**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Научный руководитель,  доцент департамента  программной инженерии факультета компьютерных наук,  канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Дударев  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **МАТЕМАТИЧЕСКАЯ ВИКТОРИНА ДЛЯ ДВУХ ИГРОКОВ «МАТЕМАТИЧЕСКАЯ ДУЭЛЬ»**  **ТЕКСТ ПРОГРАММЫ**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.01-01 12 01-1-ЛУ** | | |
|  |  | |
| Исполнитель  студент группы БПИ 186  \_\_\_\_\_\_\_\_\_\_\_\_ / М.А. Семенов /  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | |
|  | | |
|  | |  |

**Москва 2019**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.04.01-01 12 01-1-ЛУ |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **МАТЕМАТИЧЕСКАЯ ВИКТОРИНА ДЛЯ ДВУХ ИГРОКОВ «МАТЕМАТИЧЕСКАЯ ДУЭЛЬ»**  **Текст программы**  **RU.17701729.04.01-01 12 01-1**  **Листов 66** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

**Москва 2019**

ОГЛАВЛЕНИЕ

[1. ТЕКСТ ПРОГРАММЫ 4](#_Toc9292699)

[1.1. Серверная часть приложения 4](#_Toc9292700)

[1.1.1. Класс ServerObject.cs 4](#_Toc9292701)

[1.1.2. Класс ClientObject.cs 16](#_Toc9292702)

[1.1.3. Класс Game.cs 24](#_Toc9292703)

[1.1.4. Класс Program.cs 25](#_Toc9292704)

[1.2. Клиентская часть приложения 25](#_Toc9292705)

[1.2.1. Класс User.cs 25](#_Toc9292706)

[1.2.2. Класс CurrentUser.cs 31](#_Toc9292707)

[1.2.3. Класс AcceptPage.xaml.cs 31](#_Toc9292708)

[1.2.4. AcceptPage.xaml 32](#_Toc9292709)

[1.2.5. Класс App.xaml.cs 34](#_Toc9292710)

[1.2.6. App.xaml 34](#_Toc9292711)

[1.2.7. Класс ChooseAnswers.xaml.cs 35](#_Toc9292712)

[1.2.8. ChooseAnswers.xaml 36](#_Toc9292713)

[1.2.9. Класс ExitPage.xaml.cs 37](#_Toc9292714)

[1.2.10. ExitPage.xaml 37](#_Toc9292715)

[1.2.11. Класс GetActiveUsers.xaml.cs 38](#_Toc9292716)

[1.2.12 GetActiveUsers.xaml 39](#_Toc9292717)

[1.2.13 Класс GetStatistic.xaml.cs 39](#_Toc9292718)

[1.2.14 GetStatistic.xaml 40](#_Toc9292719)

[1.2.15 Класс MainMenu.xaml.cs 41](#_Toc9292720)

[1.2.16 MainMenu.xaml 43](#_Toc9292721)

[1.2.17 Класс MainMenuDetail.xaml.cs 44](#_Toc9292722)

[1.2.18 MainMenuDetail.xaml 45](#_Toc9292723)

[1.2.19 Класс MainMenuMaster.xaml.cs 46](#_Toc9292724)

[1.2.20 MainMenuMaster.xaml 47](#_Toc9292725)

[1.2.21 Класс MainMenuMenuItem.cs 48](#_Toc9292726)

[1.2.22 Класс MainPage.xaml.cs 48](#_Toc9292727)

[1.2.23 MainPage.xaml 51](#_Toc9292728)

[1.2.24 Класс PlayGamePage.xaml.cs 53](#_Toc9292729)

[1.2.25 PlayGamePage.xaml 53](#_Toc9292730)

[1.2.26 Класс ShowErPage.xaml.cs 54](#_Toc9292731)

[1.2.27 ShowErPage.xaml 55](#_Toc9292732)

[1.2.28 Класс ShowResaltPage.xaml.cs 55](#_Toc9292733)

[1.2.29 ShowResaltPage.xaml 58](#_Toc9292734)

[1.2.30 Класс TasksPage.xaml.cs 59](#_Toc9292735)

[1.2.31 TasksPage.xaml 62](#_Toc9292736)

# ТЕКСТ ПРОГРАММЫ

## Серверная часть приложения

### Класс ServerObject.cs

public class ServerObject

{

static TcpListener tcpListener; // сервер для прослушивания

public List<ClientObject> clients = new List<ClientObject>(); // все подключения

const string ip = "172.20.10.10";

const string connectionString = @"Data Source=LAPTOP-OORPDL13\SQLEXPRESS;Initial Catalog=MathQuiz;Integrated Security=True";

static Random rnd = new Random();

public List<Game> games = new List<Game>();

protected internal void AddConnection(ClientObject clientObject)

{

clients.Add(clientObject);

}

protected internal void RemoveConnection(string id)

{

// получаем по id закрытое подключение

ClientObject client = clients.FirstOrDefault(c => c.Id == id);

if (client != null)

{

Game g = games.FirstOrDefault(c => c.player1 == client || c.player2 == client);

if(g != null)

{

ClientObject opon = g.player1 == client ? g.player2 : g.player1;

if(opon!= null)

{

CallBack("opex", opon);

RemoveInGame(opon.GetName());

games.Remove(g);

DeleteGame(g.idGame);

g = null;

}

}

}

// и удаляем его из списка подключений

if (client != null)

clients.Remove(client);

}

// прослушивание входящих подключений

protected internal void Listen()

{

try

{

tcpListener = new TcpListener(IPAddress.Parse(ip), 8888);

tcpListener.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

while (true)

{

TcpClient tcpClient = tcpListener.AcceptTcpClient();

ClientObject clientObject = new ClientObject(tcpClient, this);

Thread clientThread = new Thread(new ThreadStart(clientObject.Process));

clientThread.Start();

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

Disconnect();

}

}

// отключение всех клиентов

protected internal void Disconnect()

{

tcpListener.Stop(); //остановка сервера

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

{

clients[i].Close(); //отключение клиента

}

Environment.Exit(0); //завершение процесса

}

protected internal bool SignIn(string msg, string id)

{

string[] s = msg.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlDataReader sqlReader = null;

SqlCommand command = new SqlCommand("SELECT \* FROM [Users] WHERE [Nickname] = @Nickname", sqlConnection);

command.Parameters.AddWithValue("Nickname", s[0]);

try

{

sqlReader = command.ExecuteReader();

if (sqlReader.Read()&& sqlReader["Password"].ToString() == s[1]&& sqlReader["IsActive"].ToString() == "False")

{

sqlReader.Close();

SqlCommand update = new SqlCommand("UPDATE [Users] SET [IsActive] = @IsActive WHERE [Nickname] = @Nickname", sqlConnection);

update.Parameters.AddWithValue("Nickname", s[0]);

update.Parameters.AddWithValue("IsActive", true);

update.ExecuteNonQuery();

return true;

}

else

{

return false;

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return false;

}

finally

{

if (sqlReader != null)

{

sqlReader.Close();

}

if (sqlConnection != null)

{

sqlConnection.Close();

}

}

}

//

protected internal bool SignUp(string msg, string id)

{

string[] s = msg.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlDataReader sqlReader = null;

SqlCommand command = new SqlCommand("SELECT \* FROM [Users] WHERE [Nickname] = @Nickname", sqlConnection);

command.Parameters.AddWithValue("Nickname", s[0]);

sqlReader = command.ExecuteReader();

int count = 1;

try

{

if (!sqlReader.Read())

{

if (sqlReader != null)

{

sqlReader.Close();

}

command = new SqlCommand("SELECT \* FROM [Users]", sqlConnection);

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

count += 1;

}

if (sqlReader != null)

{

sqlReader.Close();

}

SqlCommand insertUser = new SqlCommand("INSERT [Users] (Id\_User, Nickname, Password, InGame, IsActive)VALUES(@Id\_User,@Nickname, @Password, @InGame, @IsActive)", sqlConnection);

insertUser.Parameters.AddWithValue("Nickname", s[0]);

insertUser.Parameters.AddWithValue("Password", s[1]);

insertUser.Parameters.AddWithValue("InGame", false);

insertUser.Parameters.AddWithValue("IsActive", true);

insertUser.Parameters.AddWithValue("Id\_User", count);

insertUser.ExecuteNonQuery();

return true;

}

else

{

return false;

}

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

return false;

}

finally

{

if (sqlReader != null)

{

sqlReader.Close();

}

}

}

//отправляет клиенту ответ от сервера

protected internal void CallBack(string message, ClientObject client)

{

try

{

byte[] data = Encoding.Unicode.GetBytes(message);

if (client != null)

{

client.Stream.Write(data, 0, data.Length);

}

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

}

}

//снимает активность пользователя в базе данных

protected internal void RemoveActive(string msg)

{

string[] s = msg.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

try

{

SqlCommand update = new SqlCommand("UPDATE [Users] SET [IsActive] = @IsActive WHERE [Nickname] = @Nickname", sqlConnection);

update.Parameters.AddWithValue("Nickname", s[0]);

update.Parameters.AddWithValue("IsActive", false);

update.ExecuteNonQuery();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

protected internal string GetActiveUsers()

{

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlDataReader sqlReader = null;

SqlDataReader sqlReaderOpon = null;

SqlCommand command = new SqlCommand("SELECT \* FROM [Users] WHERE [IsActive] = @IsActive", sqlConnection);

command.Parameters.AddWithValue("IsActive", true);

sqlReader = command.ExecuteReader();

string s = "AU";

try

{

while (sqlReader.Read())

{

s += sqlReader["Nickname"].ToString() + " ";

}

return s;

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

return "-10";

}

finally

{

if (sqlReader != null)

{

sqlReader.Close();

}

if (sqlReaderOpon != null)

{

sqlReaderOpon.Close();

}

if (sqlConnection != null)

{

sqlConnection.Close();

}

}

}

protected internal string GetStatistic(string nickname)

{

int counttasks = 0;

int countcortasks = 0;

using (SqlConnection sqlConnection = new SqlConnection(connectionString))

{

sqlConnection.Open();

SqlDataReader sqlReader = null;

SqlCommand command = new SqlCommand("SELECT s. \* FROM [Statistic] as s JOIN [Tasks] AS t ON s.Id\_Task = t.Id\_Task WHERE s.Id\_User = @Id\_User", sqlConnection);

command.Parameters.AddWithValue("@Id\_User", GetIdUser(nickname));

using (sqlReader = command.ExecuteReader())

{

while (sqlReader.Read())

{

counttasks++;

}

}

}

if(counttasks == 0)

{

return $"ST{counttasks} {countcortasks}";

}

using (SqlConnection sqlConnection = new SqlConnection(connectionString))

{

sqlConnection.Open();

SqlDataReader sqlReader = null;

SqlCommand command = new SqlCommand("SELECT s. \* FROM [Statistic] as s JOIN[Tasks] AS t ON s.User\_Answer = t.Answer AND s.Id\_Task = t.Id\_Task WHERE s.Id\_User = @Id\_User", sqlConnection);

command.Parameters.AddWithValue("@Id\_User", GetIdUser(nickname));

using (sqlReader = command.ExecuteReader())

{

while (sqlReader.Read())

{

countcortasks++;

}

}

}

return $"ST{counttasks} {countcortasks}";

}

protected internal bool PossibleInvite(string usName)

{

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlDataReader sqlReader = null;

SqlCommand command = new SqlCommand("SELECT \* FROM [Users] WHERE [Nickname] = @Nickname", sqlConnection);

command.Parameters.AddWithValue("Nickname", usName);

sqlReader = command.ExecuteReader();

try

{

while (sqlReader.Read())

{

if (sqlReader["IsActive"].ToString() == "True" && sqlReader["InGame"].ToString() == "False")

{

return true;

}

else

return false;

}

return false;

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

return false;

}

finally

{

if (sqlReader != null)

{

sqlReader.Close();

}

if (sqlConnection != null)

{

sqlConnection.Close();

}

}

}

protected internal void GetTasks(string s, Game game)

{

List<string> tasks = new List<string>();

List<string> answers = new List<string>();

List<string> IdTasks = new List<string>();

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlDataReader sqlReader = null;

SqlCommand command = new SqlCommand("SELECT \* FROM [Tasks] WHERE [Id\_Type] = @Id\_Type", sqlConnection);

command.Parameters.AddWithValue("Id\_Type", s);

try

{

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

tasks.Add(sqlReader["Task"].ToString());

answers.Add(sqlReader["Answer"].ToString());

IdTasks.Add(sqlReader["Id\_Task"].ToString());

}

int[] numTasks = new int[5];

numTasks = GetNumbers("", tasks.Count);

string[] tasks1 = new string[5];

string[] answ1 = new string[5];

string[] IdTask1 = new string[5];

for (int i = 0; i < 5; i++)

{

int num = numTasks[i];

tasks1[i] = tasks[num];

answ1[i] = answers[num];

IdTask1[i] = IdTasks[num];

}

game.tasks = tasks1;

game.answ = answ1;

game.idTasks = IdTask1;

game.player1.idTasks = IdTask1;

game.player2.idTasks = IdTask1;

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

if (sqlReader != null)

{

sqlReader.Close();

}

if (sqlConnection != null)

{

sqlConnection.Close();

}

}

}

public void SetInGame(string nikcname)

{

List<string> tasks = new List<string>();

List<string> answers = new List<string>();

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

try

{

SqlCommand update = new SqlCommand("UPDATE [Users] SET [InGame] = @InGame WHERE [Nickname] = @Nickname", sqlConnection);

update.Parameters.AddWithValue("Nickname", nikcname);

update.Parameters.AddWithValue("InGame", true);

update.ExecuteNonQuery();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

if (sqlConnection != null)

{

sqlConnection.Close();

}

}

}

public void RemoveInGame(string nikcname)

{

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

List<string> tasks = new List<string>();

List<string> answers = new List<string>();

SqlCommand update = new SqlCommand("UPDATE [Users] SET [InGame] = @InGame WHERE [Nickname] = @Nickname", sqlConnection);

update.Parameters.AddWithValue("Nickname", nikcname);

update.Parameters.AddWithValue("InGame", false);

update.ExecuteNonQuery();

if (sqlConnection != null)

{

sqlConnection.Close();

}

}

public void AddStatistic(string nick, string[] idT, string[] answ)

{

string id = GetIdUser(nick);

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

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

{

try

{

SqlCommand insertUser = new SqlCommand("INSERT [Statistic] (Id\_User, Id\_Task, User\_Answer)VALUES(@Id\_User,@Id\_Task, @User\_Answer)", sqlConnection);

insertUser.Parameters.AddWithValue("Id\_User", id);

insertUser.Parameters.AddWithValue("Id\_Task", idT[i]);

insertUser.Parameters.AddWithValue("User\_Answer", answ[i]);

insertUser.ExecuteNonQuery();

}

catch

{

SqlCommand update = new SqlCommand("UPDATE [Statistic] SET [User\_Answer] = @User\_Answer WHERE [Id\_User] = @Id\_User AND [Id\_Task] = @Id\_Task", sqlConnection);

update.Parameters.AddWithValue("Id\_User", id);

update.Parameters.AddWithValue("Id\_Task", idT[i]);

update.Parameters.AddWithValue("User\_Answer", answ[i]);

update.ExecuteNonQuery();

}

}

if (sqlConnection != null)

{

sqlConnection.Close();

}

}

public string GetIdUser(string nickname)

{

string id = "";

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlDataReader sqlReader = null;

SqlCommand command = new SqlCommand("SELECT \* FROM [Users] WHERE [Nickname] = @Nickname", sqlConnection);

command.Parameters.AddWithValue("Nickname", nickname);

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

id = sqlReader["Id\_User"].ToString();

}

if (sqlReader != null)

{

sqlReader.Close();

}

if (sqlConnection != null)

{

sqlConnection.Close();

}

return id;

}

protected internal void AddGame(string nickname1, string nickname2, string idGame)

{

string id1 = GetIdUser(nickname1);

string id2 = GetIdUser(nickname2);

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand insertUser = new SqlCommand("INSERT [Games] (Id\_Game, Id\_Player1, Id\_Player2)VALUES(@Id\_Game,@Id\_Player1, @Id\_Player2)", sqlConnection);

insertUser.Parameters.AddWithValue("Id\_Player1", id1);

insertUser.Parameters.AddWithValue("Id\_Player2", id2);

insertUser.Parameters.AddWithValue("Id\_Game", idGame);

insertUser.ExecuteNonQuery();

if (sqlConnection != null)

{

sqlConnection.Close();

}

}

protected internal void DeleteGame(string idGame)

{

SqlConnection sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

try

{

SqlCommand insertUser = new SqlCommand("DELETE FROM [Games] WHERE Id\_Game = @Id\_Game", sqlConnection);

insertUser.Parameters.AddWithValue("Id\_Game", idGame);

insertUser.ExecuteNonQuery();

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

if (sqlConnection != null)

{

sqlConnection.Close();

}

}

}

private int[] GetNumbers(string s, int count)

{

int k = 0;

int[] numTasks = new int[5];

while (k != 5)

{

int randNum = rnd.Next(0,count);

if (!s.Contains(randNum.ToString()))

{

numTasks[k] = randNum;

k++;

s += randNum.ToString() + " ";

}

}

return numTasks;

}

}

### Класс ClientObject.cs

public class ClientObject

{

protected internal string Id { get; private set; }

protected internal NetworkStream Stream { get; private set; }

public string userName;

public string acception = "0";

public int numinGame = 0;

public string[] answers = null;

public string[] idTasks = null;

TcpClient client;

ServerObject server;// объект сервера

public ClientObject(TcpClient tcpClient, ServerObject serverObject)

{

Id = Guid.NewGuid().ToString();

client = tcpClient;

server = serverObject;

}

public void Process()

{

try

{

bool flag = false;

do

{

Stream = client.GetStream();

// получаем имя пользователя

string message = GetMessage();

if (message.Length >= 1)

{

userName = message.Substring(1);

}

else

{

return;

}

if (message[0] == '2')

{

if (server.SignUp(userName, this.Id))

{

server.SignIn(userName, this.Id);

message = "0";

flag = true;

Console.WriteLine($"User {userName} enter in game");

}

else

{

message = "-1";

}

}

else if (message[0] == '1')

{

if (server.SignIn(userName, this.Id))

{

message = "0";

flag = true;

Console.WriteLine($"User {userName} enter in game");

}

else

{

message = "-1";

}

}

else

{

message = "not correct data";

}

// посылаем сообщение о входе

server.CallBack(message, this);

}

while (!flag);

server.AddConnection(this);

// в бесконечном цикле получаем сообщения от клиента

while (true)

{

int k = 0;

try

{

string message = GetMessage();

string msg = String.Format("{0}: {1}", userName, message);

if (message != "")

{

Console.WriteLine(msg);

}

if(message == "exit")

{

Close();

}

if(message == "" && k<1)

{

message = String.Format("{0}: leave game 1", userName);

Console.WriteLine(message);

k++;

k = 0;

break;

}

//parsing messages

//работа с получением списка активных пользователей

if (message == "3")

{

string s = server.GetActiveUsers();

server.CallBack(s, this);

}

//получаем статистику пользователя

else if(message == "4")

{

string[] str = userName.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

string s = server.GetStatistic(str[0]);

server.CallBack(s, this);

}

//приглашаем опонентав игру

else if(message.Length>1 && message[0] == '5')

{

//отправляем пользователю приглашение

string uName = message.Substring(1);

bool ans = server.PossibleInvite(uName);

if (ans)

{

ClientObject client = server.clients.FirstOrDefault(c => c.GetName() == uName);

if (client==null)

{

server.CallBack("-2", this);

}

else

{

if (client != null)

{

InviteInGame(client);

}

else

server.CallBack("-2", this);

}

}

else

{

server.CallBack("-2", this);

}

}

else if (message.Contains("type"))

{

//получаем список задач заданного типа

string s = message[4].ToString();

Game game = server.games.FirstOrDefault(c => c.player1 == this || c.player2 == this);

if(game!= null)

{

server.GetTasks(s, game);

game.SendTasks();

}

else

{

server.CallBack("er", this);

}

}

//принятие приглашения

else if (message.Contains("acp"))

{

AcceptInvating(message);

}

//отправляем ответы противникy

else if (message.Length>2 && message[0] == 'U' && message[1] == 'A')

{

Game game = server.games.FirstOrDefault(c => c.player1 == this || c.player2 == this);

if (game != null)

{

ClientObject player = game.player1 == this ? game.player2 : game.player1;

server.CallBack(message, player);

}

}

//добавляем ответы в статистику

else if (message.Length > 2 && message[0] == 'M' && message[1] == 'A')

{

string[] UsAns = ParseAnsw(message);

server.AddStatistic(this.GetName(),idTasks, UsAns);

}

//удаляем поля InGame и Game

else if(message == "remGame")

{

server.RemoveInGame(GetName());

try

{

Game g = server.games.FirstOrDefault(c => c.player1 == this || c.player2 == this);

if (g != null)

{

server.games.Remove(g);

server.DeleteGame(g.idGame);

g = null;

}

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

string message = String.Format("{0}: leave game 2", userName);

Console.WriteLine(message);

server.CallBack("exit", this);

Close();

break;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

// в случае выхода из цикла закрываем ресурсы

server.RemoveConnection(this.Id);

Close();

}

}

// чтение входящего сообщения и преобразование в строку

private string GetMessage()

{

byte[] data = new byte[64]; // буфер для получаемых данных

StringBuilder builder = new StringBuilder();

int bytes = 0;

do

{

bytes = Stream.Read(data, 0, data.Length);

builder.Append(Encoding.Unicode.GetString(data, 0, bytes));

}

while (Stream.DataAvailable);

return builder.ToString();

}

// закрытие подключения

protected internal void Close()

{

if (userName != string.Empty && userName != null)

{

server.RemoveInGame(GetName());

server.RemoveActive(userName);

}

if (Stream != null)

Stream.Close();

if (client != null)

client.Close();

}

protected internal void InviteInGame(ClientObject client)

{

string message = "inv" + " " + GetName();

server.CallBack(message, client);

while (client.acception == "0")

{

Thread.Sleep(20);

}

if (client.acception == "1")

{

//создаем комнату для игроков

client.acception = "0";

server.games.Add(new Game(this, client, server, (server.games.Count+1).ToString()));

server.CallBack("-4", this);

}

if (client.acception == "-1")

{

//отправляем пригласившему отказ

client.acception = "0";

server.CallBack("-3", this);

}

//пользователь не ожидает игры

if(client.acception == "2")

{

client.acception = "0";

server.CallBack("-5", this);

}

}

protected internal void AcceptInvating(string message)

{

if (message[3] == '1')

{

acception = "1";

}

else if(message[3]=='3')

{

acception = "2";

}

else

{

acception = "-1";

}

}

protected internal string GetName()

{

string[] s = userName.Split(new char[] { ' ' });

return s[0];

}

protected internal string[] ParseAnsw(string s)

{

s = s.Substring(2);

string[] ans = s.Split(new string[] {"//"}, StringSplitOptions.RemoveEmptyEntries);

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

{

double d;

if(!double.TryParse(ans[i], out d))

{

ans[i] = "-1";

}

else

{

ans[i] = d.ToString();

}

}

string[] ans1 = new string[5];

for (int i = 0; i < 5; i++)

{

ans1[i] = ans[i+1];

}

return ans1;

}

}

### Класс Game.cs

public class Game

{

public ClientObject player1 = null;

public ServerObject server = null;

public ClientObject player2 = null;

public string[] tasks = new string[5];

public string[] answ = new string[5];

public string[] idTasks = new string[5];

public string idGame = "";

public Game(ClientObject client1,ClientObject client2, ServerObject server, string idGame)

{

player1 = client1;

player1.numinGame = 1;

player2 = client2;

player2.numinGame = 2;

this.server = server;

this.idGame = idGame;

//добавить в БД информацию что пользователи в игре

server.SetInGame(player1.GetName());

server.SetInGame(player2.GetName());

server.AddGame(player1.GetName(), player2.GetName(), idGame);

}

public void SendTasks()

{

string tasksStr = "t/asks";

for (int i = 0; i < tasks.Length-1; i++)

{

tasksStr += tasks[i] + "//";

}

tasksStr += tasks[tasks.Length - 1];

string answersStr = "";

for (int i = 0; i < answ.Length - 1; i++)

{

answersStr += answ[i] + "//";

}

answersStr += answ[answ.Length - 1];

string comMsg = tasksStr + "\_\_" + answersStr;

server.CallBack(comMsg, player1);

server.CallBack(comMsg, player2);

}

}

### Класс Program.cs

class Program

{

static ServerObject server; // сервер

static Thread listenThread; // потока для прослушивания

static void Main(string[] args)

{

try

{

server = new ServerObject();

listenThread = new Thread(new ThreadStart(server.Listen));

listenThread.Start(); //старт потока

}

catch (Exception ex)

{

server.Disconnect();

Console.WriteLine(ex.Message);

}

}

}

## Клиентская часть приложения

### Класс User.cs

public class User

{

public string userName;

private const string host = "172.20.10.10";

private const int port = 8888;

public string opponenet = "";

public string iswait = "";

public string opflag = "";

public string ActUsers = "";

public string stat = "";

public string[] tasks = null;

public string[] answers = null;

public string[] useranswers = new string[5];

public bool flag = false;

public string s = "";

static TcpClient client;

static NetworkStream stream;

public User()

{

client = new TcpClient();

client.Connect(host, port); //подключение клиента

stream = client.GetStream(); // получаем поток

Thread receiveThread = new Thread(new ThreadStart(ReceiveMessage));

receiveThread.Start();//старт потока

}

~User()

{

Disconnect();

}

// отправка сообщений

public void SendMessage(string message)

{

try

{

if (message != "")

{

byte[] data = Encoding.Unicode.GetBytes(message);

stream.Write(data, 0, data.Length);

}

}

catch

{

}

}

// получение сообщений

public void ReceiveMessage()

{

while (true)

{

try

{

byte[] data = new byte[64]; // буфер для получаемых данных

StringBuilder builder = new StringBuilder();

int bytes = 0;

do

{

bytes = stream.Read(data, 0, data.Length);

builder.Append(Encoding.Unicode.GetString(data, 0, bytes));

}

while (stream.DataAvailable);

string message = builder.ToString();

if (message == "-1")

{

flag = false;

s = "NO";

}

else if (message == "0")

{

flag = true;

s = "OK";

}

if (message == "-10")

{

s = "error";

}

else if(message[0] == 'A' && message.Length>=2 && message[1] == 'U')//принимает активных пользоваелей

{

s = "OKAU";

ActUsers = message.Substring(2);

}

else if(message[0] == 'S' && message.Length >= 2 && message[1] == 'T')//принимает статистику

{

s = "OKST";

stat = message.Substring(2);

}

//работа с приглашением пользователей

else if(message == "-2")

{

CurrentUser.currentUser.opflag = message;

}

else if (message == "-3")

{

CurrentUser.currentUser.opflag = message;

}

else if(message == "-5")

{

CurrentUser.currentUser.opflag = message;

}

else if (message == "-4")

{

CurrentUser.currentUser.opflag = message;

}

else if(message == "er")

{

App.Current.MainPage = new MainPage();

}

else if (message.Contains("inv"))

{

if (CurrentUser.IsWaiing == true && CurrentUser.currentUser.iswait == "")

{

iswait = message;

}

else

{

CurrentUser.currentUser.SendMessage("acp3");

}

}

else if(message == "exit")

{

Disconnect();

}

else if (message.Contains("t/asks"))

{

TransformTasks(message);

App.Current.MainPage = new TasksPage(1, tasks[0]);

//показать окно с вопросами

}

else if (message.Contains("opex"))

{

opponenet = "";

iswait = "";

opflag = "";

tasks = null;

answers = null;

CurrentUser.OponEx = true;

App.Current.MainPage = new ShowErPage();

}

else if (message.Contains("UA"))

{

ParseAnsw(message.Substring(2));

CurrentUser.GetOponAnsw = true;

}

}

catch

{

Disconnect();

}

}

}

public void Disconnect()

{

SendMessage("exit");

if (stream != null)

stream.Close();//отключение потока

if (client != null)

client.Close();//отключение клиента

System.Diagnostics.Process.GetCurrentProcess().CloseMainWindow(); //завершение процесса

}

public void SignIn(string message)

{

message = "1" + message;

byte[] data = Encoding.Unicode.GetBytes(message);

stream.Write(data, 0, data.Length);

}

public void SignUp(string message)

{

message = "2" + message;

byte[] data = Encoding.Unicode.GetBytes(message);

stream.Write(data, 0, data.Length);

}

public void GetActiveusers()

{

string message = "3";

byte[] data = Encoding.Unicode.GetBytes(message);

stream.Write(data, 0, data.Length);

}

public void GetStatistic()

{

string message = "4";

byte[] data = Encoding.Unicode.GetBytes(message);

stream.Write(data, 0, data.Length);

}

public void InviteUser(string s)

{

string message = "5"+s;

byte[] data = Encoding.Unicode.GetBytes(message);

stream.Write(data, 0, data.Length);

}

public void AcceptionInvating(string s)

{

string message = "acp" + s;

byte[] data = Encoding.Unicode.GetBytes(message);

stream.Write(data, 0, data.Length);

}

public void TransformTasks(string message)

{

message = message.Substring(6);

string[] sep = {"\_\_"};

string[] s = message.Split(sep,StringSplitOptions.RemoveEmptyEntries);

sep[0] = "//";

answers = s[1].Split(sep, StringSplitOptions.RemoveEmptyEntries);

tasks = s[0].Split(sep, StringSplitOptions.RemoveEmptyEntries);

}

public string[] ParseAnsw(string answ)

{

string[] s = answ.Split(new string[] { "//" }, StringSplitOptions.RemoveEmptyEntries);

CurrentUser.OponAnsw = s;

return s;

}

}

### Класс CurrentUser.cs

public class CurrentUser

{

public static User currentUser = null;

public static bool IsWaiing = false;

public static string[] OponAnsw = null;

public static bool GetOponAnsw = false;

public static bool OponEx = false;

}

### Класс AcceptPage.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class AcceptPage : ContentPage

{

public AcceptPage (string usName)

{

InitializeComponent ();

NavigationPage.SetHasNavigationBar(this, false);

InviteLabel.Text = $"user {usName} invite you";

CurrentUser.currentUser.opponenet = usName;

}

public void Yes\_Click(object sender, EventArgs e)

{

WaitLab.Text = "wait while your opponent chose type of tasks";

CurrentUser.currentUser.SendMessage("acp1");

if (WaitMethod())

{

App.Current.MainPage = new ShowErPage();

return;

}

//App.Current.MainPage = new TasksPage(1, CurrentUser.currentUser.tasks[0]);

}

public void No\_Click(object sender, EventArgs e)

{

CurrentUser.currentUser.opponenet = "";

CurrentUser.currentUser.iswait = "";

CurrentUser.currentUser.opflag = "";

CurrentUser.currentUser.tasks = null;

CurrentUser.currentUser.answers = null;

CurrentUser.IsWaiing = false;

CurrentUser.OponAnsw = null;

CurrentUser.GetOponAnsw = false;

CurrentUser.OponEx = false;

CurrentUser.currentUser.SendMessage("acp0");

Navigation.PushAsync(new MainMenu());

}

private bool WaitMethod()

{

while (CurrentUser.currentUser.tasks == null)

{

if (CurrentUser.OponEx)

{

CurrentUser.OponEx = false;

return true;

}

Thread.Sleep(20);

}

return false;

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

}

### AcceptPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.AcceptPage"

BackgroundColor="LightBlue">

<ContentPage.Content>

<StackLayout>

<Label Text=""

HeightRequest="70"

/>

<Label Text=""

x:Name="InviteLabel"

FontSize="30"

TextColor="DarkBlue"

FontAttributes="Bold"

VerticalOptions="Center"

HorizontalOptions="Center"

Margin="20"

/>

<Label Text="Do you want play game?"

FontSize="30"

TextColor="Yellow"

FontAttributes="Bold"

VerticalOptions="Center"

HorizontalOptions="Center"

Margin="10"/>

<Button Text="Yes"

Clicked="Yes\_Click"

VerticalOptions="Center"

HorizontalOptions="Center"

BackgroundColor="DarkBlue"

TextColor="White"

HeightRequest="50"

WidthRequest="100"/>

<Label Text=""

FontSize="20"

x:Name="WaitLab"

TextColor="Yellow"

HeightRequest="70"

/>

<Button Text="No"

Clicked="No\_Click"

VerticalOptions="Center"

HorizontalOptions="Center"

TextColor="White"

BackgroundColor="DarkBlue"

HeightRequest="50"

WidthRequest="100"/>

<Label Text=""

TextColor="Red"

VerticalOptions="Center"

HorizontalOptions="Center"/>

</StackLayout>

</ContentPage.Content>

</ContentPage>

### Класс App.xaml.cs

public partial class App : Application

{

public App()

{

InitializeComponent();

MainPage = new NavigationPage(new MainPage());

}

protected override void OnStart()

{

// Handle when your app starts

}

protected override void OnSleep()

{

// Handle when your app sleeps

}

protected override void OnResume()

{

// Handle when your app resumes

}

}

### App.xaml

<?xml version="1.0" encoding="utf-8" ?>

<Application xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.App">

<Application.Resources>

</Application.Resources>

</Application>

### Класс ChooseAnswers.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ChooseAnswers : ContentPage

{

int k = 0;

public ChooseAnswers()

{

NavigationPage.SetHasNavigationBar(this, false);

InitializeComponent();

ChooseComplexyty.Toggled += ChooseComplexyty\_Toggled;

}

private void ChooseComplexyty\_Toggled(object sender, ToggledEventArgs e)

{

k = (k + 1) % 2;

}

public void Algebra\_click(object sender, EventArgs e)

{

CurrentUser.currentUser.SendMessage($"type{1 + k}");

//ShowTasks();

}

public void Geom\_click(object sender, EventArgs e)

{

CurrentUser.currentUser.SendMessage($"type{3 + k}");

ShowTasks();

}

public void Comb\_click(object sender, EventArgs e)

{

CurrentUser.currentUser.SendMessage($"type{5 + k}");

ShowTasks();

}

public void ShowTasks()

{

while (CurrentUser.currentUser.tasks == null)

{

Thread.Sleep(50);

}

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### ChooseAnswers.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.ChooseAnswers" BackgroundColor="LightBlue">

<ContentPage.Content>

<StackLayout>

<Label Text="Complexyty tasks?"

TextColor="DarkBlue"

FontSize="20"

HorizontalOptions="Center"

VerticalOptions="EndAndExpand"

HeightRequest="30"/>

<Switch x:Name="ChooseComplexyty"

OnColor="Yellow"

HorizontalOptions="Center"

VerticalOptions="Center"/>

<Label Text="Choose topic of five questions?"

FontSize="20"

TextColor="DarkBlue"

HorizontalOptions="Center"

/>

<Button Text="Algebra"

BackgroundColor="DarkBlue"

VerticalOptions="Center"

Clicked="Algebra\_click"

TextColor="White"

HorizontalOptions="Center"

WidthRequest="150"

Margin="25"

HeightRequest="50"/>

<Button Text="Geometry"

BackgroundColor="DarkBlue"

VerticalOptions="Center"

TextColor="White"

Clicked="Geom\_click"

HorizontalOptions="Center"

WidthRequest="150"

Margin="12"

HeightRequest="50"/>

<Button Text="Combinatorics"

BackgroundColor="DarkBlue"

VerticalOptions="Center"

Clicked="Comb\_click"

TextColor="White"

HorizontalOptions="Center"

WidthRequest="150"

Margin="25"

HeightRequest="50"/>

</StackLayout>

</ContentPage.Content>

</ContentPage>

### Класс ExitPage.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ExitPage : ContentPage

{

public ExitPage ()

{

InitializeComponent ();

}

public void Yes\_click(object sender, EventArgs e)

{

CurrentUser.currentUser.Disconnect();

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### ExitPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.ExitPage"

BackgroundColor="LightBlue">

<ContentPage.Content>

<StackLayout>

<Label Text=""

TextColor="Red"

HeightRequest="70"

/>

<Label Text="Are you sure?"

FontSize="30"

TextColor="DarkBlue"

FontAttributes="Bold"

VerticalOptions="Center"

HorizontalOptions="End"

HeightRequest="170"

WidthRequest="270"/>

<Button Text="Yes"

Clicked="Yes\_click"

VerticalOptions="Center"

HorizontalOptions="Center"

BackgroundColor="DarkBlue"

TextColor="White"

HeightRequest="50"

WidthRequest="100"/>

<Label Text=""

x:Name="SignUpInfo"

TextColor="Red"

VerticalOptions="Center"

HorizontalOptions="Center"/>

</StackLayout>

</ContentPage.Content>

</ContentPage>

### Класс GetActiveUsers.xaml.cs

public partial class GetActiveUsers : ContentPage

{

public ObservableCollection<string> Items { get; set; }

public GetActiveUsers()

{

Show();

}

public void Show()

{

List<string> collect = new List<string>();

InitializeComponent();

string[] s1 = CurrentUser.currentUser.ActUsers.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

MyListView.ItemsSource = s1;

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### GetActiveUsers.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.GetActiveUsers"

BackgroundColor="White">

<ListView x:Name="MyListView"

BackgroundColor="LightBlue"

SeparatorColor="DarkBlue"

ItemsSource="{Binding Items}"

CachingStrategy="RecycleElement">

<!--Built in Cells-->

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding .}"

TextColor="DarkBlue"/>

</DataTemplate>

</ListView.ItemTemplate>

<!--Custom View Cells-->

<!--

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<StackLayout>

<Label Text="{Binding Text}"

Style="{DynamicResource ListItemTextStyle}" />

<Label Text="{Binding Detail}"

Style="{DynamicResource ListItemDetailTextStyle}"/>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

-->

</ListView>

</ContentPage>

### Класс GetStatistic.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class GetStatistic : ContentPage

{

public GetStatistic ()

{

InitializeComponent ();

}

public void Show()

{

string[] s = CurrentUser.currentUser.stat.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

Answers.Text = s[0];

CorrectAns.Text = s[1];

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### GetStatistic.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.GetStatistic"

BackgroundColor="LightBlue">

<ContentPage.Content>

<StackLayout>

<Label

Text="you answered on"

FontSize="20"

HeightRequest="30"

TextColor="DarkBlue"

WidthRequest="250"

VerticalOptions="Center"

HorizontalOptions="Center" />

<Label

x:Name="Answers"

Text="0"

FontSize="40"

HeightRequest="50"

TextColor="Yellow"

WidthRequest="250"

VerticalOptions="Center"

HorizontalOptions="Center" />

<Label

Text="tasks"

FontSize="20"

HeightRequest="30"

TextColor="DarkBlue"

WidthRequest="250"

VerticalOptions="Center"

HorizontalOptions="Center" />

<Label

Text="you answered correct on"

FontSize="20"

HeightRequest="30"

TextColor="DarkBlue"

WidthRequest="250"

VerticalOptions="Center"

HorizontalOptions="Center" />

<Label

x:Name="CorrectAns"

Text="0"

FontSize="40"

HeightRequest="50"

TextColor="Yellow"

WidthRequest="250"

VerticalOptions="Center"

HorizontalOptions="Center" />

<Label

Text="tasks"

FontSize="20"

HeightRequest="30"

TextColor="DarkBlue"

WidthRequest="250"

VerticalOptions="Center"

HorizontalOptions="Center" />

</StackLayout>

</ContentPage.Content>

</ContentPage>

### Класс MainMenu.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class MainMenu : MasterDetailPage

{

public MainMenu()

{

InitializeComponent();

NavigationPage.SetHasNavigationBar(this, false);

MasterPage.ListView.ItemSelected += ListView\_ItemSelected;

}

private void ListView\_ItemSelected(object sender, SelectedItemChangedEventArgs e)

{

var item = e.SelectedItem as MainMenuMenuItem;

if (item == null)

return;

//invite player

if (item.Id == 0)

{

var page = (Page)Activator.CreateInstance(item.TargetType);

page.Title = item.Title;

NavigationPage np = new NavigationPage(page);

np.BarBackgroundColor = Color.DarkBlue;

np.BarTextColor = Color.White;

Detail = np;

IsPresented = false;

}

//get statistic

else if(item.Id == 1)

{

CurrentUser.currentUser.stat = "";

CurrentUser.currentUser.GetStatistic();

while (CurrentUser.currentUser.stat == "")

{

Thread.Sleep(50);

}

GetStatistic page = new GetStatistic();

page.Show();

page.Title = item.Title;

NavigationPage np = new NavigationPage(page);

np.BarBackgroundColor = Color.DarkBlue;

np.BarTextColor = Color.White;

Detail = np;

IsPresented = false;

}

//get active users

else if (item.Id == 2)

{

CurrentUser.currentUser.ActUsers = "";

CurrentUser.currentUser.GetActiveusers();

while (CurrentUser.currentUser.ActUsers == "")

{

Thread.Sleep(50);

}

Page p = new GetActiveUsers();

p.Title = item.Title;

NavigationPage np = new NavigationPage(p);

np.BarBackgroundColor = Color.DarkBlue;

np.BarTextColor = Color.White;

Detail = np;

IsPresented = false;

}

//exit

else if (item.Id == 3)

{

var page = (Page)Activator.CreateInstance(item.TargetType);

page.Title = item.Title;

NavigationPage np = new NavigationPage(page);

np.BarBackgroundColor = Color.DarkBlue;

np.BarTextColor = Color.White;

Detail = np;

IsPresented = false;

}

else if (item.Id == 4)

{

Page page = new PlayGamePage();

page.Title = item.Title;

NavigationPage np = new NavigationPage(page);

np.BarBackgroundColor = Color.DarkBlue;

np.BarTextColor = Color.White;

Detail = np;

IsPresented = false;

}

MasterPage.ListView.SelectedItem = null;

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### MainMenu.xaml

<?xml version="1.0" encoding="utf-8" ?>

<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.MainMenu"

xmlns:pages="clr-namespace:MathDuel"

BackgroundColor="LightBlue">

<MasterDetailPage.Master>

<pages:MainMenuMaster x:Name="MasterPage" />

</MasterDetailPage.Master>

<MasterDetailPage.Detail>

<NavigationPage>

<x:Arguments>

<pages:MainMenuDetail />

</x:Arguments>

</NavigationPage>

</MasterDetailPage.Detail>

</MasterDetailPage>

### Класс MainMenuDetail.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class MainMenuDetail : ContentPage

{

public MainMenuDetail()

{

InitializeComponent();

}

public void Invite\_click(object sender, EventArgs e)

{

string opname = InputOponnentNick.Text == null ? "" : InputOponnentNick.Text;

if(opname == CurrentUser.currentUser.userName)

{

IviteLabel.Text = "you can not invite yourself";

return;

}

else if(opname == "")

{

IviteLabel.Text = "nickname can nor be empty";

return;

}

else if(opname.Contains(" ") || opname.Contains("/"))

{

IviteLabel.Text = "nickname can nor contain space and / symbols";

return;

}

else

{

CurrentUser.currentUser.InviteUser(opname);

CurrentUser.currentUser.opponenet = opname;

CurrentUser.currentUser.opflag = "";

while (CurrentUser.currentUser.opflag == "")

{

Thread.Sleep(50);

}

if(CurrentUser.currentUser.opflag == "-2")

{

IviteLabel.Text = "impossible invite this user in game";

}

else if(CurrentUser.currentUser.opflag == "-3")

{

IviteLabel.Text = "opponent decline your inviting";

}

else if (CurrentUser.currentUser.opflag == "-5")

{

IviteLabel.Text = "opponent do not wait game";

}

else

{

Navigation.PushAsync(new ChooseAnswers());

}

}

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### MainMenuDetail.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.MainMenuDetail"

BackgroundColor="LightBlue"

Title="Inviting">

<StackLayout Padding="10">

<Label Text=""

WidthRequest="100"

HeightRequest="100"/>

<Entry Placeholder="Input Nickname of oponnent"

x:Name="InputOponnentNick"

BackgroundColor="White"/>

<Label Text=""

x:Name="IviteLabel"

TextColor="Red"

WidthRequest="200"

HeightRequest="100"

HorizontalOptions="Center"

VerticalOptions="Center"/>

<Button Text="Invite"

Clicked="Invite\_click"

BackgroundColor="DarkBlue"

TextColor="White"

VerticalOptions="Center"

HorizontalOptions="Center"

WidthRequest="80"

HeightRequest="40"/>

</StackLayout>

</ContentPage>

### Класс MainMenuMaster.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class MainMenuMaster : ContentPage

{

public ListView ListView;

public MainMenuMaster()

{

InitializeComponent();

BindingContext = new MainMenuMasterViewModel();

ListView = MenuItemsListView;

}

class MainMenuMasterViewModel : INotifyPropertyChanged

{

public ObservableCollection<MainMenuMenuItem> MenuItems { get; set; }

public MainMenuMasterViewModel()

{

MenuItems = new ObservableCollection<MainMenuMenuItem>(new[]

{

new MainMenuMenuItem { Id = 0, Title = "Inviting" },

new MainMenuMenuItem { Id = 1, Title = "Statistic"},

new MainMenuMenuItem { Id = 2, Title = "ActiveUsers",TargetType = typeof(GetActiveUsers)},

new MainMenuMenuItem { Id = 4, Title = "PlayGame", TargetType = typeof(ExitPage)},

new MainMenuMenuItem { Id = 3, Title = "Exit", TargetType = typeof(ExitPage)},

});

}

#region INotifyPropertyChanged Implementation

public event PropertyChangedEventHandler PropertyChanged;

void OnPropertyChanged([CallerMemberName] string propertyName = "")

{

if (PropertyChanged == null)

return;

PropertyChanged.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

#endregion

}

### MainMenuMaster.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.MainMenuMaster"

BackgroundColor="LightBlue"

Title="Master">

<StackLayout>

<ListView x:Name="MenuItemsListView"

SeparatorVisibility="None"

HasUnevenRows="true"

ItemsSource="{Binding MenuItems}">

<ListView.Header>

<Grid BackgroundColor="DarkBlue">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="10"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="10"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="30"/>

<RowDefinition Height="80"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="10"/>

</Grid.RowDefinitions>

<Label

Grid.Column="1"

Grid.Row="2"

Text="MathDuel"

FontSize="25"

TextColor="White"

Style="{DynamicResource SubtitleStyle}"/>

</Grid>

</ListView.Header>

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<StackLayout Padding="15,10" HorizontalOptions="FillAndExpand">

<Label VerticalOptions="FillAndExpand"

VerticalTextAlignment="Center"

Text="{Binding Title}"

BackgroundColor="LightBlue"

TextColor="DarkBlue"

FontSize="24"/>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

</ContentPage>

### Класс MainMenuMenuItem.cs

public class MainMenuMenuItem

{

public MainMenuMenuItem()

{

TargetType = typeof(MainMenuDetail);

}

public int Id { get; set; }

public string Title { get; set; }

public Type TargetType { get; set; }

}

### Класс MainPage.xaml.cs

public partial class MainPage : ContentPage

{

public MainPage()

{

NavigationPage.SetHasNavigationBar(this, false);

InitializeComponent();

}

public void SignIn\_Click(object sender, EventArgs e)

{

SignUpInfo.Text = "";

SignInInfo.Text = "";

User testUser = null;

string nickname = InputNickname.Text == null ? "" : InputNickname.Text;

string password = InputPassword.Text == null ? "" : InputPassword.Text; ;

if (IsCorrectNick(nickname) != "OK")

{

SignInInfo.Text = IsCorrectNick(nickname);

return;

}

if (IsCorrectPassword(password) != "OK")

{

SignInInfo.Text = IsCorrectPassword(password);

return;

}

try

{

testUser = new User();

}

catch (Exception ex)

{

string st = ex.Message;

st += "";

}

testUser.SignIn(InputNickname.Text + " " + InputPassword.Text);

while (testUser.s == "")

{

Thread.Sleep(20);

}

if (testUser.flag)

{

CurrentUser.currentUser = testUser;

CurrentUser.currentUser.userName = nickname;

SignInInfo.Text = "you enter in game";

Navigation.PushAsync(new MainMenu());

}

else

{

SignInInfo.Text = "not correct data";

}

}

public void SignUp\_Click(object sender, EventArgs e)

{

SignUpInfo.Text = "";

SignInInfo.Text = "";

User testUser = null;

string nickname = InputNickname.Text == null?"": InputNickname.Text;

string password = InputPassword.Text == null ? "" : InputPassword.Text; ;

if (IsCorrectNick(nickname) != "OK")

{

SignUpInfo.Text = IsCorrectNick(nickname);

return;

}

if ( IsCorrectPassword(password) != "OK")

{

SignUpInfo.Text = IsCorrectPassword(password);

return;

}

try

{

testUser = new User();

}

catch (Exception ex)

{

string st = ex.Message;

st += "";

}

testUser.SignUp(InputNickname.Text + " " + InputPassword.Text);

while (testUser.s == "")

{

Thread.Sleep(20);

}

if (testUser.flag)

{

CurrentUser.currentUser = testUser;

CurrentUser.currentUser.userName = nickname;

SignUpInfo.Text = "you SignUp and enter in game";

Navigation.PushAsync(new MainMenu());

}

else

{

SignUpInfo.Text = "that user already exist";

}

}

public string IsCorrectNick(string message)

{

if (message.Length > 8)

{

return "nickname must contein less than 8 symbols";

}

if (message.Contains(" ")|| message.Contains("/"))

{

return "nicknew can not contain space sympols and / symbol";

}

if (message == "")

{

return "Nickname can not be empty";

}

else

{

return "OK";

}

}

public string IsCorrectPassword(string message)

{

if (message.Length > 12)

{

return "password must contein less than 12 symbols";

}

if (message.Contains(" "))

{

return "password can not contain space symbols";

}

if (message == "")

{

return "password can not be empty";

}

else

{

return "OK";

}

}

}

### MainPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:MathDuel"

BackgroundColor="LightBlue"

x:Class="MathDuel.MainPage">

<StackLayout>

<Label Text="Math Duel"

FontSize="45"

TextColor="DarkBlue"

FontAttributes="Bold"

HorizontalOptions="Center"

WidthRequest="125"/>

<Entry Placeholder="Input your Nickname"

x:Name="InputNickname"

BackgroundColor="White"

HeightRequest="40"

WidthRequest="125"/>

<Label Text="" HeightRequest="30"/>

<Entry Placeholder="Input your Password"

x:Name="InputPassword"

BackgroundColor="White"/>

<Button Text="Sign In"

Clicked="SignIn\_Click"

VerticalOptions="Center"

HorizontalOptions="Center"

TextColor="White"

BackgroundColor="DarkBlue"

HeightRequest="50"

WidthRequest="100"/>

<Label Text=""

x:Name="SignInInfo"

TextColor="Red"

HeightRequest="30"

VerticalOptions="Center"

HorizontalOptions="Center"/>

<Button Text="Sign Up"

Clicked="SignUp\_Click"

TextColor="White"

VerticalOptions="Center"

HorizontalOptions="Center"

BackgroundColor="DarkBlue"

HeightRequest="50"

WidthRequest="100"/>

<Label Text=""

x:Name="SignUpInfo"

TextColor="Red"

VerticalOptions="Center"

HorizontalOptions="Center"/>

</StackLayout>

</ContentPage>

### Класс PlayGamePage.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class PlayGamePage : ContentPage

{

public PlayGamePage()

{

InitializeComponent();

}

public void WaitBut\_Click(object sender, EventArgs e)

{

CurrentUser.IsWaiing = true;

int k = 0;

while(CurrentUser.currentUser.iswait == "" && k!=1000)

{

Thread.Sleep(20);

k++;

}

if (k == 1000 && CurrentUser.currentUser.iswait=="")

{

//выводим информациюна экран

CurrentUser.IsWaiing = false;

PlayGameInfo.Text = "you was not invaited";

}

else

{

string[] s = CurrentUser.currentUser.iswait.Split(new char[] { ' ' });

Navigation.PushAsync(new AcceptPage(s[1]));

}

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### PlayGamePage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.PlayGamePage"

BackgroundColor="LightBlue">

<ContentPage.Content>

<StackLayout>

<Label Text=""

TextColor="Red"

HeightRequest="70"

/>

<Label Text="Are you ready?"

FontSize="30"

TextColor="DarkBlue"

FontAttributes="Bold"

VerticalOptions="Center"

HorizontalOptions="End"

HeightRequest="170"

WidthRequest="270"/>

<Button Text="Yes"

Clicked="WaitBut\_Click"

VerticalOptions="Center"

HorizontalOptions="Center"

BackgroundColor="DarkBlue"

TextColor="White"

HeightRequest="50"

WidthRequest="100"/>

<Label Text=""

x:Name="PlayGameInfo"

TextColor="Red"

VerticalOptions="Center"

HorizontalOptions="Center"/>

</StackLayout>

</ContentPage.Content>

</ContentPage>

### Класс ShowErPage.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ShowErPage : ContentPage

{

public ShowErPage ()

{

InitializeComponent ();

NavigationPage.SetHasNavigationBar(this, false);

}

public void Ok\_click(object sender, EventArgs e)

{

App.Current.MainPage = new MainMenu();

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### ShowErPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.ShowErPage" BackgroundColor="LightBlue">

<ContentPage.Content>

<StackLayout>

<Label Text="your opponent close the application"

Margin="40"

TextColor="DarkBlue"

HorizontalOptions="CenterAndExpand"

FontSize="25"/>

<Button

Text="OK"

BackgroundColor="DarkBlue"

TextColor="White"

HorizontalOptions="Center"

WidthRequest="60"

Clicked="Ok\_click"

/>

</StackLayout>

</ContentPage.Content>

</ContentPage>

### Класс ShowResaltPage.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ShowResaltPage : ContentPage

{

public ObservableCollection<string> Items { get; set; }

int yourCount = 0;

int countoponnent = 0;

public ShowResaltPage(string[] opansw)

{

InitializeComponent();

Title = "Result";

Items = new ObservableCollection<string>

{

"Task №",

"1",

"2",

"3",

"4",

"5",

};

string[] answ = new string[6];

answ[0] = "Right Answers";

for (int i = 1; i < 6; i++)

{

answ[i] = CurrentUser.currentUser.answers[i - 1];

}

MyListView2.ItemsSource = GetArrAnsw(CurrentUser.currentUser.useranswers, CurrentUser.currentUser.userName);

MyListView4.ItemsSource = answ;

MyListView1.ItemsSource = Items;

MyListView3.ItemsSource = GetArrAnsw(opansw, CurrentUser.currentUser.opponenet);

GetWinner();

}

public void OK\_Click(object sender, EventArgs e)

{

CurrentUser.currentUser.opponenet = "";

CurrentUser.currentUser.iswait = "";

CurrentUser.currentUser.opflag = "";

CurrentUser.IsWaiing = false;

CurrentUser.OponAnsw = null;

CurrentUser.GetOponAnsw = false;

CurrentUser.OponEx = false;

CurrentUser.currentUser.tasks = null;

CurrentUser.currentUser.answers = null;

CurrentUser.currentUser.SendMessage("remGame");

App.Current.MainPage = new MainMenu();

//отправить результаты на сервер

}

private string[] GetArrAnsw(string[] answers, string name)

{

string[] coransw = new string[answers.Length + 1];

coransw[0] = name;

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

{

double d;

if(!double.TryParse(answers[i], out d))

{

coransw[i + 1] = "incorrect";

}

else

{

coransw[i + 1] = d.ToString();

if(double.Parse(CurrentUser.currentUser.answers[i]) == d)

{

if(name == CurrentUser.currentUser.userName)

{

yourCount += 1;

}

else

{

countoponnent += 1;

}

}

}

}

//отправить ответы и добавить в статистику

if (name == CurrentUser.currentUser.userName)

{

string s = "MA";

for (int i = 0; i < coransw.Length-1; i++)

{

s += coransw[i].ToString() + "//";

}

s += coransw[coransw.Length - 1].ToString();

CurrentUser.currentUser.SendMessage(s);

}

return coransw;

}

private void GetWinner()

{

if(yourCount == countoponnent)

{

WinLabel.Text = "DRAW";

}

else if(yourCount > countoponnent)

{

WinLabel.Text = "YOU WIN!";

}

else

{

WinLabel.Text = "YOUR OPPONENT WIN";

}

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### ShowResaltPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.ShowResaltPage" BackgroundColor="LightBlue">

<StackLayout>

<StackLayout Orientation="Horizontal" >

<ListView x:Name="MyListView1"

ItemsSource="{Binding Items}"

CachingStrategy="RecycleElement"

BackgroundColor="White"

SeparatorColor="DarkBlue"

>

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding .}" TextColor="Black"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<ListView x:Name="MyListView4"

ItemsSource="{Binding Items}"

CachingStrategy="RecycleElement"

BackgroundColor="White"

SeparatorColor="DarkBlue"

>

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding .}" TextColor="Black"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<ListView x:Name="MyListView2"

ItemsSource="{Binding Items}"

CachingStrategy="RecycleElement"

SeparatorColor="DarkBlue"

BackgroundColor="White">

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding .}" TextColor="Black"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<ListView x:Name="MyListView3"

ItemsSource="{Binding Items}"

CachingStrategy="RecycleElement"

BackgroundColor="White"

SeparatorColor="DarkBlue"

>

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding .}" TextColor="Black" />

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

<Label x:Name="WinLabel"

TextColor="Red"

FontSize="20"

HorizontalOptions="Center"

/>

<Button x:Name="ButOk"

Text="OK"

Clicked="OK\_Click"

Margin="100"

TextColor="White"

BackgroundColor="DarkBlue"

VerticalOptions="Center"

HorizontalOptions="Center"

WidthRequest="50"/>

</StackLayout>

</ContentPage>

### Класс TasksPage.xaml.cs

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class TasksPage : ContentPage

{

public TasksPage(int number, string task)

{

InitializeComponent();

NavigationPage.SetHasNavigationBar(this, false);

NumberTask.Text = number.ToString();

TaskLabel.Text = task;

}

public void But\_Click(object sender, EventArgs e)

{

Button but = (Button)sender;

if(but.Text == "del")

{

AnswerLab.Text = DelLastSymbol(AnswerLab.Text);

}

else if(but.Text == "Enter")

{

if (CurrentUser.OponEx)

{

CurrentUser.currentUser.opponenet = "";

CurrentUser.currentUser.iswait = "";

CurrentUser.currentUser.opflag = "";

CurrentUser.currentUser.tasks = null;

CurrentUser.currentUser.answers = null;

CurrentUser.IsWaiing = false;

CurrentUser.OponAnsw = null;

CurrentUser.GetOponAnsw = false;

CurrentUser.OponEx = false;

App.Current.MainPage = new ShowErPage();

}

if (int.Parse(NumberTask.Text) == 5)

{

if (AnswerLab.Text == "")

{

CurrentUser.currentUser.useranswers[int.Parse(NumberTask.Text) - 1] = "-";

TaskLabel.Text = CurrentUser.currentUser.tasks[int.Parse(NumberTask.Text) - 1];

}

else

{

CurrentUser.currentUser.useranswers[int.Parse(NumberTask.Text) - 1] = AnswerLab.Text;

CurrentUser.currentUser.SendMessage(AnswToStr());

}

//сравниваем результаты и выбрасыывем другую старницу

while (!CurrentUser.GetOponAnsw)

{

Thread.Sleep(20);

if (CurrentUser.OponEx)

{

CurrentUser.OponEx = false;

App.Current.MainPage = new ShowErPage();

return;

}

}

App.Current.MainPage = new ShowResaltPage(CurrentUser.OponAnsw);

return;

}

else

{

if(AnswerLab.Text == "")

{

CurrentUser.currentUser.useranswers[int.Parse(NumberTask.Text) - 1] = "-";

NumberTask.Text = (int.Parse(NumberTask.Text) + 1).ToString();

TaskLabel.Text = CurrentUser.currentUser.tasks[int.Parse(NumberTask.Text) - 1];

}

else

{

CurrentUser.currentUser.useranswers[int.Parse(NumberTask.Text) - 1] = AnswerLab.Text;

NumberTask.Text = (int.Parse(NumberTask.Text) + 1).ToString();

TaskLabel.Text = CurrentUser.currentUser.tasks[int.Parse(NumberTask.Text) - 1];

}

AnswerLab.Text = "";

}

}

else if (AnswerLab.Text.Length<6)

{

AnswerLab.Text += but.Text;

}

}

private string DelLastSymbol(string s)

{

string str = "";

for (int i = 0; i < s.Length-1; i++)

{

str += s[i];

}

return str;

}

private string AnswToStr()

{

string s = "UA";

for (int i = 0; i < CurrentUser.currentUser.useranswers.Length-1; i++)

{

s += CurrentUser.currentUser.useranswers[i] + "//";

}

s += CurrentUser.currentUser.useranswers[CurrentUser.currentUser.useranswers.Length - 1];

return s;

}

protected override bool OnBackButtonPressed()

{

base.OnBackButtonPressed();

return true;

}

}

### TasksPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MathDuel.TasksPage" BackgroundColor="LightBlue">

<ContentPage.Content>

<StackLayout>

<Label

Text="1"

FontSize="20"

x:Name="NumberTask"

TextColor="Gold"

HorizontalOptions="Center"

/>

<Label x:Name="Timerlabel"

TextColor="WhiteSmoke"

HorizontalOptions="Center"

FontSize="12"

/>

<Label

x:Name="TaskLabel"

Text=""

FontSize="18"

HorizontalOptions="Center"

TextColor="Black"

BackgroundColor="White"

/>

<Label

Text="input your answer"

HorizontalOptions="Center"

TextColor="DarkBlue"/>

<Label

Text=""

x:Name="AnswerLab"

BackgroundColor="White"

TextColor="Black"

HeightRequest="30"/>

<StackLayout Spacing="8" Orientation="Horizontal" HorizontalOptions="Center">

<Button x:Name="but1" BackgroundColor="DarkBlue" Text="1" Clicked="But\_Click" TextColor="White"/>

<Button x:Name="but2" BackgroundColor="DarkBlue" Text="2" Clicked="But\_Click" TextColor="White"/>

<Button x:Name="but3" BackgroundColor="DarkBlue" Text="3" Clicked="But\_Click" TextColor="White"/>

</StackLayout>

<StackLayout Spacing="8" Orientation="Horizontal" HorizontalOptions="Center">

<Button x:Name="but4" BackgroundColor="DarkBlue" Text="4" Clicked="But\_Click" TextColor="White"/>

<Button x:Name="but5" BackgroundColor="DarkBlue" Text="5" Clicked="But\_Click" TextColor="White"/>

<Button x:Name="but6" BackgroundColor="DarkBlue" Text="6" Clicked="But\_Click" TextColor="White"/>

</StackLayout>

<StackLayout Spacing="8" Orientation="Horizontal" HorizontalOptions="Center">

<Button x:Name="but7" BackgroundColor="DarkBlue" Text="7" Clicked="But\_Click" TextColor="White"/>

<Button x:Name="but8" BackgroundColor="DarkBlue" Text="8" Clicked="But\_Click" TextColor="White"/>

<Button x:Name="but9" BackgroundColor="DarkBlue" Text="9" Clicked="But\_Click" TextColor="White"/>

</StackLayout>

<StackLayout Spacing="8" Orientation="Horizontal" HorizontalOptions="Center">

<Button x:Name="but0" BackgroundColor="DarkBlue" Text="0" Clicked="But\_Click" TextColor="White"/>

<Button x:Name="butComma" BackgroundColor="DarkBlue" Text="." Clicked="But\_Click" TextColor="White"/>

</StackLayout>

<StackLayout Spacing="8" Orientation="Horizontal" HorizontalOptions="Center">

<Button x:Name="del" BackgroundColor="DarkBlue" Text="del" Clicked="But\_Click" TextColor="White"/>

<Button x:Name="Enter" BackgroundColor="Gold" Text="Enter" Clicked="But\_Click" TextColor="Red"/>

</StackLayout>

</StackLayout>

</ContentPage.Content>

</ContentPage>

# 

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |