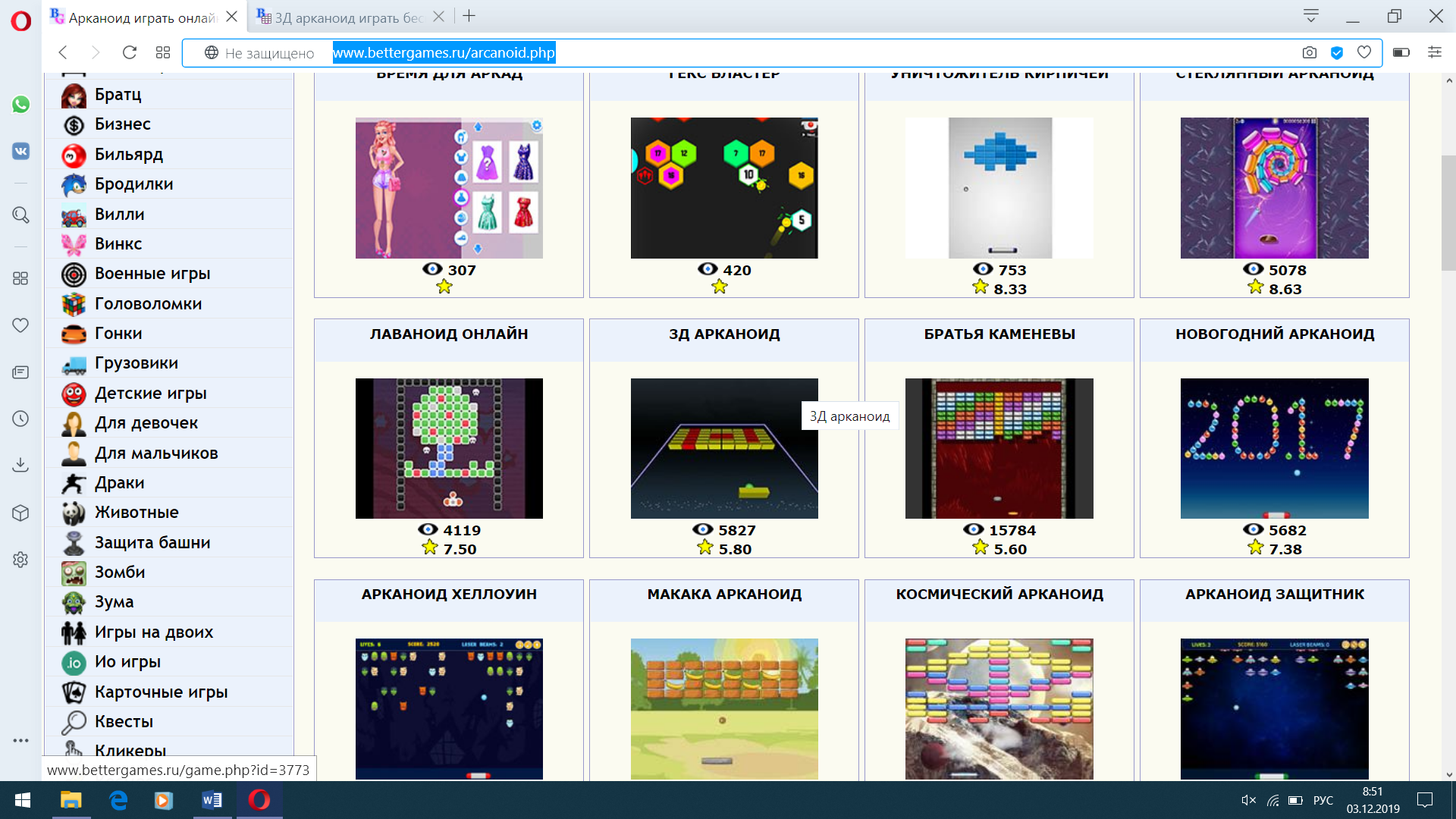


Рисунок 1 - «Сайт игр. Раздел Арканоид»

На сайте присутствует множество игр подобного жанра, остальных направлений.

Игра имеет псевдо 3D дизайн, в ней работает подсчёт баллов. Но не реализована авторизация и регистрация пользователей, результаты не сохраняются. При этом весь необходимый для создания программного продукта механизм работает отлично.

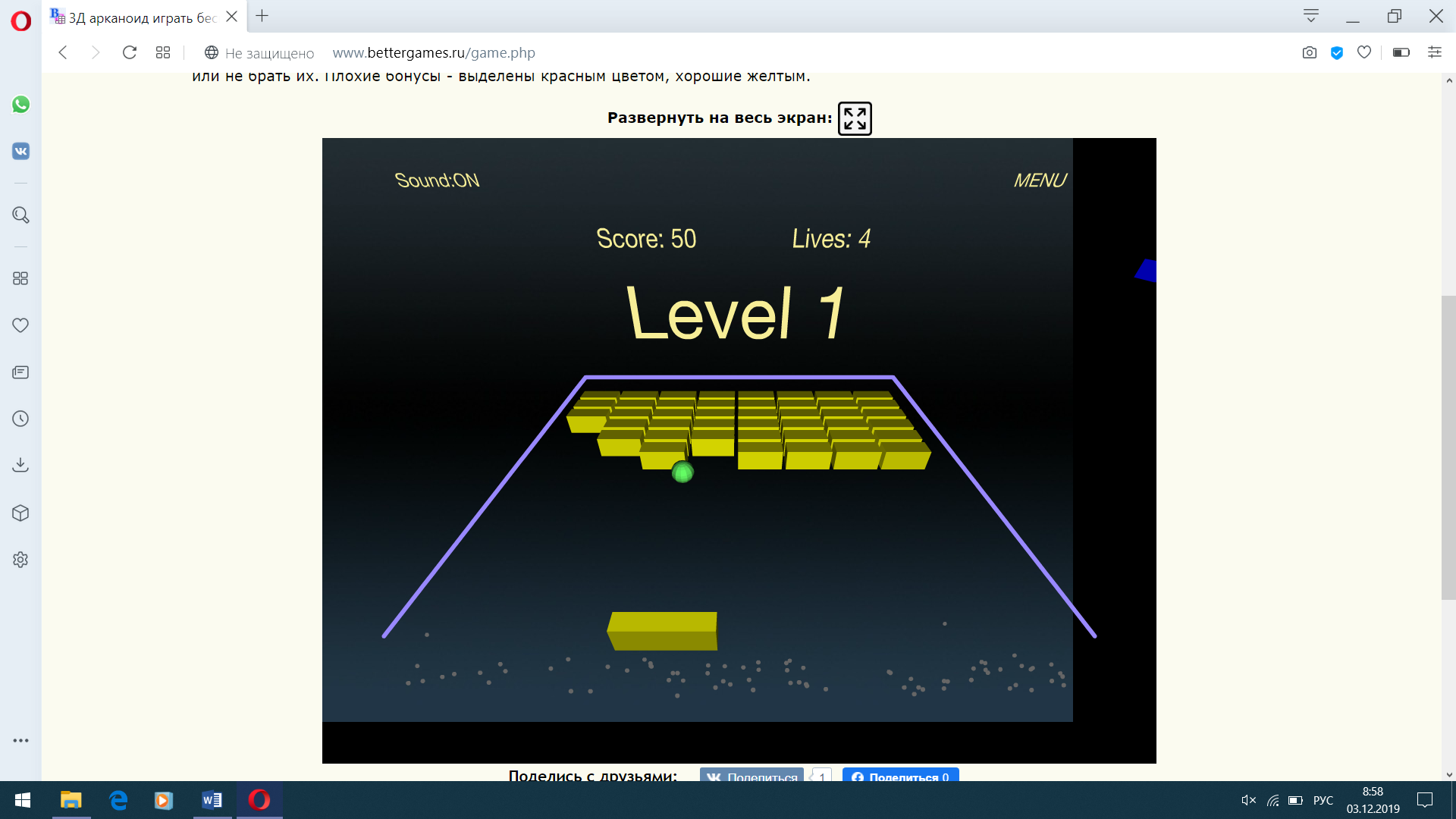


Рисунок 2 - «Игра 3DNOID»

# **3 СПЕЦИАЛЬНАЯ ЧАСТЬ**

## **3.1 Разработка алгоритма и проектирование структуры базы данных**

База данных «ARCANOID» была создана на основании задания, представленного в приложения А.

Модель базы данных «ARCANOID» создана в дизайнере Microsoft SQL Server Management Studio 18.

При создании таблиц базы данных использовался режим конструктора MS SQL, где были установлены свойства полей всех таблиц данной базы данных.

База данных «ARCANOID» состоит из двух таблиц:

Для создания таблиц использовался режим конструктора, позволяющий установить свойства полей таблицы.

Модель базы данных «ARCANOID» состоит из двух таблиц, каждая из которых имеет свою структуру, необходимую для установления связи между всеми таблицами:

* «USERS» (Пользователи);
* «PLAYERS» (Игроки);

Таблица «USERS» предназначена для входа в систему пользователей, при вводе логина и пароля. Ключевым полем является id\_user (Код пользователя), тип данных int. Также имеются поля login\_user (Логин пользователя), тип данных nvarchar(50) и password\_user (Пароль пользователя), тип данных nvarchar(50).

Таблица «PLAYERS» имеет связь с таблицей «USERS». Ключевым полем является поле id (Код) тип данных int. Также имеются поля id\_players (Код игрока), по которому проходит соединение с таблицей «USERS».

Целостность данных *–* это значит, что данные будут надёжно хранится в базе данных. СУБД автоматически поддерживает целостность данных с помощью установленных связей между таблицами. Таблицы базы данных связаны связями «один ко многим»; для каждой из связей установлена поддержка целостности данных, каскадные обновление и удаление. Схема связей таблиц представлена в соответствии с рисунком 3.

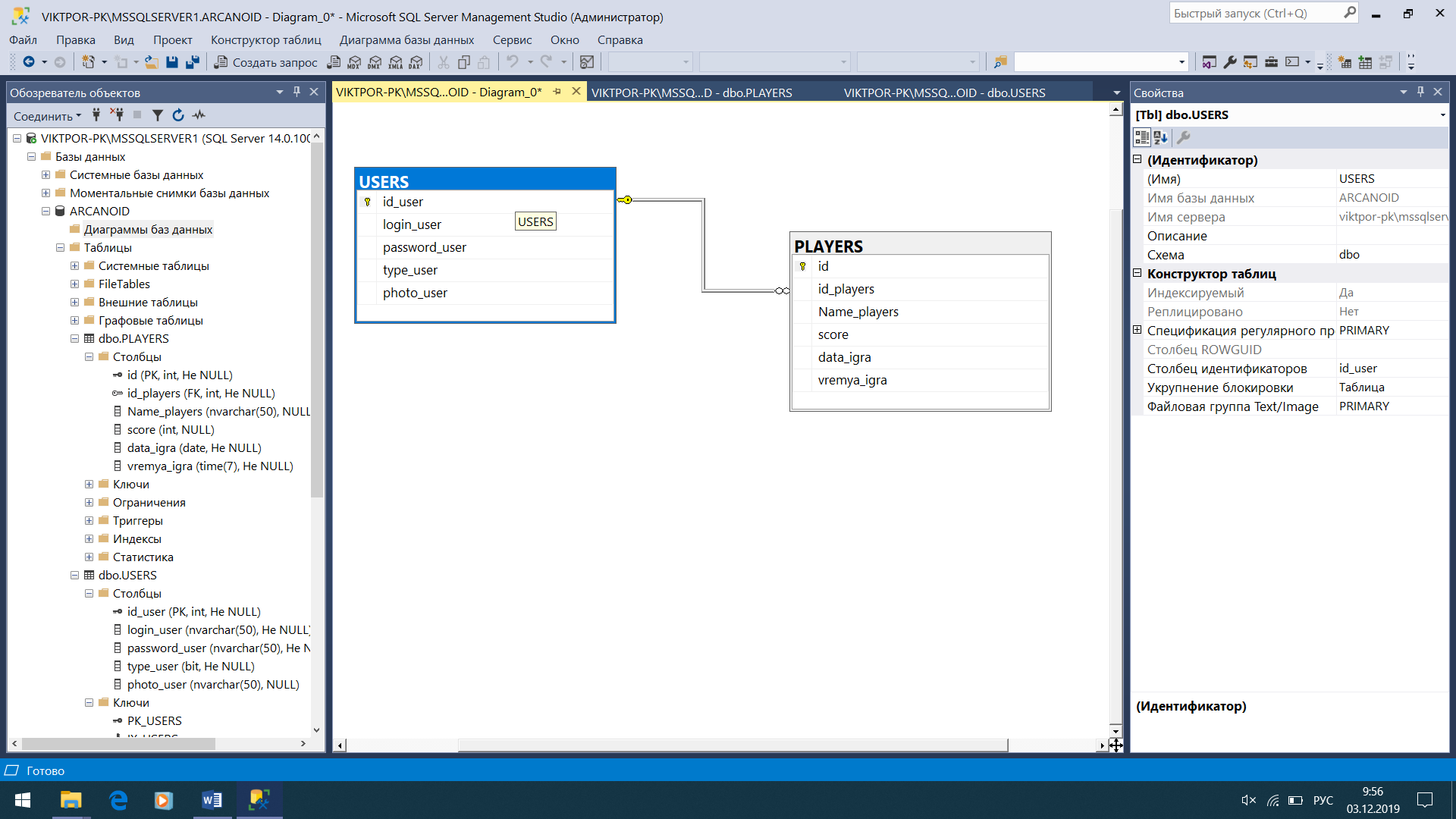


Рисунок 3 - «Схема связей таблиц»

## **3.2 Разработка программного продукта**

При разработке данного программного продукта было реализовано несколько подсистем. Первая из них – система авторизации. В ней реализован механизм, где работа происходит с зарегистрированными пользователями. Алгоритм работы выглядит так, что при загрузке приложения, происходит проверка на регистрацию пользователя в базе программы. Если она имеется, то авторизация не предлагается, а в противном случае, необходимо её пройти. Данный механизм реализован посредством простой проверкой существования записи в таблице. Проверка идет SQL запросом в присоединенную к программе базу данных. Код данной части программного продукта представлен на листинге 1.

Листинг 1. Поиск юзера

//строка н поиск юзера

string s = "select \* from users where login\_user = '" + login\_user + "' and password\_user = '" + password\_user + "'";

Стоит отметить, что в данном действии, как и во всей системе происходит работа с базой данных. После ввода логина и пароля происходит отправка запроса на выборку имени пользователя (login\_user) и пароля (password\_user) после чего возвращает результат выполнения. В программе предусмотрен вывод предупреждения в случае ошибочных данных. представлен на листинге 2.

Листинг 2. Вывод ошибки пользователя

else

{

MessageBox.Show("Пользователь не найден");

}

Вторая из подсистем – регистрация. При заполнении необходимых полей происходит добавление нового пользователя в базу данных, код предоставлен на листинге 3.

Листинг 3. Запрос на регистрацию

SqlConnection connection = new SqlConnection(Program.st\_connect);

try

{

connection.Open();

SqlCommand command = new SqlCommand($"INSERT INTO[dbo].[USERS]" +

$"([login\_user],[password\_user],[type\_user],[photo\_user])" +

$"VALUES" +

$"('{LoginBox.Text}','{PasswordBox.Text}',0,'{pathImage}')");

command.Connection = connection;

int rows = command.ExecuteNonQuery();

if(rows > 0)

{

MessageBox.Show($"Добавлено {rows} пользователей");

}

После выполнения команды в базу данных попадает запись о новом пользователе. После чего выводится сообщение о добавлении нового пользователя.

Далее открывается форма просмотра и изменения статистики. Переход в нее осуществляется через проверку типа пользователя. Код данной части представлен на листинге 4.

. Если авторизуемый тип пользователя «Администратор», то открывается форма администратора, если «Пользователь», соответственно открывается форма для пользователя.

Листинг 4. Проверка типа пользователя

if (Program.type\_user == false)

{

USERS\_FORM f = new USERS\_FORM();

f.Show();

}

else

{

Form3 f = new Form3();

f.Show();

}

Из этих форм можно попасть в формы «Изменение пользователя», «Статистика пользователя» или запустить саму игру.

Форма изменения пользователя реализована так же, как и регистрация, так же там добавлена возможность смены типа пользователя. С кодом можно ознакомиться на листинге 5.

Листинг 5. Тип пользователя

//Запрос на обновление данных в таблице пользователей

int k = dataGridView1.CurrentRow.Index;

Program.id\_user = Convert.ToInt32(dataGridView1[0, k].Value);//запоминаем id выбранного юзера

SqlConnection connect = new SqlConnection(Program.st\_connect);

connect.Open();

string s = "UPDATE USERS set " +

"login\_user='" + LoginBox.Text.Trim() + "', " +

"password\_user='" + PasswordBox.Text.Trim() + "'," +

"type\_user='" + comboBox1.SelectedIndex + "'," +

"photo\_user='" + label3.Text + "' " +

"WHERE id\_user = " + Program.id\_user.ToString();

SqlCommand comm = new SqlCommand(s, connect);

comm.ExecuteScalar();

connect.Close();

//Запрос на обновление видимых данных в таблице после обновления данных в базе

SqlConnection conn1 = new SqlConnection(Program.st\_connect);

conn1.Open();

string s1 = "SELECT \* FROM USERS";

//Делаем запрос к БД через адаптер, что бы потом поместить даные в DATASET

SqlDataAdapter adap = new SqlDataAdapter(s1, conn1);

//создаем пустой датасет

DataSet ds1 = new DataSet();

//соединяем датасет с нашим набором данных

adap.Fill(ds1);

//соединяем датасет с визуальным компонентом

dataGridView1.DataSource = ds1.Tables[0];

if (ds1.Tables[0].Rows.Count >= 10)

{

button2.Enabled = true;

}

conn1.Close();

dataGridView1.Columns[0].Visible = false;

LoginBox.Text = "";

PasswordBox.Text = "";

comboBox1.Text = "";

//Вывод сообщения об изменении

MessageBox.Show("Изменено");

Далее идет форма просмотра и изменения статистики. Существует форма изменения статистики, открываемая только для администратора, и упрощенная форма администратора, открываемая для пользователя, с меньшим функционалом.

В данной форме показывается таблица всех игроков, которую можно перелистывать. В ней так же можно удалять игроков (Данный отрезок кода представлен в листинге 6), менять его игровые данные (Отрезки кода для смены данных можно увидеть в листингах 7, 8, 9), формировать отчет в Excell (с этим кодом можно ознакомиться в листинге 10).

Листинг 6. Удаление

int k = dataGridView1.CurrentRow.Index;

Program.id\_user = Convert.ToInt32(dataGridView1[0, k].Value);

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

string s = "DELETE FROM PLAYERS WHERE id=" + Program.id\_user.ToString();

SqlCommand comm = new SqlCommand(s, conn);

comm.ExecuteScalar();

conn.Close();

MessageBox.Show("Чел удален");

При нажатии кнопки изменения, происходит изменение размеров формы, открываются дополнительные поля

Листинг 7. Изменение размеров формы

this.Width = 900;

this.Height = 465;

В текстбоксы переносятся данные выделенной в таблице строки, которые можно изменять.

Листинг 8. Данные в текстбоксах

textBox2.SelectionStart = 0;

textBox2.SelectionLength = textBox2.Text.Length;

textBox2.Focus();

textBox1.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

textBox2.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

Листинг 9. Изменение данных

int k = dataGridView1.CurrentRow.Index;

Program.Id = Convert.ToInt32(dataGridView1[0, k].Value);//запоминаем id выбранного юзера

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

string s = "UPDATE PLAYERS " +

" SET " + "Name\_players= " + "'" + textBox1.Text.Trim() + "'," +

"score = " + "'" + textBox2.Text.Trim() + "'," +

"data\_igra = " + "'" + dateTimePicker1.Value.ToShortDateString() + "'," +

"vremya\_igra = " + "'" + dateTimePicker2.Value.ToShortTimeString() + "' WHERE Id=" + Program.Id.ToString();

SqlCommand comm = new SqlCommand(s, conn);

comm.ExecuteScalar();

conn.Close();

Измененные данные попадают в таблицу и их сразу можно увидеть.

Листинг 10. Формирование отчета в Excel

//создаем виртуальный объект Excel

Microsoft.Office.Interop.Excel.Application myExcel = new Microsoft.Office.Interop.Excel.Application();

//создать книгу в объекте Excel

myExcel.Application.Workbooks.Add(Type.Missing);

//название страницы

DateTime dt = DateTime.Now;

string curDate = dt.ToShortDateString();

myExcel.ActiveSheet.Name = "Отчет за " + curDate;

//Настраиваем ячейки

myExcel.Columns.ColumnWidth = 20;

//Пишем заголовки ячеек

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

myExcel.Cells[1, i + 1] = dataGridView1.Columns[i].HeaderText;

// заголовки программно

}

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

for (int j = 0; j < dataGridView1.RowCount - 1; j++)

{

myExcel.Cells[j + 2, i + 1] = dataGridView1[i, j].Value.ToString();

}

}

//скрываем столбец А

Range range = (Range)myExcel.Columns[1, Type.Missing];

range.EntireColumn.Hidden = true;

myExcel.Visible = true;

}

В отчете показывается имя игрока, его счет, дата и время игры. Первый столбец с айди игрока скрывается программно. Готовый отчет можно сохранить.

Так же в этой форме можно формировать списки по дате (Код представлен в листинге 11), по счету (отрезок кода можно просмотреть в листинге 12), и по имени текущего пользователя (данную часть кода можно просмотреть в листинге 13).

Формирование списков проходит через запросы к базе данных. После отправки запроса можно увидеть измененную таблицу. Списки показываются выбором определенного радиобаттона.

Листинг 11. Формирование списка по дате игры

else if (radioButton2.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', vremya\_igra as 'Время' FROM PLAYERS ORDER BY data\_igra DESC OFFSET " + n.ToString() + "ROWS FETCH NEXT 10 ROWS ONLY ";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

button2.Enabled = false;

}

else

{

button1.Enabled = false;

button2.Enabled = true;

}

Листинг 12. Формирование списка по счету

else if (radioButton1.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

button2.Enabled = false;

}

else

{

button1.Enabled = false;

button2.Enabled = true;

}

}

Листинг 13. Демонстрация счета авторизированного игрока

SqlConnection connect = new SqlConnection(Program.st\_connect);

string s = "SELECT Id\_user, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', vremya\_igra as 'Время' FROM PLAYERS INNER JOIN USERS on PLAYERS.id\_players=USERS.id\_user WHERE Id\_user= " + Program.id\_user.ToString();

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

Изменения игроков и формирования списков завершено, перейдем к самой игре.

Игра представляет собой механизм разбивания блоков, подсчета очков, так же необходимо не упустить шарик, орудие игры. Внизу экрана игроком передвигается платформа, которой нужно отбивать шарик. Блоки для разбивания строятся циклом, цвет блоков подбирается случайно. Перед инициализацией формы необходимо указать название рандома, на который ссылается цикл Foreach. Эту часть кода можно увидеть в листинге 14.

Листинг 14. Расстановка блоков

private Random rnd = new Random();

public Form1()

{

InitializeComponent();

foreach (Control x in this.Controls)

{

if (x is PictureBox && x.Tag == "block")

{ Color RandomColor = Color.FromArgb(rnd.Next(256), rnd.Next(256),

x.BackColor = RandomColor;

}

}

}

В листинге 15 можно увидеть механизм игры. ball – это шарик, ballx ballly – его координаты по осям (необходимы для столкновений с блоками или границами окна игры), player – платформа, отбивающая шарик, goleft и gorigt – направления движения платформы, право и лево. Speed – скорость движения платформы.

Листинг 15.

private void timer1\_Tick(object sender, EventArgs e)

{

ball.Left += ballx;

ball.Top += bally;

label1.Text = "Очки " + score;

if (goLeft)

{

player.Left -= speed;

}

if (goRight)

{

player.Left += speed;

}

if (player.Left < 1)

{

goLeft = false;

}

else if (player.Left + player.Width > 670)

{

goRight = false;

}

if (ball.Left + ball.Width > ClientSize.Width || ball.Left < 0)

{

ballx = -ballx;

}

if (ball.Top < 0 || ball.Bounds.IntersectsWith(player.Bounds))

{

bally = -bally;

}

if (ball.Top +ball.Height > ClientSize.Height)

{

GameOver();

MessageBox.Show("Вы проиграли");

timer1.Stop();

}

foreach(Control x in this.Controls)

{

if (x is PictureBox && x.Tag == "block")

{

if (ball.Bounds.IntersectsWith(x.Bounds))

{

this.Controls.Remove(x);

bally = -bally;

score++;

}

}

}

if (score > 34)

{

GameOver();

MessageBox.Show("Вы победили");

timer1.Stop();

}

}

this.Text = "Имя " + player.Text.ToString() + ", Счет " + score;

По завершении игры (если мячик упущен или разбиты все блоки) останавливается таймер далее происходит запись в базу данных. Записывается id игрока, его имя, счет, дата и время игры. Окно игры закрывается и выводится сообщение о проигрыше или выигрыше, открывается форма, из которой моно посмотреть свой счет или начать игру заново. В листинге 16 можно увидеть эту часть кода.

Листинг 16. Конец игры

void GameOver()

{

timer1.Stop();

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

DateTime dateTimeVariable = DateTime.Now;

string date1 = dateTimeVariable.ToString("yyyy-MM-dd");

string date2 = dateTimeVariable.ToString("H:mm:ss");

string s = "INSERT INTO PLAYERS " +

"(id\_players, Name\_players, score, data\_igra, vremya\_igra) VALUES" +

"('" + Program.id\_user + "'," +

"'" + player.Text.ToString() + "'," +

"'" + score + "', " +

"'" + date1 + "', " +

"'" + date2 + "')";

SqlCommand command = new SqlCommand(s, conn);

SqlDataReader reader = command.ExecuteReader();

conn.Close();

this.Close();

}

С кодом всей программы, всех ее форм и кнопок можно ознакомиться в приложении B.

## **3.3 Руководство пользователя**

Запуск программы. Программа запускается с ярлыка приложения. Первое окно программы – это форма авторизации.

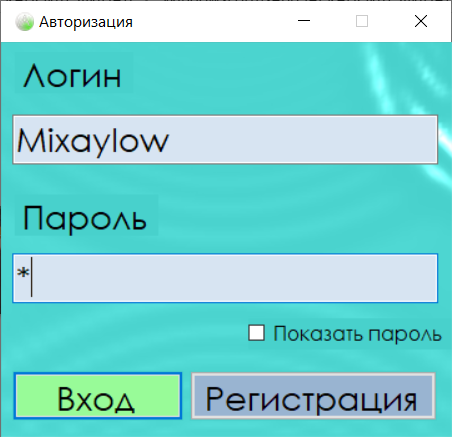


Рисунок 4 - «Окно авторизации»

Для входа необходимо ввести уже зарегистрированные учетные данные, либо пройти регистрацию, нажав по кнопке «Регистрация»

Кнопка «Регистрация» открывает окно регистрации нового пользователя. В ней необходимо внести логин, пароль и предоставить свое фото. Данные записываются в базу данных игры и хранятся там. Сразу после регистрации можно авторизоваться под новыми учетными данными.

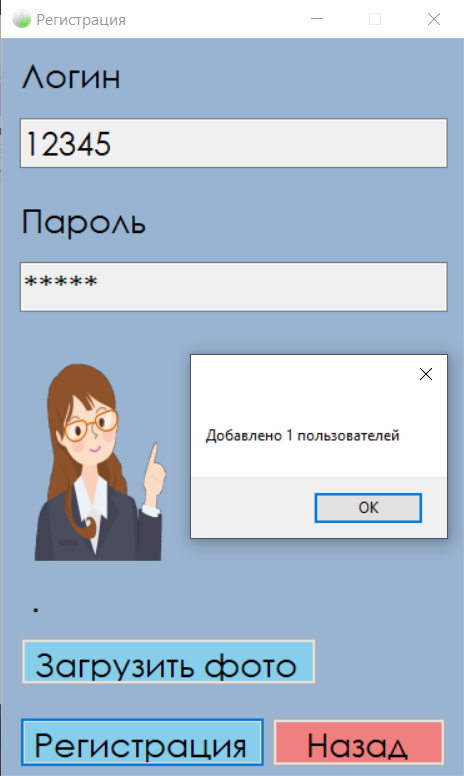


Рисунок 5 - «Регистрация нового пользователя»

После успешной авторизации открывается окно пользователя, она представлена на рисунке 11 (или администратора – на рисунке 6), в котором можно выбрать игру, посмотреть статистику и т.д.



Рисунок 6 - «Форма администратора»

В форме администратора реализована возможность изменения данных пользователя, просмотр и изменения игровых итогов игроков, запустить игру и покинуть форму.

Кнопка «Изменить юзера» на форме администратора осуществляет переход на форму изменения пользователей. В ней можно менять логин, пароль, тип юзера (админ, не админ) и фотографию (аватарку). Рисунок 7 демонстрирует это окно.

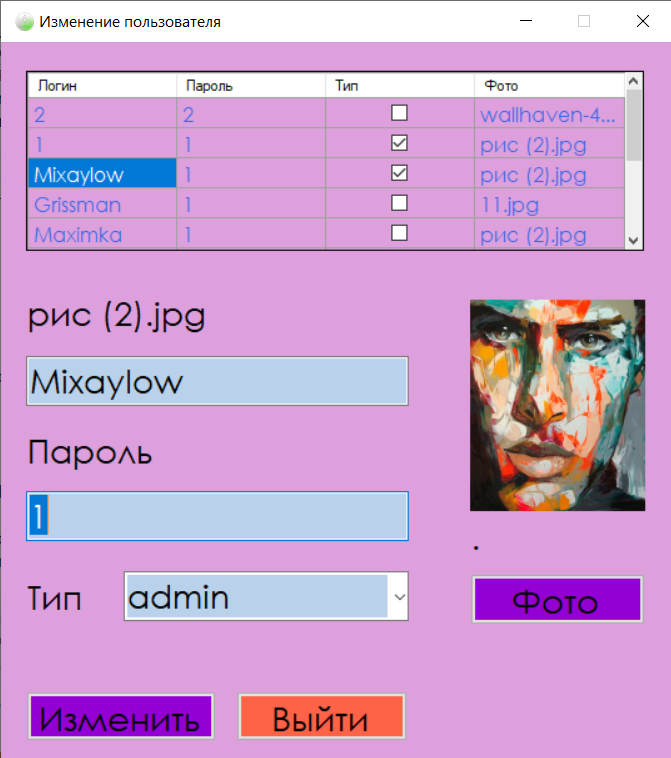


Рисунок 7 - «Форма изменения пользователей»

Кнопка «Изменить игрока» на форме администратора.

В окне, которое мы видим на рисунке 8, формируется таблица с данными игроков, учувствовавших в игре. Здесь можно увидеть имя игрока, его счет, дату и время игры. Здесь осуществляется листание игроков (кнопки << и >>), игрока можно удалить (кнопка «Удалить»), изменить его данные (кнопка «Изменить»). Список можно сформировать по счету, по дате (Радиобаттоны). Данные можно вывести в Excel документ (кнопка «Отчет в Excel», вывод результата в Excell можно увидеть на рисунке 10).

При нажатии кнопки «Изменить» окно изменяется и открывается дополнительное меню, это можно увидеть на рисунке 9. В нем можно менять данные игрока. Измененные данные сохраняются в базе.

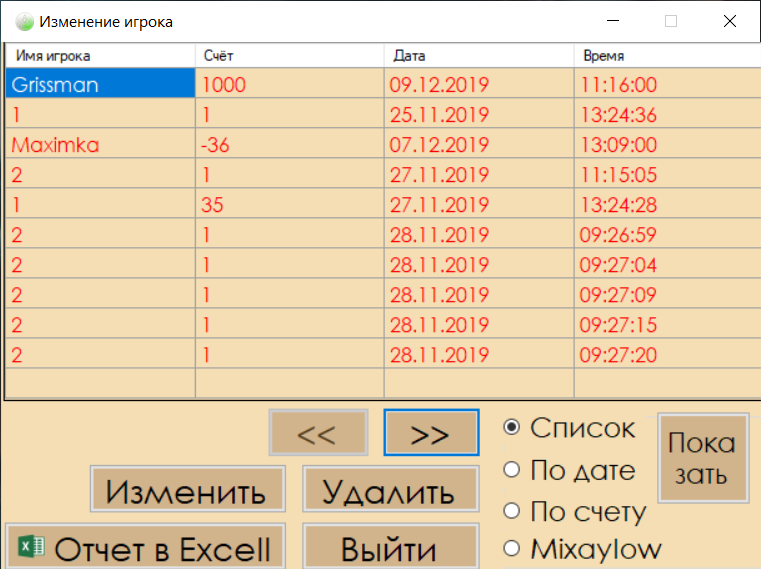


Рисунок 8 - «Форма изменения игроков»

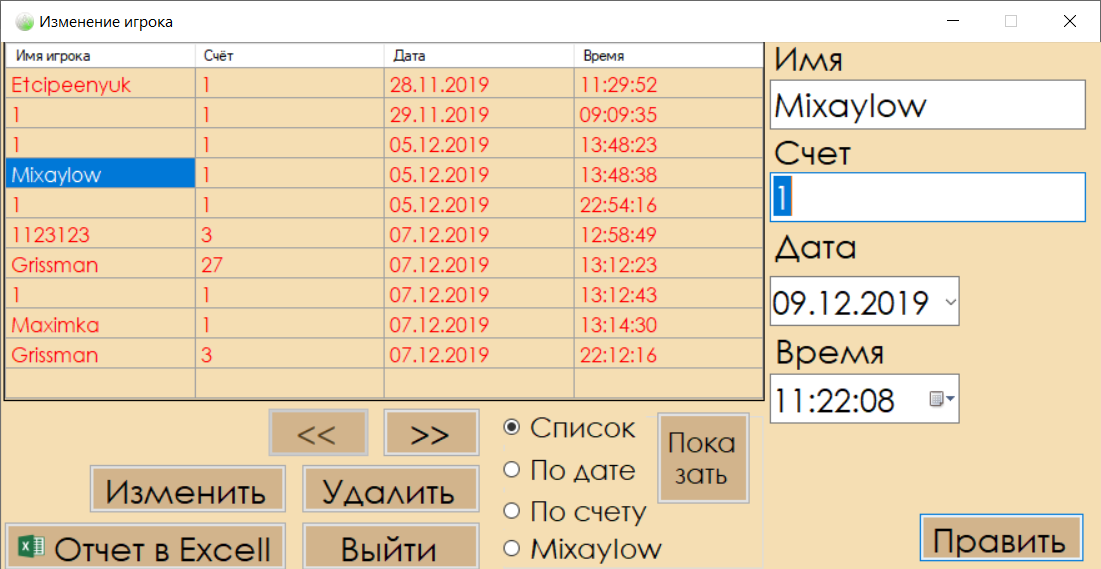


Рисунок 9 - «Расширенная форма изменения игроков»

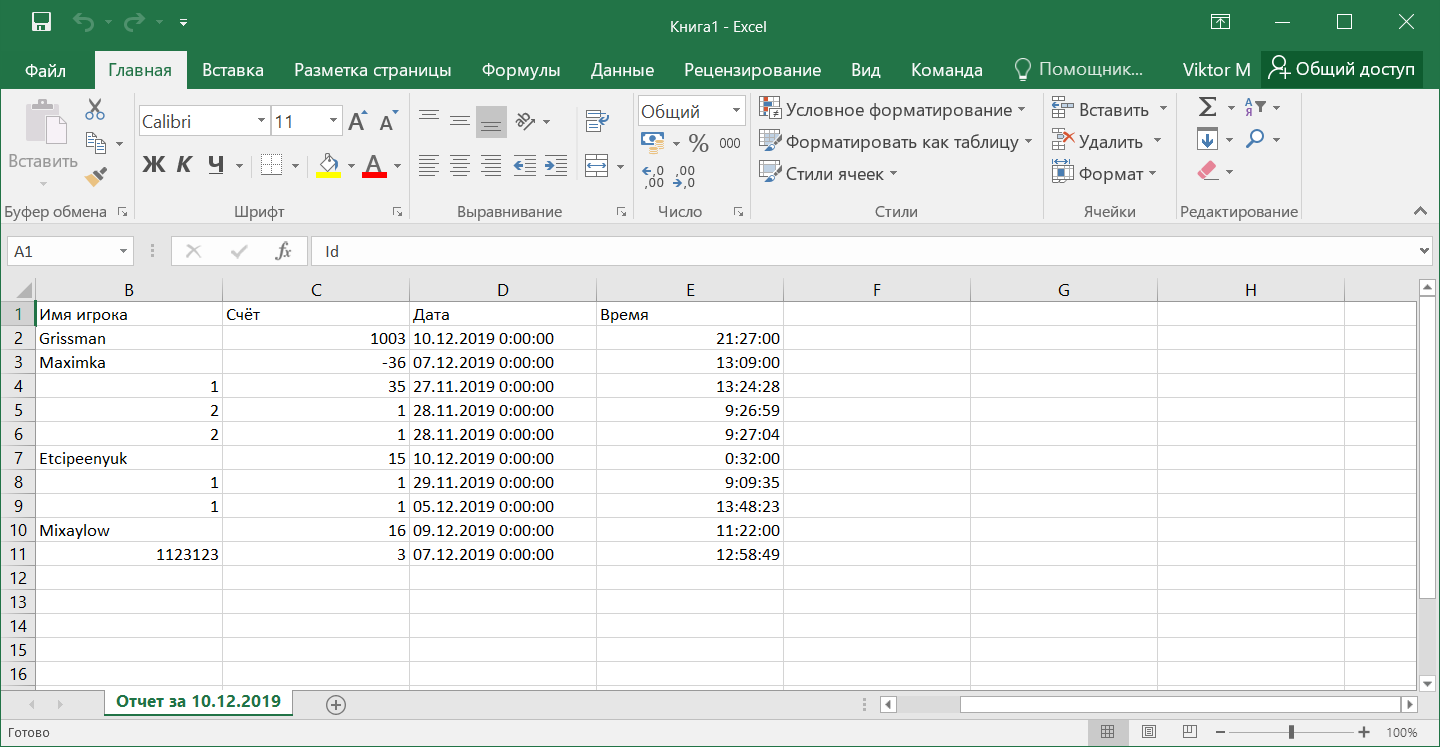


Рисунок 10 - «Формирование отчета в Excell »

Кнопка «Игра» начинает саму игру, рисунок 11. Игра представляет собой несложный процесс разбивания блоков, с ведением счета и отбивание шарика плавающей снизу экрана платформой. На платформе видно имя игрока, так же оно написано в заголовке формы. Там же можно увидеть и свой текущий счет. Данные так же видны и на самой форме, в верхнем левом углу.



Рисунок 11 - «Игра»

После завершения игры (разбиты все блоки или шарик упущен) в базу поступают данные игрока. Его имя, счет, дата и время игры, которые потом можно просмотреть или изменить (если вы админ) в статистике.

Кнопка «Выйти» закрывает форму, после открывается форма авторизации.

Форма игрока гораздо проще. Как это видно на рисунке 12, в ней по сравнению с формой админа функционал ограничен. Здесь можно посмотреть статистику, не изменяя её и запустить игру.



Рисунок 12 - «Форма пользователя»

Кнопка «Статистика» открывает упрощенную форму изменения игроков. Она предоставлена на рисунке 13. В ней скрыты кнопки, которые были доступны в режиме админа. Оставлена только возможно листания таблицы, формирования рейтингов, и вывод статистики в Excel.

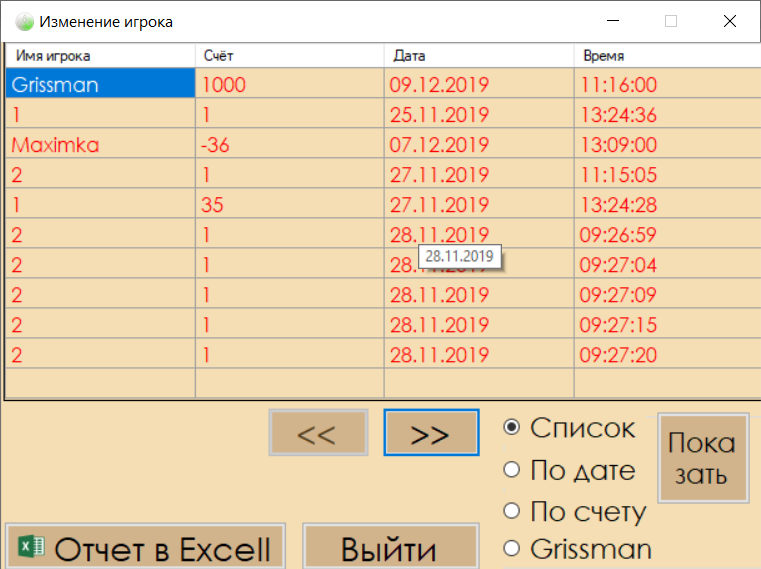


Рисунок 13 - «Упрощенная форма изменения игроков»

Остальные кнопки в форме игрока выполняют аналогичные роли, что и в форме администратора.

**3.4 О программе**

В игре имеется меню «О программе». Вызывается кнопкой знак вопроса в области кнопок управления окном.



Рисунок 14 - «Кнопка вызова меню «О программе»»

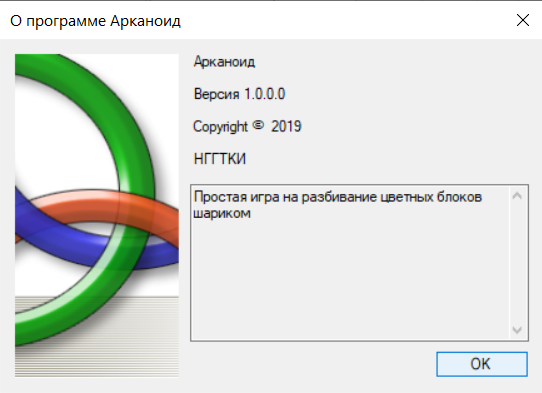


Рисунок 15 - «Кнопка вызова меню «О программе»»

В этой форме можно найти информацию о программе, и нехитрое описание игры.

# **ЗАКЛЮЧЕНИЕ**

В результате курсовой работы было проведено исследование предметной области, разработано техническое задание, выбрано и обосновано средство реализации БД, проанализирован прототип, проведен анализ и была спроектирована база данных. Так же выбрано и обосновано программное средство реализации, была разработана игра «Арканоид» с использованием СУБД MS SQL-server, изучена среда разработки, разработан интерфейс проекта с привлекательным дизайном, проведено тестирование и отладка программного обеспечения, произведен сбор данных об объектах и было разработано руководство пользователя.

Разработанный проект позволит пользователям системы играть в игру «Арканоид», сохранять и просматривать свои результаты.

В дальнейшем планируется провести более детальную проработку интерфейса приложения. Планируется улучшить работоспособность программы, добавить различные уровни сложности, разработать веб-интерфейс для программы для возможности игры на ее сайте.

Цель курсового проекта, а именно разработка игры «Арканоид» с использованием СУБД MSSQL-server достигнута.

Программа разработана, процессы, связанные с обработкой и информации и ее обменом с пользователем изучены, обработаны и исправно работают в данной программе. Регистрация игроков, просмотр и изменение их данных, все данные об игроках попадают в таблицы базы данных и с успехом извлекаются оттуда в необходимые окна разработанного приложения.

Цель достигнута, задачи выполнены.

# **СПИСОК ЛИТЕРАТУРЫ**

1. Васильев, Алексей C#. Объектно-ориентированное программирование / Алексей Васильев. - М.: Питер, 2012. - 320 c.
2. Подбельский, В. В. Язык С#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, 2013. - 408 c.
3. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 633 c
4. Вишневский, Алексей Microsoft SQL Server. Эффективная работа / Алексей Вишневский. - М.: Питер, 2009. - 143 c.
5. Ицик, Бен-Ган Microsoft SQL Server 2012. Высокопроизводительный код T-SQL. Оконные функции / Бен-Ган Ицик. - М.: Русская Редакция, 2013. - 861 c.

# **ПРИЛОЖЕНИЯ**

ПРИЛОЖЕНИЕ B

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9АРКАНОИД\_ФИНАЛ

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

///

//строка подключения к БД

public static string st\_connect = @"Data Source=VIKTPOR-PK\MSSQLSERVER1;Initial Catalog=ARCANOID;Integrated Security=True";

//переменная для хранения id авторизовавшегося пользователя

public static int id\_user;

//переменая для хранения типа пользователя, который сейчас авторизован

public static bool type\_user = false;

public static int id\_players;

public static int Id;

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new AUTHORISATION());

}

}

}

Форма авторизации

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.Data.SqlClient;

using System.IO;

using System.Diagnostics;

namespace WindowsFormsApp9АРКАНОИД\_ФИНАЛ

{

public partial class AUTHORISATION : Form

{

public AUTHORISATION()

{

InitializeComponent();

textBox1.Focus();

}

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

{

}

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

{

REGISTRSTION f = new REGISTRSTION();

f.Show();

textBox1.Enabled = true;

}

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

{

}

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

{

//получаем логин и пароль в переменные

//сразу чистим краевые пробелы (трим)

string login\_user = textBox1.Text.Trim();

string password\_user = textBox2.Text.Trim();

if (login\_user == "" || password\_user == "")

{

MessageBox.Show("Вы не ввели данные");

}

else

{

//создаем подключение к БД

SqlConnection connect = new SqlConnection(Program.st\_connect);

//открываем подключение

connect.Open();

//строка н поиск юзера

string s = "select \* from users where login\_user = '" + login\_user + "' and password\_user = '"

+ password\_user + "'";

//делаем запрос к БД

SqlCommand command = new SqlCommand(s, connect);

//получаем данные которые вернул запрос

SqlDataReader reader = command.ExecuteReader();

//читаем одну строку из набора данных

if (reader.HasRows)

{

reader.Read();

//получаем данные. В скобках это номера столбцов как в запросе(0-8)

Program.id\_user = reader.GetInt32(0);

Program.type\_user = reader.GetBoolean(3);

if (Program.type\_user == false)

{

USERS\_FORM f = new USERS\_FORM();

f.Show();

//this.Close();

}

else

{

Form3 f = new Form3();

f.Show();

}

}

else

{

MessageBox.Show("Пользователь не найден");

}

connect.Close();

}

this.Hide();

}

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

{

REGISTRSTION f = new REGISTRSTION();

f.Show();

textBox1.Enabled = true;

this.Hide();

}

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

{

}

private void timer1\_Tick(object sender, EventArgs e)

{

}

private void AUTHORISATION\_FormClosing(object sender, FormClosingEventArgs e)

{

//Application.Exit();

//this.Close();

//this.Hide();

Process[] List;

List = Process.GetProcessesByName("EXCEL");

foreach (Process proc in List)

{

proc.Kill();

}

}

private void AUTHORISATION\_FormClosed(object sender, FormClosedEventArgs e)

{

string name = "Арканоид";//процесс, который нужно убить

System.Diagnostics.Process[] etc = System.Diagnostics.Process.GetProcesses();//получим процессы

foreach (System.Diagnostics.Process anti in etc)//обойдем каждый процесс

if (anti.ProcessName.ToLower().Contains(name.ToLower())) anti.Kill();//найдем нужный и убьем

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

//кнопка показать скрытый пароль при авторизации

if (textBox2.PasswordChar == '\*')

{

textBox2.PasswordChar = '\0';

}

else

{

textBox2.PasswordChar = '\*';

}

}

}

}

Форма регистрации

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.Data.SqlClient;

using System.IO;

namespace WindowsFormsApp9АРКАНОИД\_ФИНАЛ

{

public partial class REGISTRSTION : Form

{

public string pathImage { get; set; }

public REGISTRSTION()

{

InitializeComponent();

}

void Registerr()

{

if(LoginBox.Text != string.Empty)

{

if(PasswordBox.Text != string.Empty)

{

SqlConnection connection = new SqlConnection(Program.st\_connect);

try

{

connection.Open();

SqlCommand command = new SqlCommand($"INSERT INTO[dbo].[USERS]" +

$"([login\_user],[password\_user],[type\_user],[photo\_user])" +

$"VALUES" +

$"('{LoginBox.Text}','{PasswordBox.Text}',0,'{pathImage}')");

command.Connection = connection;

int rows = command.ExecuteNonQuery();

if(rows > 0)

{

MessageBox.Show($"Добавлено {rows} пользователей");

}

}

catch (Exception e)

{

MessageBox.Show(e.Message);

}

}

}

}

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

{

Registerr();

this.Close();

AUTHORISATION f = new AUTHORISATION();

f.Show();

}

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

{

OpenFileDialog filedialog = new OpenFileDialog();

filedialog.Multiselect = false;

if(filedialog.ShowDialog() == DialogResult.OK)

{

File.Copy(

filedialog.FileName,

AppDomain.CurrentDomain.BaseDirectory + "/" + Path.GetFileName(filedialog.FileName)

);

pathImage = Path.GetFileName(filedialog.FileName);

pictureBox1.Image = Image.FromFile(filedialog.FileName);

}

}

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

{

Form1 f = new Form1();

f.Show();

this.Close();

}

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

{

}

}

}

Форма администратора

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;

namespace WindowsFormsApp9АРКАНОИД\_ФИНАЛ

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

}

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

{

Form6 f = new Form6();

f.Show();

this.Hide();

}

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

{

Form1 ff = new Form1();

ff.Show();

this.Hide();

}

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

{

Form2 fff = new Form2();

fff.Show();

this.Hide();

}

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

{

this.Close();

}

private void Form3\_FormClosing(object sender, FormClosingEventArgs e)

{

AUTHORISATION f = new AUTHORISATION();

f.Show();

}

}

}

Форма пользователя

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;

namespace WindowsFormsApp9АРКАНОИД\_ФИНАЛ

{

public partial class USERS\_FORM : Form

{

public USERS\_FORM()

{

InitializeComponent();

}

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

{

Form f = new Form2();

f.Show();

this.Hide();

}

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

{

this.Close();

}

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

{

Form1 f = new Form1();

f.Show();

this.Hide();

}

private void USERS\_FORM\_FormClosing(object sender, FormClosingEventArgs e)

{

AUTHORISATION f = new AUTHORISATION();

f.Show();

}

}

}

Форма изменения данных пользователя

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.Data.SqlClient;

using System.IO;

namespace WindowsFormsApp9АРКАНОИД\_ФИНАЛ

{

public partial class Form6 : Form

{

public string pathImage { get; set; }

public Form6()

{

InitializeComponent();

this.dataGridView1.DefaultCellStyle.Font = new Font("Century Gothic", 12);

this.dataGridView1.DefaultCellStyle.ForeColor = Color.RoyalBlue;

this.dataGridView1.DefaultCellStyle.BackColor = Color.Plum;

}

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

{

int k = dataGridView1.CurrentRow.Index;

Program.id\_user = Convert.ToInt32(dataGridView1[0, k].Value);//запоминаем id выбранного юзера

SqlConnection connect = new SqlConnection(Program.st\_connect);

connect.Open();

string s = "UPDATE USERS set " +

"login\_user='" + LoginBox.Text.Trim() + "', " +

"password\_user='" + PasswordBox.Text.Trim() + "'," +

"type\_user='" + comboBox1.SelectedIndex + "'," +

"photo\_user='" + label3.Text + "' " +

"WHERE id\_user = " + Program.id\_user.ToString();

SqlCommand comm = new SqlCommand(s, connect);

comm.ExecuteScalar();

connect.Close();

SqlConnection conn1 = new SqlConnection(Program.st\_connect);

conn1.Open();

string s1 = "SELECT \* FROM USERS";

SqlDataAdapter adap = new SqlDataAdapter(s1, conn1);

DataSet ds1 = new DataSet();

adap.Fill(ds1);

dataGridView1.DataSource = ds1.Tables[0];

if (ds1.Tables[0].Rows.Count >= 10)

{

button2.Enabled = true;

}

conn1.Close();

dataGridView1.Columns[0].Visible = false;

LoginBox.Text = "";

PasswordBox.Text = "";

comboBox1.Text = "";

MessageBox.Show("Изменено");

}

private void Form6\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "aRCANOIDDataSet.USERS". При необходимости она может быть перемещена или удалена.

this.uSERSTableAdapter.Fill(this.aRCANOIDDataSet.USERS);

dataGridView1.CurrentCell = dataGridView1[1, 0];

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

int k = dataGridView1.CurrentRow.Index;

Program.id\_user = Convert.ToInt32(dataGridView1[0, k].Value);

string s = "SELECT photo\_user FROM USERS WHERE Id\_user = " + Program.id\_user;

SqlCommand comm = new SqlCommand(s, conn);

SqlDataReader read = comm.ExecuteReader();

read.Read();

string s1 = read.GetString(0);

pictureBox1.Image = Image.FromFile(s1);

}

private void Form6\_FormClosed(object sender, FormClosedEventArgs e)

{

Form3 ff = new Form3();

ff.Show();

}

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

{

this.Close();

}

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

{

LoginBox.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

PasswordBox.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

if (Convert.ToBoolean(dataGridView1.CurrentRow.Cells[3].Value) ==true)

{

comboBox1.SelectedIndex = 1;

}

else

{

comboBox1.SelectedIndex = 0;

}

label3.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString();

PasswordBox.SelectionStart = 0;

PasswordBox.SelectionLength = LoginBox.Text.Length;

PasswordBox.Focus();

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

int k = dataGridView1.CurrentRow.Index;

Program.id\_user = Convert.ToInt32(dataGridView1[0, k].Value);

string s = "SELECT photo\_user FROM USERS WHERE Id\_user = " + Program.id\_user;

SqlCommand comm = new SqlCommand(s, conn);

SqlDataReader read = comm.ExecuteReader();

read.Read();

string s1 = read.GetString(0);

pictureBox1.Image = Image.FromFile(s1);

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

LoginBox.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

PasswordBox.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

if (Convert.ToBoolean(dataGridView1.CurrentRow.Cells[3].Value) == true)

{

comboBox1.SelectedIndex = 1;

}

else

{

comboBox1.SelectedIndex = 0;

}

label3.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString();

PasswordBox.SelectionStart = 0;

PasswordBox.SelectionLength = LoginBox.Text.Length;

PasswordBox.Focus();

}

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

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

label3.Text = Path.GetFileName(openFileDialog1.FileName);

File.Copy(openFileDialog1.FileName, AppDomain.CurrentDomain.BaseDirectory + label3.Text, true);//true = перезапись имени файла без ошибок

pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);

}

}

}

}

Форма изменения или просмотра статистки игрока

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.Data.SqlClient;

using Microsoft.Office.Interop.Excel;

using Excel = Microsoft.Office.Interop.Excel;

namespace WindowsFormsApp9АРКАНОИД\_ФИНАЛ

{

public partial class Form1 : Form

{

private object ARCANOIDDataSet;

int n = 0;

public Form1()

{

InitializeComponent();

this.Width = 628;

this.Height = 465;

this.dataGridView1.DefaultCellStyle. Font = new System.Drawing.Font(“Century Gothic”, 12);

this.dataGridView1.DefaultCellStyle.ForeColor = Color.Red;

this.dataGridView1.DefaultCellStyle.BackColor = Color.Wheat;

}

Private void Form1\_Load(object sender, EventArgs e)

{

radioButton3.Checked = true;

button2.Enabled = false;

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

//Делаем запрос к БД через адаптер, что бы потом поместить данные в DATASET

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

//создаем пустой датасет

DataSet ds = new DataSet();

//соединяем датасет с нашим набором данных

adap.Fill(ds);

//соединяем датасет с визуальным компонентом

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

}

else

{

button1.Enabled = false;

button2.Enabled = false;

}

conn.Close();

if (Program.type\_user == false)

{

button4.Visible = false;

button5.Visible = false;

}

else

{

button4.Visible = true;

button5.Visible = true;

}

dataGridView1.CurrentCell = dataGridView1[1, 0];

SqlConnection conn1 = new SqlConnection(Program.st\_connect);

conn1.Open();

string s1 = "SELECT login\_user FROM USERS WHERE id\_user= " + Program.id\_user;

//Делаем запрос к БД через адаптер, что бы потом поместить даные в DATASET

SqlCommand command = new SqlCommand(s1, conn1);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

radioButton4.Text = reader.GetString(0);

dataGridView1.Columns[0].Visible = false;

}

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

{

textBox2.SelectionStart = 0;

textBox2.SelectionLength = textBox2.Text.Length;

textBox2.Focus();

textBox1.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

textBox2.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

this.Width = 900;

this.Height = 465;

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

Microsoft.Office.Interop.Excel.Application myExcel = new Microsoft.Office.Interop.Excel.Application();

//создаем виртуальный объект Excel

//создать книгу в объекте Excel

myExcel.Application.Workbooks.Add(Type.Missing);

//Настраиваем ячейки

myExcel.Quit();

if (Program.type\_user == false)

{

USERS\_FORM f = new USERS\_FORM();

if (!f.Visible)

f.Show();

}

else

{

Form3 f = new Form3();

if (!f.Visible)

f.Show();

}

}

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

{

n = n - 10;

if (n <= 0)

{

button2.Enabled = false;

}

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

if (radioButton3.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

//button2.Enabled = false;

}

else

{

button1.Enabled = false;

//button2.Enabled = true;

}

}

else if (radioButton2.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY data\_igra DESC OFFSET " + n.ToString() + " ROWS " +

"FETCH NEXT 10 ROWS ONLY ";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

//button2.Enabled = false;

}

else

{

button1.Enabled = false;

//button2.Enabled = true;

}

}

else if (radioButton1.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

//button2.Enabled = false;

}

else

{

button1.Enabled = false;

//button2.Enabled = true;

}

}

conn.Close();

dataGridView1.Columns[0].Visible = false;

}

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

{

if (n <= 0)

{

button2.Enabled = false;

}

n = n + 10;

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

if (radioButton3.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

//button1.Enabled = true;

button2.Enabled = false;

}

else

{

//button1.Enabled = false;

button2.Enabled = true;

}

}

else if (radioButton2.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY data\_igra DESC OFFSET " + n.ToString() + " ROWS " +

"FETCH NEXT 10 ROWS ONLY ";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

//button1.Enabled = true;

button2.Enabled = false;

}

else

{

//button1.Enabled = false;

button2.Enabled = true;

}

}

else if (radioButton1.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

//button1.Enabled = true;

button2.Enabled = false;

}

else

{

//button1.Enabled = false;

button2.Enabled = true;

}

}

conn.Close();

dataGridView1.Columns[0].Visible = false;

}

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

{

int k = dataGridView1.CurrentRow.Index;

Program.id\_user = Convert.ToInt32(dataGridView1[0, k].Value);

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

string s = "DELETE FROM PLAYERS WHERE id=" + Program.id\_user.ToString();

SqlCommand comm = new SqlCommand(s, conn);

comm.ExecuteScalar();

conn.Close();

MessageBox.Show("Чел удален");

n = n + 10;

button2.Enabled = true;

SqlConnection conn1 = new SqlConnection(Program.st\_connect);

conn1.Open();

string s1 = "SELECT Id ,Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

SqlDataAdapter adap = new SqlDataAdapter(s1, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

}

else

{

button1.Enabled = false;

}

conn1.Close();

dataGridView1.Columns[0].Visible = false;

}

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

{

try

{

if (textBox1.Text.Trim() == "" ||//проверка на пустоту

textBox2.Text.Trim() == "")

{

MessageBox.Show("Не все поля заполнены");

}

else

{

int k = dataGridView1.CurrentRow.Index;

Program.Id = Convert.ToInt32(dataGridView1[0, k].Value);//запоминаем id выбранного юзера

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

string s = "UPDATE PLAYERS " +

" SET " + "Name\_players= " + "'" + textBox1.Text.Trim() + "'," +

"score = " + "'" + textBox2.Text.Trim() + "'," +

"data\_igra = " + "'" + dateTimePicker1.Value.ToShortDateString() + "'," +

"vremya\_igra = " + "'" + dateTimePicker2.Value.ToShortTimeString() + "' WHERE Id=" + Program.Id.ToString();

SqlCommand comm = new SqlCommand(s, conn);

comm.ExecuteScalar();

conn.Close();

SqlConnection conn1 = new SqlConnection(Program.st\_connect);

conn1.Open();

string s1 = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

SqlDataAdapter adap = new SqlDataAdapter(s1, conn);

DataSet ds1 = new DataSet();

adap.Fill(ds1);

dataGridView1.DataSource = ds1.Tables[0];

if (ds1.Tables[0].Rows.Count >= 10)

{

button2.Enabled = true;

}

conn1.Close();

dataGridView1.Columns[0].Visible = false;

}

}

catch

{

MessageBox.Show("Что то пошло не так");

}

this.Width = 628;

this.Height = 465;

}

private void button3\_Click\_1(object sender, EventArgs e)

{

this.Close();

}

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

{

textBox1.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

textBox2.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

textBox2.SelectionStart = 0;

textBox2.SelectionLength = textBox2.Text.Length;

textBox2.Focus();

}

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

{

{

Microsoft.Office.Interop.Excel.Application myExcel = new Microsoft.Office.Interop.Excel.Application();

myExcel.Application.Workbooks.Add(Type.Missing);

DateTime dt = DateTime.Now;

string curDate = dt.ToShortDateString();

myExcel.ActiveSheet.Name = "Отчет за " + curDate;

//Настраиваем ячейки

myExcel.Columns.ColumnWidth = 20;

//Пишем заголовки ячеек

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

myExcel.Cells[1, i + 1] = dataGridView1.Columns[i].HeaderText;// заголовки программно

}

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

for (int j = 0; j < dataGridView1.RowCount - 1; j++)

{

myExcel.Cells[j + 2, i + 1] = dataGridView1[i, j].Value.ToString();

}

}

//скрываем столбец с id (столбец A)

Range range = (Range)myExcel.Columns[1, Type.Missing];

range.EntireColumn.Hidden = true;

myExcel.Visible = true;

}

}

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

{

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

if (radioButton3.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

button2.Enabled = false;

}

else

{

button1.Enabled = false;

button2.Enabled = true;

}

}

else if (radioButton2.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY data\_igra DESC OFFSET " + n.ToString() + "ROWS " +

"FETCH NEXT 10 ROWS ONLY ";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

button2.Enabled = false;

}

else

{

button1.Enabled = false;

button2.Enabled = true;

}

}

else if (radioButton1.Checked)

{

string s = "SELECT Id, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS ORDER BY id OFFSET " + n.ToString() + " ROWS FETCH NEXT 10 ROWS ONLY";

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

if (ds.Tables[0].Rows.Count >= 10)

{

button1.Enabled = true;

button2.Enabled = false;

}

else

{

button1.Enabled = false;

button2.Enabled = true;

}

}

else if (radioButton4.Checked)

{

SqlConnection connect = new SqlConnection(Program.st\_connect);

string s = "SELECT Id\_user, Name\_players as 'Имя игрока', score as 'Счёт', data\_igra as 'Дата', " +

"vremya\_igra as 'Время' FROM PLAYERS INNER JOIN USERS on " +

"PLAYERS.id\_players=USERS.id\_user WHERE Id\_user= " + Program.id\_user.ToString();

SqlDataAdapter adap = new SqlDataAdapter(s, conn);

DataSet ds = new DataSet();

adap.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

button2.Enabled = false;

button1.Enabled = false;

}

conn.Close();

dataGridView1.Columns[0].Visible = false;

}

}

}