ПРИЛОЖЕНИЕ 1



Рис.1.1

ПРИЛОЖЕНИЕ 2

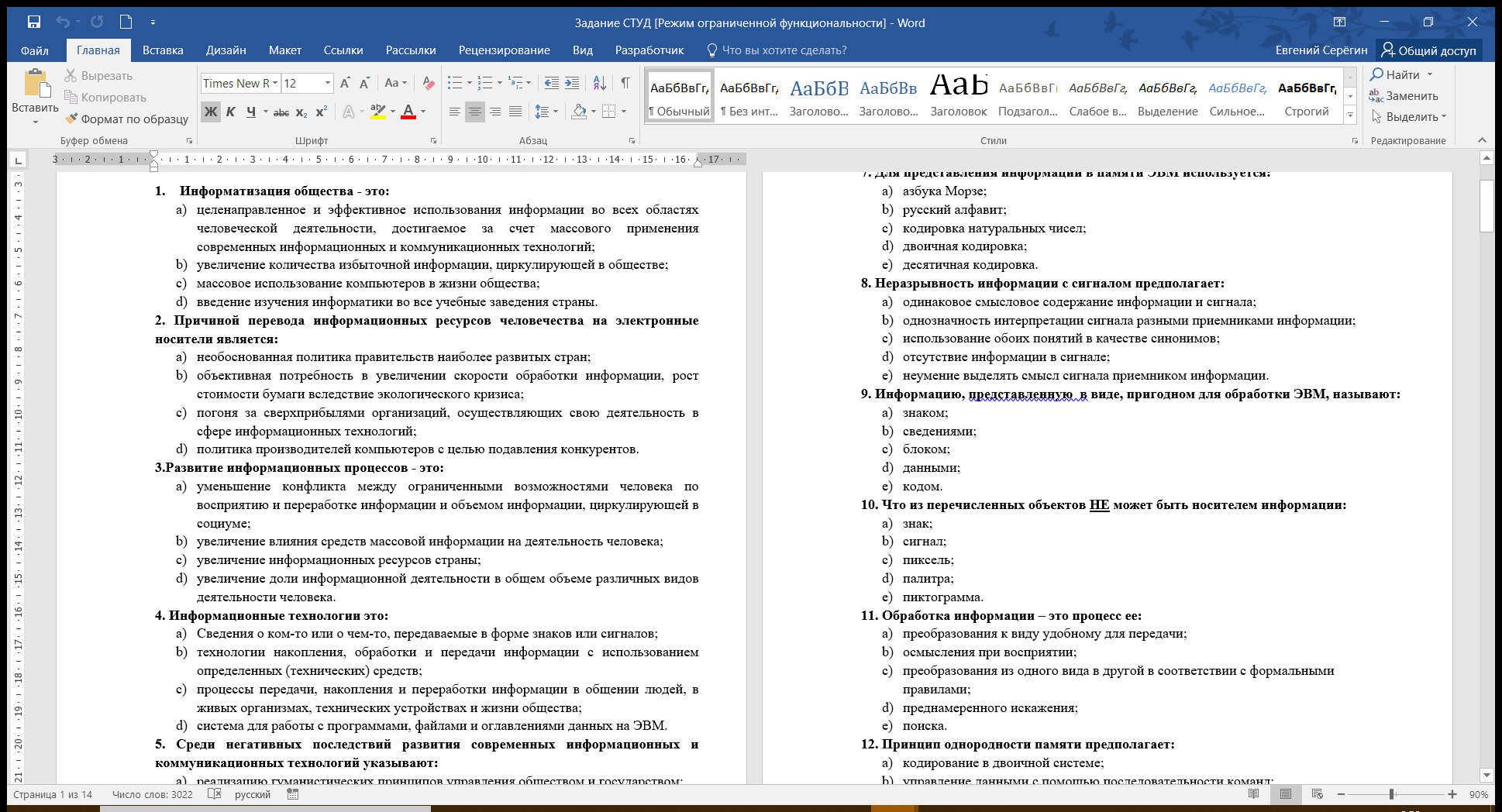


Рис 2.1

ПРИЛОЖЕНИЕ 3

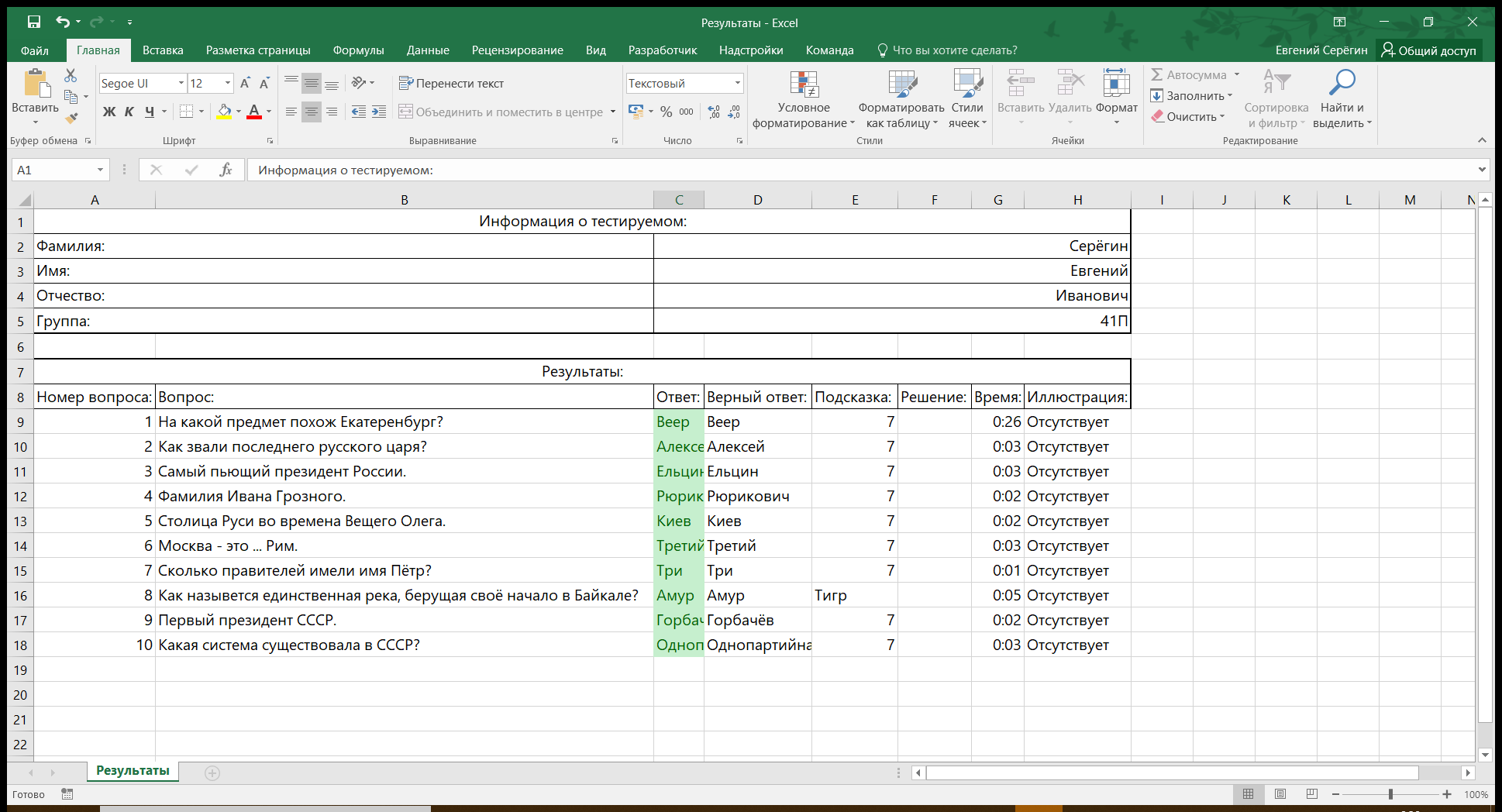


Рис 3.1

ПРИЛОЖЕНИЕ 4

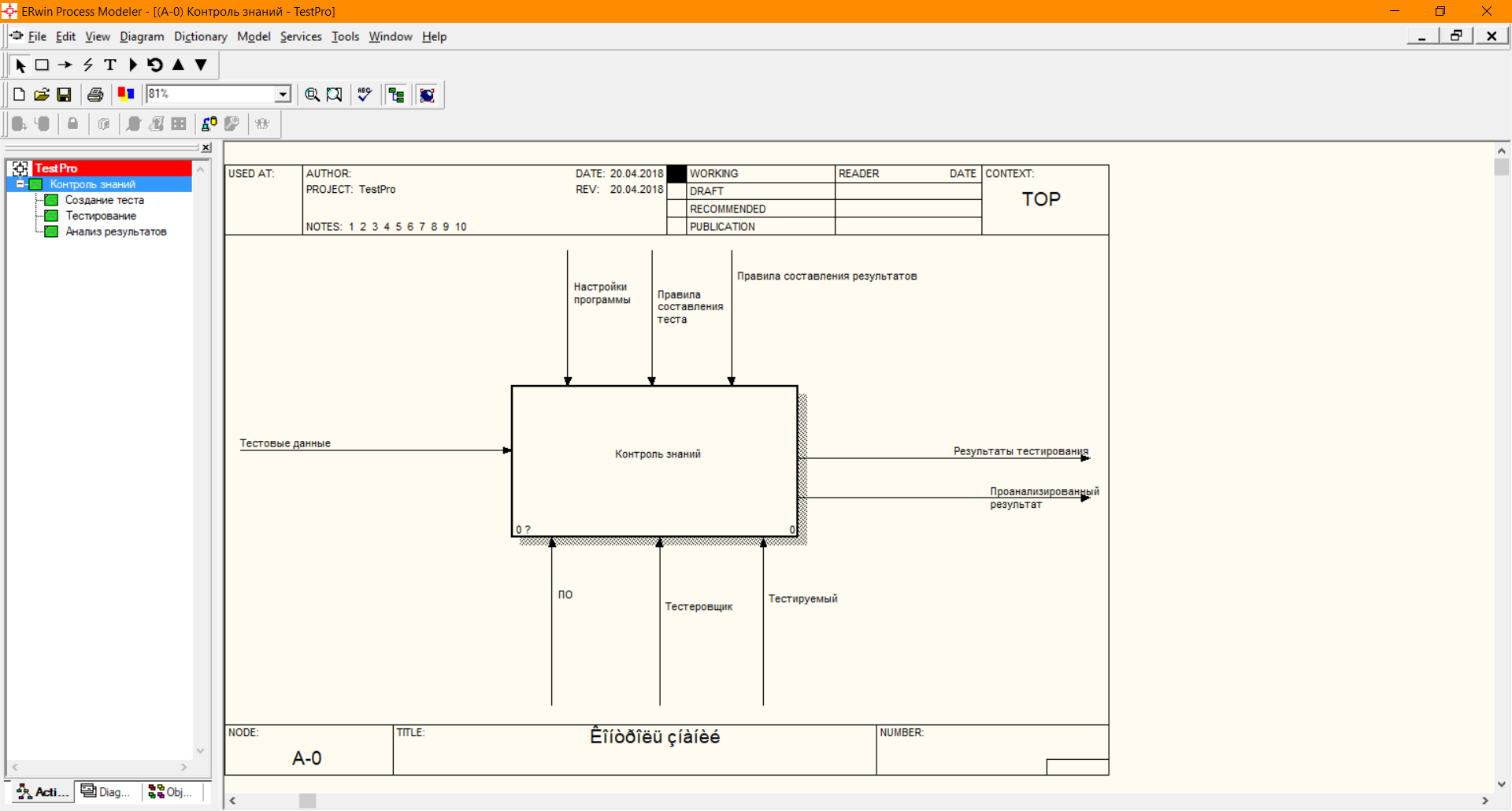


Рис 4.1

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ 4

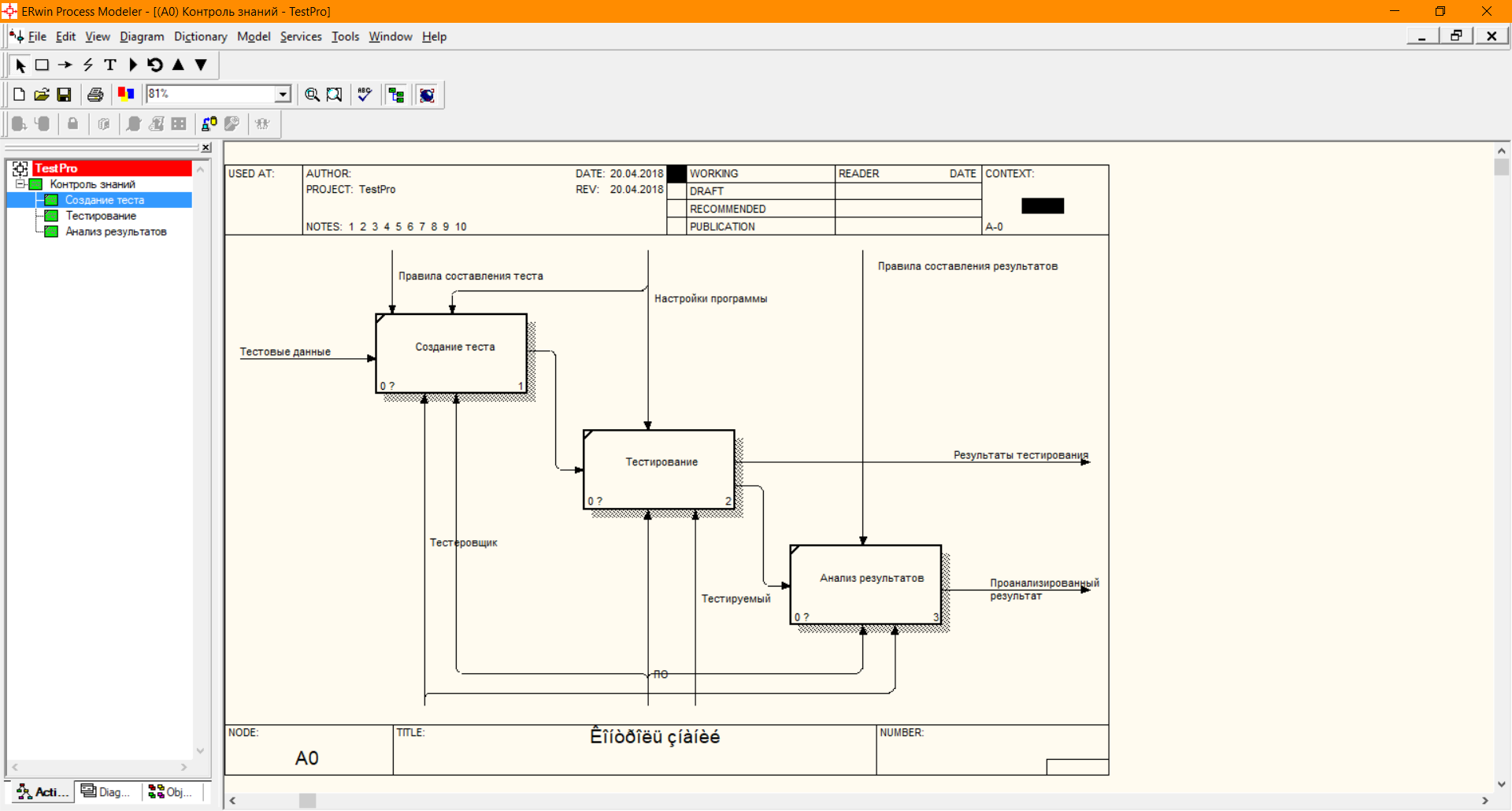


Рис 4.2

ПРИЛОЖЕНИЕ 5

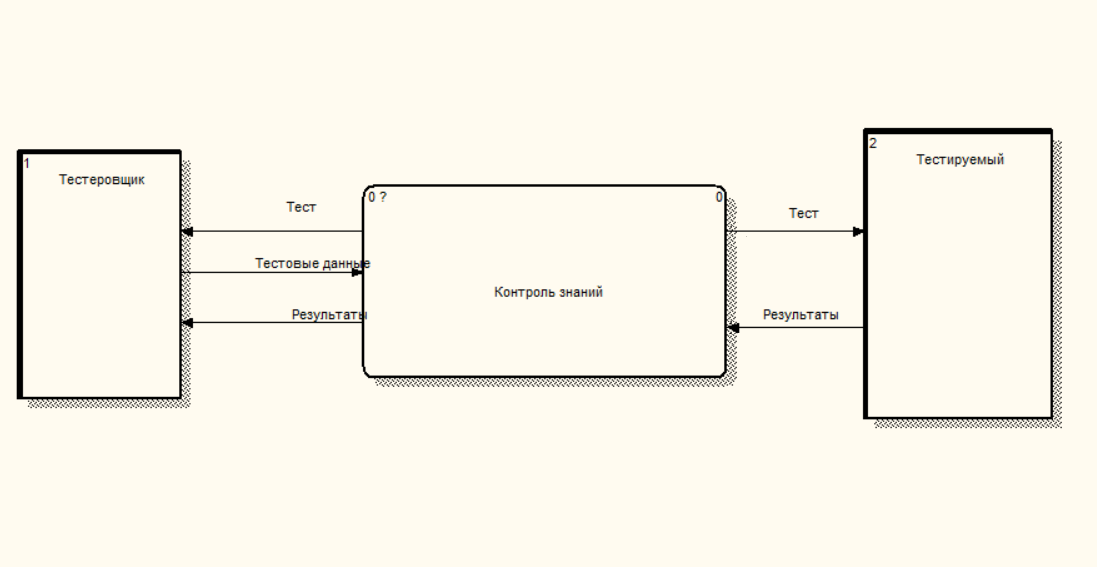
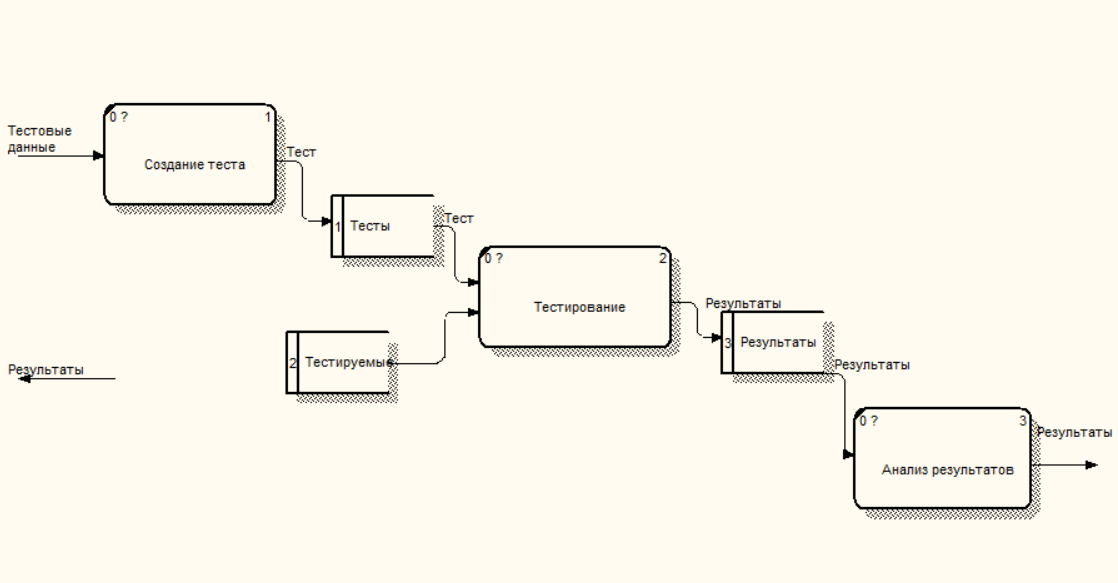


Рис 5,1

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ 5

Рис 5,2



ПРИЛОЖЕНИЕ 6

 Рис. 6.1

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ 5



Рис. 6.2

ПРИЛОЖЕНИЕ 7

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestPro.Raven

{

public class PictureClass

{

public string Path { get; set; }

public string Description { get; set; }

public bool Status { get; set; }

public object Media { get; set; }

}

public class Questions

{

public string $type { get; set; }

public string[] Answers { get; set; }

public string Question { get; set; }

public int RightAnswer { get; set; }

public string Hint { get; set; }

public string Solution { get; set; }

public bool AnotherAnswer { get; set; }

public PictureClass Picture { get; set; }

}

public class TestInfo

{

public string Name { get; set; }

public string Author { get; set; }

public string Date { get; set; }

public string Version { get; set; }

public string ProgramVersion { get; set; }

public string Subject { get; set; }

public bool TaskManager { get; set; }

public bool TaskPanel { get; set; }

public bool OtherApps { get; set; }

public bool QuestionMeshing { get; set; }

public bool AnswersMeshing { get; set; }

public bool Draft { get; set; }

public bool Calc { get; set; }

public bool Back { get; set; }

public bool Hint { get; set; }

public bool Stopwatch { get; set; }

public bool Timer { get; set; }

public int TimerValue { get; set; }

Рис. 7.1

public int QuestionsCount { get; set; }

public bool Inc { get; set; }

public bool ElseAllow { get; set; }

public Questions[] Questions { get; set; }

}

}

Рис. 7.2

ПРИЛОЖЕНИЕ 8

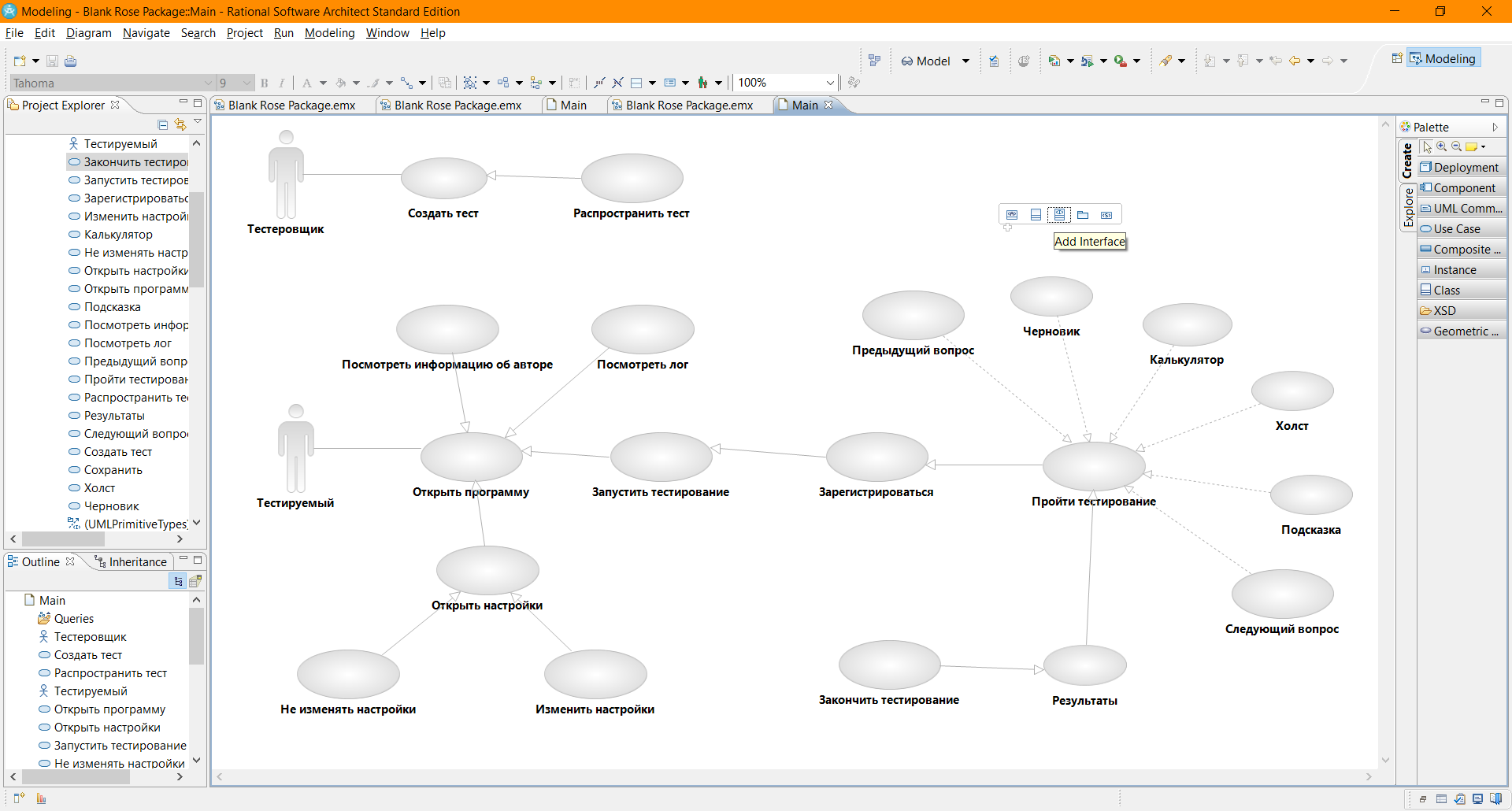


Рис 8.1

ПРИЛОЖЕНИЕ 9



Рис. 9.1

ПРИЛОЖЕНИЕ 10



Рис. 10.1

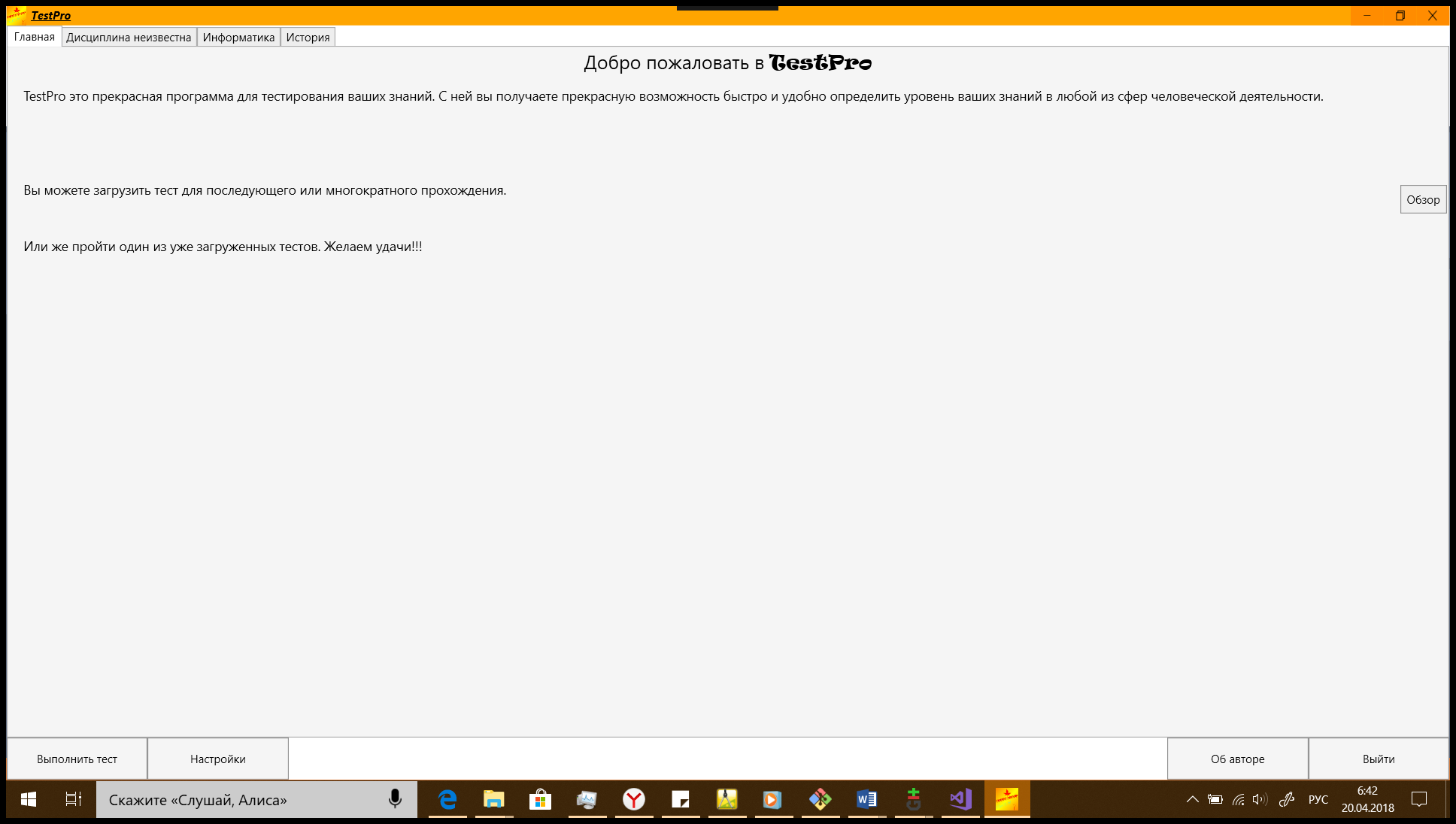


Рис. 10.2

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ 10

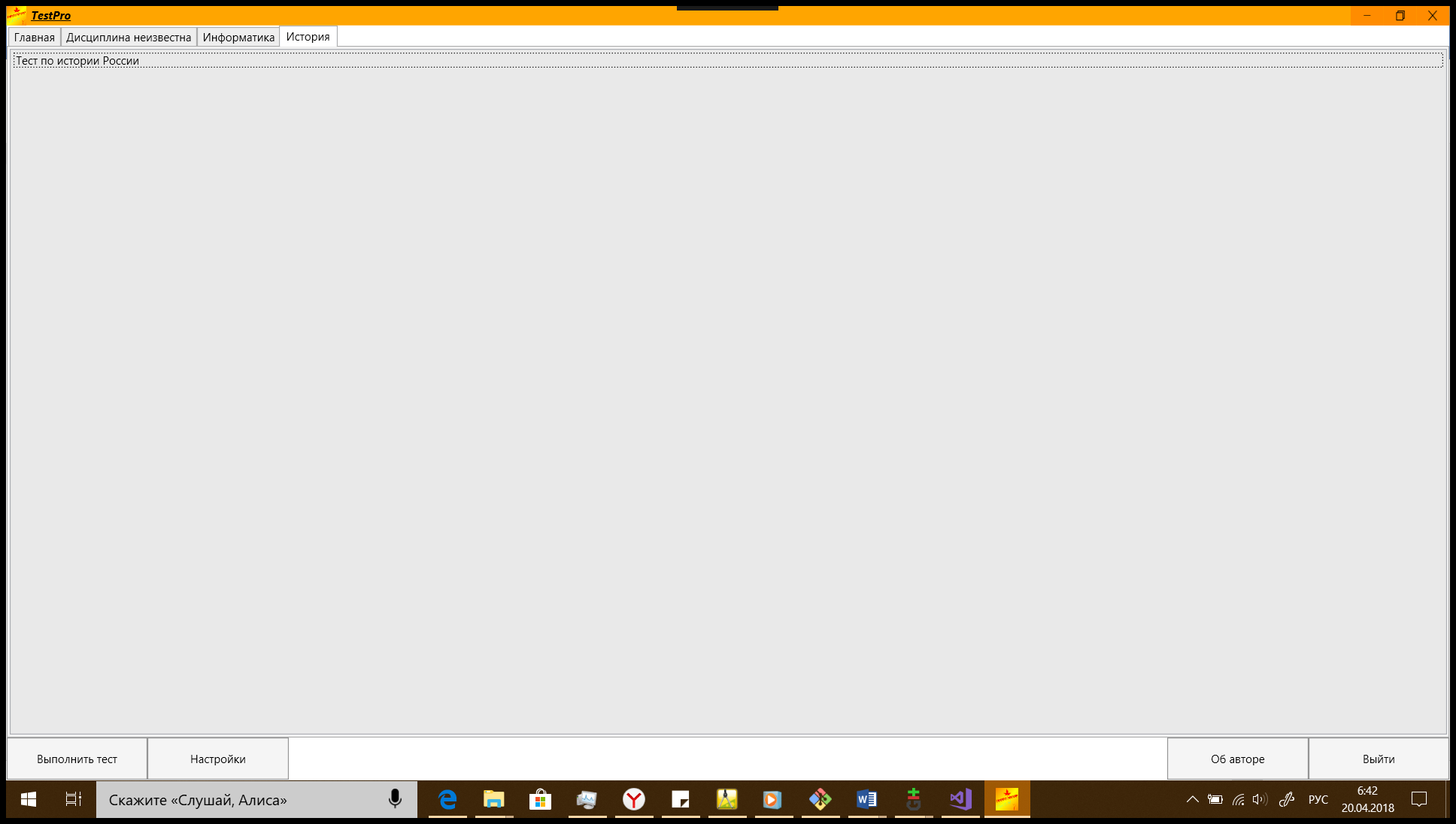


Рис. 10.3



Рис. 10.4

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ 10

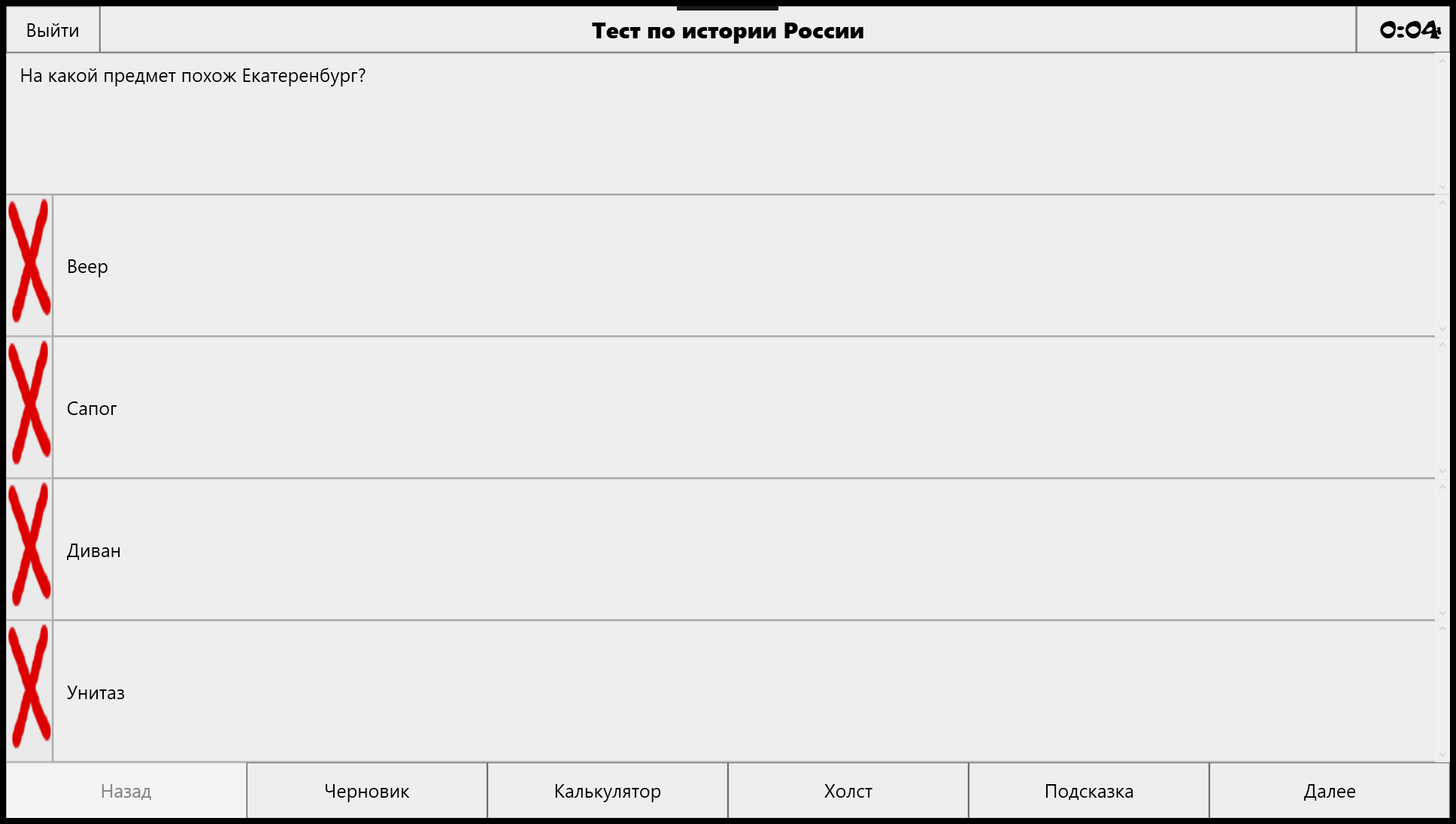


Рис. 10.5

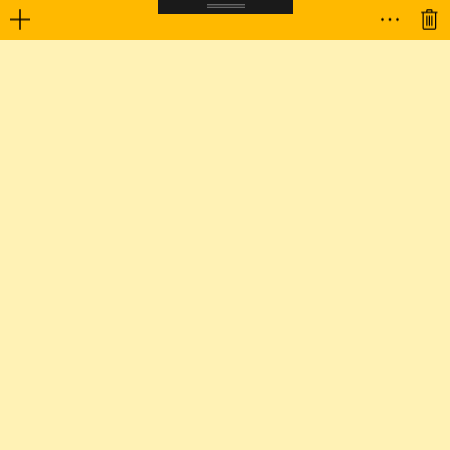


Рис. 10.6

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ 10

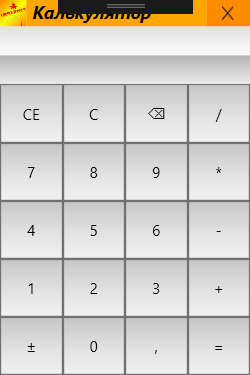


Рис. 10.7

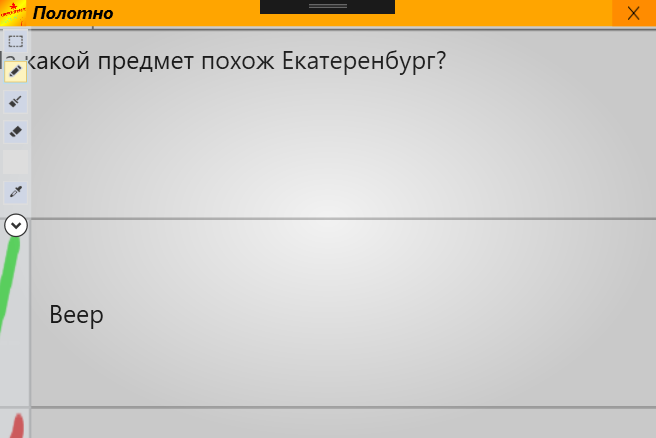


Рис. 10.8

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ 10

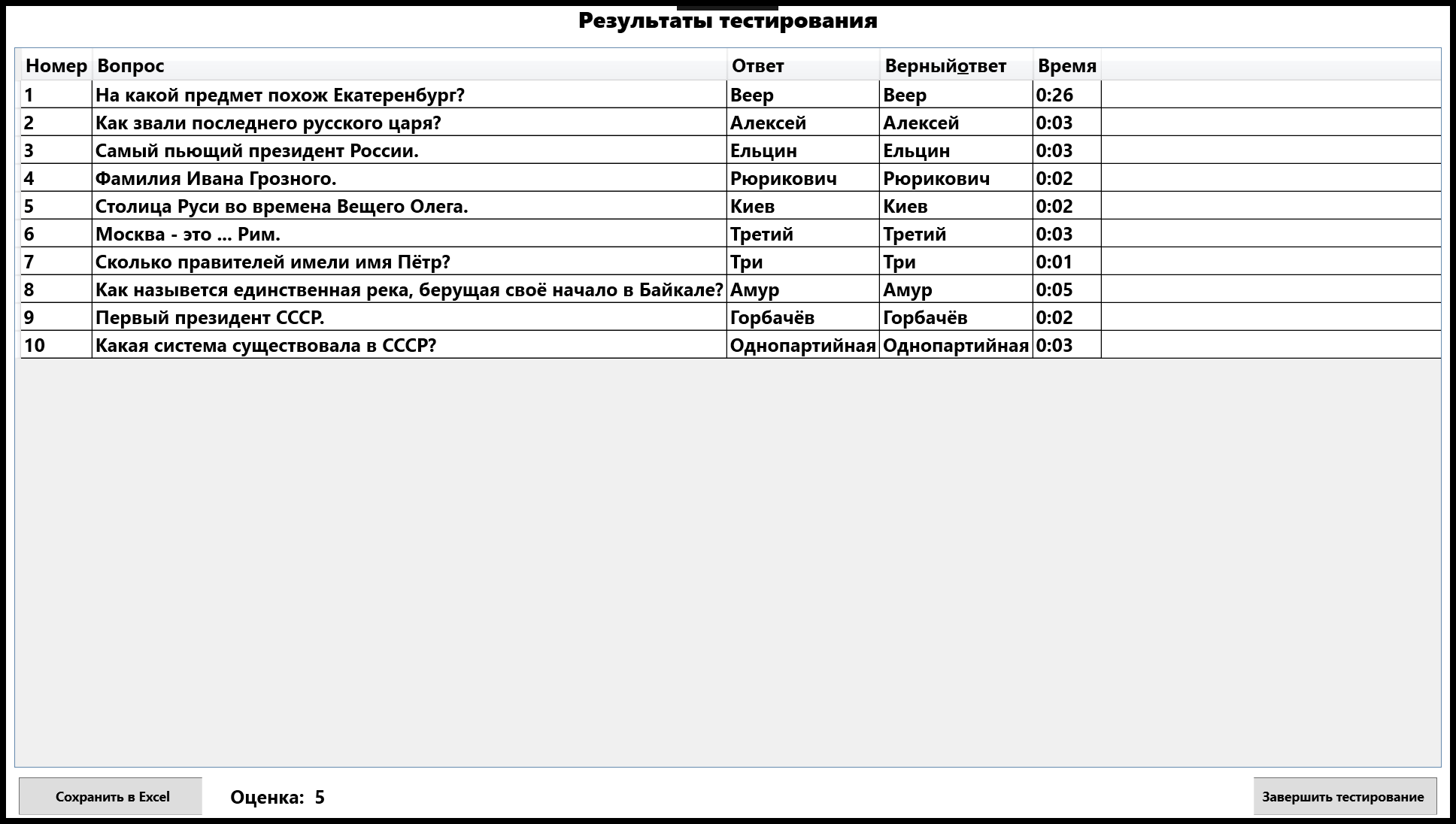


Рис. 10.9

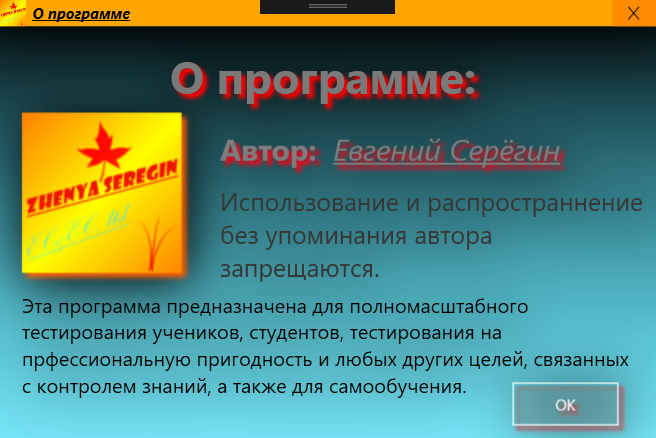


Рис. 10.10

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ 10

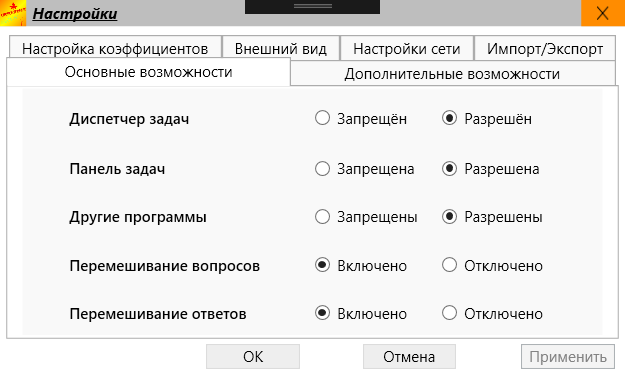


Рис. 10.11

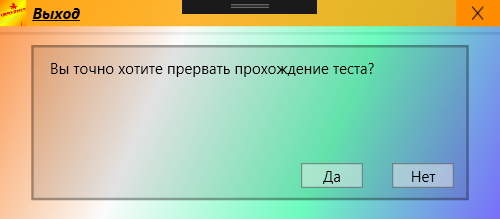


Рис. 10.12

ПРИЛОЖЕНИЕ 11

Листинг модуля Network.Network.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Net;

using System.IO;

using System.Net.Sockets;

using System.Windows;

using TestPro.Test.Main;

namespace TestPro.Network

{

/// <summary>

/// Клиентская часть сетевого стэка приложения

/// </summary>

public class Network

{

/// <summary>

/// Базовый конструктор класса

/// </summary>

public Network()

{

if (int.TryParse(Settings.ServerPort, out int Port))

{

Uri u;

if (Uri.TryCreate(Settings.ServerAdress, UriKind.RelativeOrAbsolute, out u))

{

Address = $"{Settings.ServerAdress}:{Settings.ServerPort}";

}

else

{

Address = $"{Settings.ServerAdress}:8090";

}

}

else

{

Uri u;

if (Uri.TryCreate(Settings.ServerAdress, UriKind.RelativeOrAbsolute, out u))

{

Address = $"http://localhost:{Settings.ServerPort}";

}

else

{

Address = @"http://localhost:8090";

}

}

}

/// <summary>

/// Тест соединения с сервером

/// </summary>

/// <returns>Успех соединения</returns>

public bool TestConnection()

{

try

{

HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(Address.TrimEnd(new char[1] { '/'}) + "?testconnection");

HttpWebResponse response = (HttpWebResponse)request.GetResponse();

Stream responseStream = response.GetResponseStream();

StreamReader responseReader = new StreamReader(responseStream);

string responseString = responseReader.ReadToEnd();

return true;

}

catch

{

return false;

}

}

string Address;

/// <summary>

/// Запрос списка тестов

/// </summary>

/// <returns>Список тестов</returns>

public List<string> GetTestsList()

{

WebRequest request = WebRequest.Create(Address + "?gettestslist");

WebResponse response = request.GetResponse();

using (Stream responseStream = response.GetResponseStream())

{

StreamReader sr = new StreamReader(responseStream);

string ss = sr.ReadToEnd();

using (MemoryStream ms = new MemoryStream(Encoding.UTF8.GetBytes(ss)))

{

TestPro.Server.Main.TestsListMeta tl = TestPro.Server.Main.TestsListMeta.Deserialize(ms);

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

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

s.Add(tl[i].Name);

tl.TestList.RemoveAt(1);

request = WebRequest.Create(Address + "?synchronizetests");

request.Method = "POST";

var va = tl.Serialize();

request.ContentLength = va.Length;

var rs = request.GetRequestStream();

va.CopyTo(rs);

tl.Serialize().CopyTo(request.GetRequestStream());

response = request.GetResponse();

using (Stream rStream = response.GetResponseStream())

{

StreamReader ssr = new StreamReader(rStream);

string sss = ssr.ReadToEnd();

using (MemoryStream mss = new MemoryStream(Encoding.Default.GetBytes(ss)))

{

TestPro.Server.Main.TestsListMeta tls = TestPro.Server.Main.TestsListMeta.Deserialize(mss);

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

MessageBox.Show(tls[i].Name);

}

}

return s;

}

}

}

/// <summary>

/// Запрос конкретного теств

/// </summary>

/// <param name="TestName"></param>

/// <returns></returns>

public string GetTest(string TestName)

{

WebRequest request = WebRequest.Create(Address + "?gettest&name=" + TestName);

WebResponse response = request.GetResponse();

Stream responseStream = response.GetResponseStream();

StreamReader responseReader = new StreamReader(responseStream);

return responseReader.ReadToEnd();

}

}

}

Листинг модуля Network.ServerNetwork.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Net;

using System.Net.Sockets;

using System.Windows;

using System.IO;

using TestPro.Utils;

using TestPro.Server.Main;

namespace TestPro.Network

{

/// <summary>

/// Серверная часть сетевого стека приложения

/// </summary>

public class ServerNetwork

{

private HttpListener server;

private bool active;

/// <summary>

/// Текущее состояние сервера

/// </summary>

public bool Active { get => active; set => active = value; }

/// <summary>

/// Базовый конструктор класса

/// </summary>

public ServerNetwork()

{

}

/// <summary>

/// Запуск сервера

/// </summary>

public void Start()

{

if (server == null || !server.IsListening)

{

server = new HttpListener();

server.Prefixes.Add("http://localhost:8090/"); // Probably 0.0.0.0:8090

Active = true;

Thread netThread = new Thread(new ThreadStart(Main));

netThread.Start();

}

}

//public Window w;

/// <summary>

/// Главный цикл сервера

/// </summary>

private void Main()

{

server.Start();

try

{

while (Active)

{

HttpListenerContext context = server.GetContext();

HttpListenerRequest request = context.Request;

var query = request.QueryString;

MemoryStream ms;

HttpListenerResponse response = context.Response;

Stream responseStream = (response.OutputStream as Stream);

if (query.Count == 1 && query.Get(0) == "testconnection" && String.IsNullOrWhiteSpace(query.GetKey(0)))

{

response.KeepAlive = true;

response.StatusCode = 200;

response.ContentType = "text/plain";

byte[] b = Encoding.UTF8.GetBytes("Connection created succesfully");

response.ContentLength64 = b.Length;

responseStream.Write(b, 0, b.Length);

responseStream.Close();

Thread.Sleep(1000);

}

if (query.Count == 1 && query.Get(0) == "gettestslist" && String.IsNullOrWhiteSpace(query.GetKey(0)))

{

MemoryStream fss = null; Settings.TestList.Serialize();

byte[] b = new byte[fss.Length];

fss.Read(b, 0, b.Length);

string result = Encoding.UTF8.GetString(b);

b = Encoding.UTF8.GetBytes(result);

response.ContentLength64 = b.Length;

responseStream.Write(b, 0, b.Length);

responseStream.Close();

}

//!

if (query.Count == 1 && query.Get(0) == "gettestslist0" && String.IsNullOrWhiteSpace(query.GetKey(0)))

{

ms = new MemoryStream();

foreach (var e in Settings.TestList.TestList)

{

byte[] buf = Encoding.Default.GetBytes(e.Name + Environment.NewLine);

ms.Write(buf, 0, buf.Length);

}

ms.Position = 0;

ms.CopyTo(response.OutputStream);

ms.Close();

}

//!

if (query.Count == 1 && query.Get(0) == "synchronizetests" && String.IsNullOrWhiteSpace(query.GetKey(0)))

{

StreamReader sr = new StreamReader(request.InputStream);

string r = sr.ReadToEnd();

Stream mmm = request.InputStream;

var t = TestsListMeta.Deserialize(request.InputStream as MemoryStream);

var tt = Settings.TestList.Compare(t);

MemoryStream fss = null; tt.Serialize();

byte[] b = new byte[fss.Length];

fss.Read(b, 0, b.Length);

string result = Encoding.Default.GetString(b);

b = Encoding.UTF8.GetBytes(result);

response.ContentLength64 = b.Length;

responseStream.Write(b, 0, b.Length);

responseStream.Close();

}

}

}

catch { }

if (server != null && server.IsListening)

{

server.Stop();

server.Close();

}

}

/// <summary>

/// Остановка сервера

/// </summary>

public void Stop()

{

if (server != null && server.IsListening)

{

server.Stop();

Active = false;

server.Close();

}

}

}

}

Листинг модуля Raven.RavenDb.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Raven.Client;

using Raven.Client.Embedded;

using TestPro.Test.Main;

using TestPro.Utils;

using System.IO;

namespace TestPro.Raven

{

/// <summary>

/// Класс обеспечивающий работу с бд

/// </summary>

public class RavenDB : IDisposable

{

private IDocumentStore store;

private IDocumentSession session;

private readonly string path = $@"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\TestPro\Test\RavenDB\System";

/// <summary>

/// Базовый конструктор класса

/// </summary>

public RavenDB()

{

store = new EmbeddableDocumentStore()

{

DataDirectory = path,

DefaultDatabase = "Tests"

}.Initialize();

session = store.OpenSession();

session.Advanced.MaxNumberOfRequestsPerSession = 128;

}

#region Tester's methods

/// <summary>

/// Добавление нового тестируемого в бд

/// </summary>

/// <param name="tester"></param>

public void AddTester(Tester tester)

{

session.Store(tester, tester.Id);

session.SaveChanges();

Reinitialize();

}

/// <summary>

/// Проверка существования тестируемого

/// </summary>

/// <param name="tester"></param>

/// <returns></returns>

public bool TesterExists(Tester tester)

{

return session.Query<Tester>().Any(x => x.Id == tester.Id);

}

/// <summary>

/// Запрос списка тестируемых

/// </summary>

/// <returns>Список тестируемых</returns>

public List<Tester> Testers()

{

List<Tester> results = new List<Tester>(), buffer;

int pageNumber = 0;

const int pageSize = 512;

do

{

buffer = session

.Query<Tester>()

.Select(x => new Tester()

{

Group = x.Group,

Middlename = x.Middlename,

Name = x.Name,

Surname = x.Surname,

Results = x.Results

})

.Skip((pageNumber \* pageSize))

.Take(pageSize)

.ToList();

pageNumber++;

results.AddRange(buffer);

}

while (buffer.Count > 0);

Reinitialize();

return results;

}

public void UpdateTester(Tester tester)

{

Tester value = session.Load<Tester>(tester.Id);

value = tester;

session.SaveChanges();

}

#endregion

#region Test's methods

/// <summary>

/// Добавление теста в бд

/// </summary>

/// <param name="path"></param>

public void AddTest(string path)

{

TestInfo test = new TestInfo();

if (Path.GetExtension(path) == ".tsz")

{

Packager up = new Packager();

var list = up.ReadFiles(path);

foreach (var e in list)

if (Path.GetExtension(e) == ".tst")

{

string text = "";

using (var fs = new FileStream(e, FileMode.Open))

{

byte[] buffer = new byte[fs.Length];

fs.Read(buffer, 0, buffer.Length);

text = Encoding.Default.GetString(buffer);

if (text.IndexOf("{Тип документа}Тест{/Тип документа}") == -1)

{

text = Encoding.Unicode.GetString(buffer);

if (text.IndexOf("{Тип документа}Тест{/Тип документа}") == -1)

throw new InvalidDataException();

}

}

test.Fill(text);

break;

}

session.Store(test);

}

if ((Path.GetExtension(path) == ".tse"))

{

Files files = new Files();

test.Fill(files.Read(path, true));

session.Store(test);

}

if ((Path.GetExtension(path) == ".tst"))

{

string text = "";

using (var fs = new FileStream(path, FileMode.Open))

{

byte[] buffer = new byte[fs.Length];

fs.Read(buffer, 0, buffer.Length);

text = Encoding.Default.GetString(buffer);

if (text.IndexOf("{Тип документа}Тест{/Тип документа}") == -1)

{

text = Encoding.Unicode.GetString(buffer);

if (text.IndexOf("{Тип документа}Тест{/Тип документа}") == -1)

throw new InvalidDataException();

}

}

test.Fill(text);

session.Store(test);

}

session.SaveChanges();

Reinitialize();

}

/// <summary>

/// Запрос списка тестов

/// </summary>

/// <returns></returns>

public List<TestInfo> Tests()

{

List<TestInfo> results = new List<TestInfo>(), buffer;

int pageNumber = 0;

const int pageSize = 512;

do

{

buffer = session

.Query<TestInfo>()

.Select(x => new TestInfo())

.Skip((pageNumber \* pageSize))

.Take(pageSize)

.ToList();

pageNumber++;

results.AddRange(buffer);

}

while (buffer.Count > 0);

Reinitialize();

return results;

}

/// <summary>

/// Запрос списка дисциплин

/// </summary>

/// <returns></returns>

public List<string> GetSubjects()

{

List<string> results = new List<string>(), buffer;

int pageNumber = 0;

const int pageSize = 512;

do

{

buffer = session

.Query<TestInfo>()

.Select(x => x.Subject)

.Skip((pageNumber \* pageSize))

.Take(pageSize)

.ToList();

pageNumber++;

results.AddRange(buffer);

}

while (buffer.Count > 0);

Reinitialize();

return results;

}

/// <summary>

/// Запрос тестов по предмету

/// </summary>

/// <param name="subject"></param>

/// <returns></returns>

public List<TestInfo> SubjectTests(string subject)

{

List<TestInfo> results = new List<TestInfo>(), buffer;

int pageNumber = 0;

const int pageSize = 512;

do

{

buffer = session

.Query<TestInfo>()

.Where(x => x.Subject == subject)

.Select(x => new TestInfo())

.Skip((pageNumber \* pageSize))

.Take(pageSize)

.ToList();

pageNumber++;

results.AddRange(buffer);

}

while (buffer.Count > 0);

Reinitialize();

return results;

}

#endregion

#region IDisposable Support

private bool disposedValue = false;

/// <summary>

/// Метод очистки ресурсов класса

/// </summary>

/// <param name="disposing">Проверка избыточности вызовов</param>

protected virtual void Dispose(bool disposing)

{

if (!disposedValue)

{

if (disposing)

{

session.Dispose();

store.Dispose();

}

// TODO: освободить неуправляемые ресурсы (неуправляемые объекты) и переопределить ниже метод завершения.

// TODO: задать большим полям значение NULL.

disposedValue = true;

}

}

// TODO: переопределить метод завершения, только если Dispose(bool disposing) выше включает код для освобождения неуправляемых ресурсов.

// ~RavenDB() {

// Dispose(false);

// }

/// <summary>

/// Обёртка для метода очистки ресурсов класса

/// </summary>

public void Dispose()

{

Dispose(true);

// TODO: раскомментировать следующую строку, если метод завершения переопределен выше.

// GC.SuppressFinalize(this);

}

#endregion

/// <summary>

/// Переподключение к серверу бд по истечению пула запросов

/// </summary>

public void Reinitialize(bool forced = false)

{

if (session.Advanced.NumberOfRequests > 120 || forced)

session = store.OpenSession();

}

}

}

Листинг модуля Raven.TestInfo.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using TestPro.Test.Main;

using TestPro.TestKinds;

using TestPro.Utils;

namespace TestPro.Raven

{

/// <summary>

/// Модель теста

/// </summary>

public class TestInfo

{

public string Id { get; set; }

public string Name { get; set; }

public string Author { get; set; }

public string Date { get; set; }

public string Version { get; set; }

public string ProgramVersion { get; set; }

public string Subject { get; set; }

public bool? TaskManager { get; set; }

public bool? TaskPanel { get; set; }

public bool? OtherApps { get; set; }

public bool? QuestionMeshing { get; set; }

public bool? AnswersMeshing { get; set; }

public bool? Draft { get; set; }

public bool? Calc { get; set; }

public bool? Back { get; set; }

public bool? Hint { get; set; }

public bool? Stopwatch { get; set; }

public bool? Timer { get; set; }

public int? TimerValue { get; set; }

public int? QuestionsCount { get; set; }

public bool? Inc { get; set; }

public bool? ElseAllow { get; set; }

public BasicTest[] Questions { get; set; }

/// <summary>

/// Извлечение данных теста из файла

/// </summary>

/// <param name="FillString">Строка теста</param>

/// <returns>Успех операции</returns>

public bool Fill(string FillString)

{

if (String.IsNullOrWhiteSpace(FillString))

throw new ArgumentNullException();

FillString = FillString.Trim();

int i = FillString.IndexOf("<Информация>");

int j = FillString.IndexOf("</Информация>");

if (i != -1 && j != -1)

{

MetadataFill(FillString.Substring(i + 12, j - i - 12).Trim());

}

i = FillString.IndexOf("<Задания>");

j = FillString.IndexOf("</Задания>");

if (i != -1 && j != -1)

{

string TestString = FillString.Substring(i + 9, j - i - 9).Trim();

if (QuestionsCount == -1)

{

int f = TestString.Split(new string[] { "[Вопрос]" }, StringSplitOptions.None).Count() - 1;

int s = TestString.Split(new string[] { "[/Вопрос]" }, StringSplitOptions.None).Count() - 1;

QuestionsCount = (f <= s) ? f : s;

}

Clear();

Questions = new BasicTest[(int)QuestionsCount];

int LastIndex = 0;

string Buffer = "";

for (int k = 0; k < QuestionsCount; k++)

{

i = TestString.IndexOf("[Вопрос]", LastIndex);

j = TestString.IndexOf("[/Вопрос]", LastIndex);

LastIndex = j + 8;

Buffer = TestString.Substring(i + 8, j - i - 8);

if (Buffer.IndexOf("<Обычный тест>") != -1)

{

Questions[k] = new BigTest(Buffer.Trim());

continue;

}

if (Buffer.IndexOf("<Вопрос - ответ>") != -1)

{

Questions[k] = new AnswerTest(Buffer.Trim());

continue;

}

if (Buffer.IndexOf("<Изменение порядка>") != -1)

{

Questions[k] = new OrderTest(Buffer.Trim());

continue;

}

if (Buffer.IndexOf("<Большой тест>") != -1)

{

Questions[k] = new BigTest(Buffer.Trim());

continue;

}

}

return true;

}

else return false;

}

/// <summary>

/// Заполнение метаданных

/// </summary>

/// <param name="FillString">Строка метаданных</param>

/// <returns>Успех операции</returns>

private bool MetadataFill(string FillString)

{

bool Flag = true;

if (String.IsNullOrWhiteSpace(FillString))

throw new ArgumentNullException();

FillString = FillString.Trim();

try

{

int ii = (QuestionsCount == null) ? 0 : (int)QuestionsCount;

if (!int.TryParse(Utils.System.ExecTestValue(ref FillString, "Количество вопросов"), out ii))

Flag = false;

QuestionsCount = ii;

}

catch

{

QuestionsCount = -1;

Flag = false;

}

int i, j;

i = FillString.IndexOf("<Название теста>");

j = FillString.IndexOf("</Название теста>");

string NString;

if (Utils.System.NotExistsCheck(j) && Utils.System.NotExistsCheck(i))

{

NString = FillString.Substring(i + 16, j - i - 16);

}

else

{

NString = "Название неизвестно";

}

try

{

Author = Utils.System.ExecTestValueStrict(ref FillString, "Автор теста");

}

catch

{

Author = "Автор неизвестен";

}

try

{

Subject = Utils.System.ExecTestValueStrict(ref FillString, "Дисциплина");

}

catch

{

Subject = "Дисциплина неизвестна";

}

try

{

Date = Utils.System.ExecTestValueStrict(ref FillString, "Дата создания");

}

catch

{

Date = "Дата создания неизвестна";

}

try

{

Version = Utils.System.ExecTestValueStrict(ref FillString, "Версия документа");

}

catch

{

Version = "Версия документа неизвестна";

}

try

{

ProgramVersion = Utils.System.ExecTestValueStrict(ref FillString, "Версия программы");

}

catch

{

ProgramVersion = "Версия программы неизвестна";

}

Name = NString;

try

{

TaskManager = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Диспетчер задач"));

}

catch

{

TaskManager = null;

}

try

{

TaskPanel = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Панель задач"));

}

catch

{

TaskPanel = null;

}

try

{

OtherApps = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Другие приложения"));

}

catch

{

OtherApps = null;

}

try

{

QuestionMeshing = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Перемешивание вопросов"));

}

catch

{

QuestionMeshing = null;

}

try

{

AnswersMeshing = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Перемешивание ответов"));

}

catch

{

AnswersMeshing = null;

}

try

{

Draft = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Черновик"));

}

catch

{

Draft = null;

}

try

{

Calc = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Калькулятор"));

}

catch

{

Calc = null;

}

try

{

Inc = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Холст"));

}

catch

{

Inc = null;

}

try

{

Hint = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Подсказки"));

}

catch

{

Hint = null;

}

try

{

Back = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Перемещение назад"));

}

catch

{

Back = null;

}

try

{

Stopwatch = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Секундомер"));

}

catch

{

Stopwatch = null;

}

try

{

Timer = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Таймер"));

}

catch

{

Timer = null;

}

i = FillString.IndexOf("<Значение таймера>");

j = FillString.IndexOf("</Значение таймера>");

if (Utils.System.NotExistsCheck(i) && Utils.System.NotExistsCheck(j))

{

TimerValue = Convert.ToInt32(FillString.Substring(i + 18, j - i - 18));

}

else

{

if (Timer != true)

{

TimerValue = 0;

}

else

{

if (QuestionsCount != -1)

{

TimerValue = QuestionsCount \* Settings.SecondsOnQuestion;

}

else

{

TimerValue = 0;

Timer = false;

}

}

}

try

{

ElseAllow = Convert.ToBoolean(Utils.System.ExecTestValueStrict(ref FillString, "Другие ответы"));

}

catch

{

ElseAllow = null;

}

return Flag;

}

private void Clear()

{

if (Questions != null)

{

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

Questions[i] = null;

Questions = null;

}

}

/// <summary>

/// Загрузка теста

/// </summary>

public void Load()

{

Metadata.Id = Id;

Metadata.Name = Name;

Metadata.Author = Author;

Metadata.Date = Date;

Metadata.Version = Version;

Metadata.ProgramVersion = ProgramVersion;

Metadata.Subject = Subject;

Metadata.TaskManager = TaskManager != null? (bool)TaskManager : Settings.TaskManager;

Metadata.TaskPanel = TaskPanel != null ? (bool)TaskPanel : Settings.TaskPanel;

Metadata.OtherApps = OtherApps != null ? (bool)OtherApps : Settings.OtherApps;

Metadata.QuestionMeshing = QuestionMeshing != null ? (bool)QuestionMeshing : Settings.QuestionMeshing;

Metadata.AnswersMeshing = AnswersMeshing != null ? (bool)AnswersMeshing : Settings.AnswersMeshing;

Metadata.Draft = Draft != null ? (bool)Draft : Settings.Draft;

Metadata.Calc = Calc != null ? (bool)Calc : Settings.Calc;

Metadata.Back = Back != null ? (bool)Back : Settings.Back;

Metadata.Hint = Hint != null ? (bool)Hint : Settings.Hint;

Metadata.Stopwatch = Stopwatch != null ? (bool)Stopwatch : Settings.Stopwatch;

Metadata.Timer = Timer != null ? (bool)Timer : Settings.Timer;

Metadata.Inc = Inc != null ? (bool)Inc : Settings.Inc;

Metadata.TimerValue = TimerValue != null ? (int)TimerValue : Settings.TimerValue;

Metadata.QuestionsCount = QuestionsCount != null ? (int)QuestionsCount : 0;

Metadata.ElseAllow = ElseAllow;

Test.Main.Test.Answers = new BasicAnswer[(int)QuestionsCount];

Test.Main.Test.Questions = new BasicTest[(int)QuestionsCount];

Test.Main.Test.Questions = Questions;

}

}

}

Листинг модуля Test.AboutWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using TestPro.Utils;

namespace TestPro.Test

{

/// <summary>

/// Логика взаимодействия для AboutWindow.xaml

/// </summary>

public partial class AboutWindow : Window

{

/// <summary>

/// Количество созданных экземпляров класса

/// </summary>

public static int Count = 0;

/// <summary>

/// Базовый конструктор окна "О программе"

/// </summary>

public AboutWindow()

{

Logger.Log("TP.T.AW - Инициализация окна о программе");

InitializeComponent();

Count++;

Logger.Log("TP.T.AW - Инициализация завершена");

}

/// <summary>

/// Закрытие окна по нажатию на кнопку ОК

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void OK\_Click(object sender, RoutedEventArgs e)

{

Count--;

Close();

Logger.Log("TP.T.AW - Окно закрыто");

}

/// <summary>

/// Закрытие окна по нажатию на кнопку закрытия окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void EscapeActive\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Count--;

Close();

Logger.Log("TP.T.AW - Окно закрыто");

}

}

}

Листинг модуля Test.AboutWindow.xaml

<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Test"

xmlns:ed="http://schemas.microsoft.com/expression/2010/drawing" x:Class="TestPro.Test.AboutWindow"

mc:Ignorable="d"

Title="О программе" Height="350" Width="525" Foreground="#00000000" AllowsTransparency="True"

WindowStyle="None" ResizeMode="NoResize" Topmost="True">

<Window.Resources>

<Style x:Key="ButtonStyle1" TargetType="{x:Type Button}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Grid>

<ed:RegularPolygon Fill="#FFF4F4F5" InnerRadius="1" PointCount="3" RenderTransformOrigin="0.5,0.5" Stretch="Fill" Stroke="Black">

<ed:RegularPolygon.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform Angle="56.863"/>

<TranslateTransform/>

</TransformGroup>

</ed:RegularPolygon.RenderTransform>

</ed:RegularPolygon>

<ContentPresenter HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}" RecognizesAccessKey="True" SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}" VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsFocused" Value="True"/>

<Trigger Property="IsDefaulted" Value="True"/>

<Trigger Property="IsMouseOver" Value="True"/>

<Trigger Property="IsPressed" Value="True"/>

<Trigger Property="IsEnabled" Value="False"/>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="Escape">

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Image.Source" Value="../Images/General/Escape/Escape.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="EscapeActive\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="../Images/General/Escape/EscapeActive.png" />

</Trigger>

</Style.Triggers>

</Style>

</Window.Resources>

<Window.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Black"/>

<GradientStop Color="#7F00DFFF" Offset="1"/>

</LinearGradientBrush>

</Window.Background>

<WindowChrome.WindowChrome>

<WindowChrome CaptionHeight="21"/>

</WindowChrome.WindowChrome>

<Grid>

<!-- Свойства Grid'а -->

<Grid.Effect>

<DropShadowEffect BlurRadius="60"/>

</Grid.Effect>

<Grid.RowDefinitions>

<RowDefinition Height="21"/>

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="21" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="35" />

</Grid.ColumnDefinitions>

<!-- Заголовок -->

<Rectangle x:Name="TitleBlock">

<Rectangle.Margin>0,0,0,0</Rectangle.Margin>

<Rectangle.Stroke>Orange</Rectangle.Stroke>

<Rectangle.Fill>Orange</Rectangle.Fill>

<Grid.ColumnSpan>3</Grid.ColumnSpan>

</Rectangle>

<TextBlock x:Name="TitleText" Margin="5,0,0,0" TextWrapping="Wrap" Text="О программе" Foreground="Black" FontWeight="Bold" FontStyle="Italic" TextDecorations="Underline" Grid.Column="1" Padding="0,2,0,0"/>

<Image x:Name="Escape" WindowChrome.IsHitTestVisibleInChrome="True" Style="{StaticResource ResourceKey=Escape}" Grid.Column="2"/>

<Image x:Name="TitleLogo" Margin="0,0,0,0" Source="../Images/TestProLogo.ico" Stretch="Fill"/>

<!-- Тело -->

<TextBlock x:Name="AboutBlock">

<TextBlock.Margin>115,15,100,260</TextBlock.Margin>

<TextBlock.FontSize>36</TextBlock.FontSize>

<TextBlock.Foreground>#FF7A7A7A</TextBlock.Foreground>

<TextBlock.RenderTransformOrigin>0.5,0.5</TextBlock.RenderTransformOrigin>

<TextBlock.TextWrapping>Wrap</TextBlock.TextWrapping>

<TextBlock.FontWeight>Bold</TextBlock.FontWeight>

<TextBlock.Effect>

<DropShadowEffect Color="Red"/>

</TextBlock.Effect>

<Run Text="О программе"/><Run Text=":"/><LineBreak/>

<Grid.Row>1</Grid.Row>

<Grid.Column>1</Grid.Column>

</TextBlock>

<Image x:Name="Logo" Margin="17.5,69,344.5,132" Source="../Images/TestProLogo.ico" Stretch="Fill" RenderTransformOrigin="0.5,0.5" Grid.Row="1">

<Image.Effect>

<DropShadowEffect Color="Orange" BlurRadius="7" RenderingBias="Quality"/>

</Image.Effect>

<Grid.Column>0</Grid.Column>

<Grid.ColumnSpan>2</Grid.ColumnSpan>

</Image>

<TextBlock x:Name="Author" Margin="155,82,15,210" Foreground="#FF7D7D7D" FontSize="24">

<TextBlock.Effect>

<DropShadowEffect Color="#FFFF0052"/>

</TextBlock.Effect>

<Run FontWeight="Bold" Text="Автор: "/>

<Run FontStyle="Italic" Text="Евгений Серёгин" TextDecorations="Underline"/>

<LineBreak/><Run/>

<Grid.Row>1</Grid.Row>

<Grid.Column>1</Grid.Column>

<TextBlock.TextWrapping>Wrap</TextBlock.TextWrapping>

</TextBlock>

<TextBlock x:Name="UsingInfo" Margin="155,126,10,125" Text="Использование и распространнение без упоминания автора запрещаются.">

<Grid.Row>1</Grid.Row>

<Grid.Column>1</Grid.Column>

<Grid.ColumnSpan>2</Grid.ColumnSpan>

<TextBlock.FontSize>20</TextBlock.FontSize>

<TextBlock.Foreground>#FF373737</TextBlock.Foreground>

<TextBlock.TextWrapping>Wrap</TextBlock.TextWrapping>

</TextBlock>

<TextBlock x:Name="Description" Margin="17.5,212,10,23" Foreground="Black" FontSize="16">

<Run Text="Эта программа предназначена для полномасштабного тестирования учеников, студентов, тестирования на прфессиональную пригодность и любых других целей, связанных с контролем знаний, а также для самообучения."/>

<LineBreak/>

<Run/>

<Grid.Row>1</Grid.Row>

<Grid.Column>0</Grid.Column>

<Grid.ColumnSpan>3</Grid.ColumnSpan>

<TextBlock.TextWrapping>Wrap</TextBlock.TextWrapping>

</TextBlock>

<Button x:Name="OK" Content="ОК" Click="OK\_Click">

<Button.Effect>

<DropShadowEffect Color="Red"/>

</Button.Effect>

<Button.Margin>389,285,30,10</Button.Margin>

<Button.Foreground>White</Button.Foreground>

<Button.BorderBrush>White</Button.BorderBrush>

<Button.Background>#3300FAFF</Button.Background>

<Button.ClickMode>Press</Button.ClickMode>

<Grid.Row>1</Grid.Row>

<Grid.Column>1</Grid.Column>

<Grid.ColumnSpan>2</Grid.ColumnSpan>

</Button>

</Grid>

</Window>

Листинг модуля Test.MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.IO;

using TestPro.Utils;

using TestPro.TestKinds;

using TestPro.Visual.Core;

using TestPro.Test.Main;

using TestPro.Visual.Common;

using TestPro.Raven;

using TestPro.Try;

using Microsoft.Win32;

namespace TestPro.Test

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

/// <summary>

/// Конструктор главной формы

/// </summary>

public MainWindow(List<string> testsList)

{

Logger.Log("TP.T.MW - Инициализация главного окна");

InitializeComponent();

Utils.System.mainWindow = this;

TestProMain\_StateChanged(this, null);

try

{

if (Settings.UseBackground)

{

if (Settings.Gradient.Active)

{

MainGrid.Background = Settings.Gradient.GetGradient();

}

else if (!String.IsNullOrWhiteSpace(Settings.BackgroundName))

{

MainGrid.Background = new ImageBrush(new BitmapImage(new Uri(Settings.BackgroundName, UriKind.Relative)));

}

}

}

catch { }

Logger.Log("TP.T.MW - Установка фона");

Settings.BackgroundNameChange += Settings\_BackgroundNameChange;

Settings.Gradient.GradientOff += Settings\_BackgroundNameChange;

Settings.GradientChange += Settings\_GradientChange;

Settings.Gradient.GradientOn += Settings\_GradientChange;

Settings.BackgroundOn += Settings\_Background;

Settings.BackgroundOff += Settings\_Background;

Logger.Log("TP.T.MW - Подписка на события");

using (var raven = new RavenDB())

{

testsList?.Sort();

foreach (var e in testsList)

{

tests.Add(e, raven.SubjectTests(e));

ListBox l = new ListBox()

{

Tag = e,

ItemsSource = tests[e].Select(x => x.Name)

};

l.MouseRightButtonUp += (s, er) =>

{

if (s is ListBox lb)

{

new TestInfoWindow(tests[e].ElementAt(lb.SelectedIndex)).ShowDialog();

}

};

l.MouseDoubleClick += (s, er) =>

{

StartTesting\_Click(s, er);

};

var tb = new TabItem()

{

Header = e,

Content = l

};

tb.MouseLeftButtonUp += (s, er) =>

{

currentListBox = (s as TabItem).Content as ListBox;

};

tbMain.Items.Add(tb);

}

}

Logger.Log("TP.T.MW - Загрузка тестов");

Logger.Log("TP.T.MW - Инициализация завершена");

}

/// <summary>

/// Словарь тестов

/// </summary>

private Dictionary<string, List<TestInfo>> tests = new Dictionary<string, List<TestInfo>>();

/// <summary>

/// Текущий ListBox

/// </summary>

private ListBox currentListBox = null;

/// <summary>

/// Закрытие окна по нажатию на кнопку закрытия окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void EscapeActive\_MouseLeftButtonUp(object sender, EventArgs e)

{

Close();

Logger.Log("TP.T.MW - Закрытие главного окна");

}

/// <summary>

/// Закрытие окна по нажатию на кнопку выйти

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Exit\_Click(object sender, RoutedEventArgs e)

{

Close();

Logger.Log("TP.T.MW - Закрытие главного окна");

}

/// <summary>

/// Открытие окна Лога

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Log\_Click(object sender, RoutedEventArgs e)

{

if (LogWindow.Count == 0)

{

LogWindow logWindow = new LogWindow();

logWindow.Show();

LogWindow.Count++;

}

}

/// <summary>

/// Начало тестирования

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void StartTesting\_Click(object sender, RoutedEventArgs e)

{

if (currentListBox is ListBox lb)

{

if (lb.SelectedIndex == -1) return;

Main.Test.Clear();

tests[lb.Tag as string].ElementAt(lb.SelectedIndex).Load();

if (Metadata.QuestionMeshing) Main.Test.Mix();

while (Main.Test.Questions[0] is OrderTest)

Main.Test.Mix();

Random random = new Random();

if (Metadata.AnswersMeshing)

foreach (var er in Main.Test.Questions)

if (er is OrderTest ot) ot.Mix(random);

else if (er is BigTest bt) bt.Mix(random);

new RegistrateTesterWindow().Show();

}

}

/// <summary>

/// Открытие окна настроек

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Settings\_Click(object sender, RoutedEventArgs e)

{

new SettingsWindow().ShowDialog();

}

/// <summary>

/// Обработчик события изменения градиента фона

/// </summary>

private void Settings\_GradientChange()

{

if (Settings.UseBackground)

{

if (Settings.Gradient.Active)

{

MainGrid.Background = Settings.Gradient.GetGradient();

}

}

Logger.Log("TP.T.MW - Изменение фона");

}

/// <summary>

/// Обработчик события изменения изображения фона

/// </summary>

private void Settings\_BackgroundNameChange()

{

if (Settings.UseBackground)

{

if (!Settings.Gradient.Active)

{

MainGrid.Background = new ImageBrush(new BitmapImage(new Uri(Settings.BackgroundName, UriKind.Relative)));

}

}

Logger.Log("TP.T.MW - Изменение фона");

}

/// <summary>

/// Обработчик события отключения фона

/// </summary>

private void Settings\_Background()

{

if (!Settings.UseBackground)

{

MainGrid.Background = new SolidColorBrush(new GradientCorner("ffffff").GetColor());

}

Logger.Log("TP.T.MW - Изменение фона");

}

/// <summary>

/// Открытие окна о программе

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void About\_Click(object sender, RoutedEventArgs e)

{

if (AboutWindow.Count == 0)

new AboutWindow().Show();

}

/// <summary>

/// Закрытие окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void MainForm\_Closed(object sender, EventArgs e)

{

Utils.System.ClearAppData();

Settings.BackgroundNameChange -= Settings\_BackgroundNameChange;

Settings.Gradient.GradientOff -= Settings\_BackgroundNameChange;

Settings.GradientChange -= Settings\_GradientChange;

Settings.Gradient.GradientOn -= Settings\_GradientChange;

Settings.BackgroundOn -= Settings\_Background;

Settings.BackgroundOff -= Settings\_Background;

Logger.Log("TP.T.MW - Отписка от событий");

}

/// <summary>

/// Придание окну нормальных размеров

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Compact\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

WindowState = WindowState.Normal;

}

/// <summary>

/// Разворачивание окна на весь экран

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Restore\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

WindowState = WindowState.Maximized;

}

/// <summary>

/// Сворачивание окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Min\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

WindowState = WindowState.Minimized;

Logger.Log("TP.T.MW - Минимизация");

}

/// <summary>

/// Изменение неклиентской части окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TestProMain\_StateChanged(object sender, EventArgs e)

{

if (WindowState == WindowState.Maximized)

{

BottomRow.Height = new GridLength(

SystemParameters.VirtualScreenHeight - SystemParameters.WorkArea.Height + 7);

Logger.Log("TP.T.MW - Максимизация");

}

else

{

BottomRow.Height = new GridLength(0);

Logger.Log("TP.T.MW - Нормализация");

}

}

/// <summary>

/// Добавление теста

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void AddTest\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog OpenDialog = new OpenFileDialog()

{

CheckFileExists = true,

Filter = "Тесты (\*.tst, \*.tse, \*.tsz)|\*.tst;\*.tse;\*.tsz|Все файлы (\*.\*)|\*.\*"

};

Logger.Log("TP.T.MW - Открытие окна добавления фона");

if (OpenDialog.ShowDialog() == true)

try

{

Thread thread = new Thread(

() =>

{

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

using (var raven = new RavenDB())

{

raven.AddTest(OpenDialog.FileName);

Thread.Sleep(1000); // Задержка, чтобы данные успели отправиться на сервер

var buf = raven.GetSubjects();

Dispatcher.Invoke(() => testsList = buf);

testsList?.Sort();

for (int i = 1; i < tbMain.Items.Count; i++)

Dispatcher.Invoke(() => tbMain.Items.RemoveAt(i));

foreach (var ee in testsList)

{

if (!tests.Any(x => x.Key == ee))

tests.Add(ee, raven.SubjectTests(ee));

Dispatcher.Invoke(() =>

{

ListBox l = new ListBox()

{

Tag = ee,

ItemsSource = tests[ee].Select(x => x.Name)

};

l.MouseRightButtonUp += (s, er) =>

{

if (s is ListBox lb)

{

new TestInfoWindow(tests[ee].ElementAt(lb.SelectedIndex)).ShowDialog();

}

};

l.MouseDoubleClick += (s, er) =>

{

StartTesting\_Click(s, er);

};

var tb = new TabItem()

{

Header = ee,

Content = l

};

tb.MouseLeftButtonUp += (s, er) =>

{

currentListBox = (s as TabItem).Content as ListBox;

};

tbMain.Items.Add(tb);

InvalidateVisual();

});

//Dispatcher.Invoke(() => tbMain.Items.Add(tb));

Dispatcher.Invoke(() => Logger.Log("TP.T.MW - Завершение потока добавления теста"));

}

}

})

{

Name = "TestPro.Load",

IsBackground = true

};

thread.Start();

Logger.Log("TP.T.MW - Запуск потока добавления теста");

}

catch

{

CustomMessageBox.Show("Ошибка. Выберите другой файл", "Ошибка загрузки", MessageBoxButton.OK, MessageBoxImage.Error);

Logger.Log("TP.T.MW - Ошибка загрузки файла");

AddTest\_Click(sender, e);

}

}

}

}

Листинг модуля Test.MainWindow.xaml

<Window x:Class="TestPro.Test.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Test"

mc:Ignorable="d"

Title="TestPro" Height="300" Width="500" MinHeight="300" MinWidth="500" AllowsTransparency="True"

WindowStartupLocation="CenterScreen" WindowStyle="None" x:Name="TestProMain" WindowState="Maximized"

StateChanged="TestProMain\_StateChanged" Background="Transparent" Closed="MainForm\_Closed">

<WindowChrome.WindowChrome>

<WindowChrome CaptionHeight="21" ResizeBorderThickness="5"/>

</WindowChrome.WindowChrome>

<Window.Resources>

<Style x:Key="BaseImageStyle">

<Setter Property="WindowChrome.IsHitTestVisibleInChrome" Value="True" />

<Setter Property="Control.Margin" Value="0" />

<Setter Property="Control.Width" Value="35" />

<Setter Property="Image.Stretch" Value="Fill" />

</Style>

<Style x:Key="Escape" BasedOn="{StaticResource ResourceKey=BaseImageStyle}">

<Setter Property="Image.Source" Value="../Images/General/Escape/Escape.png" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="EscapeActive\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="../Images/General/Escape/EscapeActive.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="Compact" BasedOn="{StaticResource ResourceKey=BaseImageStyle}">

<Setter Property="Image.Source" Value="../Images/General/Compact/Compact.png"/>

<EventSetter Event="Control.MouseLeftButtonUp" Handler="Compact\_MouseLeftButtonUp"/>

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=TestProMain,Path=WindowState}" Value="Normal">

<DataTrigger.Setters>

<Setter Property="Control.Visibility" Value="Hidden"/>

</DataTrigger.Setters>

</DataTrigger>

<Trigger Property="Control.IsMouseOver" Value="True">

<Trigger.Setters>

<Setter Property="Image.Source" Value="../Images/General/Compact/CompactHover.png"/>

</Trigger.Setters>

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="Restore" BasedOn="{StaticResource ResourceKey=BaseImageStyle}">

<Setter Property="Image.Source" Value="../Images/General/Restore/Restore.png"/>

<EventSetter Event="Control.MouseLeftButtonUp" Handler="Restore\_MouseLeftButtonUp"/>

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=TestProMain,Path=WindowState}" Value="Maximized">

<DataTrigger.Setters>

<Setter Property="Control.Visibility" Value="Hidden"/>

</DataTrigger.Setters>

</DataTrigger>

<Trigger Property="Control.IsMouseOver" Value="True">

<Trigger.Setters>

<Setter Property="Image.Source" Value="../Images/General/Restore/RestoreHover.png"/>

</Trigger.Setters>

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="Min" BasedOn="{StaticResource ResourceKey=BaseImageStyle}">

<Setter Property="Image.Source" Value="../Images/General/Min/Min.png"/>

<EventSetter Event="Control.MouseLeftButtonUp" Handler="Min\_MouseLeftButtonUp"/>

<Style.Triggers>

<Trigger Property="Control.IsMouseOver" Value="True">

<Trigger.Setters>

<Setter Property="Image.Source" Value="../Images/General/Min/MinHover.png"/>

</Trigger.Setters>

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="ButtonStyle">

<Setter Property="Control.Background" Value="#7FEAEAEA"/>

<Setter Property="Control.BorderBrush" Value="#BF707070"/>

</Style>

<Style TargetType="TabItem">

<Setter Property="Background" Value="#7FDDDDDD"/>

</Style>

<Style TargetType="ListBox">

<Setter Property="Background" Value="#7FDDDDDD"/>

</Style>

<Style x:Key="DefinitionStyle">

<Setter Property="RowDefinition.Height" Value="0"/>

<Setter Property="ColumnDefinition.Width" Value="0"/>

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=TestProMain,Path=WindowState}" Value="Maximized">

<DataTrigger.Setters>

<Setter Property="RowDefinition.Height" Value="6"/>

<Setter Property="ColumnDefinition.Width" Value="7"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Style="{StaticResource ResourceKey=DefinitionStyle}"/>

<RowDefinition Height="\*"/>

<RowDefinition x:Name="BottomRow" Height="0"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Style="{StaticResource ResourceKey=DefinitionStyle}"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Style="{StaticResource ResourceKey=DefinitionStyle}"/>

</Grid.ColumnDefinitions>

<Grid x:Name="MainGrid" Grid.Row="1" Grid.Column="1" Background="White">

<Grid.RowDefinitions>

<RowDefinition Height="21"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Background="Orange">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="21"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="35"/>

<ColumnDefinition Width="35"/>

<ColumnDefinition Width="35"/>

</Grid.ColumnDefinitions>

<Image Source="../Images/TestProLogo.ico"/>

<TextBlock Text="{Binding ElementName=TestProMain,Path=Title}" Grid.Column="1" TextWrapping="Wrap" Foreground="Black" FontWeight="Bold" FontStyle="Italic" TextDecorations="Underline" Padding="5,2,0,15"/>

<Image x:Name="Min" Style="{StaticResource Min}" Grid.Column="2"/>

<Image x:Name="Compact" Style="{StaticResource Compact}" Grid.Column="3"/>

<Image x:Name="Restore" Style="{StaticResource Restore}" Grid.Column="3"/>

<Image x:Name="Escape" Style="{StaticResource Escape}" Grid.Column="4"/>

</Grid>

<Grid Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition Height="8\*"/>

<RowDefinition Height="\*" MaxHeight="45"/>

</Grid.RowDefinitions>

<TabControl x:Name="tbMain" Background="#7FEAEAEA">

<TabItem Header="Главная">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="30"/>

<RowDefinition Height="100"/>

<RowDefinition Height="60"/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBlock FontSize="20" TextAlignment="Center"><Run Text="Добро пожаловать в "/><Run FontFamily="Ravie" Text="TestPro"/></TextBlock>

<TextBlock TextWrapping="Wrap" Margin="15,10" Grid.Row="1" Text="TestPro это прекрасная программа для тестирования ваших знаний. С ней вы получаете прекрасную возможность быстро и удобно определить уровень ваших знаний в любой из сфер человеческой деятельности. " FontSize="14"/>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="50"/>

</Grid.ColumnDefinitions>

<TextBlock TextWrapping="Wrap" Margin="15,10,0,10" Text="Вы можете загрузить тест для последующего или многократного прохождения." FontSize="14"/>

<Button x:Name="bAddTest" Style="{StaticResource ResourceKey=ButtonStyle}" MaxHeight="30" Grid.Column="1" Content="Обзор" HorizontalAlignment="Right" Margin="0,0,-0.4,0" Width="50" Click="AddTest\_Click"/>

</Grid>

<TextBlock Margin="15,10,15,0" FontSize="14" Grid.Row="3" Text="Или же пройти один из уже загруженных тестов. Желаем удачи!!!"/>

</Grid>

</TabItem>

</TabControl>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" MaxWidth="150"/>

<ColumnDefinition Width="\*" MaxWidth="150"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*" MaxWidth="150"/>

<ColumnDefinition Width="\*" MaxWidth="150"/>

</Grid.ColumnDefinitions>

<Button x:Name="bStartTesting" Content="Выполнить тест" Style="{StaticResource ResourceKey=ButtonStyle}" Click="StartTesting\_Click"/>

<Button x:Name="bSettings" Content="Настройки" Grid.Column="1" Style="{StaticResource ResourceKey=ButtonStyle}" Click="Settings\_Click"/>

<Button x:Name="bAbout" Content="Об авторе" Grid.Column="3" Style="{StaticResource ResourceKey=ButtonStyle}" Click="About\_Click"/>

<Button x:Name="bExit" Content="Выйти" Grid.Column="4" Style="{StaticResource ResourceKey=ButtonStyle}" Click="Exit\_Click"/>

<Button x:Name="bLog" Content="Лог" Grid.Column="2" Foreground="#00000000" BorderBrush="#00707070" Background="#00DDDDDD" Click="Log\_Click" Focusable="False" Opacity="0" OpacityMask="#00000000"/>

</Grid>

</Grid>

</Grid>

</Grid>

</Window>

Листинг модуля Test.SettingsWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using Microsoft.Win32;

using System.IO;

using TestPro.Visual.Core;

using TestPro.Test.Main;

using TestPro.Utils;

namespace TestPro.Test

{

/// <summary>

/// Логика взаимодействия для SettingsWindow.xaml

/// </summary>

public partial class SettingsWindow : Window

{

/// <summary>

/// Конструктор окна Настройки

/// </summary>

public SettingsWindow()

{

Logger.Log("TP.T.SW - Инициализация окна настроек");

InitializeComponent();

if (Settings.UseBackground)

{

if (Settings.Gradient.Active)

{

Background = Settings.Gradient.GetGradient(191);

}

else if (String.IsNullOrWhiteSpace(Settings.BackgroundName))

{

Background = new ImageBrush(new BitmapImage(new Uri(Settings.BackgroundName, UriKind.Relative)));

}

}

Logger.Log("TP.T.SW - Установка фона");

if (String.IsNullOrWhiteSpace(PictureName))

{

if (File.Exists(PictureName))

{

BitmapImage bitmapImage = new BitmapImage(new Uri(PictureName, UriKind.Relative));

ggPicture.Background = new ImageBrush(new BitmapImage(new Uri(PictureName, UriKind.Relative)));

(ggPicture.Background as ImageBrush).Stretch = Stretch.Uniform;

Logger.Log("TP.T.SW - Загрузка изображения фона");

}

}

LoadSettings();

Settings.BackgroundNameChange += Settings\_BackgroundNameChange;

Settings.Gradient.GradientOff += Settings\_BackgroundNameChange;

Settings.Gradient.GradientChange += Settings\_GradientChange;

Settings.Gradient.GradientOn += Settings\_GradientChange;

Logger.Log("TP.T.SW - Подписка на события");

}

// Local Variables

private string PictureName = Settings.BackgroundName;

bool loaded = false;

Gradient newGradient = new Gradient();

//MainFunctions

#region Functions

/// <summary>

/// Сохранение настроек

/// </summary>

private void SaveChanges()

{

if (TaskMgrOn.IsChecked == true)

Settings.TaskManager = true;

else

Settings.TaskManager = false;

if (TaskPanelOn.IsChecked == true)

Settings.TaskPanel = true;

else

Settings.TaskPanel = false;

if (OtherAppsOn.IsChecked == true)

Settings.OtherApps = true;

else

Settings.OtherApps = false;

var buf = (Convert.ToInt32(CMark.Text) / 100f, Convert.ToInt32(BMark.Text) / 100f, Convert.ToInt32(AMark.Text) / 100f);

if (buf.Item1 < buf.Item2 && buf.Item2 < buf.Item3)

Settings.Mark = buf;

else

{

CustomMessageBox.Show("Указаны неверные коэффициенты. Коэффициент для оценки \"5\" должен быть больше, чем для оценки \"4\", коэффициент которой, в свою очередь, должен быть больше, чем для оценки \"3\"", "Неверные данные");

Logger.Log("TP.T.SW - Введены неверные коээфициенты оценивания");

}

if (QuestionsMeshingOn.IsChecked == true)

Settings.QuestionMeshing = true;

else

Settings.QuestionMeshing = false;

if (AnswersMeshingOn.IsChecked == true)

Settings.AnswersMeshing = true;

else

Settings.AnswersMeshing = false;

if (DraftOn.IsChecked == true)

Settings.Draft = true;

else

Settings.Draft = false;

if (CalcOn.IsChecked == true)

Settings.Calc = true;

else

Settings.Calc = false;

if (IncOn.IsChecked == true)

Settings.Inc = true;

else

Settings.Inc = false;

if (HintOn.IsChecked == true)

Settings.Hint = true;

else

Settings.Hint = false;

if (NonLinearOn.IsChecked == true)

Settings.Back = true;

else

Settings.Back = false;

if (BackgroundOn.IsChecked == true)

{

Settings.UseBackground = true;

if (Gradient.IsChecked == true)

{

Settings.Gradient = newGradient.Clone();

Settings.Gradient.Active = true;

Settings\_GradientChange();

}

else

Settings.Gradient.Active = false;

if (Picture.IsChecked == true)

Settings.BackgroundName = PictureName;

}

else

{

Settings.UseBackground = false;

Background = new SolidColorBrush(new GradientCorner("ffffff").GetColor());

}

if (StopwatchOn.IsChecked == true)

Settings.Stopwatch = true;

else

Settings.Stopwatch = false;

if (TimerOn.IsChecked == true)

Settings.Timer = true;

else

Settings.Timer = false;

Settings.TimerValue = Convert.ToInt32(TestSeconds.Text);

if (UseWebOn.IsChecked == true)

Settings.UseWeb = true;

else

Settings.UseWeb = false;

if (AutomationWebSettingsOn.IsChecked == true)

Settings.UseWebDefault = true;

else

Settings.UseWebDefault = false;

Settings.ServerAdress = tbServerName.Text;

Settings.ServerPort = tbServerPort.Text;

Logger.Log("TP.T.SW - Сохранение настроек");

}

/// <summary>

/// Загрузка настроек

/// </summary>

private void LoadSettings()

{

if (Settings.TaskManager) TaskMgrOn.IsChecked = true; else TaskMgrOff.IsChecked = true;

if (Settings.TaskPanel) TaskPanelOn.IsChecked = true; else TaskPanelOff.IsChecked = true;

if (Settings.OtherApps) OtherAppsOn.IsChecked = true; else OtherAppsOff.IsChecked = true;

AMark.Text = (Settings.Mark.B \* 100).ToString();

BMark.Text = (Settings.Mark.C \* 100).ToString();

CMark.Text = (Settings.Mark.D \* 100).ToString();

if (Settings.QuestionMeshing) QuestionsMeshingOn.IsChecked = true; else QuestionsMeshingOff.IsChecked = true;

if (Settings.AnswersMeshing) AnswersMeshingOn.IsChecked = true; else AnswersMeshingOff.IsChecked = true;

if (Settings.Draft) DraftOn.IsChecked = true; else DraftOff.IsChecked = true;

if (Settings.Calc) CalcOn.IsChecked = true; else CalcOff.IsChecked = true;

if (Settings.Inc) IncOn.IsChecked = true; else IncOff.IsChecked = true;

if (Settings.Hint) HintOn.IsChecked = true; else HintOff.IsChecked = true;

if (Settings.Back) NonLinearOn.IsChecked = true; else NonLinearOff.IsChecked = true;

if (Settings.UseBackground) BackgroundOn.IsChecked = true; else BackgroundOff.IsChecked = true;

if (Settings.Gradient.Active) Gradient.IsChecked = true; else Picture.IsChecked = true;

if (Settings.Gradient.Active) gPicture.Visibility = Visibility.Hidden; else gGradient.Visibility = Visibility.Hidden;

if (Settings.Stopwatch) StopwatchOn.IsChecked = true; else StopwatchOff.IsChecked = true;

if (Settings.Timer) TimerOn.IsChecked = true; else TimerOff.IsChecked = true;

newGradient = Settings.Gradient.Clone();

newGradient.Active = true;

StyleButton\_Click(new object(), new RoutedEventArgs());

if (Settings.UseWeb)

{

tbServerName.IsEnabled = true;

tbServerPort.IsEnabled = true;

bTestConnection.IsEnabled = true;

UseWebOn.IsChecked = true;

}

else

{

tbServerName.IsEnabled = false;

tbServerPort.IsEnabled = false;

bTestConnection.IsEnabled = false;

UseWebOff.IsChecked = true;

}

if (Settings.UseWebDefault)

{

gWeb.IsEnabled = false;

AutomationWebSettingsOn.IsChecked = true;

}

else

{

gWeb.IsEnabled = true;

AutomationWebSettingsOff.IsChecked = true;

}

TestSeconds.Text = Settings.TimerValue.ToString();

tbServerName.Text = Settings.ServerAdress;

tbServerPort.Text = Settings.ServerPort;

loaded = true;

Logger.Log("TP.T.SW - Загрузка настроек");

}

/// <summary>

/// Обработчик события изменения градиента фона

/// </summary>

private void Settings\_GradientChange()

{

if (Settings.UseBackground)

{

if (Settings.Gradient.Active)

{

Background = Settings.Gradient.GetGradient(191);

}

}

Logger.Log("TP.T.SW - Изменение фона");

}

/// <summary>

/// Обработчик события изменения изображения фона

/// </summary>

private void Settings\_BackgroundNameChange()

{

if (Settings.UseBackground)

{

if (!Settings.Gradient.Active)

{

Background = new ImageBrush(new BitmapImage(new Uri(Settings.BackgroundName, UriKind.Relative)));

}

}

Logger.Log("TP.T.SW - Изменение фона");

}

#endregion

//Standard interface functions

#region StandardInterfaceFunctions

/// <summary>

/// Закрытие окна по нажатию на кнопку отмена

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Cancel\_Click(object sender, RoutedEventArgs e)

{

Close();

Logger.Log("TP.T.SW - Отмена изменения настроек и закрытие окна");

}

/// <summary>

/// Закрытие окна по нажатию на кнопку закрытия окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Escape\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Close();

Logger.Log("TP.T.SW - Закрытие окна");

}

#endregion

//Intarface

#region Interface

/// <summary>

/// Проверка на наличие текущих изменений

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void RadioButton\_Click(object sender, RoutedEventArgs e)

{

Logger.Log("TP.T.SW - Изменение активности кнопки применить");

if (!((Settings.TaskManager && TaskMgrOn.IsChecked == true) || (!Settings.TaskManager && TaskMgrOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.TaskPanel && TaskPanelOn.IsChecked == true) || (!Settings.TaskPanel && TaskPanelOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.OtherApps && OtherAppsOn.IsChecked == true) || (!Settings.OtherApps && OtherAppsOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

var buf = (Convert.ToInt32(CMark.Text) / 100f, Convert.ToInt32(BMark.Text) / 100f, Convert.ToInt32(AMark.Text) / 100f);

if (buf.Item1 != Settings.Mark.D || buf.Item2 != Settings.Mark.C || buf.Item3 != Settings.Mark.B)

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.QuestionMeshing && QuestionsMeshingOn.IsChecked == true) || (!Settings.QuestionMeshing && QuestionsMeshingOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.AnswersMeshing && AnswersMeshingOn.IsChecked == true) || (!Settings.AnswersMeshing && AnswersMeshingOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.Draft && DraftOn.IsChecked == true) || (!Settings.Draft && DraftOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.Calc && CalcOn.IsChecked == true) || (!Settings.Calc && CalcOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.Inc && IncOn.IsChecked == true) || (!Settings.Inc && IncOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.Hint && HintOn.IsChecked == true) || (!Settings.Hint && HintOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.Back && NonLinearOn.IsChecked == true) || (!Settings.Back && NonLinearOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.UseBackground && BackgroundOn.IsChecked == true) || (!Settings.UseBackground && BackgroundOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.Gradient.Active && Gradient.IsChecked == true) || (!Settings.Gradient.Active && Picture.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.UseWebDefault && AutomationWebSettingsOn.IsChecked == true) || (!Settings.UseWebDefault && AutomationWebSettingsOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.UseWeb && UseWebOn.IsChecked == true) || (!Settings.UseWeb && UseWebOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (tbServerName.Text != Settings.ServerAdress)

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (tbServerPort.Text != Settings.ServerPort)

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.Stopwatch && StopwatchOn.IsChecked == true) || (!Settings.Stopwatch && StopwatchOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (!((Settings.Timer && TimerOn.IsChecked == true) || (!Settings.Timer && TimerOff.IsChecked == true)))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

if (Settings.TimerValue != Convert.ToInt32(TestSeconds.Text))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

}

/// <summary>

/// Нажатие на кнопку применить

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Apply\_Click(object sender, RoutedEventArgs e)

{

SaveChanges();

Settings.Save();

bApply.IsEnabled = false;

}

/// <summary>

/// Нажатие на кнопку ОК

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void OK\_Click(object sender, RoutedEventArgs e)

{

SaveChanges();

Settings.Save();

Close();

Logger.Log("TP.T.SW - Сохранение настроек и закрытие окна");

}

/// <summary>

/// Выбор изображения фона

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Picture\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog OpenDialog = new OpenFileDialog()

{

CheckFileExists = true,

Filter = "Картинки(\*.jpg;\*.jpeg;\*.gif;\*.png;\*.bmp)|\*.JPG;\*.JPEG;\*.GIF;\*.PNG;\*.BMP"

};

OpenDialog.Filter += "|PNG(\*.png)|\*.PNG" + "|JPEG(\*.jpg;\*.jpeg)|\*.JPG;\*.JPEG";

OpenDialog.Filter += "|GIF(\*.gif)|\*.GIF" + "|Точечный рисунок(\*.bmp)|\*.BMP";

OpenDialog.Filter += "|Все файлы (\*.\*)|\*.\*";

if (OpenDialog.ShowDialog() == true)

{

string actualPath = System.IO.Path.GetDirectoryName(OpenDialog.FileName);

string standardPath = System.IO.Path.GetFullPath("Backgrounds");

PictureName = @"Backgrounds\" + System.IO.Path.GetFileName(OpenDialog.FileName);

if (actualPath != standardPath)

using (FileStream originalFileStream = new FileStream(OpenDialog.FileName, FileMode.Open))

{

using (FileStream newFileStream = new FileStream(PictureName, FileMode.Create))

{

originalFileStream.CopyTo(newFileStream);

}

}

BitmapImage bitmapImage = new BitmapImage(new Uri(PictureName, UriKind.Relative));

ggPicture.Background = new ImageBrush(new BitmapImage(new Uri(PictureName, UriKind.Relative)));

(ggPicture.Background as ImageBrush).Stretch = Stretch.Uniform;

}

if (PictureName != Settings.BackgroundName)

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

Logger.Log("TP.T.SW - Изменение фона");

}

/// <summary>

/// Изменение настроек пользовательского фона

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Background\_Click(object sender, RoutedEventArgs e)

{

if ((sender as RadioButton).Name == "BackgroundOn")

{

Gradient.IsEnabled = true;

Picture.IsEnabled = true;

gPicture.IsEnabled = true;

gGradient.IsEnabled = true;

}

else

{

Gradient.IsEnabled = false;

Picture.IsEnabled = false;

gPicture.IsEnabled = false;

gGradient.IsEnabled = false;

}

RadioButton\_Click(sender, e);

Logger.Log("TP.T.SW - Изменение фона");

}

/// <summary>

/// Изменение текущего варианта фона

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void GradientPicture\_Click(object sender, RoutedEventArgs e)

{

if ((sender as RadioButton).Name == "Gradient")

{

gPicture.Visibility = Visibility.Hidden;

gGradient.Visibility = Visibility.Visible;

}

else

{

gPicture.Visibility = Visibility.Visible;

gGradient.Visibility = Visibility.Hidden;

}

RadioButton\_Click(sender, e);

Logger.Log("TP.T.SW - Изменение фона");

}

/// <summary>

/// Загрузка данных о текущем градиенте

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void StyleButton\_Click(object sender, RoutedEventArgs e)

{

loaded = false;

if (LT.IsChecked == true)

{

Red.Value = newGradient.LeftTop.Red;

Green.Value = newGradient.LeftTop.Green;

Blue.Value = newGradient.LeftTop.Blue;

Alpha.Value = 3 \* newGradient.LeftTop.Alpha;

}

if (RT.IsChecked == true)

{

Red.Value = newGradient.RightTop.Red;

Green.Value = newGradient.RightTop.Green;

Blue.Value = newGradient.RightTop.Blue;

Alpha.Value = 3 \* newGradient.RightTop.Alpha;

}

if (RB.IsChecked == true)

{

Red.Value = newGradient.RightBottom.Red;

Green.Value = newGradient.RightBottom.Green;

Blue.Value = newGradient.RightBottom.Blue;

Alpha.Value = 3 \* newGradient.RightBottom.Alpha;

}

if (LB.IsChecked == true)

{

Red.Value = newGradient.LeftBottom.Red;

Green.Value = newGradient.LeftBottom.Green;

Blue.Value = newGradient.LeftBottom.Blue;

Alpha.Value = 3 \* newGradient.LeftBottom.Alpha;

}

loaded = true;

}

/// <summary>

/// Изменение текущего градиента

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Color\_Change(object sender, RoutedPropertyChangedEventArgs<double> e)

{

if (!loaded) return;

Gradient tempGradient = newGradient.Clone();

if (LT.IsChecked == true)

{

newGradient.LeftTop.Red = (byte)Red.Value;

newGradient.LeftTop.Green = (byte)Green.Value;

newGradient.LeftTop.Blue = (byte)Blue.Value;

newGradient.LeftTop.Alpha = (byte)Alpha.Value;

}

if (RT.IsChecked == true)

{

newGradient.RightTop.Red = (byte)Red.Value;

newGradient.RightTop.Green = (byte)Green.Value;

newGradient.RightTop.Blue = (byte)Blue.Value;

newGradient.RightTop.Alpha = (byte)Alpha.Value;

}

if (RB.IsChecked == true)

{

newGradient.RightBottom.Red = (byte)Red.Value;

newGradient.RightBottom.Green = (byte)Green.Value;

newGradient.RightBottom.Blue = (byte)Blue.Value;

newGradient.RightBottom.Alpha = (byte)Alpha.Value;

}

if (LB.IsChecked == true)

{

newGradient.LeftBottom.Red = (byte)Red.Value;

newGradient.LeftBottom.Green = (byte)Green.Value;

newGradient.LeftBottom.Blue = (byte)Blue.Value;

newGradient.LeftBottom.Alpha = (byte)Alpha.Value;

}

tempGradient.LeftTop.Alpha = Convert.ToByte(3 \* newGradient.LeftTop.Alpha / 4);

tempGradient.RightTop.Alpha = Convert.ToByte(3 \* newGradient.RightTop.Alpha / 4);

tempGradient.RightBottom.Alpha = Convert.ToByte(3 \* newGradient.RightBottom.Alpha / 4);

tempGradient.LeftBottom.Alpha = Convert.ToByte(3 \* newGradient.LeftBottom.Alpha / 4);

Background = tempGradient.GetGradient();

if (!Settings.Gradient.IsIdentiacal(newGradient))

{

bApply.IsEnabled = true;

return;

}

else

bApply.IsEnabled = false;

RadioButton\_Click(sender, new RoutedEventArgs());

Logger.Log("TP.T.SW - Изменение фона");

}

/// <summary>

/// Проверка соединения с сервером

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TestConnection\_Click(object sender, RoutedEventArgs e)

{

Network.Network network = new Network.Network();

if (network.TestConnection())

CustomMessageBox.Show("Проверка соединения с сервером выполнена успешно", "Проверка соединения", MessageBoxButton.OK, MessageBoxImage.Information, MessageBoxResult.OK);

else if (CustomMessageBox.Show("Не удалось установить связь с сервером, попробовать ещё раз", "Проверка соединения", MessageBoxButton.YesNo, MessageBoxImage.Question, MessageBoxResult.No) == MessageBoxResult.No)

CustomMessageBox.Show("Ошибка соединения", "Проверка соединения", MessageBoxButton.OK, MessageBoxImage.Error);

else

TestConnection\_Click(sender, e);

Logger.Log("TP.T.SW - Тест соединения с сервером");

}

/// <summary>

/// Изменение статуса использования сети

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void WebSettings\_Click(object sender, RoutedEventArgs e)

{

if ((sender as RadioButton).Name == "AutomationWebSettingsOn")

gWeb.IsEnabled = false;

else

gWeb.IsEnabled = true;

RadioButton\_Click(sender, e);

}

/// <summary>

/// Изменение режима использования сети

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void UseWeb\_Click(object sender, RoutedEventArgs e)

{

if ((sender as RadioButton).Name == "UseWebOn")

{

tbServerName.IsEnabled = true;

tbServerPort.IsEnabled = true;

bTestConnection.IsEnabled = true;

}

else

{

tbServerName.IsEnabled = false;

tbServerPort.IsEnabled = false;

bTestConnection.IsEnabled = false;

}

RadioButton\_Click(sender, e);

}

/// <summary>

/// Изменение порта, используемого для сетевых возможностей

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void ServerPort\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

string value = (sender as TextBox).Text;

int i;

int.TryParse(e.Text, out i);

i = Convert.ToInt32(value + i.ToString());

if (!Char.IsDigit(e.Text, 0) || value.Length >= 5 || i > 65535)

e.Handled = true;

}

/// <summary>

/// Изменение адреса сервера

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Server\_TextChanged(object sender, TextChangedEventArgs e)

{

RadioButton\_Click(e, new RoutedEventArgs());

}

/// <summary>

/// Импорт настроек

/// </summary>

/// <param name="OpenDialog"></param>

private void TryImport(OpenFileDialog OpenDialog = null)

{

if (OpenDialog == null)

{

OpenDialog = new OpenFileDialog()

{

CheckFileExists = true,

Filter = "Файл настроек(\*.fts)|\*.fts|Все файлы (\*.\*)|\*.\*"

};

}

if (OpenDialog.ShowDialog() == true)

{

try

{

if (Settings.Check(OpenDialog.FileName))

{

string source;

using (FileStream fs = new FileStream(OpenDialog.FileName, FileMode.Open))

{

using (StreamReader sr = new StreamReader(fs))

{

source = sr.ReadToEnd();

}

}

Settings.Fill(source);

LoadSettings();

bApply.IsEnabled = false;

}

else

{

{

if (MessageBoxResult.Yes == CustomMessageBox.Show("Выбран некорректный файл, попробовать ещё раз?", "Некорректный файл", MessageBoxButton.YesNo, MessageBoxImage.Warning))

TryImport(OpenDialog);

}

}

Logger.Log("TP.T.SW - Импорт настроек");

}

catch

{

if (MessageBoxResult.Yes == CustomMessageBox.Show("Произошла ошибка, попробовать ещё раз?", "Ошибка", MessageBoxButton.YesNo, MessageBoxImage.Warning))

{

TryImport(OpenDialog);

Logger.Log("TP.T.SW - Ошибка импорта настроек");

}

}

}

}

/// <summary>

/// Нажатие импорта настроек

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void bImport\_Click(object sender, RoutedEventArgs e)

{

TryImport();

}

/// <summary>

/// Экспорт настроек

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Export\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog SaveDialog = new SaveFileDialog()

{

CheckFileExists = false,

Filter = "Файл настроек (\*.fts)|\*.fts|Все файлы (\*.\*)|\*.\*"

};

if (SaveDialog.ShowDialog() == true)

{

using (FileStream SettingsStream = new FileStream(Settings.SettingsPath, FileMode.Open))

{

using (FileStream ExportFileStream = new FileStream(SaveDialog.FileName, FileMode.Create))

{

SettingsStream.CopyTo(ExportFileStream);

}

}

Logger.Log("TP.T.SW - Экспорт настроек");

}

}

#endregion

/// <summary>

/// Изменение времени на тестирование

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TestSeconds\_TextChanged(object sender, TextChangedEventArgs e)

{

RadioButton\_Click(e, new RoutedEventArgs());

}

/// <summary>

/// Закрытие окна, отписка от события

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void SettingsForm\_Closed(object sender, EventArgs e)

{

Settings.BackgroundNameChange -= Settings\_BackgroundNameChange;

Settings.Gradient.GradientOff -= Settings\_BackgroundNameChange;

Settings.Gradient.GradientChange -= Settings\_GradientChange;

Settings.Gradient.GradientOn -= Settings\_GradientChange;

Logger.Log("TP.T.SW - Отписка от событий");

}

}

}

Листинг модуля Test.SettingsWindow.xaml

<Window x:Name="SettingsForm" x:Class="TestPro.Test.SettingsWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Test"

mc:Ignorable="d"

Title="Настройки" Height="300" Width="500" AllowsTransparency="True" WindowStyle="None" ResizeMode="NoResize"

Closed="SettingsForm\_Closed">

<WindowChrome.WindowChrome>

<WindowChrome CaptionHeight="21"/>

</WindowChrome.WindowChrome>

<Window.Resources>

<Style x:Key="TabItemGrid">

<Setter Property="Control.Background" Value="#3FE5E5E5" />

<Setter Property="Control.Margin" Value="10,0,10,0" />

</Style>

<Style x:Key="TabItem">

<Setter Property="Control.Background" Value="#3FE5E5E5" />

</Style>

<Style x:Key="ButtonStyle">

<Setter Property="Control.Background" Value="#7FDDDDDD" />

<Setter Property="Control.BorderBrush" Value="#3F707070" />

<Setter Property="Control.HorizontalAlignment" Value="Left" />

<Setter Property="Control.VerticalAlignment" Value="Top" />

<Setter Property="Control.Width" Value="75" />

</Style>

<Style x:Key="TaskMgr">

<Setter Property="Control.ToolTip" Value="Запрет открытия диспетчера задач во время прохождения теста" />

</Style>

<Style x:Key="TaskPanel">

<Setter Property="Control.ToolTip" Value="Скрытие панели задач на время прохождения теста" />

</Style>

<Style x:Key="OtherApps">

<Setter Property="Control.ToolTip" Value="Запрет открытия других программ во время прохождения теста" />

</Style>

<Style x:Key="QuestionsMeshing">

<Setter Property="Control.ToolTip" Value="Случайный порядок вопросов" />

</Style>

<Style x:Key="AnswersMeshing">

<Setter Property="Control.ToolTip" Value="Случайный порядок ответов" />

</Style>

<Style x:Key="Draft">

<Setter Property="Control.ToolTip" Value="Всплывающее окно, позволяющее делать заметки" />

</Style>

<Style x:Key="Calc">

<Setter Property="Control.ToolTip" Value="Всплывающее окно калькулятора" />

</Style>

<Style x:Key="Inc">

<Setter Property="Control.ToolTip" Value="Всплывающее окно, позволяющее рисовать" />

</Style>

<Style x:Key="Hint">

<Setter Property="Control.ToolTip" Value="Всплывающее окно, с подсказками" />

</Style>

<Style x:Key="NonLinear">

<Setter Property="Control.ToolTip" Value="Свободная навигация по вопросам в тесте" />

</Style>

<Style x:Key="Background">

<Setter Property="Control.ToolTip" Value="Использование кастомного фона" />

<EventSetter Event="RadioButton.Click" Handler="RadioButton\_Click" />

</Style>

<Style x:Key="Gradient">

<Style.Triggers>

<Trigger Property="RadioButton.IsChecked" Value="True">

<Setter Property="Control.ToolTip" Value="Выбранный фон" />

</Trigger>

<Trigger Property="RadioButton.IsChecked" Value="False">

<Setter Property="Control.ToolTip" Value="Доступный фон" />

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="Slider">

<EventSetter Event="Slider.ValueChanged" Handler="Color\_Change" />

<Setter Property="Maximum" Value="255" />

<Setter Property="SmallChange" Value="2" />

<Setter Property="LargeChange" Value="8" />

<Setter Property="Height" Value="18" />

</Style>

<Style x:Key="StyleButton">

<EventSetter Event="RadioButton.Click" Handler="StyleButton\_Click" />

<Style.Triggers>

<Trigger Property="RadioButton.IsChecked" Value="True">

<Setter Property="RadioButton.ToolTip" Value="Редактируемый угол"></Setter>

</Trigger>

<Trigger Property="RadioButton.IsChecked" Value="False">

<Setter Property="RadioButton.ToolTip" Value="Доступный для редактирования угол"></Setter>

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="AutomationWebSettings">

<Setter Property="Control.ToolTip" Value="Автоматический поиск сервера и настройка соединения" />

</Style>

<Style x:Key="UseWeb">

<Setter Property="Control.ToolTip" Value="Получение тестов, отправка результатов по локальной сети" />

</Style>

<Style x:Key="ServerName">

<Setter Property="Control.ToolTip" Value="Сетевое имя сервера. Например, http://www.yandex.ru, localhost, 192.168.1.1 и т.д." />

</Style>

<Style x:Key="ServerPort">

<Setter Property="Control.ToolTip" Value="Порт, использующийся сервером. В промежутке от 1 до 65 535" />

</Style>

<Style x:Key="Escape">

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Image.Source" Value="../Images/General/Escape/Escape.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="Escape\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="../Images/General/Escape/EscapeActive.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="TextBloxStyle">

<Setter Property="TextBlock.FontWeight" Value="Medium"/>

<Setter Property="Control.VerticalAlignment" Value="Center"/>

<Setter Property="Control.Margin" Value="15,0"/>

<Setter Property="TextBox.HorizontalContentAlignment" Value="Right"/>

</Style>

<Style x:Key="Stopwatch" BasedOn="{StaticResource ResourceKey=TextBloxStyle}">

<Setter Property="RadioButton.FontWeight" Value="Normal"/>

<Setter Property="TextBox.HorizontalContentAlignment" Value="Left"/>

<Setter Property="Control.ToolTip" Value="Включение режима прохождения с фиксацией времени"/>

</Style>

<Style x:Key="Timer" BasedOn="{StaticResource ResourceKey=TextBloxStyle}">

<Setter Property="RadioButton.FontWeight" Value="Normal"/>

<Setter Property="TextBox.HorizontalContentAlignment" Value="Left"/>

<Setter Property="Control.ToolTip" Value="Включение режима контроля времени"/>

</Style>

<Style x:Key="TimerField" BasedOn="{StaticResource ResourceKey=TextBloxStyle}">

<Setter Property="Control.IsEnabled" Value="False"/>

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=TimerOn,Path=IsChecked}" Value="True">

<DataTrigger.Setters>

<Setter Property="Control.IsEnabled" Value="True"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="21" />

<RowDefinition Height="\*" />

<RowDefinition Height="29" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="21" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="35" />

</Grid.ColumnDefinitions>

<!-- Заголовок -->

<Rectangle x:Name="Rectangle" Margin="0,0,0,0" Fill="#7F7D7D7D" Grid.ColumnSpan="3" StrokeThickness="0"/>

<Image x:Name="TitleLogo" Margin="0,0,0,0" Source="../Images/TestProLogo.ico" Stretch="Fill"/>

<TextBlock x:Name="TitleText" Margin="5,0,0,0" TextWrapping="Wrap" Text="Настройки" Foreground="Black" FontWeight="Bold" FontStyle="Italic" TextDecorations="Underline" Grid.Column="1" Padding="0,2,0,0"/>

<Image x:Name="Escape" Style="{StaticResource Escape}" Grid.Column="2" WindowChrome.IsHitTestVisibleInChrome="True"/>

<!-- Тело -->

<TabControl Height="245" Margin="5,5,5,0" Background="Transparent" Grid.Row="1" Grid.ColumnSpan="3">

<TabItem Header="Основные возможности" Style="{StaticResource TabItem}">

<Grid Style="{StaticResource TabItemGrid}">

<RadioButton x:Name="TaskMgrOn" Style="{StaticResource TaskMgr}" GroupName="TaskMgr" Content="Запрещён" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,16.392,0,0" Click="RadioButton\_Click"/>

<RadioButton x:Name="TaskMgrOff" Style="{StaticResource TaskMgr}" GroupName="TaskMgr" Content="Разрешён" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,16.392,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource TaskMgr}" Content="Диспетчер задач" HorizontalAlignment="Left" Margin="32.408,10.392,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<RadioButton x:Name="TaskPanelOn" Style="{StaticResource TaskPanel}" GroupName="TaskPanel" Content="Запрещена" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,56.555,0,0" Click="RadioButton\_Click"/>

<RadioButton x:Name="TaskPanelOff" Style="{StaticResource TaskPanel}" GroupName="TaskPanel" Content="Разрешена" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,56.555,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource TaskPanel}" Content="Панель задач" HorizontalAlignment="Left" Margin="32.408,50.555,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<RadioButton x:Name="OtherAppsOn" Style="{StaticResource OtherApps}" GroupName="OtherApps" Content="Запрещены" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,95.079,0,0" Click="RadioButton\_Click"/>

<RadioButton x:Name="OtherAppsOff" Style="{StaticResource OtherApps}" GroupName="OtherApps" Content="Разрешены" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,95.079,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource OtherApps}" Content="Другие программы" HorizontalAlignment="Left" Margin="32.408,89.079,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<RadioButton x:Name="QuestionsMeshingOn" Style="{StaticResource QuestionsMeshing}" GroupName="QuestionsMeshing" Content="Включено" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,133.596,0,0" Click="RadioButton\_Click"/>

<RadioButton x:Name="QuestionsMeshingOff" Style="{StaticResource QuestionsMeshing}" GroupName="QuestionsMeshing" Content="Отключено" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,133.596,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource QuestionsMeshing}" Content="Перемешивание вопросов" HorizontalAlignment="Left" Margin="32.408,127.596,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<RadioButton x:Name="AnswersMeshingOn" Style="{StaticResource AnswersMeshing}" GroupName="AnswersMeshing" Content="Включено" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,172.12,0,0" Click="RadioButton\_Click"/>

<RadioButton x:Name="AnswersMeshingOff" Style="{StaticResource AnswersMeshing}" GroupName="AnswersMeshing" Content="Отключено" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,172.12,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource AnswersMeshing}" Content="Перемешивание ответов" HorizontalAlignment="Left" Margin="32.408,166.12,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

</Grid>

</TabItem>

<TabItem Header="Дополнительные возможности" Style="{StaticResource TabItem}">

<Grid Style="{StaticResource TabItemGrid}">

<RadioButton Style="{StaticResource Draft}" x:Name="DraftOn" GroupName="Draft" Content="Включен" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,16.392,0,0" Click="RadioButton\_Click"/>

<RadioButton Style="{StaticResource Draft}" x:Name="DraftOff" GroupName="Draft" Content="Отключен" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,16.392,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource Draft}" Content="Черновик" HorizontalAlignment="Left" Margin="32.408,10.392,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<RadioButton Style="{StaticResource Calc}" x:Name="CalcOn" GroupName="Calc" Content="Включен" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,56.555,0,0" Click="RadioButton\_Click"/>

<RadioButton Style="{StaticResource Calc}" x:Name="CalcOff" GroupName="Calc" Content="Отключен" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,56.555,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource Calc}" Content="Калькулятор" HorizontalAlignment="Left" Margin="32.408,50.555,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<RadioButton Style="{StaticResource Inc}" x:Name="IncOn" GroupName="Ink" Content="Включен" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,95.079,0,0" Click="RadioButton\_Click"/>

<RadioButton Style="{StaticResource Inc}" x:Name="IncOff" GroupName="Ink" Content="Отключен" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,95.079,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource Inc}" Content="Холст" HorizontalAlignment="Left" Margin="32.408,89.079,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<RadioButton Style="{StaticResource Hint}" x:Name="HintOn" GroupName="Hints" Content="Включены" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,133.596,0,0" Click="RadioButton\_Click"/>

<RadioButton Style="{StaticResource Hint}" x:Name="HintOff" GroupName="Hints" Content="Отключены" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,133.596,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource Hint}" Content="Подсказки" HorizontalAlignment="Left" Margin="32.408,127.596,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<RadioButton Style="{StaticResource NonLinear}" x:Name="NonLinearOn" GroupName="NonLinear" Content="Включено" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,172.12,0,0" Click="RadioButton\_Click"/>

<RadioButton Style="{StaticResource NonLinear}" x:Name="NonLinearOff" GroupName="NonLinear" Content="Отключено" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,172.12,0,0" RenderTransformOrigin="0.04,0.759" Click="RadioButton\_Click"/>

<Label Style="{StaticResource NonLinear}" Content="Свободное перемещение" HorizontalAlignment="Left" Margin="32.408,166.12,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

</Grid>

</TabItem>

<TabItem Header="Настройка коэффициентов" Style="{StaticResource TabItem}">

<Grid Style="{StaticResource TabItemGrid}">

<Grid.RowDefinitions>

<RowDefinition Height="0\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Секундомер" Grid.Row="1" Style="{StaticResource ResourceKey=Stopwatch}" FontWeight="Medium"/>

<TextBlock Text="Таймер" Grid.Row="2" Style="{StaticResource ResourceKey=Timer}" FontWeight="Medium"/>

<Grid Grid.Row="1" Grid.Column="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<RadioButton x:Name="StopwatchOn" Style="{StaticResource ResourceKey=Stopwatch}" Content="Включён" Click="RadioButton\_Click"/>

<RadioButton x:Name="StopwatchOff" Style="{StaticResource ResourceKey=Stopwatch}" Content="Выключен" Grid.Column="1" Click="RadioButton\_Click"/>

</Grid>

<Grid Grid.Row="2" Grid.Column="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<RadioButton x:Name="TimerOn" Style="{StaticResource ResourceKey=Timer}" Content="Включён" Click="RadioButton\_Click"/>

<RadioButton x:Name="TimerOff" Style="{StaticResource ResourceKey=Timer}" Content="Выключен" Grid.Column="1" Click="RadioButton\_Click"/>

</Grid>

<TextBlock Text="Секунд на прохождение теста:" Grid.Row="3" Style="{StaticResource ResourceKey=TextBloxStyle}"/>

<TextBlock Text="Выполнено заданий на 5 (%):" Grid.Row="4" Style="{StaticResource ResourceKey=TextBloxStyle}"/>

<TextBlock Text="Выполнено заданий на 4 (%):" Grid.Row="5" Style="{StaticResource ResourceKey=TextBloxStyle}"/>

<TextBlock Text="Выполнено заданий на 3 (%):" Grid.Row="6" Style="{StaticResource ResourceKey=TextBloxStyle}"/>

<TextBox x:Name="TestSeconds" Grid.Row="3" Grid.Column="1" Style="{StaticResource ResourceKey=TimerField}" TextChanged="TestSeconds\_TextChanged"/>

<TextBox x:Name="AMark" Grid.Row="4" Grid.Column="1" Text="90" Style="{StaticResource ResourceKey=TextBloxStyle}"/>

<TextBox x:Name="BMark" Grid.Row="5" Grid.Column="1" Text="75" Style="{StaticResource ResourceKey=TextBloxStyle}"/>

<TextBox x:Name="CMark" Grid.Row="6" Grid.Column="1" Text="50" Style="{StaticResource ResourceKey=TextBloxStyle}"/>

</Grid>

</TabItem>

<TabItem Header="Внешний вид" Style="{StaticResource TabItem}">

<Grid Style="{StaticResource TabItemGrid}" >

<RadioButton Style="{StaticResource Background}" x:Name="BackgroundOn" GroupName="UseBackground" Content="Включен" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,16.392,0,0" Click="Background\_Click"/>

<RadioButton Style="{StaticResource Background}" x:Name="BackgroundOff" GroupName="UseBackground" Content="Отключен" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,16.392,0,0" RenderTransformOrigin="0.04,0.759" Click="Background\_Click"/>

<Label Style="{StaticResource Background}" Content="Нестандартный фон" HorizontalAlignment="Left" Margin="32.408,10.392,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<RadioButton Style="{StaticResource Gradient}" x:Name="Gradient" GroupName="GradientPicture" Content="Использовать градиент" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="45,56.555,0,0" Click="GradientPicture\_Click"/>

<RadioButton Style="{StaticResource Gradient}" x:Name="Picture" GroupName="GradientPicture" Content="Использовать изображение" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="230,56.555,0,0" RenderTransformOrigin="0.04,0.759" Click="GradientPicture\_Click"/>

<Grid x:Name="gPicture" Margin="30,90,30,10" Background="#3FB4B4B4">

<Button x:Name="bPicture" Content="Выбрать изображение" HorizontalAlignment="Left" Margin="10,67,0,0" VerticalAlignment="Top" Width="132.377" Click="Picture\_Click"/>

<Grid x:Name="ggPicture" Background="#00000000" Margin="194.213,18.646,10,10"/>

</Grid>

<Grid x:Name="gGradient" Margin="30,90,30,10" Background="#3FB4B4B4">

<Slider x:Name="Alpha" Margin="200,9.835,10,66.244"/>

<Slider x:Name="Red" Margin="200,30.835,10,48.244">

<Slider.Background>

<LinearGradientBrush>

<GradientStop Color="#00FF0000" Offset="0" />

<GradientStop Color="Red" Offset="1" />

</LinearGradientBrush>

</Slider.Background>

</Slider>

<Slider x:Name="Green" Margin="200,51.835,10,27.244">

<Slider.Background>

<LinearGradientBrush>

<GradientStop Color="#0000FF00" Offset="0" />

<GradientStop Color="#FF00FF00" Offset="1" />

</LinearGradientBrush>

</Slider.Background>

</Slider>

<Slider x:Name="Blue" Margin="200,72.835,10,6.244">

<Slider.Background>

<LinearGradientBrush>

<GradientStop Color="#000000FF" Offset="0" />

<GradientStop Color="#FF0000FF" Offset="1" />

</LinearGradientBrush>

</Slider.Background>

</Slider>

<RadioButton x:Name="LT" IsChecked="True" Style="{StaticResource StyleButton}" Content="Левый верхний угол" Margin="10,10,250,70"/>

<RadioButton x:Name="RT" Style="{StaticResource StyleButton}" Content="Правый верхний угол" Margin="10,30,250,50"/>

<RadioButton x:Name="RB" Style="{StaticResource StyleButton}" Content="Правый нижний угол" Margin="10,50,250,30"/>

<RadioButton x:Name="LB" Style="{StaticResource StyleButton}" Content="Левый нижний угол" Margin="10,70,250,10"/>

</Grid>

</Grid>

</TabItem>

<TabItem Header="Настройки сети" Style="{StaticResource TabItem}">

<Grid Style="{StaticResource TabItemGrid}" >

<RadioButton x:Name="AutomationWebSettingsOn" Style="{StaticResource AutomationWebSettings}" GroupName="AutomationWebSettings" Content="Включена" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="232.786,16.392,0,0" Click="WebSettings\_Click"/>

<RadioButton x:Name="AutomationWebSettingsOff" Style="{StaticResource AutomationWebSettings}" GroupName="AutomationWebSettings" Content="Отключена" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="334.425,16.392,0,0" RenderTransformOrigin="0.04,0.759" Click="WebSettings\_Click"/>

<Label Style="{StaticResource AutomationWebSettings}" Content="Автоматическая настройка" HorizontalAlignment="Left" Margin="32.408,10.392,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<Grid x:Name="gWeb" Background="#3FD7D7D7">

<Grid.Margin>

<Thickness>15,50,15,5</Thickness>

</Grid.Margin>

<RadioButton x:Name="UseWebOn" Style="{StaticResource UseWeb}" GroupName="UseWeb" Content="Включены" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="217.786,16,0,0" Click="UseWeb\_Click"/>

<RadioButton x:Name="UseWebOff" Style="{StaticResource UseWeb}" GroupName="UseWeb" Content="Отключены" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="319.425,16,0,0" RenderTransformOrigin="0.04,0.759" Click="UseWeb\_Click"/>

<Label Style="{StaticResource UseWeb}" Content="Сетевые возможности" HorizontalAlignment="Left" Margin="17.408,10,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<TextBox x:Name="tbServerName" Style="{StaticResource ServerName}" TextChanged="Server\_TextChanged">

<TextBox.Margin>220, 47, 30, 68</TextBox.Margin>

<TextBox.Padding>0,5,0,0</TextBox.Padding>

</TextBox>

<Label Style="{StaticResource ServerName}" Content="Адрес сервера" HorizontalAlignment="Left" Margin="17.408,49.079,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<TextBox x:Name="tbServerPort" Style="{StaticResource ServerPort}" PreviewTextInput="ServerPort\_PreviewTextInput" TextChanged="Server\_TextChanged">

<TextBox.Margin>220, 86, 30, 29</TextBox.Margin>

<TextBox.Padding>0,5,0,0</TextBox.Padding>

</TextBox>

<Label Style="{StaticResource ServerPort}" Content="Порт сервера" HorizontalAlignment="Left" Margin="17.408,87.596,0,0" VerticalAlignment="Top" FontWeight="Medium"/>

<Button x:Name="bTestConnection" Click="TestConnection\_Click">

<Button.Margin>295, 118, 5, 5</Button.Margin>

<Button.Content>Проверить соединение</Button.Content>

<Button.Background>#7FDDDDDD</Button.Background>

<Button.BorderBrush>#3F707070</Button.BorderBrush>

</Button>

</Grid>

</Grid>

</TabItem>

<TabItem Header="Импорт/Экспорт" Style="{StaticResource TabItem}">

<Grid>

<Button x:Name="bImport" Click="bImport\_Click">

<Button.Background>#7FDDDDDD</Button.Background>

<Button.BorderBrush>#3F707070</Button.BorderBrush>

<Button.Content>Импорт</Button.Content>

<Button.Margin>10,100,300,50</Button.Margin>

</Button>

<Button x:Name="bExport" Click="Export\_Click">

<Button.Background>#7FDDDDDD</Button.Background>

<Button.BorderBrush>#3F707070</Button.BorderBrush>

<Button.Content>Экспорт</Button.Content>

<Button.Margin>300,100,10,50</Button.Margin>

</Button>

</Grid>

</TabItem>

</TabControl>

<!-- Подвал -->

<Button x:Name="bOK" Grid.Row="2" Grid.Column="1" Content="ОК" Margin="144,4,0,0" Style="{StaticResource ButtonStyle}" Click="OK\_Click" />

<Button x:Name="bCancel" Grid.Row="2" Grid.Column="1" Content="Отмена" Margin="269,4,0,0" Style="{StaticResource ButtonStyle}" Click="Cancel\_Click" />

<Button x:Name="bApply" Grid.Row="2" Grid.Column="1" Grid.ColumnSpan="2" Content="Применить" Margin="396,4,0,0" Style="{StaticResource ButtonStyle}" IsEnabled="False" Click="Apply\_Click" />

</Grid>

</Window>

Листинг модуля Test.SplashScreenWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using TestPro.Raven;

using TestPro.Utils;

using TestPro.Test.Main;

using TestPro.TestKinds;

namespace TestPro.Test

{

/// <summary>

/// Логика взаимодействия для SplashScreenWindow.xaml

/// </summary>

public partial class SplashScreenWindow : Window

{

/// <summary>

/// Базовый конструктор окна заставки

/// </summary>

public SplashScreenWindow()

{

Logger.Log("TP.T.SSW - Инициализация");

InitializeComponent();

EndInitialize += SplashScreenWindow\_Initialized;

Logger.Log("TP.T.SSW - Подписка на события");

}

/// <summary>

/// Обработчик события завершения инициализации

/// </summary>

private void SplashScreenWindow\_Initialized()

{

if (showed && loaded)

{

Logger.Log("TP.T.SSW - Инициализация приложения завершена");

new MainWindow(tests).Show();

Close();

}

}

delegate void EndInitializeHandler();

event EndInitializeHandler EndInitialize;

bool showed = false, loaded = false;

List<string> tests;

/// <summary>

/// Активизация окна, изменение прозрачности

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_Activated(object sender, EventArgs e)

{

System.Timers.Timer timer = new System.Timers.Timer(30);

timer.Elapsed += (s, er) =>

{

double opacity = 0;

Dispatcher.Invoke(() => { Opacity += 0.01; opacity = Opacity; });

if ((int)opacity == 1)

{

timer.Stop();

Dispatcher.Invoke(() =>

{

showed = true;

EndInitialize();

Logger.Log("TP.T.SSW - Инициализация завершена");

});

}

};

timer.Start();

Load();

}

/// <summary>

/// Загрузка данных из бд

/// </summary>

private void Load()

{

Thread thread = new Thread(

() =>

{

using (var raven = new RavenDB())

{

//raven.AddTest(@"C:\Users\zheny\Documents\Visual Studio 2017\Projects\WPFTest\WpfApp1\bin\Debug\Tests\NewTest2.tsz");

Dispatcher.Invoke(() => tests = raven.GetSubjects());

}

Dispatcher.Invoke(() =>

{

loaded = true;

EndInitialize();

Logger.Log("TP.T.SSW - Загрузка данных из БД закончена");

});

})

{

Name = "TestPro.Load",

IsBackground = true

};

thread.Start();

Logger.Log("TP.T.SSW - Загрузка данных из БД начата");

}

}

}

Листинг модуля Test.SplashScreenWindow.xaml

<Window x:Class="TestPro.Test.SplashScreenWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Test"

mc:Ignorable="d"

Title="TestPro" Height="300" Width="600" WindowStyle="None" AllowsTransparency="True" Topmost="True"

Activated="Window\_Activated" OpacityMask="White" ResizeMode="NoResize" WindowStartupLocation="CenterScreen"

Opacity="0">

<Window.Background>

<ImageBrush ImageSource="../Images/SplashScreen.jpg"/>

</Window.Background>

<Window.TaskbarItemInfo>

<TaskbarItemInfo ProgressState="Indeterminate"/>

</Window.TaskbarItemInfo>

<Grid>

<TextBlock x:Name="textBlock" HorizontalAlignment="Left" Margin="33,22,0,0" TextWrapping="Wrap" Text="TestPro" VerticalAlignment="Top" Height="67" Width="227" FontFamily="Ravie" FontSize="36"/>

<Image x:Name="image" Source="../Images/TestProLogo.ico" HorizontalAlignment="Left" Height="100" Margin="500,200,0,0" VerticalAlignment="Top" Width="100"/>

</Grid>

</Window>

Листинг модуля Test.TestInfoWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using TestPro.Raven;

using TestPro.Utils;

namespace TestPro.Test

{

/// <summary>

/// Логика взаимодействия для TestInfoWindow.xaml

/// </summary>

public partial class TestInfoWindow : Window

{

/// <summary>

/// Базовый конструктор окна информации о тесте

/// </summary>

public TestInfoWindow()

{

Logger.Log("TP.T.TIW - Инициализация окна информации о тесте");

InitializeComponent();

}

/// <summary>

/// Конструктор окна по заданному тесту

/// </summary>

/// <param name="info">Заданный тест</param>

public TestInfoWindow(TestInfo info) : this()

{

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

list.Add($"Автор: {info.Author}");

list.Add($"Название: {info.Name}");

list.Add($"Предмет: {info.Subject}");

list.Add($"Количество вопросов: {info.QuestionsCount}");

list.Add($"Дата создания: {info.Date}");

list.Add($"Таймер: {info.Timer.ToString().Replace("True", "Включён").Replace("False", "Выключен")}");

list.Add($"Секундомер: {info.Stopwatch.ToString().Replace("True", "Включён").Replace("False", "Выключен")}");

string s = "";

foreach (var e in list)

s += e + Environment.NewLine;

LogInfo.Text = s;

Logger.Log("TP.T.TIW - Загрузка информации о тесте");

}

/// <summary>

/// Закрытие окна по нажатию на кнопку закрытия окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void EscapeActive\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Close();

Logger.Log("TP.T.TIW - Закрытие окна");

}

}

}

Листинг модуля Test.TestInfoWindow.xaml

<Window x:Class="TestPro.Test.TestInfoWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Test"

mc:Ignorable="d"

Title="Информация о тесте" Height="225" Width="400" MinHeight="225" MinWidth="400" AllowsTransparency="True" WindowStyle="None" Foreground="#00000000" Background="Transparent" Topmost="True" WindowStartupLocation="CenterScreen">

<WindowChrome.WindowChrome>

<WindowChrome x:Name="Chrome" CaptionHeight="21" ResizeBorderThickness="2.5"/>

</WindowChrome.WindowChrome>

<Window.Resources>

<Style x:Key="Escape">

<Setter Property="WindowChrome.IsHitTestVisibleInChrome" Value="True" />

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Control.Width" Value="35" />

<Setter Property="Control.HorizontalAlignment" Value="Right" />

<Setter Property="Image.Source" Value="../Images/General/Escape/Escape.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="EscapeActive\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="../Images/General/Escape/EscapeActive.png" />

</Trigger>

</Style.Triggers>

</Style>

</Window.Resources>

<Grid x:Name="MainGrid" Background="#00000000">

<Grid.RowDefinitions>

<RowDefinition x:Name="TitleRow" Height="21"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="21"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="35"/>

</Grid.ColumnDefinitions>

<ScrollViewer Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3">

<TextBlock x:Name="LogInfo" TextWrapping="Wrap" Foreground="#FF01151A" FontSize="20" ScrollViewer.VerticalScrollBarVisibility="Auto" Padding="15,10">

<TextBlock.Background>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="#1921B6DA" Offset="1"/>

<GradientStop Color="#33000CFF"/>

</LinearGradientBrush>

</TextBlock.Background>

</TextBlock>

</ScrollViewer>

<Rectangle x:Name="TitleBlock" Margin="0,0,0,0" Stroke="Orange" Fill="Orange" Grid.ColumnSpan="5" Grid.IsSharedSizeScope="True"/>

<Image x:Name="TitleLogo" Margin="0" Source="../Images/TestProLogo.ico" Stretch="Fill" HorizontalAlignment="Left" Width="21"/>

<Image x:Name="Escape" WindowChrome.IsHitTestVisibleInChrome="True" Style="{StaticResource Escape}" Grid.Column="2"/>

<TextBlock x:Name="TitleText" Margin="5,0,0,0" TextWrapping="Wrap" Text="Информация о тесте" Foreground="Black" FontWeight="Bold" FontStyle="Italic" TextDecorations="Underline" Grid.Column="1" Padding="0,2,0,0"/>

</Grid>

</Window>

Листинг модуля Test.WindowsApi.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Runtime.InteropServices;

using System.Windows;

using System.Windows.Interop;

using TestPro.Test.Main;

namespace TestPro.Test

{

class NativeMethods

{

[DllImport("user32.dll", SetLastError = true, CharSet = CharSet.Unicode)]

public static extern uint SetForegroundWindow(IntPtr hWnd);

[DllImport("user32.dll", SetLastError = true, CharSet = CharSet.Unicode)]

public static extern IntPtr FindWindow(string lpClassName, string lpWindowName);

[DllImport("user32.dll", SetLastError = true, CharSet = CharSet.Unicode)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);

[DllImport("user32.dll", SetLastError = true, CharSet = CharSet.Unicode)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool ShowWindowAsync(IntPtr hWnd, int nCmdShow);

enum showWindow

{

Hide = 0,

ShowNormal = 1,

Normal = 1,

ShowMinimized = 2,

ShowMaximized = 3,

Maximized = 3,

ShowNoActivate = 4,

Show = 5,

Minimize = 6,

ShowMinNoActive = 7,

ShowNa = 8,

Restore = 9,

ShowDefault = 10,

ForceMinimize = 11,

Max = 11,

}

enum Status

{

Hidding,

Showing

}

Status status = Status.Showing;

public void SetForeground(Window window)

{

unsafe

{

WindowInteropHelper w = new WindowInteropHelper(window);

var u = SetForegroundWindow(w.Handle);

}

}

public void Hide()

{

if (status == Status.Hidding) return;

if (Settings.TaskManager)

{

IntPtr taskManager = FindWindow("TaskManagerWindow", null);

if (taskManager != IntPtr.Zero)

{

bool result = ShowWindow(taskManager, (int)showWindow.Hide);

result = ShowWindowAsync(taskManager, (int)showWindow.Show);

}

}

if (Settings.TaskPanel)

{

IntPtr taskPanel = FindWindow("Shell\_TrayWnd", null);

if (taskPanel != IntPtr.Zero)

{

bool result = ShowWindow(taskPanel, (int)showWindow.Hide);

result = ShowWindowAsync(taskPanel, (int)showWindow.Show);

}

}

status = Status.Hidding;

}

public void Show()

{

if (status == Status.Showing) return;

if (Settings.TaskManager)

{

IntPtr taskManager = FindWindow("TaskManagerWindow", null);

if (taskManager != IntPtr.Zero)

{

ShowWindow(taskManager, (int)showWindow.Show);

}

}

if (Settings.TaskPanel)

{

IntPtr taskPanel = FindWindow("Shell\_TrayWnd", null);

if (taskPanel != IntPtr.Zero)

{

ShowWindow(taskPanel, (int)showWindow.Show);

}

}

status = Status.Showing;

}

}

}

Листинг модуля Test.Main.BasicAnswer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestPro.Test.Main

{

public class BasicAnswer : ICloneable

{

public BasicAnswer()

{

Answer = "";

StopwatchBegin = 0;

StopwatchEnd = 0;

TimerBegin = 0;

TimerEnd = 0;

DecideChanges = 0;

Done = false;

}

public BasicAnswer(string aAnswer, int aStopwatchBegin, int aStopwatchEnd, int aTimerBegin, int aTimerEnd, int aDecideChanges, bool aDone)

{

Answer = aAnswer;

StopwatchBegin = aStopwatchBegin;

StopwatchEnd = aStopwatchEnd;

TimerBegin = aTimerBegin;

TimerEnd = aTimerEnd;

DecideChanges = aDecideChanges;

Done = aDone;

}

private string answer;

public string Answer

{

get { return answer; }

set { answer = value; }

}

private int stopwatchBegin, stopwatchEnd, timerBegin, timerEnd;

public int StopwatchBegin

{

get { return stopwatchBegin; }

set { stopwatchBegin = value; }

}

public int StopwatchEnd

{

get { return stopwatchEnd; }

set { stopwatchEnd = value; }

}

public int TimerBegin

{

get { return timerBegin; }

set { timerBegin = value; }

}

public int TimerEnd

{

get { return timerEnd; }

set { timerEnd = value; }

}

public int GetNeedTime()

{

return TimerBegin - TimerEnd;

}

public object Clone()

{

BasicAnswer ba = new BasicAnswer();

ba.answer = answer;

ba.decideChanges = decideChanges;

ba.done = done;

ba.stopwatchBegin = stopwatchBegin;

ba.stopwatchEnd = stopwatchEnd;

ba.timerBegin = timerBegin;

ba.timerEnd = timerEnd;

return ba;

}

private int decideChanges;

public int DecideChanges

{

get { return decideChanges; }

set { decideChanges = value; }

}

public bool Done { get => done; set => done = value; }

private bool done;

}

}

Листинг модуля Test.Main.Metadata.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using TestPro.Utils;

namespace TestPro.Test.Main

{

public static class Metadata

{

public static string Id { get; set; }

private static string name, author, subject, date, version, programVersion;

private static bool taskManager, taskPanel, otherApps, questionMeshing, answersMeshing;

private static bool draft, calc, inc, hint, back;

private static bool stopwatch, timer;

private static int timerValue, questionsCount = -1;

private static bool? elseAllow;

public static string Name { get => name; set => name = value; }

public static string Author { get => author; set => author = value; }

public static string Date { get => date; set => date = value; }

public static string Version { get => version; set => version = value; }

public static string ProgramVersion { get => programVersion; set => programVersion = value; }

public static string Subject { get => subject; set => subject = value; }

public static bool TaskManager { get => taskManager; set => taskManager = value; }

public static bool TaskPanel { get => taskPanel; set => taskPanel = value; }

public static bool OtherApps { get => otherApps; set => otherApps = value; }

public static bool QuestionMeshing { get => questionMeshing; set => questionMeshing = value; }

public static bool AnswersMeshing { get => answersMeshing; set => answersMeshing = value; }

public static bool Draft { get => draft; set => draft = value; }

public static bool Calc { get => calc; set => calc = value; }

public static bool Back { get => back; set => back = value; }

public static bool Hint { get => hint; set => hint = value; }

public static bool Stopwatch { get => stopwatch; set => stopwatch = value; }

public static bool Timer { get => timer; set => timer = value; }

public static int TimerValue { get => timerValue; set => timerValue = value; }

public static int QuestionsCount { get => questionsCount; set => questionsCount = value; }

public static bool Inc { get => inc; set => inc = value; }

public static bool? ElseAllow { get => elseAllow; set => elseAllow = value; }

}

}

Листинг модуля Test.Main.Report.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using Microsoft.Office.Interop.Excel;

using System.Runtime.InteropServices;

using TestPro.TestKinds;

namespace TestPro.Test.Main

{

/// <summary>

/// Изменение этапа создания отчёта

/// </summary>

/// <param name="sender">Экземпляр класса, сгенерировавший событие</param>

/// <param name="NewValue">Текущее этап сохранения</param>

public delegate void ValueChangeHandler(object sender, double NewValue);

/// <summary>

/// Начало формирования отчёта

/// </summary>

/// <param name="sender">Экземпляр класса, сгенерировавший событие</param>

/// <param name="Pass">Разрешение, либо запрет формирования отчёта</param>

public delegate void SaveStartHandler(object sender, ref bool Pass);

/// <summary>

/// Завершение формирования отчёта

/// </summary>

/// <param name="sender">Экземпляр класса, сгенерировавший событие</param>

public delegate void SaveEndHandler(object sender);

/// <summary>

/// Отмена формирования отчёта

/// </summary>

/// <param name="sender"></param>

/// <param name="quest"></param>

public delegate void SaveAbortedHandler(object sender, string quest);

public class Report : IDisposable

{

public Report()

{

excelApp = new Application();

WorkbookIndex = excelApp.Workbooks.Count;

ValueChange += ValueListner;

SaveStart += Starting;

SaveEnd += s => { };

SaveAborted += (s, q) => { };

}

public event ValueChangeHandler ValueChange;

public event SaveStartHandler SaveStart;

public event SaveEndHandler SaveEnd;

public event SaveAbortedHandler SaveAborted;

Application excelApp;

Workbook workbook;

Worksheet worksheet;

int WorkbookIndex;

private double \_Value = 0;

public double Value { get => \_Value; set => \_Value = value; }

private string \_ProgressInfo;

public string ProgressInfo { get => \_ProgressInfo; set => \_ProgressInfo = value; }

public void Save()

{

bool Pass = true;

SaveStart(this, ref Pass);

if (!Pass)

{

Value = 0;

return;

}

try

{

Forming();

ValueChange(this, Value);

ProgressInfo = "Применение стилей";

CellsFormating(9, 9 + Metadata.QuestionsCount);

Value = 55;

ValueChange(this, Value);

ProgressInfo = "Выгрузка данных в таблицу";

for (int i = 0; i < Metadata.QuestionsCount; i++)

{

worksheet.Cells[i + 9, 1].Value = Convert.ToString(i + 1);

worksheet.Cells[i + 9, 2].Value = Test.Questions[i].Question;

worksheet.Cells[i + 9, 3].Value = (Test.Questions[i] is BigTest btt) ? btt.Answers[Convert.ToInt32(Test.Answers[i].Answer) - 1] : Test.Answers[i].Answer;

worksheet.Cells[i + 9, 4].Value = (Test.Questions[i] is BigTest bt) ? bt.Answers[Convert.ToInt32(bt.RightAnswer) - 1] : Test.Questions[i].RightAnswer;

worksheet.Cells[i + 9, 5].Value = Test.Questions[i].Hint;

worksheet.Cells[i + 9, 6].Value = Test.Questions[i].Solution;

worksheet.Cells[i + 9, 7].Value = Settings.IntToTime(Test.Answers[i].StopwatchEnd - Test.Answers[i].StopwatchBegin);

worksheet.Cells[i + 9, 8].Value = Test.Questions[i].Picture.Status ? "Имеется" : "Отсутствует";

}

Value = 80;

ValueChange(this, Value++);

ProgressInfo = "Завершение формирования отчёта";

excelApp.DisplayInfoWindow = false;

ValueChange(this, Value++);

excelApp.Application.DisplayAlerts = false;

ValueChange(this, Value++);

workbook.Author = "Test";

ValueChange(this, Value++);

worksheet.Protect(Password: "1234", DrawingObjects: true, Contents: true, Scenarios: true, AllowFormattingCells: true, AllowFormattingColumns: true, AllowFormattingRows: true);

ValueChange(this, Value++);

worksheet.EnableSelection = XlEnableSelection.xlNoSelection;

ValueChange(this, Value++);

workbook.Protect("1234", true, true);

ValueChange(this, Value++);

workbook.Final = true;

ValueChange(this, Value);

excelApp.UserName = "Test";

Value = 90;

ValueChange(this, Value);

}

catch (COMException)

{

ProgressInfo = "Отмена создания книги Excel";

Value = 100;

ValueChange(this, Value);

SaveAborted(this, ProgressInfo);

}

catch

{

SaveAborted(this, ProgressInfo);

}

ProgressInfo = "Отчёт создан";

Value = 100;

ValueChange(this, Value);

SaveEnd(this);

}

private void Forming()

{

try

{

ProgressInfo = "Создание книги Excel";

workbook = excelApp.Workbooks.Add();

worksheet = workbook.Worksheets[1];

Value = 1;

ValueChange(this, Value);

WorkbookIndex = excelApp.Workbooks.Count;

ProgressInfo = "Начало формирования таблицы";

ValueChange(this, ++Value);

excelApp.Range["A1:H1"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

Value = 5;

ValueChange(this, Value++);

ProgressInfo = "Работа с оформлением";

excelApp.Range["A2:B2"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

ValueChange(this, Value++);

excelApp.Range["A3:B3"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

ValueChange(this, Value++);

excelApp.Range["A4:B4"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

ValueChange(this, Value++);

excelApp.Range["A5:B5"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

ValueChange(this, Value++);

excelApp.Range["C2:H2"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignRight;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

ValueChange(this, Value++);

excelApp.Range["C3:H3"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignRight;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

ValueChange(this, Value++);

excelApp.Range["C4:H4"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignRight;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

ValueChange(this, Value++);

excelApp.Range["C5:H5"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignRight;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

ValueChange(this, Value++);

excelApp.Range["A7:H7"].Select();

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.Merge();

ValueChange(this, Value++);

excelApp.Range["A8"].Select();

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

ValueChange(this, Value++);

excelApp.Range["B8"].Select();

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

ValueChange(this, Value++);

excelApp.Range["C8"].Select();

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

ValueChange(this, Value++);

excelApp.Range["D8"].Select();

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

ValueChange(this, Value++);

excelApp.Range["E8"].Select();

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

ValueChange(this, Value++);

excelApp.Range["F8"].Select();

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

ValueChange(this, Value++);

excelApp.Range["G8"].Select();

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

ValueChange(this, Value++);

excelApp.Range["H8"].Select();

excelApp.Selection.VerticalAlignment = XlHAlign.xlHAlignCenter;

excelApp.Selection.HorizontalAlignment = XlHAlign.xlHAlignLeft;

ValueChange(this, Value++);

excelApp.Range["A1:H5"].Select();

excelApp.Selection.NumberFormat = "@";

ValueChange(this, Value++);

excelApp.Range["A7:H7"].Select();

excelApp.Selection.NumberFormat = "@";

ProgressInfo = "Внесение общей информации";

ValueChange(this, Value++);

worksheet.Cells[1, 1].Value = "Информация о тестируемом:";

ValueChange(this, Value++);

worksheet.Cells[2, 1].Value = "Фамилия:";

worksheet.Cells[2, 3].Value = string.IsNullOrWhiteSpace(Test.CurrentTester.Surname) ? "Не указано" : Test.CurrentTester.Surname;

ValueChange(this, Value++);

worksheet.Cells[3, 1].Value = "Имя:";

worksheet.Cells[3, 3].Value = string.IsNullOrWhiteSpace(Test.CurrentTester.Name) ? "Не указано" : Test.CurrentTester.Name;

ValueChange(this, Value++);

worksheet.Cells[4, 1].Value = "Отчество:";

worksheet.Cells[4, 3].Value = string.IsNullOrWhiteSpace(Test.CurrentTester.Middlename) ? "Не указано" : Test.CurrentTester.Middlename;

ValueChange(this, Value++);

worksheet.Cells[5, 1].Value = "Группа:";

worksheet.Cells[5, 3].Value = string.IsNullOrWhiteSpace(Test.CurrentTester.Group) ? "Не указано" : Test.CurrentTester.Group;

ValueChange(this, Value++);

worksheet.Cells[7, 1].Value = "Результаты:";

ValueChange(this, Value++);

worksheet.Cells[8, 1].Value = "Номер вопроса:";

ValueChange(this, Value++);

worksheet.Cells[8, 2].Value = "Вопрос:";

ValueChange(this, Value++);

worksheet.Cells[8, 3].Value = "Ответ:";

ValueChange(this, Value++);

worksheet.Cells[8, 4].Value = "Верный ответ:";

ValueChange(this, Value++);

worksheet.Cells[8, 5].Value = "Подсказка:";

ValueChange(this, Value++);

worksheet.Cells[8, 6].Value = "Решение:";

ValueChange(this, Value++);

worksheet.Cells[8, 7].Value = "Время:";

ValueChange(this, Value++);

worksheet.Cells[8, 8].Value = "Иллюстрация:";

ValueChange(this, Value++);

excelApp.Cells.Select();

excelApp.Selection.Font.Name = "Segoe UI";

ProgressInfo = "Форматирование документа";

ValueChange(this, Value++);

excelApp.Selection.Font.Size = 12;

excelApp.Selection.Font.Strikethrough = false;

excelApp.Selection.Font.Superscript = false;

excelApp.Selection.Font.Subscript = false;

excelApp.Selection.Font.OutlineFont = false;

excelApp.Selection.Font.Shadow = false;

excelApp.Selection.Font.Underline = XlUnderlineStyle.xlUnderlineStyleNone;

excelApp.Selection.Font.ThemeColor = XlThemeColor.xlThemeColorLight1;

excelApp.Selection.Font.TintAndShade = 0;

excelApp.Selection.Font.ThemeFont = XlThemeFont.xlThemeFontNone;

ValueChange(this, Value++);

excelApp.Columns["A:A"].ColumnWidth = 16.7;

excelApp.Columns["D:D"].ColumnWidth = 18;

excelApp.Columns["D:D"].ColumnWidth = 14.7;

excelApp.Columns["E:E"].ColumnWidth = 10;

excelApp.Columns["E:E"].ColumnWidth = 12.4;

excelApp.Columns["E:E"].ColumnWidth = 11.5;

excelApp.Columns["F:F"].ColumnWidth = 10.6;

excelApp.Columns["F:F"].ColumnWidth = 9.8;

excelApp.Columns["G:G"].ColumnWidth = 6.8;

excelApp.Columns["H:H"].ColumnWidth = 14.6;

excelApp.Columns["C:C"].ColumnWidth = 6.4;

ValueChange(this, Value++);

excelApp.Range["A1:H5"].Select();

excelApp.Selection.Borders(XlBordersIndex.xlEdgeLeft).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeLeft).Weight = XlBorderWeight.xlMedium;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeTop).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeTop).Weight = XlBorderWeight.xlMedium;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeBottom).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeBottom).Weight = XlBorderWeight.xlMedium;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeRight).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeRight).Weight = XlBorderWeight.xlMedium;

excelApp.Selection.Borders(XlBordersIndex.xlInsideVertical).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlInsideVertical).Weight = XlBorderWeight.xlThin;

excelApp.Selection.Borders(XlBordersIndex.xlInsideHorizontal).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlInsideHorizontal).Weight = XlBorderWeight.xlThin;

excelApp.Selection.Borders(XlBordersIndex.xlDiagonalDown).LineStyle = XlLineStyle.xlLineStyleNone;

excelApp.Selection.Borders(XlBordersIndex.xlDiagonalUp).LineStyle = XlLineStyle.xlLineStyleNone;

ValueChange(this, Value++);

excelApp.Range["A7:H8"].Select();

excelApp.Selection.Borders(XlBordersIndex.xlDiagonalDown).LineStyle = XlLineStyle.xlLineStyleNone;

excelApp.Selection.Borders(XlBordersIndex.xlDiagonalUp).LineStyle = XlLineStyle.xlLineStyleNone;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeLeft).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeLeft).Weight = XlBorderWeight.xlMedium;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeTop).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeTop).Weight = XlBorderWeight.xlMedium;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeBottom).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeBottom).Weight = XlBorderWeight.xlThin;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeRight).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlEdgeRight).Weight = XlBorderWeight.xlMedium;

excelApp.Selection.Borders(XlBordersIndex.xlInsideVertical).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlInsideVertical).Weight = XlBorderWeight.xlThin;

excelApp.Selection.Borders(XlBordersIndex.xlInsideHorizontal).LineStyle = XlLineStyle.xlContinuous;

excelApp.Selection.Borders(XlBordersIndex.xlInsideHorizontal).Weight = XlBorderWeight.xlThin;

ValueChange(this, Value++);

worksheet.Name = "Результаты";

ValueChange(this, Value++);

excelApp.Range["A1:H1"].Select();

ValueChange(this, Value);

}

catch

{

SaveAborted(this, ProgressInfo);

}

}

private void CellFormating(int Index)

{

string Cell = "=$D$" + Convert.ToString(Index);

try

{

excelApp.Range["$C$" + Convert.ToString(Index)].Select();

excelApp.Selection.FormatConditions.Add(XlFormatConditionType.xlCellValue, XlFormatConditionOperator.xlEqual, Cell);

excelApp.Selection.FormatConditions(excelApp.Selection.FormatConditions.Count).SetFirstPriority();

excelApp.Selection.FormatConditions(1).Font.Color = -16752384;

excelApp.Selection.FormatConditions(1).Font.TintAndShade = 0;

excelApp.Selection.FormatConditions(1).Interior.PatternColorIndex = Constants.xlAutomatic;

excelApp.Selection.FormatConditions(1).Interior.Color = 13561798;

excelApp.Selection.FormatConditions(1).Interior.TintAndShade = 0;

excelApp.Selection.FormatConditions(1).StopIfTrue = false;

excelApp.Selection.FormatConditions.Add(XlFormatConditionType.xlCellValue, XlFormatConditionOperator.xlGreater, Cell);

excelApp.Selection.FormatConditions(excelApp.Selection.FormatConditions.Count).SetFirstPriority();

excelApp.Selection.FormatConditions(1).Font.Color = -16383844;

excelApp.Selection.FormatConditions(1).Font.TintAndShade = 0;

excelApp.Selection.FormatConditions(1).Interior.PatternColorIndex = Constants.xlAutomatic;

excelApp.Selection.FormatConditions(1).Interior.Color = 13551615;

excelApp.Selection.FormatConditions(1).Interior.TintAndShade = 0;

excelApp.Selection.FormatConditions(1).StopIfTrue = false;

excelApp.Selection.FormatConditions.Add(XlFormatConditionType.xlCellValue, XlFormatConditionOperator.xlLess, Cell);

excelApp.Selection.FormatConditions(excelApp.Selection.FormatConditions.Count).SetFirstPriority();

excelApp.Selection.FormatConditions(1).Font.Color = -16383844;

excelApp.Selection.FormatConditions(1).Font.TintAndShade = 0;

excelApp.Selection.FormatConditions(1).Interior.PatternColorIndex = Constants.xlAutomatic;

excelApp.Selection.FormatConditions(1).Interior.Color = 13551615;

excelApp.Selection.FormatConditions(1).Interior.TintAndShade = 0;

excelApp.Selection.FormatConditions(1).StopIfTrue = false;

excelApp.Range["A1:H1"].Select();

}

catch

{

SaveAborted(this, ProgressInfo);

}

}

private void CellsFormating(int Min, int Max)

{

try

{

if (Max <= 8 || Min <= 8) throw new IndexOutOfRangeException("Ошибка при выгрузке в Microsoft Excel");

if (Max < Min) throw new IndexOutOfRangeException("Ошибка при выгрузке в Microsoft Excel");

for (int i = Min; i < Max; i++) CellFormating(i);

}

catch (IndexOutOfRangeException)

{

SaveAborted(this, "Ошибка при выгрузке в Microsoft Excel: выход за границы");

}

catch

{

SaveAborted(this, ProgressInfo);

}

}

private void ValueListner(object sender, double NewValue)

{

//if (NewValue > 100) Value = 0;

}

private void Starting(object sender, ref bool Pass)

{

Pass = true;

}

#region IDisposable Support

private bool disposedValue = false;

protected virtual void Dispose(bool disposing)

{

if (!disposedValue)

{

if (disposing)

{

// TODO: освободить управляемое состояние (управляемые объекты).

}

if (worksheet != null)

{

Marshal.ReleaseComObject(worksheet);

worksheet = null;

}

if (workbook != null)

{

workbook.Close(false, false, false);

Marshal.ReleaseComObject(workbook);

workbook = null;

}

if (excelApp != null)

{

excelApp.Quit();

Marshal.ReleaseComObject(excelApp);

excelApp = null;

}

disposedValue = true;

}

}

~Report()

{

Dispose(false);

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

#endregion

}

}

Листинг модуля Test.Main.Settings.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Windows.Media;

using TestPro.Utils;

using TestPro.Visual.Core;

namespace TestPro.Test.Main

{

/// <summary>

/// Изменение заставки

/// </summary>

public delegate void BackgroundNameChangeHandler();

/// <summary>

/// Кастомный фон активирован

/// </summary>

public delegate void BackgroundOnHandler();

/// <summary>

/// Кастомный фон деактивирован

/// </summary>

public delegate void BackgroundOffHandler();

/// <summary>

/// Изменение настроек

/// </summary>

public delegate void SettingsChangeHandler();

public static class Settings

{

public static event GradientChangeHandler GradientChange;

public static event BackgroundNameChangeHandler BackgroundNameChange;

public static event BackgroundOnHandler BackgroundOn;

public static event BackgroundOffHandler BackgroundOff;

public static event SettingsChangeHandler SettingsChange;

static Settings()

{

SettingsChange += Settings\_SettingsChange;

GradientChange += Settings\_SettingsChange;

BackgroundNameChange += Settings\_SettingsChange;

BackgroundOn += Settings\_SettingsChange;

BackgroundOff += Settings\_SettingsChange;

if (!Check())

{

TaskManager = true;

TaskPanel = true;

OtherApps = false;

Mark = (0.5f, 0.75f, 0.9f);

QuestionMeshing = true;

AnswersMeshing = true;

Draft = true;

Calc = true;

Inc = true;

Back = true;

Hint = true;

Stopwatch = true;

Timer = false;

TimerValue = 60;

StopwatchValue = 0;

SecondsOnQuestion = 30;

UseBackground = false;

BackgroundName = "Background\_1";

Gradient = new Gradient("ff7611", "ffffff", "42ffae", "5143fe");

Gradient.GradientChange += Gradient\_GradientChange;

Gradient.Active = true;

ServerAdress = "localhost";

ServerPort = "8090";

}

}

public static readonly string SettingsPath = $@"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\TestPro\Test\Settings.fts";

private static bool taskManager, taskPanel, otherApps, questionMeshing, answersMeshing;

private static bool draft, calc, inc, hint, back;

private static bool stopwatch, timer;

private static bool useWeb, useWebDefault;

private static int secondsOnQuestion;

private static bool useBackground;

private static string backgroundName;

private static string serverAdress, serverPort;

private static int timerValue, stopwatchValue;

private static Gradient gradient;

private static (float D, float C, float B) mark;

public static Gradient Gradient

{

get

{

return gradient;

}

set

{

gradient = value;

SettingsChange();

GradientChange();

}

}

private static void Gradient\_GradientChange()

{

GradientChange();

}

public static bool TaskManager

{

get

{

return taskManager;

}

set

{

taskManager = value;

SettingsChange();

}

}

public static bool TaskPanel

{

get

{

return taskPanel;

}

set

{

taskPanel = value;

SettingsChange();

}

}

public static bool OtherApps

{

get

{

return otherApps;

}

set

{

otherApps = value;

SettingsChange();

}

}

public static bool QuestionMeshing

{

get

{

return questionMeshing;

}

set

{

questionMeshing = value;

SettingsChange();

}

}

public static bool AnswersMeshing

{

get

{

return answersMeshing;

}

set

{

answersMeshing = value;

SettingsChange();

}

}

public static bool Draft

{

get

{

return draft;

}

set

{

draft = value;

SettingsChange();

}

}

public static bool Calc

{

get

{

return calc;

}

set

{

calc = value;

SettingsChange();

}

}

public static bool Inc

{

get

{

return inc;

}

set

{

inc = value;

SettingsChange();

}

}

public static bool Back

{

get

{

return back;

}

set

{

back = value;

SettingsChange();

}

}

public static bool Hint

{

get

{

return hint;

}

set

{

hint = value;

SettingsChange();

}

}

public static bool Stopwatch

{

get

{

return stopwatch;

}

set

{

stopwatch = value;

SettingsChange();

}

}

public static bool Timer

{

get

{

return timer;

}

set

{

timer = value;

SettingsChange();

}

}

public static bool UseBackground

{

get

{

return useBackground;

}

set

{

useBackground = value;

SettingsChange();

if (value == true) BackgroundOn(); else BackgroundOff();

}

}

public static bool UseWeb

{

get

{

return useWeb;

}

set

{

useWeb = value;

SettingsChange();

}

}

public static bool UseWebDefault

{

get

{

return useWebDefault;

}

set

{

useWebDefault = value;

SettingsChange();

}

}

public static string BackgroundName

{

get

{

return backgroundName;

}

set

{

backgroundName = value;

SettingsChange();

BackgroundNameChange();

}

}

public static string ServerAdress

{

get

{

return serverAdress;

}

set

{

value = value.TrimStart(new char[9] { '/', '\\', ':', ';', '!', '?', '^', '&', '\*' });

int i = value.IndexOf(":");

int j;

if (i != -1 && value.IndexOf("://") != i)

{

string buffer = value.Substring(i);

j = buffer.IndexOf("/");

if (j == -1)

value = value.Substring(0, i - 1);

else

value = value.Substring(0, i - 1) + buffer.Substring(j);

}

i = value.IndexOf("http://");

j = value.IndexOf("https://");

if (i == -1 && j == -1)

value = "http://" + value;

serverAdress = value.Trim();

SettingsChange();

}

}

public static string ServerPort

{

get

{

return serverPort;

}

set

{

int i = 0;

if (!int.TryParse(value, out i)) throw new InvalidCastException();

if (i <= 0) throw new InvalidCastException();

if (i > Math.Abs((int)Int16.MinValue) + Int16.MaxValue) throw new InvalidCastException();

serverPort = value;

SettingsChange();

}

}

public static int TimerValue

{

get

{

return timerValue;

}

set

{

timerValue = value;

SettingsChange();

}

}

public static int StopwatchValue

{

get

{

return stopwatchValue;

}

set

{

stopwatchValue = value;

SettingsChange();

}

}

public static int SecondsOnQuestion

{

get

{

return secondsOnQuestion;

}

set

{

secondsOnQuestion = value;

SettingsChange();

}

}

public static (float D, float C, float B) Mark

{

get => mark;

set

{

mark = value;

SettingsChange();

}

}

static public string IntToTime(int Time)

{

if (Time >= 60)

{

int f = Time / 60;

int g = Time - f \* 60;

return Convert.ToString(f) + ":" + ((g >= 10) ? Convert.ToString(g) : "0" + Convert.ToString(g));

}

else

{

if (Time >= 10)

{

return "0:" + Convert.ToString(Time);

}

else

{

return "0:0" + Convert.ToString(Time);

}

}

}

static public bool Check()

{

Files Files = new Files();

if (!File.Exists(SettingsPath))

{

string Text, nl = Environment.NewLine;

Text = "<Диспетчер задач>true</Диспетчер задач>" + nl + "<Панель задач>true</Панель задач>";

Text += nl + "<Другие приложения>false</Другие приложения>";

Text += nl + "<Перемешивание вопросов>true</Перемешивание вопросов>";

Text += nl + "<Перемешивание ответов>true</Перемешивание ответов>";

Text += nl + "<Коэффициенты оценивания>(0.5, 0.75, 0.9)</Коэффициенты оценивания>";

Text += nl + "[Дополнительные возможности]";

Text += nl + "\t<Черновик>true</Черновик>" + nl + "\t<Калькулятор>true</Калькулятор>";

Text += nl + "\t<Холст>true</Холст>";

Text += nl + "\t<Подсказки>true</Подсказки>" + nl + "\t<Перемещение назад>true</Перемещение назад>";

Text += nl + "[/Дополнительные возможности]" + nl + "[Контроль времени]";

Text += nl + "\t<Секундомер>true</Секундомер>" + nl + "\t<Таймер>false</Таймер>";

Text += nl + "\t<Секунд на вопрос>30</Секунд на вопрос>" + nl + "\t<Секунд на тест>60</Секунд на тест>";

Text += nl + "[/Контроль времени]" + nl + "[Фон]";

Text += nl + "\t<Задний фон>true</Задний фон>" + nl + "\t<Имя фона>Background\_1</Имя фона>";

Text += nl + "\t{Градиент}";

Text += nl + "\t\t<Использовать градиент>false</Использовать градиент>";

Text += nl + "\t\t<Левый верхний угол>ff7611</Левый верхний угол>";

Text += nl + "\t\t<Правый верхний угол>ffffff</Правый верхний угол>";

Text += nl + "\t\t<Правый нижний угол>42ffae</Правый нижний угол>";

Text += nl + "\t\t<Левый нижний угол>5143fe</Левый нижний угол>";

Text += nl + "\t{/Градиент}" + nl + "[/Фон]" + nl + "[Серверный функционал]";

Text += nl + "\t<Использовать сеть>true</Использовать сеть>";

Text += nl + "\t<Сеть по умолчанию>true</Сеть по умолчанию>";

Text += nl + "\t<Адрес сервера>http://localhost</Адрес сервера>";

Text += nl + "\t<Порт сервера>8090</Порт сервера>";

Text += nl + "[Серверный функционал]";

Files.Write(SettingsPath, Text, false);

return false;

}

else

{

string Text = Files.Read(SettingsPath, false);

Fill(Text);

return true;

}

}

public static bool Check(string FileName)

{

string source;

using (FileStream fs = new FileStream(FileName, FileMode.Open))

{

using (StreamReader sr = new StreamReader(fs))

{

source = sr.ReadToEnd();

source = Encoding.Unicode.GetString(Encoding.Default.GetBytes(source));

}

}

try

{

Utils.System.SettingsValidating(ref source, "Диспетчер задач");

Utils.System.SettingsValidating(ref source, "Панель задач");

Utils.System.SettingsValidating(ref source, "Другие приложения");

Utils.System.SettingsValidating(ref source, "Перемешивание вопросов");

Utils.System.SettingsValidating(ref source, "Перемешивание ответов");

Utils.System.SettingsValidating(ref source, "Черновик");

Utils.System.SettingsValidating(ref source, "Калькулятор");

Utils.System.SettingsValidating(ref source, "Холст");

Utils.System.SettingsValidating(ref source, "Подсказки");

Utils.System.SettingsValidating(ref source, "Перемещение назад");

Utils.System.SettingsValidating(ref source, "Секундомер");

Utils.System.SettingsValidating(ref source, "Таймер");

Utils.System.SettingsValidating(ref source, "Секунд на вопрос");

Utils.System.SettingsValidating(ref source, "Задний фон");

Utils.System.SettingsValidating(ref source, "Имя фона");

Utils.System.SettingsValidating(ref source, "Использовать градиент");

Utils.System.SettingsValidating(ref source, "Левый верхний угол");

Utils.System.SettingsValidating(ref source, "Правый верхний угол");

Utils.System.SettingsValidating(ref source, "Правый нижний угол");

Utils.System.SettingsValidating(ref source, "Левый нижний угол");

Utils.System.SettingsValidating(ref source, "Использовать сеть");

Utils.System.SettingsValidating(ref source, "Сеть по умолчанию");

Utils.System.SettingsValidating(ref source, "Адрес сервера");

Utils.System.SettingsValidating(ref source, "Порт сервера");

}

catch

{

return false;

}

return true;

}

static public bool Fill(string FillString)

{

if (FillString == null) { return false; }

FillString = FillString.Trim();

TaskManager = Utils.System.ExecValue(ref FillString, "Диспетчер задач");

TaskPanel = Utils.System.ExecValue(ref FillString, "Панель задач");

OtherApps = Utils.System.ExecValue(ref FillString, "Другие приложения", false);

var markString = Utils.System.ExecValue(ref FillString, "Коэффициенты оценивания", "(0.5, 0.75, 0.9)")

.Trim(new char[] { '(', ')' }).Replace('.', ',')

.Split(new string[] { ", " }, StringSplitOptions.None);

Mark = (

(float)Convert.ToDouble(markString[0]),

(float)Convert.ToDouble(markString[1]),

(float)Convert.ToDouble(markString[2])

);

QuestionMeshing = Utils.System.ExecValue(ref FillString, "Перемешивание вопросов");

AnswersMeshing = Utils.System.ExecValue(ref FillString, "Перемешивание ответов");

Draft = Utils.System.ExecValue(ref FillString, "Черновик");

Calc = Utils.System.ExecValue(ref FillString, "Калькулятор");

Inc = Utils.System.ExecValue(ref FillString, "Холст");

Hint = Utils.System.ExecValue(ref FillString, "Подсказки");

Back = Utils.System.ExecValue(ref FillString, "Перемещение назад");

Stopwatch = Utils.System.ExecValue(ref FillString, "Секундомер");

Timer = Utils.System.ExecValue(ref FillString, "Таймер", false);

SecondsOnQuestion = Convert.ToInt32(Utils.System.ExecValue(ref FillString, "Секунд на вопрос", "-1"));

UseBackground = Utils.System.ExecValue(ref FillString, "Задний фон");

Gradient = new Gradient()

{

Active = Utils.System.ExecValue(ref FillString, "Использовать градиент"),

LeftTop = new GradientCorner(Utils.System.ExecValue(ref FillString, "Левый верхний угол", "ff7611")),

RightTop = new GradientCorner(Utils.System.ExecValue(ref FillString, "Правый верхний угол", "ffffff")),

RightBottom = new GradientCorner(Utils.System.ExecValue(ref FillString, "Правый нижний угол", "42ffae")),

LeftBottom = new GradientCorner(Utils.System.ExecValue(ref FillString, "Левый нижний угол", "5143fe"))

};

UseWeb = Utils.System.ExecValue(ref FillString, "Использовать сеть");

UseWebDefault = Utils.System.ExecValue(ref FillString, "Сеть по умолчанию");

ServerAdress = Utils.System.ExecValue(ref FillString, "Адрес сервера", "http://localhost");

ServerPort = Utils.System.ExecValue(ref FillString, "Порт сервера", "8090");

//Возможно стоит удалить

BackgroundName = Utils.System.ExecValue(ref FillString, "Имя фона", "Background\_1");

if (!File.Exists(BackgroundName) && UseBackground && !Gradient.Active)

{

Gradient.Active = true;

}

TimerValue = Convert.ToInt32(Utils.System.ExecValue(ref FillString, "Секунд на тест", "0"));

if (TimerValue == 0)

Timer = false;

return true;

}

public static void Save()

{

Files Files = new Files();

string Text, nl = Environment.NewLine;

Text = $"<Диспетчер задач>{TaskManager}</Диспетчер задач>" + nl + $"<Панель задач>{TaskPanel}</Панель задач>";

Text += nl + $"<Другие приложения>{OtherApps}</Другие приложения>";

Text += nl + $"<Перемешивание вопросов>{QuestionMeshing}</Перемешивание вопросов>";

Text += nl + $"<Перемешивание ответов>{AnswersMeshing}</Перемешивание ответов>";

Text += nl + $"<Коэффициенты оценивания>{Mark.ToString().Replace(", ", "~").Replace(',', '.').Replace("~", ", ")}</Коэффициенты оценивания>";

Text += nl + "[Дополнительные возможности]";

Text += nl + $"\t<Черновик>{Draft}</Черновик>" + nl + $"\t<Калькулятор>{Calc}</Калькулятор>";

Text += nl + $"\t<Холст>{Inc}</Холст>";

Text += nl + $"\t<Подсказки>{Hint}</Подсказки>" + nl + $"\t<Перемещение назад>{Back}</Перемещение назад>";

Text += nl + "[/Дополнительные возможности]" + nl + "[Контроль времени]";

Text += nl + $"\t<Секундомер>{Stopwatch}</Секундомер>" + nl + $"\t<Таймер>{Timer}</Таймер>";

Text += nl + $"\t<Секунд на вопрос>{SecondsOnQuestion}</Секунд на вопрос>";

Text += nl + $"\t<Секунд на тест>{TimerValue}</Секунд на тест>";

Text += nl + "[/Контроль времени]" + nl + "[Фон]";

Text += nl + $"\t<Задний фон>{UseBackground}</Задний фон>" + nl + $"\t<Имя фона>{BackgroundName}</Имя фона>";

Text += nl + "\t{Градиент}";

Text += nl + $"\t\t<Использовать градиент>{Gradient.Active}</Использовать градиент>";

Text += nl + $"\t\t<Левый верхний угол>{Gradient.LeftTop.GetStringColor()}</Левый верхний угол>";

Text += nl + $"\t\t<Правый верхний угол>{Gradient.RightTop.GetStringColor()}</Правый верхний угол>";

Text += nl + $"\t\t<Правый нижний угол>{Gradient.RightBottom.GetStringColor()}</Правый нижний угол>";

Text += nl + $"\t\t<Левый нижний угол>{Gradient.LeftBottom.GetStringColor()}</Левый нижний угол>";

Text += nl + "\t{/Градиент}" + nl + "[/Фон]" + nl + "[Серверный функционал]";

Text += nl + $"\t<Использовать сеть>{UseWeb}</Использовать сеть>";

Text += nl + $"\t<Сеть по умолчанию>{UseWebDefault}</Сеть по умолчанию>";

Text += nl + $"\t<Адрес сервера>{ServerAdress}</Адрес сервера>";

Text += nl + $"\t<Порт сервера>{ServerPort}</Порт сервера>";

Text += nl + "[Серверный функционал]";

Files.Write(SettingsPath, Text, false);

}

private static void Settings\_SettingsChange()

{

return;

}

}

}

Листинг модуля Test.Main.Test.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using TestPro.TestKinds;

using TestPro.Utils;

namespace TestPro.Test.Main

{

public static class Test

{

static private BasicTest[] questions;

static private BasicAnswer[] answers;

static private Tester currentTester;

static private int time;

public static BasicTest[] Questions { get => questions; set => questions = value; }

public static BasicAnswer[] Answers { get => answers; set => answers = value; }

public static Tester CurrentTester { get => currentTester; set => currentTester = value; }

public static int Time { get => time; set => time = value; }

public static bool GetResult(int Number)

{

if (Questions != null && Answers != null && Questions[Number] != null && Answers[Number] != null)

{

if(Questions[Number].RightAnswer == Answers[Number].Answer)

{

return true;

}

else

{

return false;

}

}

else

{

return false;

}

}

public static byte GetMark()

{

if (Questions != null && Answers != null)

{

int sum = 0;

string s = "";

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

{

if((Questions[i] != null && Answers[i] != null) && GetResult(i)) { sum++; }

s += GetResult(i).ToString() + Environment.NewLine;

}

double mark = (double)sum / (double)Questions.Length;

if (mark > 0.9) { return 5; }

else if (mark > 0.75) { return 4; }

else if (mark > 0.5) { return 3; }

else { return 2; }

}

return 0;

}

public static void Mix()

{

if (questions != null)

{

Dictionary<int, BasicTest> dictionary = new Dictionary<int, BasicTest>();

Random random = new Random();

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

dictionary.Add(random.Next(int.MaxValue), questions[i]);

questions = dictionary.OrderBy(x => x.Key).Select(x => x.Value).ToList().ToArray();

}

}

public static void Clear()

{

Questions = null;

Answers = null;

}

}

}

Листинг модуля Test.Main.Tester.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Security.Cryptography;

using TestPro.Utils;

namespace TestPro.Test.Main

{

public class Tester

{

private string id;

public string Id

{

get { return id; }

}

private string surname, name, middlename, group;

public Tester()

{

Results = new List<TestResults>();

}

public string Surname

{

get { return surname; }

set { surname = value; GetHash(); }

}

public string Name

{

get { return name; }

set { name = value; GetHash(); }

}

public string Middlename

{

get { return middlename; }

set { middlename = value; GetHash(); }

}

public string Group

{

get { return group; }

set { group = value; GetHash(); }

}

public string GetHash()

{

string Result = Surname + " " + Name + " " + Middlename + " " + Group;

id = CryptoSystem.GetHash(Result);

return id;

}

public bool CompareHash(string CompareHash)

{

if(CryptoSystem.CompareHash(Id, CompareHash)) { return true; }

else { return false; }

}

public bool Compare(Tester CompareHash)

{

if(Id == CompareHash.Id)

{

if(Group == CompareHash.Group)

{

if (Surname == CompareHash.Surname)

{

if (Name == CompareHash.Name)

{

if (Middlename == CompareHash.Middlename)

{

return true;

}

else { return false; }

}

else { return false; }

}

else { return false; }

}

else { return false; }

}

else { return false; }

}

public List<TestResults> Results { get; set; }

}

}

Листинг модуля Test.Main.TestResults.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestPro.Test.Main

{

public class TestResults

{

public TestResults()

{

Answers = new List<bool>();

}

public string Test { get; set; }

public byte Mark { get; set; }

public int Time { get; set; }

public List<bool> Answers { get; set; }

}

}

Листинг модуля Utils.CryptoSystem.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Security.Cryptography;

namespace TestPro.Utils

{

/// <summary>

/// Класс для шифрования и дешифрования данных по алгоритму AES256,

/// а так же для получения получения хэша по алгоритму SHA512

/// </summary>

public static class CryptoSystem

{

/// <summary>

/// Базовый конструктор класса

/// </summary>

static CryptoSystem()

{

Encrypted = "";

Decrypted = "";

CryptoMethod = new RijndaelManaged()

{

KeySize = 256,

BlockSize = 256,

Padding = PaddingMode.ISO10126,

Mode = CipherMode.CBC,

Key = Key,

IV = IV

};

}

static private string encrypted;

/// <summary>

/// Возвращает дешифрованный текст

/// </summary>

static public string Encrypted

{

get { return encrypted; }

set { encrypted = value; if (!String.IsNullOrWhiteSpace(value)) Decrypt(); }

}

static private string decrypted;

/// <summary>

/// Возвращает шифрованный текст

/// </summary>

static public string Decrypted

{

get { return decrypted; }

set { decrypted = value; if (!String.IsNullOrWhiteSpace(value)) Encrypt(); }

}

static private byte[] Key = { 124, 18, 168, 246, 139, 216, 84, 253, 201, 211, 158, 223, 93, 63, 124, 117, 234, 155, 90, 67, 146, 223, 217, 54, 149, 106, 245, 38, 217, 22, 90, 38 };

static private byte[] IV = { 52, 103, 149, 156, 201, 248, 188, 119, 106, 92, 205, 157, 233, 141, 154, 16, 133, 113, 72, 154, 236, 129, 218, 156, 219, 195, 225, 140, 121, 171, 142, 156 };

static private RijndaelManaged CryptoMethod;

/// <summary>

/// Метод для шифрования

/// </summary>

static public void Encrypt()

{

if (String.IsNullOrWhiteSpace(Decrypted)) throw new NullStringException();

ICryptoTransform encryptor = CryptoMethod.CreateEncryptor();

using (MemoryStream msEncrypt = new MemoryStream())

{

using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))

{

using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))

{

swEncrypt.Write(Decrypted);

}

Encrypted = Encoding.Default.GetString(msEncrypt.ToArray());

}

}

}

/// <summary>

/// Метод для шифровки строки

/// </summary>

/// <param name="Decrypted">Строка, которую требуется зашифровать</param>

/// <returns></returns>

static public string Encrypt(string Decrypted)

{

if (String.IsNullOrWhiteSpace(Decrypted)) throw new NullStringException();

ICryptoTransform encryptor = CryptoMethod.CreateEncryptor();

using (MemoryStream msEncrypt = new MemoryStream())

{

using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))

{

using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))

{

swEncrypt.Write(Decrypted);

}

encrypted = Encoding.Default.GetString(msEncrypt.ToArray());

}

}

return encrypted;

}

/// <summary>

/// Метод для дешифровки

/// </summary>

static public void Decrypt()

{

if (String.IsNullOrWhiteSpace(Encrypted)) throw new NullStringException();

ICryptoTransform decryptor = CryptoMethod.CreateDecryptor();

using (MemoryStream msDecrypt = new MemoryStream(Encoding.Default.GetBytes(Encrypted)))

{

using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))

{

using (StreamReader swDecrypt = new StreamReader(csDecrypt))

{

Decrypted = swDecrypt.ReadToEnd();

}

}

}

}

/// <summary>

/// Метод для дешифровки строки

/// </summary>

/// <param name="Encrypted">Строка, которую требуется расшифровать</param>

/// <returns></returns>

static public string Decrypt(string Encrypted)

{

if (String.IsNullOrWhiteSpace(Encrypted)) throw new NullStringException();

ICryptoTransform decryptor = CryptoMethod.CreateDecryptor();

using (MemoryStream msDecrypt = new MemoryStream(Encoding.Default.GetBytes(Encrypted)))

{

using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))

{

using (StreamReader swDecrypt = new StreamReader(csDecrypt))

{

decrypted = swDecrypt.ReadToEnd();

}

}

}

return decrypted;

}

/// <summary>

/// Получение хэша из строки

/// </summary>

/// <param name="OriginalString">Искомая строка</param>

/// <returns></returns>

static public string GetHash(string OriginalString)

{

if (String.IsNullOrWhiteSpace(OriginalString)) throw new NullStringException();

using (SHA512Managed Hash = new SHA512Managed())

{

byte[] result = Encoding.Default.GetBytes(OriginalString);

string Result = Encoding.Default.GetString(Hash.ComputeHash(result));

return Result;

}

}

/// <summary>

/// Сравнение двух объектов, путём сравнения их хэшей

/// </summary>

/// <param name="OriginalHash">Первый хэш</param>

/// <param name="ComparingHash">Второй хэш</param>

/// <returns></returns>

static public bool CompareHash(string OriginalHash, string ComparingHash)

{

if (OriginalHash == ComparingHash) { return true; }

else { return false; }

}

}

}

Листинг модуля Utils.Files.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//Переделать с StreamReader/Writer'ов на FileStream

namespace TestPro.Utils

{

/// <summary>

/// Класс для инкапсуляции работы с файлами

/// </summary>

public class Files

{

/// <summary>

/// Чтение файла по заданному пути

/// </summary>

/// <param name="filePath">Путь к файлу</param>

/// <param name="encrypted">Файл зашифрован</param>

/// <returns></returns>

public string Read(string filePath, bool encrypted)

{

if (String.IsNullOrWhiteSpace(filePath)) throw new NullStringException();

if (!File.Exists(filePath))

{

throw new FileNotFoundException();

}

string FileString = "";

FileStream reader = new FileStream(filePath, FileMode.Open);

try

{

byte[] b = new byte[reader.Length];

reader.Position = 0;

reader.Read(b, 0, (int)reader.Length);

FileString = Encoding.Unicode.GetString(b);

}

catch

{

throw new ReadException();

}

finally

{

reader.Close();

}

if (encrypted)

return CryptoSystem.Decrypt(FileString);

else

return FileString;

}

/// <summary>

/// Чтение фала по заданному пути

/// </summary>

/// <param name="filePath">Путь к файлу</param>

/// <returns></returns>

public string Read(string filePath)

{

return Read(filePath, true);

}

/// <summary>

/// Запись в файл по заданному пути

/// </summary>

/// <param name="filePath">Путь файлу</param>

/// <param name="value">Текст для записи</param>

/// <param name="encrypt">Необходимость шифрования</param>

public void Write(string filePath, string value, bool encrypt)

{

if (String.IsNullOrWhiteSpace(filePath)) throw new NullStringException();

FileStream writer = new FileStream(filePath, FileMode.Create);

try

{

if (encrypt)

{

byte[] b = Encoding.Unicode.GetBytes(CryptoSystem.Encrypt(value));

writer.Write(b, 0, b.Length);

}

else

{

byte[] b = Encoding.Unicode.GetBytes(value);

writer.Write(b, 0, b.Length);

}

}

catch

{

throw new WriteException();

}

finally

{

writer.Close();

}

}

/// <summary>

/// Запись в файл по заданному пути

/// </summary>

/// <param name="filePath">Путь файлу</param>

/// <param name="value">Текст для записи</param>

public void Write(string filePath, string value)

{

Write(filePath, value, true);

}

/// <summary>

/// Сыкрать файл по пути

/// </summary>

/// <param name="FilePath">Путь к файлу</param>

public void Hide(string FilePath)

{

if (String.IsNullOrWhiteSpace(FilePath)) throw new NullStringException();

if (!File.Exists(FilePath))

{

throw new FileNotFoundException();

}

FileInfo Properties = new FileInfo(FilePath)

{

IsReadOnly = true

};

File.SetAttributes(FilePath, FileAttributes.Hidden ^ FileAttributes.ReadOnly);

}

/// <summary>

/// Отменить сокрытие файла по пути

/// </summary>

/// <param name="FilePath">Путь к файлу</param>

public void Unhide(string FilePath)

{

if (String.IsNullOrWhiteSpace(FilePath)) throw new NullStringException();

if (!File.Exists(FilePath))

{

throw new FileNotFoundException();

}

FileInfo Properties = new FileInfo(FilePath)

{

IsReadOnly = true

};

File.SetAttributes(FilePath, 0);

}

}

}

Листинг модуля Utils.Logger.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace TestPro.Utils

{

/// <summary>

/// Класс для записи поведения программы

/// </summary>

public static class Logger

{

/// <summary>

/// Базовый конструктор

/// </summary>

static Logger()

{

Log("TP.U.L - Инициализация логгера");

if (!Directory.Exists($@"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\TestPro\Test\Log"))

{

Directory.CreateDirectory($@"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\TestPro\Test\Log");

Log("TP.U.L - Создание папки лога");

}

path = $@"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\TestPro\Test\Log\Log - {DateTime.Now.ToString().Replace(':', '.')}.txt";

NewRecord += (m) => { log.Append(m + Environment.NewLine); };

NewRecord += (m) => { File.AppendAllText(path, m.PadRight(200) + $"{DateTime.Now}" + Environment.NewLine, Encoding.Unicode); };

Log("TP.U.L - Инициализация завершена");

}

private static StringBuilder log = new StringBuilder();

private static readonly string path;

/// <summary>

/// Создание новой записи в логе

/// </summary>

/// <param name="message">Сообщение</param>

public delegate void NewRecordHandler(string message);

/// <summary>

/// Событие получения сообщения

/// </summary>

public static event NewRecordHandler NewRecord;

/// <summary>

/// Запись сообщения

/// </summary>

/// <param name="message"></param>

public static void Log(string message)

{

if (!string.IsNullOrWhiteSpace(message))

NewRecord?.Invoke(message);

}

/// <summary>

/// Запись сообщения и исключения

/// </summary>

/// <param name="message"></param>

/// <param name="ex"></param>

public static void Log(string message, Exception ex)

{

if (!string.IsNullOrWhiteSpace(message) && !(ex is null))

NewRecord?.Invoke($"{message}: {ex.Message}");

}

/// <summary>

/// Запись сообщения исключения

/// </summary>

/// <param name="ex"></param>

public static void Log(Exception ex)

{

if (!(ex is null))

NewRecord?.Invoke($"{ex.Message}");

}

/// <summary>

/// Возвращает весь лог за сеанс

/// </summary>

/// <returns>Лог</returns>

public static string GetLog()

{

return log.ToString();

}

}

}

Листинг модуля Utils.Packager.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.IO.Compression;

using System.IO.Packaging;

namespace TestPro.Utils

{

/// <summary>

/// Класс для работы с упакованными данными

/// </summary>

public class Packager

{

/// <summary>

/// Чтение файла по заданному пути

/// </summary>

/// <param name="zipPackageName">Путь к файлу</param>

/// <param name="extractingFileName">Выходное название</param>

/// <param name="extractingFilePath">Выходной путь</param>

public void ReadFile(string zipPackageName, string extractingFileName, string extractingFilePath)

{

if (!File.Exists(zipPackageName))

throw new FileNotFoundException();

using (FileStream zipStream = new FileStream(zipPackageName, FileMode.Open))

{

using (Package zipPackage = ZipPackage.Open(zipStream, FileMode.Open))

{

string path = Path.GetFullPath(extractingFilePath);

string dir = Path.GetDirectoryName(path);

if (!Directory.Exists(dir))

Directory.CreateDirectory(dir);

using (FileStream file = new FileStream(path, FileMode.Create))

{

zipPackage.GetPart(new Uri(extractingFileName, UriKind.Relative)).GetStream().CopyTo(file);

}

}

}

}

/// <summary>

/// Чтение файлов по заданному путь

/// </summary>

/// <param name="zipPackageName">Путь к файлу</param>

/// <returns>Список файлов</returns>

public IEnumerable<string> ReadFiles(string zipPackageName)

{

if (!File.Exists(zipPackageName))

throw new FileNotFoundException();

using (FileStream zipStream = new FileStream(zipPackageName, FileMode.Open))

{

using (Package zipPackage = ZipPackage.Open(zipStream, FileMode.Open))

{

string fileName = Path.GetFileNameWithoutExtension(zipPackageName);

string dirName = $@"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\TestPro\Test\Tests\".Replace('\\', '/') + fileName;

if (!Directory.Exists(dirName))

Directory.CreateDirectory(dirName);

foreach (var e in zipPackage.GetParts())

{

string directories = Path.GetDirectoryName(e.Uri.OriginalString);

if (!Directory.Exists(dirName + directories))

Directory.CreateDirectory(dirName + directories);

using (FileStream file = new FileStream(dirName + e.Uri.OriginalString, FileMode.Create))

{

e.GetStream().CopyTo(file);

}

yield return dirName + e.Uri.OriginalString;

}

}

}

}

/// <summary>

/// Запись файла по заданному пути

/// </summary>

/// <param name="zipPackageName">Путь к файлу</param>

/// <param name="newFileName">Выходное название</param>

/// <param name="newFilePath">Выходной путь</param>

/// <param name="newFileType">Выходное расширение</param>

public void WriteFile(string zipPackageName, string newFileName, string newFilePath, string newFileType)

{

FileMode fileMode;

if (File.Exists(zipPackageName))

fileMode = FileMode.Open;

else

fileMode = FileMode.Create;

using (FileStream zipStream = new FileStream(zipPackageName, fileMode))

{

using (Package zipPackage = ZipPackage.Open(zipStream, fileMode))

{

zipPackage.CreatePart(new Uri(newFileName, UriKind.Relative), newFileType, CompressionOption.Maximum);

using (FileStream NewFileStream = new FileStream(newFilePath, FileMode.Open))

{

NewFileStream.CopyTo(zipPackage.GetPart(new Uri(newFileName, UriKind.Relative)).GetStream());

}

}

}

}

/// <summary>

/// Запись файлов в заданный пакет

/// </summary>

/// <param name="zipPackageName">Путь для записи</param>

/// <param name="packagingFile">Пакуемые файлы</param>

public void WriteFiles(string zipPackageName, List<(string PackagingFileName, string PackagingFilePath, string PackagingFileType)> packagingFile)

{

FileMode fileMode;

if (File.Exists(zipPackageName))

fileMode = FileMode.Open;

else

fileMode = FileMode.Create;

using (FileStream zipStream = new FileStream(zipPackageName, fileMode))

{

using (Package zipPackage = ZipPackage.Open(zipStream, fileMode))

{

foreach (var e in packagingFile)

{

zipPackage.CreatePart(new Uri(e.PackagingFileName, UriKind.Relative), e.PackagingFileType, CompressionOption.Maximum);

using (FileStream NewFileStream = new FileStream(e.PackagingFilePath, FileMode.Open))

{

NewFileStream.CopyTo(zipPackage.GetPart(new Uri(e.PackagingFileName, UriKind.Relative)).GetStream());

}

}

}

}

}

/// <summary>

/// Проверка пути на корректность

/// </summary>

/// <param name="value">путь</param>

/// <returns>Исправленный путь</returns>

public string Check(string value)

{

if (String.IsNullOrWhiteSpace(value))

throw new NullStringException();

int i = value.IndexOf(@"\");

if (i != -1)

{

value = value.Replace('\\', '/');

}

if (!Uri.TryCreate(value, UriKind.Relative, out Uri u))

throw new UriFormatException();

return value.TrimStart(new char[1] { '/' }).Trim();

}

}

}

Листинг модуля Utils.System.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Media;

using System.IO;

namespace TestPro.Utils

{

/// <summary>

/// Ошибка создания файла

/// </summary>

[Serializable]

public class CreateFileExсeption : Exception

{

/// <summary>

/// Базовый конструктор исключения

/// </summary>

public CreateFileExсeption() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="Message">Сообщения с информацией об исключении</param>

public CreateFileExсeption(string Message) : base(Message)

{

Message += (Message != "") ? ": " + Message : "";

}

/// <summary>

/// Сообщение по умолчанию

/// </summary>

public new string Message;

}

/// <summary>

/// Ошибка записи

/// </summary>

[Serializable]

public class WriteException : Exception

{

/// <summary>

/// Базовый конструктор исключения

/// </summary>

public WriteException() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="Message">Сообщения с информацией об исключении</param>

public WriteException(string Message) : base(Message)

{

this.Message += (Message != "") ? ": " + Message : "";

}

/// <summary>

/// Сообщение по умолчанию

/// </summary>

public new string Message = "Ошибка записи";

}

/// <summary>

/// Ошибка чтения

/// </summary>

[Serializable]

public class ReadException : Exception

{

/// <summary>

/// Базовый конструктор исключения

/// </summary>

public ReadException() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="Message">Сообщения с информацией об исключении</param>

public ReadException(string Message) : base(Message)

{

this.Message += (Message != "") ? ": " + Message : "";

}

/// <summary>

/// Сообщение по умолчанию

/// </summary>

public new string Message = "Ошибка чтения";

}

/// <summary>

/// Неинициализированаая строка

/// </summary>

[Serializable]

public class NullStringException : Exception

{

/// <summary>

/// Базовый конструктор исключения

/// </summary>

public NullStringException() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="Message">Сообщения с информацией об исключении</param>

public NullStringException(string Message) : base(Message)

{

this.Message += (Message != "") ? ": " + Message : "";

}

/// <summary>

/// Сообщение по умолчанию

/// </summary>

public new string Message = "Пустая или неинициализированная строка";

}

/// <summary>

/// Ошибка доступа к Microsoft Excel

/// </summary>

[Serializable]

public class ExcelAppException : Exception

{

/// <summary>

/// Базовый конструктор исключения

/// </summary>

public ExcelAppException() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="Message">Сообщения с информацией об исключении</param>

public ExcelAppException(string Message) : base(Message)

{

this.Message += (Message != "") ? ": " + Message : "";

}

/// <summary>

/// Сообщение по умолчанию

/// </summary>

public new string Message = "Ошибка при выгрузке в Microsoft Excel\n";

}

/// <summary>

/// Сохранение данных в Microsoft Excel было отменено

/// </summary>

[Serializable]

public class ExcelSaveCanseledException : Exception

{

/// <summary>

/// Базовый конструктор исключения

/// </summary>

public ExcelSaveCanseledException() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="Message">Сообщения с информацией об исключении</param>

public ExcelSaveCanseledException(string Message) : base(Message)

{

this.Message += (Message != "") ? ": " + Message : "";

}

/// <summary>

/// Сообщение по умолчанию

/// </summary>

public new string Message = "Сохранение данных в Microsoft Excel было отменено";

}

/// <summary>

/// Файл настроек не найден

/// </summary>

[Serializable]

public class SettingsNotFoundException : Exception

{

/// <summary>

/// Базовый конструктор исключения

/// </summary>

public SettingsNotFoundException() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="Message">Сообщения с информацией об исключении</param>

public SettingsNotFoundException(string Message) : base(Message)

{

this.Message += (Message != "") ? ": " + Message : "";

}

/// <summary>

/// Сообщение по умолчанию

/// </summary>

public new string Message = "Файл настроек не найден\nЗаданы настройки по умолчанию\nФайл перезаписан";

}

/// <summary>

/// Неверные данные в тесте

/// </summary>

[Serializable]

public class InvalidTestData : Exception

{

/// <summary>

/// Базовый конструктор исключения

/// </summary>

public InvalidTestData() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="Message">Сообщения с информацией об исключении</param>

public InvalidTestData(string Message) : base(Message)

{

this.Message += (Message != "") ? ": " + Message : "";

}

/// <summary>

/// Сообщение по умолчанию

/// </summary>

public new string Message = "Неверные данные в тесте";

}

/// <summary>

/// Неверные данные в настройках

/// </summary>

public class IncorrectSettings : Exception

{

/// <summary>

/// Базовый конструктор исключения

/// </summary>

public IncorrectSettings() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="Message">Сообщения с информацией об исключении</param>

public IncorrectSettings(string Message) : base(Message)

{

this.Message += (Message != "") ? ": " + Message : "";

}

/// <summary>

/// Сообщение по умолчанию

/// </summary>

public new string Message = "Неверные данные в настройках";

}

/// <summary>

/// Класс с сисемными методами приложения

/// </summary>

public static class System

{

/// <summary>

/// Главное окно

/// </summary>

public static Window mainWindow;

/// <summary>

/// Получение данных из закодированной строки

/// </summary>

/// <param name="source">Источник данных</param>

/// <param name="value">Параметр для очистки</param>

/// <param name="defaultValue">Значение по-умолчанию</param>

/// <returns>Извлечённое значение</returns>

public static string ExecValue(ref string source, string value, string defaultValue)

{

if (string.IsNullOrWhiteSpace(source) || string.IsNullOrWhiteSpace(value))

throw new ArgumentNullException();

int i = source.IndexOf($"<{value}>");

int j = source.IndexOf($"</{value}>");

if (NotExistsCheck(j) && NotExistsCheck(i))

{

return source.Substring(i + value.Length + 2, j - i - value.Length - 2);

}

else

{

return defaultValue;

}

}

/// <summary>

/// Получение данных из закодированной строки

/// </summary>

/// <param name="source">Источник данных</param>

/// <param name="value">Параметр для очистки</param>

/// <param name="defaultValue">Значение по-умолчанию</param>

/// <returns>Извлечённое значение</returns>

public static bool ExecValue(ref string source, string value, bool defaultValue = true)

{

if (string.IsNullOrWhiteSpace(source) || string.IsNullOrWhiteSpace(value))

throw new ArgumentNullException();

int i = source.IndexOf($"<{value}>");

int j = source.IndexOf($"</{value}>");

if (NotExistsCheck(j) && NotExistsCheck(i))

{

try

{

return Convert.ToBoolean(source.Substring(i + value.Length + 2, j - i - value.Length - 2));

}

catch

{

return defaultValue;

}

}

else

{

return defaultValue;

}

}

/// <summary>

/// Проверка на существование

/// </summary>

/// <param name="symbol">Значение</param>

/// <returns>Существование</returns>

public static bool NotExistsCheck(int symbol)

{

if (symbol == -1)

{

return false;

}

else

{

return true;

}

}

/// <summary>

/// Получение данных из закодированной строки

/// </summary>

/// <param name="source">Источник данных</param>

/// <param name="value">Параметр для очистки</param>

/// <returns>Извлечённое значение</returns>

public static string ExecTestValue(ref string source, string value)

{

if (String.IsNullOrWhiteSpace(source) || String.IsNullOrWhiteSpace(value))

throw new NullStringException();

int i = source.IndexOf($"<{value}>");

int j = source.IndexOf($"</{value}>");

if (!NotExistsCheck(i) || !NotExistsCheck(j))

throw new InvalidTestData("Искомое значение: " + value);

else

{

return source.Substring(i + value.Length + 2, j - i - value.Length - 2);

}

}

/// <summary>

/// Получение данных из закодированной строки

/// </summary>

/// <param name="source">Источник данных</param>

/// <param name="value">Параметр для очистки</param>

/// <returns>Извлечённое значение</returns>

public static string ExecTestValueStrict(ref string source, string value)

{

if (string.IsNullOrWhiteSpace(source) || string.IsNullOrWhiteSpace(value))

throw new ArgumentNullException();

string result = ExecTestValue(ref source, value);

if (String.IsNullOrWhiteSpace(result))

throw new NullStringException();

return result;

}

/// <summary>

/// Проверка данных на корректность

/// </summary>

/// <param name="source">Источник данных</param>

/// <param name="value">Параметр для очистки</param>

public static void SettingsValidating(ref string source, string value)

{

if (String.IsNullOrWhiteSpace(source) || String.IsNullOrWhiteSpace(value))

throw new NullStringException();

int i = source.IndexOf($"<{value}>");

int j = source.IndexOf($"</{value}>");

if (i >= j)

throw new IncorrectSettings();

}

/// <summary>

/// Очистка содержимого тестовой папки

/// </summary>

public static void ClearAppData()

{

string s = $@"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\TestPro\Test\Tests\";

if (Directory.Exists(s))

Clear(s);

return;

void Clear(string path)

{

var files = Directory.GetFiles(path);

foreach (var e in files)

try

{

File.Delete(e);

}

catch { }

var directories = Directory.GetDirectories(path);

foreach (var e in directories)

try

{

Clear(e);

Directory.Delete(e);

}

catch { }

Directory.Delete(path);

}

}

}

}

Листинг модуля Visual.Common.CalcWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using TestPro.Test.Main;

using TestPro.Visual.Core;

using TestPro.Utils;

namespace TestPro.Visual.Common

{

/// <summary>

/// Логика взаимодействия для CalcWindow.xaml

/// </summary>

public partial class CalcWindow : Window

{

public CalcWindow()

{

Logger.Log("TP.V.C.CW - Инициализация окна калькулятора");

InitializeComponent();

if (Settings.UseBackground)

{

if (Settings.Gradient.Active)

{

Background = Settings.Gradient.GetGradient();

}

else if (!String.IsNullOrWhiteSpace(Settings.BackgroundName))

{

Background = new ImageBrush(new BitmapImage(new Uri(Settings.BackgroundName, UriKind.Relative)));

}

}

Logger.Log("TP.V.C.CW - Установка фона");

foreach (UIElement c in LayoutRoot.Children)

{

if (c is Button)

{

((Button)c).Click += Button\_Click;

}

}

Logger.Log("TP.V.C.CW - Подписка на события");

}

string leftop = ""; // Левый операнд

string operation = ""; // Знак операции

string rightop = ""; // Правый операнд

private void Escape\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Close();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

// Получаем текст кнопки

if (((Button)e.OriginalSource).Content is Image)

{

if (textBlock.Text.Length <= 0) return;

textBlock.Text = textBlock.Text.TrimEnd(textBlock.Text[textBlock.Text.Length - 1]);

if (operation == "")

{

leftop = textBlock.Text;

}

else

{

rightop = textBlock.Text;

}

return;

}

string s = (string)((Button)e.OriginalSource).Content;

// Добавляем его в текстовое поле

textBlock.Text += s;

int num;

// Пытаемся преобразовать его в число

bool result = Int32.TryParse(s, out num);

// Если текст - это число

if (result == true)

{

// Если операция не задана

if (operation == "")

{

// Добавляем к левому операнду

leftop += s;

}

else

{

// Иначе к правому операнду

rightop += s;

textBlock.Text = rightop;

}

}

// Если было введено не число

else

{

// Если равно, то выводим результат операции

if (s == "=")

{

Update\_RightOp();

textBlock.Text = rightop;

oldTextBlock.Text = "";

operation = "";

}

// Очищаем поле и переменные

else if (s == "C")

{

leftop = "";

rightop = "";

operation = "";

textBlock.Text = "";

oldTextBlock.Text = "";

}

else if (s == "CE")

{

rightop = "";

textBlock.Text = "";

}

else if (s == "±")

{

if (rightop != "")

{

rightop = (-int.Parse(rightop)).ToString();

textBlock.Text = rightop;

}

else if (textBlock.Text == "±")

{

rightop = "-";

textBlock.Text = rightop;

}

else

{

leftop = (-int.Parse(leftop)).ToString();

textBlock.Text = leftop;

}

}

// Получаем операцию

else

{

// Если правый операнд уже имеется, то присваиваем его значение левому

// операнду, а правый операнд очищаем

if (rightop != "")

{

Update\_RightOp();

leftop = rightop;

rightop = "";

}

operation = s;

oldTextBlock.Text = leftop + $" {operation}";

textBlock.Text = "";

}

}

}

// Обновляем значение правого операнда

private void Update\_RightOp()

{

int num1;

int num2;

if (!int.TryParse(leftop, out num1)) return;

if (!int.TryParse(rightop, out num2)) return;

// И выполняем операцию

try

{

switch (operation)

{

case "+":

rightop = (num1 + num2).ToString();

break;

case "-":

rightop = (num1 - num2).ToString();

break;

case "\*":

rightop = (num1 \* num2).ToString();

break;

case "/":

rightop = (num1 / num2).ToString();

break;

}

}

catch

{

CustomMessageBox.Show("Неверные данные");

Logger.Log("TP.V.C.CW - Введены некорректные данные");

}

Logger.Log("TP.V.C.CW - Выполнение действия");

}

private void Window\_Closed(object sender, EventArgs e)

{

Logger.Log("TP.V.C.CW - Закрытие окна");

}

}

}

Листинг модуля Visual.Common.CalcWindow.xaml

<Window x:Class="TestPro.Visual.Common.CalcWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Common;assembly=TestPro.Visual.Common"

mc:Ignorable="d"

Title="CalcWindow" Height="300" Width="200" AllowsTransparency="True" WindowStyle="None" Topmost="True"

ResizeMode="NoResize" Closed="Window\_Closed">

<WindowChrome.WindowChrome>

<WindowChrome CaptionHeight="21"/>

</WindowChrome.WindowChrome>

<Window.Resources>

<Style x:Key="Escape">

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Image.Source" Value="Images/General/Escape/Escape.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="Escape\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/Escape/EscapeActive.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="Button">

<Setter Property="Background">

<Setter.Value>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#7F8B8B8B" Offset="0"/>

<GradientStop Color="#3FC7C7C7" Offset="1"/>

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="21"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="21"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="35"/>

</Grid.ColumnDefinitions>

<Image x:Name="TitleLogo">

<Image.Margin>0,0,0,0</Image.Margin>

<Image.Source>Images/TestProLogo.ico</Image.Source>

<Image.Stretch>Fill</Image.Stretch>

</Image>

<TextBlock>

<TextBlock.Background>Orange</TextBlock.Background>

<TextBlock.Text>Калькулятор</TextBlock.Text>

<TextBlock.FontSize>14</TextBlock.FontSize>

<TextBlock.Padding>5, 0, 0, 0</TextBlock.Padding>

<TextBlock.FontWeight>Bold</TextBlock.FontWeight>

<TextBlock.FontStyle>Italic</TextBlock.FontStyle>

<Grid.Column>1</Grid.Column>

</TextBlock>

<Image x:Name="Escape" Style="{StaticResource Escape}" Grid.Column="2" WindowChrome.IsHitTestVisibleInChrome="True"/>

</Grid>

<Grid x:Name="LayoutRoot">

<Grid.Row>1</Grid.Row>

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid>

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Grid.ColumnSpan>4</Grid.ColumnSpan>

<TextBlock x:Name="oldTextBlock" Grid.Row="0" Opacity="0.6" OpacityMask="Gray" TextAlignment="Right">

<TextBlock.Background>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="#667D7D7D" Offset="0"/>

<GradientStop Color="#33CACACA" Offset="1"/>

</LinearGradientBrush>

</TextBlock.Background>

</TextBlock>

<TextBlock x:Name="textBlock" Grid.Row="1" TextAlignment="Right" >

<TextBlock.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#997D7D7D" Offset="0"/>

<GradientStop Color="#66C8C8C8" Offset="1"/>

</LinearGradientBrush>

</TextBlock.Background>

</TextBlock>

</Grid>

<Button Grid.Column="0" Grid.Row="1">CE</Button>

<Button Grid.Column="1" Grid.Row="1">C</Button>

<Button Grid.Column="2" Grid.Row="1">

<Image Source="Images/CalcWindow/DeleteChar.png" Stretch="None"/>

</Button>

<Button Grid.Column="3" Grid.Row="1">/</Button>

<Button Grid.Column="0" Grid.Row="2">7</Button>

<Button Grid.Column="1" Grid.Row="2">8</Button>

<Button Grid.Column="2" Grid.Row="2">9</Button>

<Button Grid.Column="3" Grid.Row="2">\*</Button>

<Button Grid.Column="0" Grid.Row="3">4</Button>

<Button Grid.Column="1" Grid.Row="3">5</Button>

<Button Grid.Column="2" Grid.Row="3">6</Button>

<Button Grid.Column="3" Grid.Row="3">-</Button>

<Button Grid.Column="0" Grid.Row="4">1</Button>

<Button Grid.Column="1" Grid.Row="4">2</Button>

<Button Grid.Column="2" Grid.Row="4">3</Button>

<Button Grid.Column="1" Grid.Row="5">0</Button>

<Button Grid.Column="2" Grid.Row="5">,</Button>

<Button Grid.Column="3" Grid.Row="5">=</Button>

<Button Grid.Column="3" Grid.Row="4">+</Button>

<Button Grid.Column="0" Grid.Row="5">&#x00B1;</Button>

</Grid>

</Grid>

</Window>

Листинг модуля Visual.Common.DraftWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Windows.Media.Animation;

using TestPro.Utils;

namespace TestPro.Visual.Common

{

/// <summary>

/// Логика взаимодействия для DraftWindow.xaml

/// </summary>

public partial class DraftWindow : Window

{

public DraftWindow()

{

Logger.Log("TP.V.C.DW - Инициализация окна черновика");

InitializeComponent();

Chrome.GlassFrameThickness = new Thickness(5);

//Text.Focus();

}

private void NewDraft\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

DraftWindow draft = new DraftWindow();

draft.Show();

Logger.Log("TP.V.C.DW - Создание нового черновика");

}

private void DeleteDraft\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Close();

}

private void Colors\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

ThicknessAnimation colorsGridAnimation = new ThicknessAnimation();

colorsGridAnimation.From = gColors.Margin;

colorsGridAnimation.To = new Thickness(0, 0, 0, ActualHeight - 72);

colorsGridAnimation.Duration = TimeSpan.FromSeconds(0.25);

gColors.BeginAnimation(MarginProperty, colorsGridAnimation);

Chrome.CaptionHeight = 0;

iOrange.Focus();

Logger.Log("TP.V.C.DW - Выбор цвета");

}

private void RichTextBox\_GotFocus(object sender, RoutedEventArgs e)

{

ThicknessAnimation colorsGridAnimation = new ThicknessAnimation();

colorsGridAnimation.From = gColors.Margin;

colorsGridAnimation.To = new Thickness(0, 0, 0, ActualHeight);

colorsGridAnimation.Duration = TimeSpan.FromSeconds(0.25);

gColors.BeginAnimation(MarginProperty, colorsGridAnimation);

Chrome.CaptionHeight = NewDraft.ActualHeight;

}

private void Color\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Image Sender = (Image)sender;

RadioButton radioButton = null;

switch (Sender.Name)

{

case "iOrange":

radioButton = rbOrange;

break;

case "iGreen":

radioButton = rbGreen;

break;

case "iBlue":

radioButton = rbBlue;

break;

case "iPurple":

radioButton = rbPurple;

break;

case "iPink":

radioButton = rbPink;

break;

case "iGrey":

radioButton = rbGrey;

break;

}

radioButton.IsChecked = true;

Logger.Log("TP.V.C.DW - Изменение цвета черновика");

}

private void Window\_Closed(object sender, EventArgs e)

{

Logger.Log("TP.V.C.DW - Закрытие окна ");

}

}

}

Листинг модуля Visual.Common.DraftWindow.xaml

<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Common;assembly=TestPro.Visual.Common"

xmlns:System="clr-namespace:System;assembly=mscorlib" x:Class="TestPro.Visual.Common.DraftWindow"

mc:Ignorable="d"

Title="Черновик" Height="360" Width="360" AllowsTransparency="True" WindowStyle="None" Topmost="True"

ResizeMode="NoResize" Closed="Window\_Closed">

<Window.Resources>

<Style x:Key="NewDraftStyle">

<Setter Property="FrameworkElement.Cursor" Value="Hand"/>

<Style.Triggers>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbOrange}" Value="True"/>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="False"/>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="False"/>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="False"/>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="False"/>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="False"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/OrangeNewDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbOrange}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/OrangeNewDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/GreenNewDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/GreenNewDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/BlueNewDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/BlueNewDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/PurpleNewDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/PurpleNewDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/PinkNewDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/PinkNewDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/GreyNewDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=NewDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/NewDraft/GreyNewDraftHover.png"/>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="ColorsStyle">

<Setter Property="FrameworkElement.Cursor" Value="Hand"/>

<Style.Triggers>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbOrange}" Value="True"/>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="False"/>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="False"/>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="False"/>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="False"/>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="False"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/OrangeColors.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbOrange}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/OrangeColorsHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/GreenColors.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/GreenColorsHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/BlueColors.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/BlueColorsHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/PurpleColors.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/PurpleColorsHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/PinkColors.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/PinkColorsHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/GreyColors.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=Colors}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/Colors/GreyColorsHover.png"/>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="DeleteDraftStyle">

<Setter Property="FrameworkElement.Cursor" Value="Hand"/>

<Style.Triggers>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbOrange}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/OrangeDeleteDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbOrange}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/OrangeDeleteDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/GreenDeleteDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/GreenDeleteDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/BlueDeleteDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/BlueDeleteDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/PurpleDeleteDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/PurpleDeleteDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/PinkDeleteDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/PinkDeleteDraftHover.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="False"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/GreyDeleteDraft.png"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="True"/>

<Condition Binding="{Binding IsMouseOver, ElementName=DeleteDraft}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Image.Source" Value="Images/DraftWindow/DeleteDraft/GreyDeleteDraftHover.png"/>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="CaptionStyle">

<Style.Triggers>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbOrange}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFFFB900"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FF108904"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background">

<Setter.Value>

<SolidColorBrush>#FF0078D7</SolidColorBrush>

</Setter.Value>

</Setter>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FF5C239B"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFD900A9"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFBEBEBE"/>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="TextStyle">

<Style.Triggers>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbOrange}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFFFF2B5"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFC7EFC4"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFCAE8FF"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFE1D7ED"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFFFC7F5"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFF3F3F3"/>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="ColorsGridStyle">

<Style.Triggers>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbOrange}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFFFF8E6"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGreen}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFEFFAED"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbBlue}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFEEF7FF"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPurple}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFF6EEFF"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbPink}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFFFEEFB"/>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsChecked, ElementName=rbGrey}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter Property="Control.Background" Value="#FFFBFBFB"/>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="ImageColorsStyle">

<Setter Property="Image.Stretch" Value="Fill"/>

<Setter Property="Control.Focusable" Value="True"/>

<Setter Property="Control.Clip">

<Setter.Value>

<EllipseGeometry>

<EllipseGeometry.RadiusX>20</EllipseGeometry.RadiusX>

<EllipseGeometry.RadiusY>20</EllipseGeometry.RadiusY>

<EllipseGeometry.Center>20,20</EllipseGeometry.Center>

</EllipseGeometry>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="OrangeStyle" BasedOn="{StaticResource ImageColorsStyle}">

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=rbOrange, Path=IsChecked}" Value="False">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/Orange.png"/>

</DataTrigger.Setters>

</DataTrigger>

<DataTrigger Binding="{Binding ElementName=rbOrange, Path=IsChecked}" Value="True">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/OrangeActive.png"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="GreenStyle" BasedOn="{StaticResource ImageColorsStyle}">

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=rbGreen, Path=IsChecked}" Value="False">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/Green.png"/>

</DataTrigger.Setters>

</DataTrigger>

<DataTrigger Binding="{Binding ElementName=rbGreen, Path=IsChecked}" Value="True">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/GreenActive.png"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="BlueStyle" BasedOn="{StaticResource ImageColorsStyle}">

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=rbBlue, Path=IsChecked}" Value="False">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/Blue.png"/>

</DataTrigger.Setters>

</DataTrigger>

<DataTrigger Binding="{Binding ElementName=rbBlue, Path=IsChecked}" Value="True">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/BlueActive.png"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="PurpleStyle" BasedOn="{StaticResource ImageColorsStyle}">

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=rbPurple, Path=IsChecked}" Value="False">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/Purple.png"/>

</DataTrigger.Setters>

</DataTrigger>

<DataTrigger Binding="{Binding ElementName=rbPurple, Path=IsChecked}" Value="True">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/PurpleActive.png"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="PinkStyle" BasedOn="{StaticResource ImageColorsStyle}">

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=rbPink, Path=IsChecked}" Value="False">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/Pink.png"/>

</DataTrigger.Setters>

</DataTrigger>

<DataTrigger Binding="{Binding ElementName=rbPink, Path=IsChecked}" Value="True">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/PinkActive.png"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="GreyStyle" BasedOn="{StaticResource ImageColorsStyle}">

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=rbGrey, Path=IsChecked}" Value="False">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/Grey.png"/>

</DataTrigger.Setters>

</DataTrigger>

<DataTrigger Binding="{Binding ElementName=rbGrey, Path=IsChecked}" Value="True">

<DataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/DraftWindow/ColorImage/GreyActive.png"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

</Window.Resources>

<WindowChrome.WindowChrome>

<WindowChrome x:Name="Chrome" CaptionHeight="40" GlassFrameThickness="5" ResizeBorderThickness="0"/>

</WindowChrome.WindowChrome>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="32"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid>

<Grid.Row>

<System:Int32>0</System:Int32>

</Grid.Row>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="32"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="32"/>

<ColumnDefinition Width="32"/>

</Grid.ColumnDefinitions>

<Image x:Name="NewDraft" Stretch="Fill" MouseLeftButtonUp="NewDraft\_MouseLeftButtonUp"

WindowChrome.IsHitTestVisibleInChrome="True" Style="{StaticResource NewDraftStyle}"/>

<Border Grid.Column="1" Style="{StaticResource CaptionStyle}"/>

<Image x:Name="Colors" Grid.Column="2" WindowChrome.IsHitTestVisibleInChrome="True"

Style="{StaticResource ColorsStyle}" Stretch="Fill" MouseLeftButtonUp="Colors\_MouseLeftButtonUp"/>

<Image x:Name="DeleteDraft" Grid.Column="3" WindowChrome.IsHitTestVisibleInChrome="True"

Style="{StaticResource DeleteDraftStyle}" Stretch="Fill" MouseLeftButtonUp="DeleteDraft\_MouseLeftButtonUp"/>

</Grid>

<RichTextBox x:Name="Text" AcceptsTab="True" AcceptsReturn="True" Style="{StaticResource TextStyle}" SelectionBrush="DarkOrange" BorderThickness="0" VerticalScrollBarVisibility="Auto" FontSize="18" GotFocus="RichTextBox\_GotFocus" TabIndex="0">

<Grid.Row>

<System:Int32>1</System:Int32>

</Grid.Row>

</RichTextBox>

<!---->

<Grid x:Name="gColors" Style="{StaticResource ColorsGridStyle}">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowSpan>

<System:Int32>2</System:Int32>

</Grid.RowSpan>

<Grid.Margin>

<Thickness>0,0,0,360</Thickness>

</Grid.Margin>

<RadioButton x:Name="rbOrange" IsChecked="True" Visibility="Hidden" IsTabStop="False"/>

<RadioButton x:Name="rbGreen" Visibility="Hidden" IsTabStop="False"/>

<RadioButton x:Name="rbBlue" Visibility="Hidden" IsTabStop="False"/>

<RadioButton x:Name="rbPurple" Visibility="Hidden" IsTabStop="False"/>

<RadioButton x:Name="rbPink" Visibility="Hidden" IsTabStop="False"/>

<RadioButton x:Name="rbGrey" Visibility="Hidden" IsTabStop="False"/>

<Image x:Name="iOrange" Grid.Column="0" Margin="12,16,8,16"

MouseLeftButtonUp="Color\_MouseLeftButtonUp" Style="{StaticResource OrangeStyle}">

</Image>

<Image x:Name="iGreen" Grid.Column="1" Margin="10,16,10,16"

MouseLeftButtonUp="Color\_MouseLeftButtonUp" Style="{StaticResource GreenStyle}"/>

<Image x:Name="iBlue" Grid.Column="2" Margin="10,16,10,16"

MouseLeftButtonUp="Color\_MouseLeftButtonUp" Style="{StaticResource BlueStyle}"/>

<Image x:Name="iPurple" Grid.Column="3" Margin="10,16,10,16"

MouseLeftButtonUp="Color\_MouseLeftButtonUp" Style="{StaticResource PurpleStyle}"/>

<Image x:Name="iPink" Grid.Column="4" Margin="10,16,10,16"

MouseLeftButtonUp="Color\_MouseLeftButtonUp" Style="{StaticResource PinkStyle}"/>

<Image x:Name="iGrey" Grid.Column="5" Margin="8,16,12,16"

MouseLeftButtonUp="Color\_MouseLeftButtonUp" Style="{StaticResource GreyStyle}"/>

</Grid>

</Grid>

</Window>

Листинг модуля Visual.Common.ExcelReportWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Windows.Threading;

using TestPro.Test.Main;

using TestPro.Utils;

namespace TestPro.Visual.Common

{

/// <summary>

/// Логика взаимодействия для ExcelReport.xaml

/// </summary>

public partial class ExcelReportWindow : Window, IDisposable

{

Thread ReportThread;

Report Report = new Report();

/// <summary>

/// Конструктор окна ExcelReport

/// </summary>

public ExcelReportWindow()

{

Logger.Log("TP.V.C.ERW - Инициализация окна выгрузки в MS Excel");

InitializeComponent();

}

private void CompactOnHover\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

CompactOn.Visibility = Visibility.Hidden;

TitleBlock.Height = 22;

Chrome.CaptionHeight = 22;

Background = new SolidColorBrush(Color.FromArgb(0, 0, 0, 0));

ProgressBar.Visibility = Visibility.Hidden;

Article.Visibility = Visibility.Hidden;

Description.Visibility = Visibility.Hidden;

Current.Visibility = Visibility.Hidden;

ProgressInfo.Visibility = Visibility.Hidden;

TitleText.Visibility = Visibility.Visible;

ExcelLogo.Visibility = Visibility.Visible;

CompactOff.Visibility = Visibility.Visible;

Logger.Log("TP.V.C.ERW - Минимизация окна");

}

private void CompactOffHover\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

CompactOn.Visibility = Visibility.Visible;

TitleBlock.Height = 12;

Chrome.CaptionHeight = 12;

Background = new LinearGradientBrush(Color.FromArgb(25, 255, 165, 0), Color.FromArgb(255, 0, 58, 255), new Point( .5, 1.0), new Point(0.5, 0.0) );

ProgressBar.Visibility = Visibility.Visible;

Article.Visibility = Visibility.Visible;

Description.Visibility = Visibility.Visible;

Current.Visibility = Visibility.Visible;

ProgressInfo.Visibility = Visibility.Visible;

TitleText.Visibility = Visibility.Hidden;

ExcelLogo.Visibility = Visibility.Hidden;

CompactOff.Visibility = Visibility.Hidden;

Logger.Log("TP.V.C.ERW - Нормализация окна");

}

/// <summary>

/// Устанавливает новое значение для ProgressBar

/// </summary>

/// <param name="sender">Объект, генерирующий событие</param>

/// <param name="NewValue">Стадия создания отчёта</param>

public void SetProgress(object sender, double NewValue)

{

ProgressBar.Dispatcher.Invoke(() =>

{

ProgressBar.Value = NewValue;

ProgressInfo.Text = Report.ProgressInfo;

TaskBarInfo.ProgressValue = NewValue / 100;

});

}

private void ExcelReportForm\_Loaded(object sender, RoutedEventArgs e)

{

Report.ValueChange += SetProgress;

Report.SaveEnd += s =>

{

(s as Report).Dispose();

Dispatcher.Invoke(() => Close());

};

ReportThread = new Thread(StartReport)

{

IsBackground = true,

Priority = ThreadPriority.Normal

};

ReportThread.Start();

Logger.Log("TP.V.C.ERW - Создание потока для выгрузки");

}

private void StartReport()

{

Report.Save();

Logger.Log("TP.V.C.ERW - Начало выгрузки в MS Excel");

}

#region IDisposable Support

private bool disposedValue = false; // Для определения избыточных вызовов

protected virtual void Dispose(bool disposing)

{

if (!disposedValue)

{

if (disposing)

{

Report.Dispose();

}

disposedValue = true;

}

}

public void Dispose()

{

Dispose(true);

Logger.Log("TP.V.C.ERW - Высвобождение ресурсов");

}

#endregion

}

}

Листинг модуля Visual.Common.ExcelReportWindow.xaml

<Window x:Name="ExcelReportForm" x:Class="TestPro.Visual.Common.ExcelReportWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Common"

mc:Ignorable="d"

Title="Выгрузка результатов в MS Excel" Height="240" Width="480" AllowsTransparency="True" WindowStyle="None" Loaded="ExcelReportForm\_Loaded" Topmost="True">

<Window.CacheMode>

<BitmapCache/>

</Window.CacheMode>

<Window.TaskbarItemInfo>

<TaskbarItemInfo x:Name="TaskBarInfo" ProgressState="Normal"/>

</Window.TaskbarItemInfo>

<Window.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF003AFF"/>

<GradientStop Color="#3FFFA500" Offset="1"/>

</LinearGradientBrush>

</Window.Background>

<Window.Resources>

<Style x:Key="CompactOff">

<Setter Property="Control.Margin" Value="445,0,0,219" />

<Setter Property="Image.Source" Value="Images/General/CompactOff/CompactOff.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="CompactOffHover\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/CompactOff/CompactOffHover.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="CompactOn">

<Setter Property="Control.Margin" Value="460,0,0,229" />

<Setter Property="Image.Source" Value="Images/General/CompactOn/CompactOn.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="CompactOnHover\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/CompactOn/CompactOnHover.png" />

</Trigger>

</Style.Triggers>

</Style>

</Window.Resources>

<WindowChrome.WindowChrome>

<WindowChrome x:Name="Chrome" CaptionHeight="12"/>

</WindowChrome.WindowChrome>

<Grid>

<Rectangle x:Name="TitleBlock" Height="12" Stroke="Orange" VerticalAlignment="Top" Fill="Orange"/>

<Image x:Name="CompactOn" Style="{StaticResource CompactOn}" WindowChrome.IsHitTestVisibleInChrome="True"/>

<Image x:Name="CompactOff" Style="{StaticResource CompactOff}" Visibility="Hidden" WindowChrome.IsHitTestVisibleInChrome="True"/>

<TextBlock x:Name="TitleText" HorizontalAlignment="Left" Height="16" Margin="26,1,0,0" Text="Выгрузка результатов в Excel" VerticalAlignment="Top" Width="224" Foreground="Black" FontWeight="Bold" FontStyle="Italic" TextDecorations="Underline" FontSize="14" Visibility="Hidden"/>

<Image x:Name="ExcelLogo" Margin="0,0,458,218" Source="Images/ExcelReportWindow/ExcelLogo.png" Stretch="Fill" Width="22" Height="22" Visibility="Hidden"/>

<ProgressBar x:Name="ProgressBar" HorizontalAlignment="Left" Height="45" Margin="20,155,0,0" VerticalAlignment="Top" Width="435" Value="0">

<ProgressBar.Foreground>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="Yellow" Offset="0.0" />

<GradientStop Color="Red" Offset="0.25" />

<GradientStop Color="Blue" Offset="0.75" />

<GradientStop Color="LimeGreen" Offset="1.0" />

</LinearGradientBrush>

</ProgressBar.Foreground>

</ProgressBar>

<TextBlock x:Name="Article" HorizontalAlignment="Left" Height="43" Margin="140,22,0,0" TextWrapping="Wrap" Text="Внимание!!!" VerticalAlignment="Top" Width="200" FontSize="36"/>

<TextBlock x:Name="Description" HorizontalAlignment="Left" Height="20" Margin="20,80,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="435"><Run Text="Осуществляется выгрузка результатов в "/><Run Text="MS Excel."/><Run Text=" Не закр"/><Run Text="ы"/><Run Text="вайте "/><Run Text="программу!!!"/></TextBlock>

<TextBlock x:Name="Current" HorizontalAlignment="Left" Height="20" Margin="20,120,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="140" FontWeight="Bold" FontSize="14"><Run Text="Текущее действие"/><Run Text=":"/></TextBlock>

<TextBlock x:Name="ProgressInfo" HorizontalAlignment="Left" Height="20" Margin="160,120,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="240"/>

</Grid>

</Window>

Листинг модуля Visual.Common.HintWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using TestPro.Utils;

namespace TestPro.Visual.Common

{

/// <summary>

/// Логика взаимодействия для HintWindow.xaml

/// </summary>

public partial class HintWindow : Window

{

public HintWindow()

{

Logger.Log("TP.V.C.HW - Инициализация окна подсказки");

InitializeComponent();

Count++;

}

public HintWindow(string text) : this()

{

Hint.Text = text;

}

private void Escape\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Close();

}

public static byte Count { get; set; }

private void Window\_Closed(object sender, EventArgs e)

{

Logger.Log("TP.V.C.HW - Закрытие окна");

Count--;

}

private void Window\_StateChanged(object sender, EventArgs e)

{

WindowState = WindowState.Normal;

}

}

}

Листинг модуля Visual.Common.HintWindow.xaml

<Window x:Class="TestPro.Visual.Common.HintWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Common;assembly=TestPro.Test.Visual.Common"

mc:Ignorable="d"

Title="Подсказка" Padding="0" MinHeight="150" MinWidth="150" Height="200" Width="300" WindowStyle="None"

Topmost="True" WindowStartupLocation="CenterScreen" ResizeMode="CanResizeWithGrip" Closed="Window\_Closed"

StateChanged="Window\_StateChanged">

<WindowChrome.WindowChrome>

<WindowChrome CaptionHeight="21" ResizeBorderThickness="2.5"/>

</WindowChrome.WindowChrome>

<Window.Resources>

<Style x:Key="Escape">

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Image.Source" Value="/Images/General/Escape/Escape.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="Escape\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/Escape/EscapeActive.png" />

</Trigger>

</Style.Triggers>

</Style>

</Window.Resources>

<Grid Margin="0">

<Grid.RowDefinitions>

<RowDefinition Height="21"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Margin="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="21"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="35"/>

</Grid.ColumnDefinitions>

<Image Source="Images/TestProLogo.ico"/>

<Rectangle Fill="Orange" Grid.Column="1"/>

<TextBlock x:Name="TitleText" Margin="5,0,0,0" TextWrapping="Wrap" Text="Подсказка" Foreground="Black" FontWeight="Bold" FontStyle="Italic" TextDecorations="Underline" Grid.Column="1" Padding="0,2,0,0"/>

<Image x:Name="Escape" Style="{StaticResource Escape}" Grid.Column="2" WindowChrome.IsHitTestVisibleInChrome="True"/>

</Grid>

<ScrollViewer Padding="0" Grid.Row="1">

<TextBlock x:Name="Hint" TextWrapping="Wrap" Padding="15,10" ScrollViewer.VerticalScrollBarVisibility="Auto"/>

</ScrollViewer>

</Grid>

</Window>

Листинг модуля Visual.Common.IncWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Windows.Shell;

using System.Windows.Controls.Primitives;

using TestPro.Utils;

namespace TestPro.Visual.Common

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class IncWindow : Window

{

/// <summary>

/// Конструктор главной формы

/// </summary>

public IncWindow()

{

Logger.Log("TP.V.C.IW - Инициализация окна холста");

InitializeComponent();

}

private void Escape\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Close();

}

private void EraseByPoint\_Checked(object sender, RoutedEventArgs e)

{

MainCanvas.EditingMode = InkCanvasEditingMode.EraseByPoint;

UnChecked(ref sender);

Logger.Log("TP.V.C.IW - Изменение инструмента");

}

private void Inc\_Checked(object sender, RoutedEventArgs e)

{

MainCanvas.EditingMode = InkCanvasEditingMode.Ink;

UnChecked(ref sender);

Logger.Log("TP.V.C.IW - Изменение инструмента");

}

private void Select\_Checked(object sender, RoutedEventArgs e)

{

MainCanvas.EditingMode = InkCanvasEditingMode.Select;

UnChecked(ref sender);

Logger.Log("TP.V.C.IW - Изменение инструмента");

}

private void EraseByStroke\_Checked(object sender, RoutedEventArgs e)

{

MainCanvas.EditingMode = InkCanvasEditingMode.EraseByStroke;

UnChecked(ref sender);

Logger.Log("TP.V.C.IW - Изменение инструмента");

}

private void Clear\_Checked(object sender, RoutedEventArgs e)

{

MainCanvas.Strokes.Clear();

Clear.IsChecked = false;

Logger.Log("TP.V.C.IW - Изменение инструмента");

}

private void UnChecked(ref object sender)

{

string Name = (sender as ToggleButton).Name;

if (Name != "Inc") Inc.IsChecked = false; else Inc.IsChecked = true;

if (Name != "EraseByPoint") EraseByPoint.IsChecked = false; else EraseByPoint.IsChecked = true;

if (Name != "EraseByStroke") EraseByStroke.IsChecked = false; else EraseByStroke.IsChecked = true;

if (Name != "Select") Select.IsChecked = false; else Select.IsChecked = true;

}

private void Color\_Checked(object sender, RoutedEventArgs e)

{

ColorGrid.Visibility = Visibility.Visible;

Logger.Log("TP.V.C.IW - Выбор цвета");

}

private void ColorGrid\_MouseLeave(object sender, MouseEventArgs e)

{

ColorGrid.Visibility = Visibility.Collapsed;

Color.IsChecked = false;

}

private void Color\_Change(object sender, RoutedPropertyChangedEventArgs<double> e)

{

Color c = new Color();

c.R = (byte)Red.Value;

c.G = (byte)Green.Value;

c.B = (byte)Blue.Value;

c.A = (byte)(255 - (byte)Alpha.Value);

MainCanvas.DefaultDrawingAttributes.Color = c;

ColorBox.Fill = new SolidColorBrush(c);

}

private void Window\_Closed(object sender, EventArgs e)

{

Logger.Log("TP.V.C.IW - Закрытие окна");

}

private void Window\_StateChanged(object sender, EventArgs e)

{

WindowState = WindowState.Normal;

}

}

}

Листинг модуля Visual.Common.IncWindow.xaml

<Window x:Class="TestPro.Visual.Common.IncWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Common;assembly=TestPro.Visual.Common"

mc:Ignorable="d"

Title="Холст" Height="350" Width="525" MinHeight="300" MinWidth="515" AllowsTransparency="True" WindowStyle="None" Topmost="True"

StateChanged="Window\_StateChanged" Closed="Window\_Closed">

<WindowChrome.WindowChrome>

<WindowChrome CaptionHeight="21" ResizeBorderThickness="5"/>

</WindowChrome.WindowChrome>

<Window.Resources>

<Style x:Key="Escape">

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Image.Source" Value="Images/General/Escape/Escape.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="Escape\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/Escape/EscapeActive.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="ToggleButton">

<Setter Property="BorderThickness" Value="0" />

</Style>

<Style x:Key="Selector">

<Setter Property="Image.Source" Value="Images/IncWindow/Selector/Selector.png"/>

<Setter Property="Image.Stretch" Value="Fill" />

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=Select, Path=IsChecked}" Value="True">

<Setter Property="Image.Source" Value="Images/IncWindow/Selector/SelectorActive.png"/>

</DataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=Select, Path=IsChecked}" Value="True"/>

<Condition Binding="{Binding ElementName=Select, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/Selector/SelectorActiveHover.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=Select, Path=IsChecked}" Value="False"/>

<Condition Binding="{Binding ElementName=Select, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/Selector/SelectorActive.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="Pen">

<Setter Property="Image.Source" Value="Images/IncWindow/Pen/Pen.png"/>

<Setter Property="Image.Stretch" Value="Fill" />

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=Inc, Path=IsChecked}" Value="True">

<Setter Property="Image.Source" Value="Images/IncWindow/Pen/PenActive.png"/>

</DataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=Inc, Path=IsChecked}" Value="True"/>

<Condition Binding="{Binding ElementName=Inc, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/Pen/PenActiveHover.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=Inc, Path=IsChecked}" Value="False"/>

<Condition Binding="{Binding ElementName=Inc, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/Pen/PenActive.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="Eraser">

<Setter Property="Image.Source" Value="Images/IncWindow/Eraser/Eraser.png"/>

<Setter Property="Image.Stretch" Value="Fill" />

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=EraseByPoint, Path=IsChecked}" Value="True">

<Setter Property="Image.Source" Value="Images/IncWindow/Eraser/EraserActive.png"/>

</DataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=EraseByPoint, Path=IsChecked}" Value="True"/>

<Condition Binding="{Binding ElementName=EraseByPoint, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/Eraser/EraserActiveHover.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=EraseByPoint, Path=IsChecked}" Value="False"/>

<Condition Binding="{Binding ElementName=EraseByPoint, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/Eraser/EraserActive.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="SuperEraser">

<Setter Property="Image.Source" Value="Images/IncWindow/SuperEraser/SuperEraser.png"/>

<Setter Property="Image.Stretch" Value="Fill" />

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=EraseByStroke, Path=IsChecked}" Value="True">

<Setter Property="Image.Source" Value="Images/IncWindow/SuperEraser/SuperEraserActive.png"/>

</DataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=EraseByStroke, Path=IsChecked}" Value="True"/>

<Condition Binding="{Binding ElementName=EraseByStroke, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/SuperEraser/SuperEraserActiveHover.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=EraseByStroke, Path=IsChecked}" Value="False"/>

<Condition Binding="{Binding ElementName=EraseByStroke, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/SuperEraser/SuperEraserActive.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

</Style.Triggers>

</Style>

<!-- Clear should be hear!!! -->

<Style x:Key="Color">

<Setter Property="Image.Source" Value="Images/IncWindow/Color/Color.png"/>

<Setter Property="Image.Stretch" Value="Fill" />

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=Color, Path=IsChecked}" Value="True">

<Setter Property="Image.Source" Value="Images/IncWindow/Color/ColorActive.png"/>

</DataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=Color, Path=IsChecked}" Value="True"/>

<Condition Binding="{Binding ElementName=Color, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/Color/ColorActiveHover.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding ElementName=Color, Path=IsChecked}" Value="False"/>

<Condition Binding="{Binding ElementName=Color, Path=IsMouseOver}" Value="True" />

</MultiDataTrigger.Conditions>

<MultiDataTrigger.Setters>

<Setter Property="Image.Source" Value="Images/IncWindow/Color/ColorActive.png" />

</MultiDataTrigger.Setters>

</MultiDataTrigger>

</Style.Triggers>

</Style>

</Window.Resources>

<Window.Background>

<RadialGradientBrush>

<GradientStop Color="#3F595959" Offset="1"/>

<GradientStop Color="#3FFFFFFF"/>

</RadialGradientBrush>

</Window.Background>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="21" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid Background="Orange">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="21" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="35" />

</Grid.ColumnDefinitions>

<Image x:Name="TitleLogo">

<Image.Margin>0,0,0,0</Image.Margin>

<Image.Source>Images/TestProLogo.ico</Image.Source>

<Image.Stretch>Fill</Image.Stretch>

</Image>

<TextBlock>

<TextBlock.Text>Полотно</TextBlock.Text>

<TextBlock.FontSize>14</TextBlock.FontSize>

<TextBlock.Margin>5, 0, 0, 0</TextBlock.Margin>

<TextBlock.FontWeight>Bold</TextBlock.FontWeight>

<TextBlock.FontStyle>Italic</TextBlock.FontStyle>

<Grid.Column>1</Grid.Column>

</TextBlock>

<Image x:Name="Escape" Style="{StaticResource Escape}" Grid.Column="2" WindowChrome.IsHitTestVisibleInChrome="True"/>

</Grid>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="25"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid Background="#4FEEF5FD" Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition Height="24.2"/>

<RowDefinition Height="24.2"/>

<RowDefinition Height="24.2"/>

<RowDefinition Height="24.2"/>

<RowDefinition Height="24.2"/>

<RowDefinition Height="24.2"/>

<RowDefinition Height="150"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.Row>1</Grid.Row>

<ToggleButton x:Name="Select" Click="Select\_Checked">

<ToggleButton.Margin>2.5,2.5,2.5,2.5</ToggleButton.Margin>

<Image Style="{StaticResource Selector}"/>

</ToggleButton>

<ToggleButton x:Name="Inc" Click="Inc\_Checked" IsChecked="True" Grid.Row="1">

<ToggleButton.Margin>2.5,2.5,2.5,2.5</ToggleButton.Margin>

<Image Style="{StaticResource Pen}"/>

</ToggleButton>

<ToggleButton x:Name="EraseByPoint" Click="EraseByPoint\_Checked" Grid.Row="2">

<ToggleButton.Margin>2.5,2.5,2.5,2.5</ToggleButton.Margin>

<Image Style="{StaticResource Eraser}"/>

</ToggleButton>

<ToggleButton x:Name="EraseByStroke" Click="EraseByStroke\_Checked" Grid.Row="3">

<ToggleButton.Margin>2.5,2.5,2.5,2.5</ToggleButton.Margin>

<Image Style="{StaticResource SuperEraser}"/>

</ToggleButton>

<ToggleButton x:Name="Clear" Click="Clear\_Checked" Grid.Row="4">

<ToggleButton.Margin>2.5,2.5,2.5,2.5</ToggleButton.Margin>

<Image/>

</ToggleButton>

<ToggleButton x:Name="Color" Click="Color\_Checked" Grid.Row="5">

<ToggleButton.Margin>2.5,2.5,2.5,2.5</ToggleButton.Margin>

<Image Style="{StaticResource Color}"/>

</ToggleButton>

<Expander Margin="1.25,2.5,0,2.5" Grid.Row="6">

<Slider x:Name="Alpha" Margin="0,2.5,0,2.5" SmallChange="1" LargeChange="10" Maximum="255" ValueChanged="Color\_Change" HorizontalAlignment="Center">

<Slider.Orientation>

<Orientation>Vertical</Orientation>

</Slider.Orientation>

</Slider>

</Expander>

</Grid>

<Grid Grid.Column="1" Background="Transparent">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="230"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="85"/>

<RowDefinition Height="80"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<InkCanvas x:Name="MainCanvas" Background="{x:Null}">

<Grid.ColumnSpan>2</Grid.ColumnSpan>

<Grid.RowSpan>3</Grid.RowSpan>

</InkCanvas>

<Grid Name="ColorGrid" Visibility="Collapsed" Background="#3FD000FF" MouseLeave="ColorGrid\_MouseLeave" Panel.ZIndex="5">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.Row>1</Grid.Row>

<Slider x:Name="Red" Margin="0,0,0,42" ValueChanged="Color\_Change">

<Grid.Column>0</Grid.Column>

<Slider.Maximum>255</Slider.Maximum>

<Slider.SmallChange>2</Slider.SmallChange>

<Slider.LargeChange>8</Slider.LargeChange>

<Slider.Background>

<LinearGradientBrush>

<GradientStop Color="#00FF0000" Offset="0" />

<GradientStop Color="#FFFF0000" Offset="1" />

</LinearGradientBrush>

</Slider.Background>

<Slider.Height>18</Slider.Height>

</Slider>

<Slider x:Name="Green" Margin="0,21,0,21" ValueChanged="Color\_Change">

<Grid.Column>0</Grid.Column>

<Slider.Maximum>255</Slider.Maximum>

<Slider.SmallChange>2</Slider.SmallChange>

<Slider.LargeChange>8</Slider.LargeChange>

<Slider.Background>

<LinearGradientBrush>

<GradientStop Color="#0000FF00" Offset="0" />

<GradientStop Color="#FF00FF00" Offset="1" />

</LinearGradientBrush>

</Slider.Background>

<Slider.Height>18</Slider.Height>

</Slider>

<Slider x:Name="Blue" Margin="0,42,0,0" ValueChanged="Color\_Change">

<Grid.Column>0</Grid.Column>

<Slider.Maximum>255</Slider.Maximum>

<Slider.SmallChange>2</Slider.SmallChange>

<Slider.LargeChange>8</Slider.LargeChange>

<Slider.Background>

<LinearGradientBrush>

<GradientStop Color="#000000FF" Offset="0" />

<GradientStop Color="#FF0000FF" Offset="1" />

</LinearGradientBrush>

</Slider.Background>

<Slider.Height>18</Slider.Height>

</Slider>

<Rectangle Name="ColorBox" Fill="#000000">

<Rectangle.Margin>10,10,10,10</Rectangle.Margin>

<Grid.Column>1</Grid.Column>

</Rectangle>

</Grid>

</Grid>

</Grid>

<!--Grid Name="ColorGrid" Visibility="Hidden" Background="#3FD000FF" MouseLeave="ColorGrid\_MouseLeave">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.Margin>25,85,240,165</Grid.Margin>

<Grid.Row>1</Grid.Row>

<Grid.Column>1</Grid.Column>

<Slider x:Name="Red" Margin="0,0,0,42" ValueChanged="Color\_Change">

<Grid.Column>0</Grid.Column>

<Slider.Maximum>255</Slider.Maximum>

<Slider.SmallChange>2</Slider.SmallChange>

<Slider.LargeChange>8</Slider.LargeChange>

<Slider.Background>

<LinearGradientBrush>

<GradientStop Color="#00FF0000" Offset="0" />

<GradientStop Color="#FFFF0000" Offset="1" />

</LinearGradientBrush>

</Slider.Background>

<Slider.Height>18</Slider.Height>

</Slider>

<Slider x:Name="Green" Margin="0,21,0,21" ValueChanged="Color\_Change">

<Grid.Column>0</Grid.Column>

<Slider.Maximum>255</Slider.Maximum>

<Slider.SmallChange>2</Slider.SmallChange>

<Slider.LargeChange>8</Slider.LargeChange>

<Slider.Background>

<LinearGradientBrush>

<GradientStop Color="#0000FF00" Offset="0" />

<GradientStop Color="#FF00FF00" Offset="1" />

</LinearGradientBrush>

</Slider.Background>

<Slider.Height>18</Slider.Height>

</Slider>

<Slider x:Name="Blue" Margin="0,42,0,0" ValueChanged="Color\_Change">

<Grid.Column>0</Grid.Column>

<Slider.Maximum>255</Slider.Maximum>

<Slider.SmallChange>2</Slider.SmallChange>

<Slider.LargeChange>8</Slider.LargeChange>

<Slider.Background>

<LinearGradientBrush>

<GradientStop Color="#000000FF" Offset="0" />

<GradientStop Color="#FF0000FF" Offset="1" />

</LinearGradientBrush>

</Slider.Background>

<Slider.Height>18</Slider.Height>

</Slider>

<Rectangle Name="ColorBox" Fill="#000000">

<Rectangle.Margin>10,10,10,10</Rectangle.Margin>

<Grid.Column>1</Grid.Column>

</Rectangle>

</Grid-->

</Grid>

</Window>

Листинг модуля Visual.Common.LogWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using TestPro.Visual.Core;

using TestPro.Utils;

namespace TestPro.Visual.Common

{

/// <summary>

/// Логика взаимодействия для LogWindow.xaml

/// </summary>

public partial class LogWindow : Window

{

/// <summary>

/// Базовый конструктор окна "Лог"

/// </summary>

public LogWindow()

{

Logger.Log("TP.V.C.LW - Инициализация окна лога");

InitializeComponent();

Logger.NewRecord += Logger\_NewRecord;

LogInfo.Text = Logger.GetLog();

Logger.Log("TP.V.C.LW - Лог загружен");

Logger.Log("TP.V.C.LW - Инициализация завершена");

}

private void Logger\_NewRecord(string message)

{

LogInfo.Text += message.Trim() + Environment.NewLine;

LogInfo.Height += LogInfo.LineHeight;

}

public static byte Count = 0;

//Standard interface functions

#region StandardInterfaceFunctions

private void MinHover\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

WindowState = WindowState.Normal;

}

private void EscapeActive\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Close();

}

private void LogForm\_Closed(object sender, EventArgs e)

{

Count--;

Logger.NewRecord -= Logger\_NewRecord;

Logger.Log("TP.V.C.LW - Закрытие окна лога");

}

private void MaxHover\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

WindowState = WindowState.Maximized;

}

private void CompactOnHover\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

TitleLogo.Visibility = Visibility.Hidden;

Max.Visibility = Visibility.Hidden;

Escape.Visibility = Visibility.Hidden;

CompactOn.Visibility = Visibility.Hidden;

TitleText.Visibility = Visibility.Hidden;

CompactOff.Visibility = Visibility.Visible;

TitleRow.Height = new GridLength(10);

ScrollViewer.Visibility = Visibility.Hidden;

Background = Gradient.GetGradient("fff0", "fff0", "fff0", "fff0");

Chrome.ResizeBorderThickness = new Thickness(0);

ResizeMode = ResizeMode.NoResize;

Logger.Log("TP.V.C.LW - Минимизация окна лога");

}

private void CompactOffHover\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

TitleLogo.Visibility = Visibility.Visible;

Max.Visibility = Visibility.Visible;

Escape.Visibility = Visibility.Visible;

CompactOn.Visibility = Visibility.Visible;

TitleText.Visibility = Visibility.Visible;

CompactOff.Visibility = Visibility.Hidden;

TitleRow.Height = new GridLength(21);

ScrollViewer.Visibility = Visibility.Visible;

Background = Gradient.GetGradient("D9D9D93F", "D9D9D93F", "D9D9D93F", "D9D9D93F");

Chrome.ResizeBorderThickness = new Thickness(2.5);

ResizeMode = ResizeMode.CanResize;

Logger.Log("TP.V.C.LW - Восстановление окна лога");

}

private void LogForm\_StateChanged(object sender, EventArgs e)

{

if (WindowState == WindowState.Maximized)

{

CompactOn.Visibility = Visibility.Hidden;

Max.Visibility = Visibility.Hidden;

Min.Visibility = Visibility.Visible;

BottomRow.Height = new GridLength(

SystemParameters.VirtualScreenHeight - SystemParameters.WorkArea.Height + 7);

Logger.Log("TP.V.C.LW - Максимизация окна лога");

}

else if (WindowState == WindowState.Normal)

{

CompactOn.Visibility = Visibility.Visible;

Min.Visibility = Visibility.Hidden;

Max.Visibility = Visibility.Visible;

BottomRow.Height = new GridLength(0);

Logger.Log("TP.V.C.LW - Нормализация окна лога");

}

else if (WindowState == WindowState.Minimized)

{

WindowState = WindowState.Normal;

Logger.Log("TP.V.C.LW - Попытка минимизации окна лога");

}

}

#endregion

}

}

Листинг модуля Visual.Common.LogWindow.xaml

<Window x:Name="LogForm" x:Class="TestPro.Visual.Common.LogWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Common;assembly=TestPro.Visual.Common"

mc:Ignorable="d"

Title="Лог" Height="300" Width="525" AllowsTransparency="True" WindowStyle="None" Foreground="#00000000"

Background="Transparent" MinHeight="300" MinWidth="525" Topmost="True" Closed="LogForm\_Closed"

StateChanged="LogForm\_StateChanged">

<Window.Resources>

<Style x:Key="Escape">

<Setter Property="WindowChrome.IsHitTestVisibleInChrome" Value="True" />

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Control.Width" Value="35" />

<Setter Property="Control.HorizontalAlignment" Value="Right" />

<Setter Property="Image.Source" Value="Images/General/Escape/Escape.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="EscapeActive\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/Escape/EscapeActive.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="CompactOff">

<Setter Property="WindowChrome.IsHitTestVisibleInChrome" Value="True" />

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Image.Source" Value="Images/General/CompactOff/CompactOff.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="CompactOffHover\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/CompactOff/CompactOffHover.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="CompactOn">

<Setter Property="WindowChrome.IsHitTestVisibleInChrome" Value="True" />

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Image.Source" Value="Images/General/CompactOn/CompactOn.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="CompactOnHover\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/CompactOn/CompactOnHover.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="Max">

<Setter Property="WindowChrome.IsHitTestVisibleInChrome" Value="True" />

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Image.Source" Value="Images/General/Max/Max.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="MaxHover\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/Max/MaxHover.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="Min">

<Setter Property="WindowChrome.IsHitTestVisibleInChrome" Value="True" />

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Control.Width" Value="35" />

<Setter Property="Control.HorizontalAlignment" Value="Right" />

<Setter Property="Image.Source" Value="Images/General/Min/Min.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="MinHover\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/General/Min/MinHover.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="DefinitionStyle">

<Setter Property="RowDefinition.Height" Value="0"/>

<Setter Property="ColumnDefinition.Width" Value="0"/>

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=LogForm,Path=WindowState}" Value="Maximized">

<DataTrigger.Setters>

<Setter Property="RowDefinition.Height" Value="6"/>

<Setter Property="ColumnDefinition.Width" Value="7"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

</Window.Resources>

<WindowChrome.WindowChrome>

<WindowChrome x:Name="Chrome" CaptionHeight="21" ResizeBorderThickness="5"/>

</WindowChrome.WindowChrome>

<Grid x:Name="MainGrid" Background="Transparent">

<Grid.RowDefinitions>

<RowDefinition Style="{StaticResource ResourceKey=DefinitionStyle}"/>

<RowDefinition x:Name="TitleRow" Height="21"/>

<RowDefinition Height="\*"/>

<RowDefinition x:Name="BottomRow" Height="0"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Style="{StaticResource ResourceKey=DefinitionStyle}"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Style="{StaticResource ResourceKey=DefinitionStyle}"/>

</Grid.ColumnDefinitions>

<Grid Grid.Row="1" Grid.Column="1" Background="Orange">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="21"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="35"/>

<ColumnDefinition Width="35"/>

<ColumnDefinition Width="35"/>

</Grid.ColumnDefinitions>

<Image x:Name="TitleLogo" Margin="0" Source="Images/TestProLogo.ico" Stretch="Fill" HorizontalAlignment="Left" Width="21"/>

<Image x:Name="Escape" Style="{StaticResource Escape}" Grid.Column="4" WindowChrome.IsHitTestVisibleInChrome="True"/>

<Image x:Name="Max" Style="{StaticResource Max}" Grid.Column="3" WindowChrome.IsHitTestVisibleInChrome="True"/>

<Image x:Name="Min" Style="{StaticResource Min}" Grid.Column="3" Visibility="Hidden" WindowChrome.IsHitTestVisibleInChrome="True"/>

<Image x:Name="CompactOn" Style="{StaticResource CompactOn}" Grid.Column="2" WindowChrome.IsHitTestVisibleInChrome="True"/>

<Image x:Name="CompactOff" Style="{StaticResource CompactOff}" Grid.Column="4" Visibility="Hidden" WindowChrome.IsHitTestVisibleInChrome="True"/>

<TextBlock x:Name="TitleText" Text="Лог" Foreground="Black" FontWeight="Bold" FontStyle="Italic" TextDecorations="Underline" Grid.Column="1" Padding="5,2,0,0"/>

</Grid>

<ScrollViewer x:Name="ScrollViewer" Background="#3FD9D9D9" Grid.Row="2" Grid.Column="1">

<TextBlock x:Name="LogInfo" Padding="20, 15" TextWrapping="Wrap" ScrollViewer.HorizontalScrollBarVisibility="Visible" ScrollViewer.CanContentScroll="True" Background="#0000CDFF" Foreground="#FF00CDFF" FontSize="20"/>

</ScrollViewer>

</Grid>

</Window>

Листинг модуля Visual.Common.RegistrateTesterWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using TestPro.Test.Main;

using TestPro.Raven;

using TestPro.Utils;

namespace TestPro.Visual.Common

{

/// <summary>

/// Логика взаимодействия для RegistrateTesterWindow.xaml

/// </summary>

public partial class RegistrateTesterWindow : Window

{

public RegistrateTesterWindow()

{

Logger.Log("TP.V.C.RTW - Инициализация окна регистрации участника");

InitializeComponent();

tbSurname.Focus();

if (Settings.UseBackground)

{

if (Settings.Gradient.Active)

{

Background = Settings.Gradient.GetGradient();

}

else if (String.IsNullOrWhiteSpace(Settings.BackgroundName))

{

Background = new ImageBrush(new BitmapImage(new Uri(Settings.BackgroundName, UriKind.Relative)));

}

}

Logger.Log("TP.V.C.RTW - Установка фона");

}

private void Cancel\_Click(object sender, RoutedEventArgs e)

{

Close();

}

private void TextChanged(object sender, TextChangedEventArgs e)

{

if (String.IsNullOrWhiteSpace(tbSurname.Text) || String.IsNullOrWhiteSpace(tbName.Text)

|| String.IsNullOrWhiteSpace(tbGroup.Text))

{

if (bContinious.IsEnabled)

bContinious.IsEnabled = false;

}

else

if (!bContinious.IsEnabled)

bContinious.IsEnabled = true;

}

private void Continious\_Click(object sender, RoutedEventArgs e)

{

if (cbAnonymously.IsChecked != true)

{

Tester tester = new Tester();

tester.Surname = tbSurname.Text;

tester.Name = tbName.Text;

tester.Middlename = tbMiddlename.Text;

tester.Group = tbGroup.Text;

using (var raven = new RavenDB())

{

if (!raven.TesterExists(tester))

{

raven.AddTester(tester);

}

}

Test.Main.Test.CurrentTester = tester;

}

else

Test.Main.Test.CurrentTester = new Tester();

Logger.Log("TP.V.C.RTW - Регистрация участника");

TestWindow test = new TestWindow();

test.Show();

Close();

Logger.Log("TP.V.C.RTW - Начало тестирования");

}

private void RegistrateTesterForm\_Closed(object sender, EventArgs e)

{

Logger.Log("TP.V.C.RTW - Закрытие окна");

}

}

}

Листинг модуля Visual.Common.RegistrateTesterWindow.xaml

<Window x:Name="RegistrateTesterForm" x:Class="TestPro.Visual.Common.RegistrateTesterWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Common;assembly=TestPro.Visual.Common"

mc:Ignorable="d"

Title="Регистрация участника" Height="415" Width="800" MinHeight="415" MinWidth="800"

AllowsTransparency="True" WindowStyle="None" WindowState="Maximized" Closed="RegistrateTesterForm\_Closed">

<WindowChrome.WindowChrome>

<WindowChrome x:Name="Chrome" CaptionHeight="40" ResizeBorderThickness="2.5"/>

</WindowChrome.WindowChrome>

<Window.Resources>

<Style x:Key="FontStyle">

<Setter Property="TextBox.VerticalContentAlignment" Value="Center"/>

<Setter Property="Control.FontSize" Value="20"/>

</Style>

<Style x:Key="AttrsStyle">

<Setter Property="TextBlock.VerticalAlignment" Value="Center"/>

<Setter Property="TextBlock.Margin" Value="15,0,0,0"/>

<Setter Property="Control.FontSize" Value="20"/>

</Style>

<Style x:Key="ContinueStyle" BasedOn="{StaticResource ResourceKey=FontStyle}">

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=cbAnonymously, Path=IsChecked}" Value="False">

<DataTrigger.Setters>

<Setter Property="Control.IsEnabled" Value="False"/>

</DataTrigger.Setters>

</DataTrigger>

</Style.Triggers>

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="40"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="60"/>

<RowDefinition Height="15"/>

</Grid.RowDefinitions>

<TextBlock FontSize="24" FontWeight="ExtraBold">

<TextBlock.Text>Заполнение информации об участнике</TextBlock.Text>

<TextBlock.TextAlignment>Center</TextBlock.TextAlignment>

<TextBlock.Padding>0, 5, 0, 0</TextBlock.Padding>

</TextBlock>

<Grid Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition Height="15"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="25"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="25"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="25"/>

</Grid.ColumnDefinitions>

<Border BorderBrush="Black">

<Grid.Column>1</Grid.Column>

<Grid.Row>1</Grid.Row>

<Border.BorderThickness>2</Border.BorderThickness>

<Border.Background>

<SolidColorBrush>#0C000000</SolidColorBrush>

</Border.Background>

</Border>

<Grid Grid.Column="1" Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="150"/>

</Grid.ColumnDefinitions>

<TextBlock Style="{StaticResource ResourceKey=AttrsStyle}" Grid.Row="0">

<TextBlock.Text>Фамилия</TextBlock.Text>

<TextBlock.Padding>0,3,0,0</TextBlock.Padding>

</TextBlock>

<TextBox x:Name="tbSurname" MaxHeight="40" Style="{StaticResource FontStyle}" TextChanged="TextChanged">

<TextBox.TabIndex>0</TextBox.TabIndex>

<Grid.Column>1</Grid.Column>

<Grid.Row>0</Grid.Row>

</TextBox>

<TextBlock Style="{StaticResource ResourceKey=AttrsStyle}" Margin="0">

<TextBlock.Text>- обязательно</TextBlock.Text>

<TextBlock.Padding>5,3,0,0</TextBlock.Padding>

<Grid.Column>2</Grid.Column>

<Grid.Row>0</Grid.Row>

</TextBlock>

<TextBlock Style="{StaticResource ResourceKey=AttrsStyle}">

<TextBlock.Text>Имя</TextBlock.Text>

<TextBlock.Padding>0,3,0,0</TextBlock.Padding>

<Grid.Column>0</Grid.Column>

<Grid.Row>1</Grid.Row>

</TextBlock>

<TextBox x:Name="tbName" MaxHeight="40" Style="{StaticResource FontStyle}" TextChanged="TextChanged">

<Grid.Column>1</Grid.Column>

<Grid.Row>1</Grid.Row>

<TextBox.TabIndex>1</TextBox.TabIndex>

</TextBox>

<TextBlock Style="{StaticResource ResourceKey=AttrsStyle}" Margin="0">

<TextBlock.Text>- обязательно</TextBlock.Text>

<Grid.Column>2</Grid.Column>

<Grid.Row>1</Grid.Row>

<TextBlock.Padding>5,3,0,0</TextBlock.Padding>

</TextBlock>

<TextBlock Style="{StaticResource ResourceKey=AttrsStyle}">

<TextBlock.Text>Отчество</TextBlock.Text>

<Grid.Column>0</Grid.Column>

<Grid.Row>2</Grid.Row>

<TextBlock.Padding>0,3,0,0</TextBlock.Padding>

</TextBlock>

<TextBox x:Name="tbMiddlename" MaxHeight="40" Style="{StaticResource FontStyle}" TextChanged="TextChanged">

<Grid.Column>1</Grid.Column>

<Grid.Row>2</Grid.Row>

<TextBox.TabIndex>2</TextBox.TabIndex>

</TextBox>

<TextBlock Style="{StaticResource ResourceKey=AttrsStyle}">

<TextBlock.Text>Класс/Группа</TextBlock.Text>

<Grid.Column>0</Grid.Column>

<Grid.Row>3</Grid.Row>

<TextBlock.Padding>0,3,0,0</TextBlock.Padding>

</TextBlock>

<TextBox x:Name="tbGroup" MaxHeight="40" Style="{StaticResource FontStyle}" TextChanged="TextChanged">

<Grid.Column>1</Grid.Column>

<Grid.Row>3</Grid.Row>

<TextBox.TabIndex>3</TextBox.TabIndex>

</TextBox>

<TextBlock Style="{StaticResource ResourceKey=AttrsStyle}" Margin="0">

<TextBlock.Text>- обязательно</TextBlock.Text>

<Grid.Column>2</Grid.Column>

<Grid.Row>3</Grid.Row>

<TextBlock.Padding>5,3,0,0</TextBlock.Padding>

</TextBlock>

<Grid Grid.Row="4" Grid.ColumnSpan="3">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" MaxWidth="225"/>

<ColumnDefinition Width="\*" MaxWidth="225"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<CheckBox x:Name="cbAnonymously" Style="{StaticResource ResourceKey=AttrsStyle}" VerticalContentAlignment="Center">

<CheckBox.Content>Пройти анонимно</CheckBox.Content>

<Grid.Column>0</Grid.Column>

<CheckBox.TabIndex>4</CheckBox.TabIndex>

</CheckBox>

<CheckBox x:Name="cbFirstTest" Style="{StaticResource ResourceKey=AttrsStyle}" VerticalContentAlignment="Center">

<CheckBox.Content>Пройти обучение</CheckBox.Content>

<Grid.Column>1</Grid.Column>

<CheckBox.TabIndex>5</CheckBox.TabIndex>

</CheckBox>

</Grid>

</Grid>

</Grid>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="60"/>

<ColumnDefinition Width="240"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="240"/>

<ColumnDefinition Width="60"/>

</Grid.ColumnDefinitions>

<Button Style="{StaticResource FontStyle}" x:Name="bCancel" Click="Cancel\_Click" Background="#7FDDDDDD">

<Grid.Column>1</Grid.Column>

<Button.Content>Отмена</Button.Content>

<Button.TabIndex>7</Button.TabIndex>

</Button>

<Button x:Name="bContinious" Style="{StaticResource ContinueStyle}" Click="Continious\_Click" Background="#7FDDDDDD">

<Grid.Column>3</Grid.Column>

<Button.Content>Продолжить</Button.Content>

<Button.TabIndex>6</Button.TabIndex>

</Button>

</Grid>

</Grid>

</Window>

Листинг модуля Visual.Common.ResultWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using TestPro.Raven;

using TestPro.Test.Main;

using TestPro.TestKinds;

using TestPro.Utils;

namespace TestPro.Visual.Common

{

/// <summary>

/// Логика взаимодействия для RegistrateTesterWindow.xaml

/// </summary>

public partial class ResultWindow : Window

{

public ResultWindow()

{

Logger.Log("TP.V.C.RW - Инициализация окна результатов");

InitializeComponent();

if (Settings.UseBackground)

{

if (Settings.Gradient.Active)

{

Background = Settings.Gradient.GetGradient();

}

else if (String.IsNullOrWhiteSpace(Settings.BackgroundName))

{

Background = new ImageBrush(new BitmapImage(new Uri(Settings.BackgroundName, UriKind.Relative)));

}

}

Logger.Log("TP.V.C.RW - Установка фона");

List<dynamic> l = new List<dynamic>();

TestResults results = new TestResults()

{

Test = Metadata.Id,

Mark = Test.Main.Test.GetMark(),

Time = Test.Main.Test.Time

};

for (int i = 0; i < Test.Main.Test.Answers.Count(); i++)

{

var a = new

{

Номер = i + 1,

Вопрос = Test.Main.Test.Questions[i].Question,

Ответ = (Test.Main.Test.Questions[i] is OrderTest) ? Test.Main.Test.Answers[i].Answer : ((Test.Main.Test.Questions[i] is BigTest btt) ? btt.Answers[Convert.ToInt32(Test.Main.Test.Answers[i].Answer) - 1] : Test.Main.Test.Answers[i].Answer),

Верный\_ответ = (Test.Main.Test.Questions[i] is OrderTest ot) ? ot.RightAnswer : ((Test.Main.Test.Questions[i] is BigTest bt) ? bt.Answers[Convert.ToInt32(bt.RightAnswer) - 1] : Test.Main.Test.Questions[i].RightAnswer),

Время = (Metadata.Stopwatch) ? Settings.IntToTime(Test.Main.Test.Answers[i].StopwatchEnd - Test.Main.Test.Answers[i].StopwatchBegin) : "Время не учитывалось"

};

results.Answers.Add(Test.Main.Test.GetResult(i));

l.Add(a);

}

if (Test.Main.Test.CurrentTester.Id != null)

{

Test.Main.Test.CurrentTester.Results.Add(results);

using (var raven = new RavenDB())

{

raven.UpdateTester(Test.Main.Test.CurrentTester);

}

}

ResultGrid.ItemsSource = l;

Mark.Text += $" {Test.Main.Test.GetMark()}";

Logger.Log("TP.V.C.RW - Загрузка результатов");

}

private void SaveResults\_Click(object sender, RoutedEventArgs e)

{

new ExcelReportWindow().Show();

}

private void FinishTest\_Click(object sender, RoutedEventArgs e)

{

Close();

}

private void Window\_Closed(object sender, EventArgs e)

{

Logger.Log("TP.V.C.RW - Закрытие окна");

}

}

}

Листинг модуля Visual.Common.ResultWindow.xaml

<Window x:Class="TestPro.Visual.Common.ResultWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Common;assembly=TestPro.Visual.Common"

mc:Ignorable="d"

Title="Результаты тестирования" Height="415" Width="800" AllowsTransparency="True" WindowStyle="None"

WindowState="Maximized" Closed="Window\_Closed">

<WindowChrome.WindowChrome>

<WindowChrome x:Name="Chrome" CaptionHeight="40" ResizeBorderThickness="0"/>

</WindowChrome.WindowChrome>

<Window.Resources>

<Style x:Key="FontStyle">

<Setter Property="TextBox.VerticalContentAlignment" Value="Center"/>

<Setter Property="TextBox.FontWeight" Value="Bold"/>

<Setter Property="Control.FontSize" Value="20"/>

</Style>

<Style TargetType="Button">

<Setter Property="FontSize" Value="14"/>

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="Width" Value="200"/>

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="40"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock FontSize="24" FontWeight="ExtraBold">

<TextBlock.Text>Результаты тестирования</TextBlock.Text>

<TextBlock.TextAlignment>Center</TextBlock.TextAlignment>

<TextBlock.VerticalAlignment>Center</TextBlock.VerticalAlignment>

</TextBlock>

<Grid Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition Height="10"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="60"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="15"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="15"/>

</Grid.ColumnDefinitions>

<DataGrid x:Name="ResultGrid" Style="{StaticResource ResourceKey=FontStyle}" AutoGenerateColumns="True" Grid.Column="1" Grid.Row="1"/>

<Grid Grid.Column="1" Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="205"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="205"/>

</Grid.ColumnDefinitions>

<Button x:Name="SaveResults" HorizontalAlignment="Left" Margin="5,10" Content="Сохранить в Excel" Click="SaveResults\_Click"/>

<Button x:Name="FinishTest" Grid.Column="2" HorizontalAlignment="Right" Margin="5,10" Content="Завершить тестирование" Click="FinishTest\_Click"/>

<TextBlock x:Name="Mark" Grid.Column="1" VerticalAlignment="Center" Margin="25,0" Style="{StaticResource ResourceKey=FontStyle}" Text="Оценка: "/>

</Grid>

</Grid>

</Grid>

</Window>

Листинг модуля Visual.Common.TestWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Timers;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using TestPro.TestKinds;

using TestPro.Test.Main;

using TestPro.Visual.Core;

using System.Windows.Media.Animation;

using System.IO;

using TestPro.Utils;

namespace TestPro.Visual.Common

{

/// <summary>

/// Логика взаимодействия для TestWindow.xaml

/// </summary>

public partial class TestWindow : Window

{

public TestWindow()

{

Logger.Log("TP.T.TW - Инициализация окна тестирования");

InitializeComponent();

if (Settings.UseBackground)

{

if (Settings.Gradient.Active)

{

Background = Settings.Gradient.GetGradient();

}

else if (String.IsNullOrWhiteSpace(Settings.BackgroundName))

{

Background = new ImageBrush(new BitmapImage(new Uri(Settings.BackgroundName, UriKind.Relative)));

}

}

Logger.Log("TP.T.TW - Установка фона");

tbTestName.Text = Metadata.Name;

SetSettings();

Fill(ref Test.Main.Test.Questions[CurrentQuestion]);

Test.Main.Test.Answers[CurrentQuestion] = new BasicAnswer();

if (Metadata.Stopwatch)

Test.Main.Test.Answers[CurrentQuestion].StopwatchBegin = StopwatchValue;

if (Metadata.Timer)

Test.Main.Test.Answers[CurrentQuestion].TimerBegin = Metadata.TimerValue;

if (!Metadata.Stopwatch && !Metadata.Timer)

tbTime.Visibility = Visibility.Hidden;

var t = new System.Timers.Timer(1000);

if (Metadata.Stopwatch)

t.Elapsed += (s, e) =>

{

tbTime.Dispatcher.Invoke(new Action(() => tbTime.Text = Settings.IntToTime(StopwatchValue++)));

t.Start();

};

if (Metadata.Timer)

t.Elapsed += (s, e) =>

{

tbTime.Dispatcher.Invoke(new Action(() => tbTime.Text = Settings.IntToTime(Metadata.TimerValue--)));

t.Start();

};

Logger.Log("TP.T.TW - Настройка параметров времени");

if (Metadata.Stopwatch || Metadata.Timer)

{

t.Start();

Logger.Log("TP.T.TW - Запуск потока отсчёта времени");

}

}

private bool CanClose = true;

private int StopwatchValue = 0;

private int CurrentQuestion = 0;

private void SetSettings()

{

if (Metadata.Calc) bCalc.IsEnabled = true; else bCalc.IsEnabled = false;

if (Metadata.Draft) bDraft.IsEnabled = true; else bDraft.IsEnabled = false;

if (Metadata.Inc) bInc.IsEnabled = true; else bInc.IsEnabled = false;

if (Metadata.Hint) bHint.IsEnabled = true; else bHint.IsEnabled = false;

Logger.Log("TP.T.TW - Настройка доступных опций");

}

private void Image\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Image Sender = (Image)sender;

RadioButton radioButton = null;

switch (Sender.Name)

{

case "iFirst":

radioButton = rbFirst;

break;

case "iSecond":

radioButton = rbSecond;

break;

case "iThird":

radioButton = rbThird;

break;

case "iFourth":

radioButton = rbFourth;

break;

case "iFifth":

radioButton = rbFifth;

break;

case "iSixth":

radioButton = rbSixth;

break;

case "iSeventh":

radioButton = rbSeventh;

break;

case "iEighth":

radioButton = rbEighth;

break;

}

First\_Click(radioButton, null);

iFirst.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

iSecond.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

iThird.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

iFourth.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

iFifth.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

iSixth.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

iSeventh.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

iEighth.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

if (radioButton.IsChecked == true)

Sender.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Checked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

}

private void First\_Click(object sender, RoutedEventArgs e)

{

RadioButton Sender = (RadioButton)sender;

Sender.IsChecked = (Sender.IsChecked == true) ? false : true;

Logger.Log("TP.T.TW - Выбор ответа в тесте с выбором");

}

private void Exit\_Click(object sender, RoutedEventArgs e)

{

if (CurrentQuestion != 0 || Check())

{

if (CustomMessageBox.Show("Вы точно хотите прервать прохождение теста?", "Выход", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

CanClose = false;

Close();

Logger.Log("TP.T.TW - Отмена прохождения теста и закрытие окна");

}

}

else

{

CanClose = false;

Close();

Logger.Log("TP.T.TW - Отмена прохождения теста и закрытие окна");

}

}

private void Inc\_Click(object sender, RoutedEventArgs e)

{

IncWindow IncForm = new IncWindow();

IncForm.Show();

}

private void Draft\_Click(object sender, RoutedEventArgs e)

{

DraftWindow draft = new DraftWindow();

draft.Show();

}

private void Calc\_Click(object sender, RoutedEventArgs e)

{

CalcWindow calc = new CalcWindow();

calc.Show();

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

e.Cancel = CanClose;

}

private void Next\_Click(object sender, RoutedEventArgs e)

{

Next();

}

//----------------------------------------------------------

private void Fill(ref BasicTest value)

{

byte rows = 0;

bool optional = false, content = false;

tbQuestion.Text = value.Question;

/\*if (value is BigTest)

{

tbAnswer.Visibility = Visibility.Hidden;

gOrder.Visibility = Visibility.Hidden;

tbFirst.Visibility = Visibility.Visible;

rbFifth.Focus();

SetTab(ref value);

try

{

tbFirst.Text = (value as BigTest).Answers[0];

tbSecond.Text = (value as BigTest).Answers[1];

tbThird.Text = (value as BigTest).Answers[2];

tbFourth.Text = (value as BigTest).Answers[3];

}

catch { }

rows = 4;

optional = true;

content = true;

if (value.AnotherAnswer)

{

tbFifth.Text = "Другой ответ";

rows++;

}

}\*/

if (value is AnswerTest)

{

tbFirst.Visibility = Visibility.Hidden;

gOrder.Visibility = Visibility.Hidden;

tbAnswer.Visibility = Visibility.Visible;

tbAnswer.Focus();

SetTab(ref value);

rows = 1;

optional = false;

content = true;

}

if (value is BigTest bt)

{

tbAnswer.Visibility = Visibility.Hidden;

gOrder.Visibility = Visibility.Hidden;

tbFirst.Visibility = Visibility.Visible;

rbFifth.Focus();

SetTab(ref value);

try

{

tbFirst.Text = bt.Answers[0];

tbSecond.Text = bt.Answers[1];

tbThird.Text = bt.Answers[2];

tbFourth.Text = bt.Answers[3];

tbFifth.Text = bt.Answers[4];

tbSixth.Text = bt.Answers[5];

tbSeventh.Text = bt.Answers[6];

tbEighth.Text = bt.Answers[7];

}

catch { }

rows = (byte)(value as BigTest).Answers.Count;

optional = true;

content = true;

}

if (value is OrderTest ot)

{

tbFirst.Visibility = Visibility.Hidden;

tbAnswer.Visibility = Visibility.Hidden;

gOrder.Visibility = Visibility.Visible;

SetTab(ref value);

SetOrder();

tbOrderFirst.Text = ot.Answers[0];

tbOrderSecond.Text = ot.Answers[1];

tbOrderThird.Text = ot.Answers[2];

tbOrderFourth.Text = ot.Answers[3];

rows = 1;

optional = false;

content = true;

}

SetRows(rows);

SetCols(optional, content);

if (value.Picture.Status)

{

MemoryStream ms = new MemoryStream(value.Picture.Media);

if (string.IsNullOrWhiteSpace(value.Picture.Description))

{

tbQuestion.Text = "Что изображено на иллюстрации?";

}

SetPicture(ms);

}

Logger.Log("TP.T.TW - Заполнение экранных полей полям вопроса");

}

private void SetOrder()

{

double height = gAnswers.ActualHeight / 4 - 60;

double width = gOrder.ActualWidth - 50;

Canvas.SetLeft(tbOrderFirst, 25);

Canvas.SetLeft(tbOrderSecond, 25);

Canvas.SetLeft(tbOrderThird, 25);

Canvas.SetLeft(tbOrderFourth, 25);

Canvas.SetTop(tbOrderFirst, 15);

Canvas.SetTop(tbOrderSecond, 30 + height);

Canvas.SetTop(tbOrderThird, 45 + 2 \* height);

Canvas.SetTop(tbOrderFourth, 60 + 3 \* height);

tbOrderFirst.Height = height;

tbOrderSecond.Height = height;

tbOrderThird.Height = height;

tbOrderFourth.Height = height;

tbOrderFirst.Width = width;

tbOrderSecond.Width = width;

tbOrderThird.Width = width;

tbOrderFourth.Width = width;

order.Clear();

order.Add(tbOrderFirst, 0);

order.Add(tbOrderSecond, 1);

order.Add(tbOrderThird, 2);

order.Add(tbOrderFourth, 3);

Logger.Log("TP.T.TW - Установка порядка полей в тесте с перемещением");

}

private void SetTab(ref BasicTest value)

{

if (value is AnswerTest)

{

rbFirst.IsTabStop = false;

rbSecond.IsTabStop = false;

rbThird.IsTabStop = false;

rbFourth.IsTabStop = false;

rbFifth.IsTabStop = false;

rbSixth.IsTabStop = false;

rbSeventh.IsTabStop = false;

rbEighth.IsTabStop = false;

tbAnswer.IsTabStop = true;

}

if (value is BigTest)

{

rbFirst.IsTabStop = true;

rbSecond.IsTabStop = true;

rbThird.IsTabStop = true;

rbFourth.IsTabStop = true;

rbFifth.IsTabStop = true;

rbSixth.IsTabStop = true;

rbSeventh.IsTabStop = true;

rbEighth.IsTabStop = true;

tbAnswer.IsTabStop = false;

}

if (value is OrderTest)

{

rbFirst.IsTabStop = false;

rbSecond.IsTabStop = false;

rbThird.IsTabStop = false;

rbFourth.IsTabStop = false;

rbFifth.IsTabStop = false;

rbSixth.IsTabStop = false;

rbSeventh.IsTabStop = false;

rbEighth.IsTabStop = false;

tbAnswer.IsTabStop = false;

}

Logger.Log("TP.T.TW - Настройка TabStop");

}

private void SetCols(bool optional, bool content)

{

if (optional)

cFirst.Width = new GridLength(50);

else

cFirst.Width = new GridLength(0);

if (content)

cSecond.Width = new GridLength(1, GridUnitType.Star);

else

cSecond.Width = new GridLength(0);

cMedia.Width = new GridLength(0);

Logger.Log("TP.T.TW - Настройка экранных столбцов");

}

private void SetRows(byte rowsCount)

{

rEighth.Height = new GridLength(1, GridUnitType.Star);

rSeventh.Height = new GridLength(1, GridUnitType.Star);

rSixth.Height = new GridLength(1, GridUnitType.Star);

rFifth.Height = new GridLength(1, GridUnitType.Star);

rFourth.Height = new GridLength(1, GridUnitType.Star);

rThird.Height = new GridLength(1, GridUnitType.Star);

rSecond.Height = new GridLength(1, GridUnitType.Star);

rFirst.Height = new GridLength(1, GridUnitType.Star);

switch (rowsCount)

{

case 7:

rEighth.Height = new GridLength(0);

break;

case 6:

rEighth.Height = new GridLength(0);

rSeventh.Height = new GridLength(0);

break;

case 5:

rEighth.Height = new GridLength(0);

rSeventh.Height = new GridLength(0);

rSixth.Height = new GridLength(0);

break;

case 4:

rEighth.Height = new GridLength(0);

rSeventh.Height = new GridLength(0);

rSixth.Height = new GridLength(0);

rFifth.Height = new GridLength(0);

break;

case 3:

rEighth.Height = new GridLength(0);

rSeventh.Height = new GridLength(0);

rSixth.Height = new GridLength(0);

rFifth.Height = new GridLength(0);

rFourth.Height = new GridLength(0);

break;

case 2:

rEighth.Height = new GridLength(0);

rSeventh.Height = new GridLength(0);

rSixth.Height = new GridLength(0);

rFifth.Height = new GridLength(0);

rFourth.Height = new GridLength(0);

rThird.Height = new GridLength(0);

break;

case 1:

rEighth.Height = new GridLength(0);

rSeventh.Height = new GridLength(0);

rSixth.Height = new GridLength(0);

rFifth.Height = new GridLength(0);

rFourth.Height = new GridLength(0);

rThird.Height = new GridLength(0);

rSecond.Height = new GridLength(0);

break;

}

Logger.Log("TP.T.TW - Настройка экранных строк");

}

private void SetPicture(MemoryStream stream)

{

try

{

iPicture.Source = BitmapFrame.Create(stream);

iPicture.InvalidateVisual();

cMedia.Width = new GridLength(300);

}

catch

{

cMedia.Width = new GridLength(0);

}

Logger.Log("TP.T.TW - Настройка поля медиа-вложения");

}

private void ReadAnswer()

{

if (Test.Main.Test.Questions[CurrentQuestion] is BigTest)

{

if (rbFirst.IsChecked == true)

Test.Main.Test.Answers[CurrentQuestion].Answer = "1";

if (rbSecond.IsChecked == true)

Test.Main.Test.Answers[CurrentQuestion].Answer = "2";

if (rbThird.IsChecked == true)

Test.Main.Test.Answers[CurrentQuestion].Answer = "3";

if (rbFourth.IsChecked == true)

Test.Main.Test.Answers[CurrentQuestion].Answer = "4";

if (rbFifth.IsChecked == true)

Test.Main.Test.Answers[CurrentQuestion].Answer = "5";

if (rbSixth.IsChecked == true)

Test.Main.Test.Answers[CurrentQuestion].Answer = "6";

if (rbSeventh.IsChecked == true)

Test.Main.Test.Answers[CurrentQuestion].Answer = "7";

if (rbEighth.IsChecked == true)

Test.Main.Test.Answers[CurrentQuestion].Answer = "8";

}

if (Test.Main.Test.Questions[CurrentQuestion] is AnswerTest)

{

Test.Main.Test.Answers[CurrentQuestion].Answer = tbAnswer.Text;

}

if (Test.Main.Test.Questions[CurrentQuestion] is OrderTest)

{

string s = "";

for (byte i = 0; i < 4; i++)

s += (order.ElementAt(i).Value + 1).ToString();

Test.Main.Test.Answers[CurrentQuestion].Answer = s;

}

if (Test.Main.Test.Answers[CurrentQuestion].Done)

Test.Main.Test.Answers[CurrentQuestion].DecideChanges++;

else

Test.Main.Test.Answers[CurrentQuestion].Done = true;

if (Metadata.Stopwatch)

{

Test.Main.Test.Answers[CurrentQuestion].StopwatchEnd = StopwatchValue;

}

if (Metadata.Timer)

{

Test.Main.Test.Answers[CurrentQuestion].TimerEnd = Settings.TimerValue;

}

Logger.Log("TP.T.TW - Чтение ответа");

}

private void Clear()

{

rbFirst.IsChecked = false;

iFirst.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

rbSecond.IsChecked = false;

iSecond.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

rbThird.IsChecked = false;

iThird.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

rbFourth.IsChecked = false;

iFourth.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

rbFifth.IsChecked = false;

iFifth.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

rbSixth.IsChecked = false;

iSixth.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

rbSeventh.IsChecked = false;

iSeventh.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

rbEighth.IsChecked = false;

iEighth.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Unchecked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

tbAnswer.Text = "";

SetOrder();

Logger.Log("TP.T.TW - Очистка экранных полей");

}

private bool Check()

{

Logger.Log("TP.T.TW - Проверка допустимости ответа");

if (Test.Main.Test.Questions[CurrentQuestion] is AnswerTest)

{

if (!String.IsNullOrWhiteSpace(tbAnswer.Text)) return true;

}

if (Test.Main.Test.Questions[CurrentQuestion] is OrderTest) return true;

if (Test.Main.Test.Questions[CurrentQuestion] is BigTest)

{

if (rbFirst.IsChecked == true) return true;

else if (rbSecond.IsChecked == true) return true;

else if (rbThird.IsChecked == true) return true;

else if (rbFourth.IsChecked == true) return true;

else if (rbFifth.IsChecked == true) return true;

else if (rbSixth.IsChecked == true) return true;

else if (rbSeventh.IsChecked == true) return true;

else if (rbEighth.IsChecked == true) return true;

}

return false;

}

private void Next()

{

if (!Check())

{

if (Test.Main.Test.Questions[CurrentQuestion] is BigTest)

CustomMessageBox.Show("Сначала выберите ответ!", "Нет ответа", MessageBoxButton.OK, MessageBoxImage.Warning);

if (Test.Main.Test.Questions[CurrentQuestion] is AnswerTest)

CustomMessageBox.Show("Сначала введите ответ!", "Нет ответа", MessageBoxButton.OK, MessageBoxImage.Warning);

Logger.Log("TP.T.TW - Сообщение о необходимости ввода корректного ответа");

return;

}

ReadAnswer();

Clear();

CurrentQuestion++;

Logger.Log("TP.T.TW - Переход к следующему вопросу");

//?Выход

if (CurrentQuestion == Metadata.QuestionsCount)

{

Test.Main.Test.Time = StopwatchValue;

new ResultWindow().Show();

CanClose = false;

Close();

Logger.Log("TP.T.TW - Завершение тестирования");

return;

}

//

Fill(ref Test.Main.Test.Questions[CurrentQuestion]);

if (Test.Main.Test.Answers[CurrentQuestion] is null)

Test.Main.Test.Answers[CurrentQuestion] = new BasicAnswer();

else

RestoreAnswer();

if (Metadata.Stopwatch)

Test.Main.Test.Answers[CurrentQuestion].StopwatchBegin = StopwatchValue;

if (Metadata.Timer)

Test.Main.Test.Answers[CurrentQuestion].TimerBegin = Metadata.TimerValue;

if (Metadata.Timer || Metadata.Stopwatch)

Logger.Log("TP.T.TW - Замер времени");

if (Metadata.Back)

{

if (CurrentQuestion != 0)

bBack.IsEnabled = true;

else

bBack.IsEnabled = false;

Logger.Log("TP.T.TW - Изменение доступности кнопки назад");

}

}

#region OrderTest's Support

private TextBlock activeBlock = null;

private readonly Color activeColor = new Color() { A = 0x7F, R = 0xff, G = 0x76, B = 0x11 };

private readonly Color generalColor = new Color() { A = 0x7F, R = 0xDD, G = 0xDD, B = 0xDD };

private Dictionary<TextBlock, byte> order = new Dictionary<TextBlock, byte>();

private void Order\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

if (e.Source is TextBlock current)

{

if (activeBlock is null)

{

activeBlock = current;

activeBlock.Background = new SolidColorBrush(activeColor);

Panel.SetZIndex(activeBlock, 5);

}

else if (current == activeBlock)

{

activeBlock.Background = new SolidColorBrush(generalColor);

Panel.SetZIndex(activeBlock, 1);

activeBlock = null;

}

else

{

Panel.SetZIndex(current, 4);

DoubleAnimation fts = new DoubleAnimation(Canvas.GetTop(activeBlock), Canvas.GetTop(current), TimeSpan.FromMilliseconds(300));

DoubleAnimation stf = new DoubleAnimation(Canvas.GetTop(current), Canvas.GetTop(activeBlock), TimeSpan.FromMilliseconds(300));

fts.Completed += (s, er) =>

{

var buf = order[activeBlock];

order[activeBlock] = order[current];

order[current] = buf;

activeBlock.Background = new SolidColorBrush(generalColor);

activeBlock = null;

Panel.SetZIndex(current, 1);

};

activeBlock.BeginAnimation(Canvas.TopProperty, fts);

current.BeginAnimation(Canvas.TopProperty, stf);

}

}

}

#endregion

private void Hint\_Click(object sender, RoutedEventArgs e)

{

if (HintWindow.Count == 0)

new HintWindow(Test.Main.Test.Questions[CurrentQuestion].Hint).Show();

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

Clear();

if (Test.Main.Test.Answers[CurrentQuestion].Answer == "")

Test.Main.Test.Answers[CurrentQuestion] = null;

CurrentQuestion--;

Logger.Log("TP.T.TW - Переход к предыдущему вопросу");

Fill(ref Test.Main.Test.Questions[CurrentQuestion]);

Test.Main.Test.Answers[CurrentQuestion].DecideChanges++;

if (Metadata.Stopwatch)

Test.Main.Test.Answers[CurrentQuestion].StopwatchBegin = StopwatchValue;

if (Metadata.Timer)

Test.Main.Test.Answers[CurrentQuestion].TimerBegin = Metadata.TimerValue;

if (Metadata.Timer || Metadata.Stopwatch)

Logger.Log("TP.T.TW - Замер времени");

if (Metadata.Back)

{

if (CurrentQuestion != 0)

bBack.IsEnabled = true;

else

bBack.IsEnabled = false;

Logger.Log("TP.T.TW - Изменение доступности кнопки назад");

}

RestoreAnswer();

}

private void RestoreAnswer()

{

if (Test.Main.Test.Questions[CurrentQuestion] is OrderTest ot)

{

}

else if (Test.Main.Test.Questions[CurrentQuestion] is BigTest bt)

{

Image sen = null;

switch (Convert.ToInt32(Test.Main.Test.Answers[CurrentQuestion].Answer))

{

case 1:

rbFirst.IsChecked = true;

sen = iFirst;

break;

case 2:

rbSecond.IsChecked = true;

sen = iSecond;

break;

case 3:

rbThird.IsChecked = true;

sen = iThird;

break;

case 4:

rbFourth.IsChecked = true;

sen = iFourth;

break;

case 5:

rbFifth.IsChecked = true;

sen = iFifth;

break;

case 6:

rbSixth.IsChecked = true;

sen = iSixth;

break;

case 7:

rbSeventh.IsChecked = true;

sen = iSeventh;

break;

case 8:

rbEighth.IsChecked = true;

sen = iEighth;

break;

}

if (!(sen is null))

sen.Source = new BitmapImage(new Uri("Images/TestWindow/Checker/Checked.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

}

else if (Test.Main.Test.Questions[CurrentQuestion] is AnswerTest at)

{

tbAnswer.Text = Test.Main.Test.Answers[CurrentQuestion].Answer;

}

Logger.Log("TP.T.TW - Восстановление данного ответа на вопрос");

}

private void Window\_KeyUp(object sender, KeyEventArgs e)

{

if (tbAnswer.Visibility != Visibility.Visible)

{

if (e.Key == Key.Enter)

Next();

else if (e.Key == Key.Back && CurrentQuestion > 0)

Back\_Click(sender, e);

if ((CurrentQuestion > 0 && e.Key == Key.Enter) || e.Key == Key.Back)

Logger.Log("TP.T.TW - Переход с помощью клавиш");

if (Test.Main.Test.Questions[CurrentQuestion] is BigTest bt

&& !(Test.Main.Test.Questions[CurrentQuestion] is OrderTest))

{

if ((int)e.Key >= 35 && (int)e.Key <= 42)

{

int num = (int)e.Key - 34;

KeyChangingAnswer(num, bt.Answers.Count);

}

else if ((int)e.Key >= 75 && (int)e.Key <= 82)

{

int num = (int)e.Key - 74;

KeyChangingAnswer(num, bt.Answers.Count);

}

}

}

}

private void KeyChangingAnswer(int number, int count)

{

if (number <= count)

{

switch (number)

{

case 1:

Image\_MouseLeftButtonUp(iFirst, null);

break;

case 2:

Image\_MouseLeftButtonUp(iSecond, null);

break;

case 3:

Image\_MouseLeftButtonUp(iThird, null);

break;

case 4:

Image\_MouseLeftButtonUp(iFourth, null);

break;

case 5:

Image\_MouseLeftButtonUp(iFifth, null);

break;

case 6:

Image\_MouseLeftButtonUp(iSixth, null);

break;

case 7:

Image\_MouseLeftButtonUp(iSeventh, null);

break;

case 8:

Image\_MouseLeftButtonUp(iEighth, null);

break;

}

}

}

private void Window\_MouseUp(object sender, MouseButtonEventArgs e)

{

if (e.ChangedButton == MouseButton.XButton2)

Next();

else if (e.ChangedButton == MouseButton.XButton1 && CurrentQuestion > 0)

Back\_Click(sender, e);

if ((CurrentQuestion > 0 && e.ChangedButton == MouseButton.XButton1) || e.ChangedButton == MouseButton.XButton2)

Logger.Log("TP.T.TW - Переход с помощью дополнительных клавиш мыши");

}

}

}

Листинг модуля Visual.Common.TestWindow.xaml

<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Common;assembly=TestPro.Visual.Common"

xmlns:System="clr-namespace:System;assembly=mscorlib" x:Class="TestPro.Visual.Common.TestWindow"

mc:Ignorable="d"

Title="Тестирование" Height="720" Width="1280" AllowsTransparency="True" WindowStyle="None"

WindowState="Maximized" Closing="Window\_Closing" KeyUp="Window\_KeyUp" MouseUp="Window\_MouseUp">

<Window.Resources>

<Style x:Key="ButtonStyle">

<Setter Property="Control.IsTabStop" Value="False"/>

<Setter Property="Control.Background">

<Setter.Value>

<SolidColorBrush>#7FDDDDDD</SolidColorBrush>

</Setter.Value>

</Setter>

<Setter Property="Control.FontSize" Value="20"/>

</Style>

<Style x:Key="TextBlockStyle" BasedOn="{StaticResource ButtonStyle}">

<Setter Property="Control.BorderBrush">

<Setter.Value>

<SolidColorBrush>#FF707070</SolidColorBrush>

</Setter.Value>

</Setter>

<Setter Property="Control.BorderThickness" Value="1"/>

<Setter Property="Control.Padding" Value="3"/>

<Setter Property="TextBlock.Padding" Value="15,10"/>

</Style>

<Style x:Key="AnswersTextBlockStyle" BasedOn="{StaticResource TextBlockStyle}">

<Setter Property="TextBlock.TextWrapping" Value="Wrap"/>

<Setter Property="Control.VerticalAlignment" Value="Center"/>

<Setter Property="Control.Padding" Value="5"/>

<Style.Triggers>

<Trigger Property="Control.IsMouseOver" Value="True">

<Setter Property="Control.Background">

<Setter.Value>

<SolidColorBrush>#8FDDDDDD</SolidColorBrush>

</Setter.Value>

</Setter>

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="RadioButtonStyle">

<Setter Property="Control.HorizontalAlignment" Value="Center"/>

<Setter Property="Control.VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="ScrollViewerStyle">

<Setter Property="Control.Background">

<Setter.Value>

<SolidColorBrush>#7FDDDDDD</SolidColorBrush>

</Setter.Value>

</Setter>

<Setter Property="Control.BorderBrush">

<Setter.Value>

<SolidColorBrush>#FF707070</SolidColorBrush>

</Setter.Value>

</Setter>

<Setter Property="Control.BorderThickness" Value="1"/>

</Style>

<Style x:Key="BorderStyle">

<Setter Property="Control.BorderBrush">

<Setter.Value>

<SolidColorBrush>#FF707070</SolidColorBrush>

</Setter.Value>

</Setter>

<Setter Property="Control.BorderThickness" Value="1"/>

</Style>

<Style x:Key="OrderAnswersBlockStyle" BasedOn="{StaticResource ResourceKey=AnswersTextBlockStyle}">

<Setter Property="TextBlock.Padding" Value="25,15"/>

<Setter Property="TextBlock.VerticalAlignment" Value="Center"/>

<Setter Property="TextBlock.Background" Value="#7FDDDDDD"/>

<!-- 7FDDDDDD-->

</Style>

</Window.Resources>

<Grid>

<Grid.Margin>6</Grid.Margin>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="20\*"/>

<RowDefinition Height="80\*"/>

<RowDefinition Height="60"/>

</Grid.RowDefinitions>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="100"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="100"/>

</Grid.ColumnDefinitions>

<Grid>

<Grid.Column>1</Grid.Column>

<Grid.ColumnSpan>2</Grid.ColumnSpan>

<Grid.Background>

<SolidColorBrush>#7FDDDDDD</SolidColorBrush>

</Grid.Background>

</Grid>

<Border Grid.Column="1" Grid.Row="0" Style="{StaticResource BorderStyle}"/>

<TextBlock x:Name="tbTestName">

<TextBlock.Text>Название теста</TextBlock.Text>

<TextBlock.TextAlignment>

<TextAlignment>Center</TextAlignment>

</TextBlock.TextAlignment>

<TextBlock.VerticalAlignment>

<VerticalAlignment>Center</VerticalAlignment>

</TextBlock.VerticalAlignment>

<TextBlock.FontSize>

<System:Double>24</System:Double>

</TextBlock.FontSize>

<TextBlock.FontWeight>

<FontWeight>ExtraBold</FontWeight>

</TextBlock.FontWeight>

<Grid.Column>

<System:Int32>1</System:Int32>

</Grid.Column>

</TextBlock>

<TextBlock x:Name="tbTime">

<TextBlock.FontSize>

<System:Double>22</System:Double>

</TextBlock.FontSize>

<TextBlock.FontWeight>

<FontWeight>ExtraBold</FontWeight>

</TextBlock.FontWeight>

<TextBlock.TextAlignment>

<TextAlignment>Right</TextAlignment>

</TextBlock.TextAlignment>

<Grid.Column>

<System:Int32>2</System:Int32>

</Grid.Column>

<TextBlock.VerticalAlignment>

<VerticalAlignment>Center</VerticalAlignment>

</TextBlock.VerticalAlignment>

<TextBlock.Text>00:00</TextBlock.Text>

<TextBlock.FontFamily>Ravie</TextBlock.FontFamily>

<TextBlock.Padding>0,0,10,0</TextBlock.Padding>

</TextBlock>

<Border Grid.Column="2" Grid.Row="0" Style="{StaticResource BorderStyle}"/>

<Button x:Name="bExit" Style="{StaticResource ButtonStyle}" Click="Exit\_Click">

<Grid.Column>

<System:Int32>0</System:Int32>

</Grid.Column> Выйти

</Button>

</Grid>

<Border Grid.Column="0" Grid.Row="1" Style="{StaticResource BorderStyle}"/>

<ScrollViewer>

<Grid.Row>1</Grid.Row>

<ScrollViewer.Background>

<SolidColorBrush>#7FDDDDDD</SolidColorBrush>

</ScrollViewer.Background>

<ScrollViewer.BorderBrush>

<SolidColorBrush>#FF707070</SolidColorBrush>

</ScrollViewer.BorderBrush>

<ScrollViewer.BorderThickness>1</ScrollViewer.BorderThickness>

<TextBlock x:Name="tbQuestion" Style="{StaticResource TextBlockStyle}" TextWrapping="Wrap">

<TextBlock.Text>Вопрос</TextBlock.Text>

</TextBlock>

</ScrollViewer>

<Grid>

<Grid.Row>2</Grid.Row>

<Grid.ColumnDefinitions>

<ColumnDefinition x:Name="cAnswers" Width="\*"/>

<ColumnDefinition x:Name="cMedia" Width="300"/>

</Grid.ColumnDefinitions>

<Grid x:Name="gAnswers">

<Grid.RowDefinitions>

<RowDefinition x:Name="rFirst" Height="\*"/>

<RowDefinition x:Name="rSecond" Height="\*"/>

<RowDefinition x:Name="rThird" Height="\*"/>

<RowDefinition x:Name="rFourth" Height="\*"/>

<RowDefinition x:Name="rFifth" Height="\*"/>

<RowDefinition x:Name="rSixth" Height="\*"/>

<RowDefinition x:Name="rSeventh" Height="\*"/>

<RowDefinition x:Name="rEighth" Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition x:Name="cFirst" Width="50"/>

<ColumnDefinition x:Name="cSecond" Width="\*"/>

</Grid.ColumnDefinitions>

<Border Grid.Column="0" Grid.Row="0" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="0" Grid.Row="1" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="0" Grid.Row="2" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="0" Grid.Row="3" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="0" Grid.Row="4" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="0" Grid.Row="5" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="0" Grid.Row="6" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="0" Grid.Row="7" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="1" Grid.Row="0" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="1" Grid.Row="1" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="1" Grid.Row="2" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="1" Grid.Row="3" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="1" Grid.Row="4" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="1" Grid.Row="5" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="1" Grid.Row="6" Style="{StaticResource BorderStyle}"/>

<Border Grid.Column="1" Grid.Row="7" Style="{StaticResource BorderStyle}"/>

<!-- Обычный тест -->

<RadioButton x:Name="rbFirst" Foreground="#00000000" BorderBrush="#00707070" Background="Transparent" Click="First\_Click">

<Grid.Row>0</Grid.Row>

<RadioButton.Visibility>Hidden</RadioButton.Visibility>

</RadioButton>

<Image x:Name="iFirst" Source="Images/TestWindow/Checker/Unchecked.png" MouseLeftButtonUp="Image\_MouseLeftButtonUp">

<Image.Stretch>Fill</Image.Stretch>

<Grid.Row>0</Grid.Row>

<Image.Cursor>Hand</Image.Cursor>

</Image>

<ScrollViewer Style="{StaticResource ScrollViewerStyle}">

<Grid.Column>

<System:Int32>1</System:Int32>

</Grid.Column>

<Grid.Row>

<System:Int32>0</System:Int32>

</Grid.Row>

<TextBlock x:Name="tbFirst" Style="{StaticResource AnswersTextBlockStyle}">

<TextBlock.VerticalAlignment>

<VerticalAlignment>Center</VerticalAlignment>

</TextBlock.VerticalAlignment>

<TextBlock.Text>Первый</TextBlock.Text>

</TextBlock>

</ScrollViewer>

<RadioButton Style="{StaticResource RadioButtonStyle}" x:Name="rbSecond" Height="15.0983400344849">

<Grid.Row>1</Grid.Row>

<RadioButton.Visibility>Hidden</RadioButton.Visibility>

</RadioButton>

<Image x:Name="iSecond" Source="Images/TestWindow/Checker/Unchecked.png" MouseLeftButtonUp="Image\_MouseLeftButtonUp">

<Image.Stretch>Fill</Image.Stretch>

<Grid.Row>1</Grid.Row>

<Image.Cursor>Hand</Image.Cursor>

</Image>

<ScrollViewer Style="{StaticResource ScrollViewerStyle}">

<Grid.Column>1</Grid.Column>

<Grid.Row>1</Grid.Row>

<TextBlock x:Name="tbSecond" Style="{StaticResource AnswersTextBlockStyle}">

<TextBlock.Text>Второй</TextBlock.Text>

</TextBlock>

</ScrollViewer>

<RadioButton Style="{StaticResource RadioButtonStyle}" x:Name="rbThird">

<Grid.Row>2</Grid.Row>

<RadioButton.Visibility>Hidden</RadioButton.Visibility>

</RadioButton>

<Image x:Name="iThird" Source="Images/TestWindow/Checker/Unchecked.png" MouseLeftButtonUp="Image\_MouseLeftButtonUp">

<Image.Stretch>Fill</Image.Stretch>

<Grid.Row>2</Grid.Row>

<Image.Cursor>Hand</Image.Cursor>

</Image>

<ScrollViewer Style="{StaticResource ScrollViewerStyle}">

<Grid.Column>1</Grid.Column>

<Grid.Row>2</Grid.Row>

<TextBlock x:Name="tbThird" Style="{StaticResource AnswersTextBlockStyle}">

<TextBlock.Text>Третий</TextBlock.Text>

</TextBlock>

</ScrollViewer>

<RadioButton Style="{StaticResource RadioButtonStyle}" x:Name="rbFourth">

<Grid.Row>3</Grid.Row>

<RadioButton.Visibility>Hidden</RadioButton.Visibility>

</RadioButton>

<Image x:Name="iFourth" Source="Images/TestWindow/Checker/Unchecked.png" MouseLeftButtonUp="Image\_MouseLeftButtonUp">

<Image.Stretch>Fill</Image.Stretch>

<Grid.Row>3</Grid.Row>

<Image.Cursor>Hand</Image.Cursor>

</Image>

<ScrollViewer Style="{StaticResource ScrollViewerStyle}">

<Grid.Column>1</Grid.Column>

<Grid.Row>3</Grid.Row>

<TextBlock x:Name="tbFourth" Style="{StaticResource AnswersTextBlockStyle}">

<TextBlock.Text>Четвёртый</TextBlock.Text>

</TextBlock>

</ScrollViewer>

<RadioButton Style="{StaticResource RadioButtonStyle}" x:Name="rbFifth">

<Grid.Row>4</Grid.Row>

<RadioButton.Visibility>Hidden</RadioButton.Visibility>

</RadioButton>

<Image x:Name="iFifth" Source="Images/TestWindow/Checker/Unchecked.png" MouseLeftButtonUp="Image\_MouseLeftButtonUp">

<Image.Stretch>Fill</Image.Stretch>

<Grid.Row>4</Grid.Row>

<Image.Cursor>Hand</Image.Cursor>

</Image>

<ScrollViewer Style="{StaticResource ScrollViewerStyle}">

<Grid.Column>1</Grid.Column>

<Grid.Row>4</Grid.Row>

<TextBlock x:Name="tbFifth" Style="{StaticResource AnswersTextBlockStyle}">

<TextBlock.Text>Пятый</TextBlock.Text>

</TextBlock>

</ScrollViewer>

<RadioButton Style="{StaticResource RadioButtonStyle}" x:Name="rbSixth">

<Grid.Row>5</Grid.Row>

<RadioButton.Visibility>Hidden</RadioButton.Visibility>

</RadioButton>

<Image x:Name="iSixth" Source="Images/TestWindow/Checker/Unchecked.png" MouseLeftButtonUp="Image\_MouseLeftButtonUp">

<Image.Stretch>Fill</Image.Stretch>

<Grid.Row>5</Grid.Row>

<Image.Cursor>Hand</Image.Cursor>

</Image>

<ScrollViewer Style="{StaticResource ScrollViewerStyle}">

<Grid.Column>1</Grid.Column>

<Grid.Row>5</Grid.Row>

<TextBlock x:Name="tbSixth" Style="{StaticResource AnswersTextBlockStyle}">

<TextBlock.Text>Шестой</TextBlock.Text>

</TextBlock>

</ScrollViewer>

<RadioButton Style="{StaticResource RadioButtonStyle}" x:Name="rbSeventh">

<Grid.Row>6</Grid.Row>

<RadioButton.Visibility>Hidden</RadioButton.Visibility>

</RadioButton>

<Image x:Name="iSeventh" Source="Images/TestWindow/Checker/Unchecked.png" MouseLeftButtonUp="Image\_MouseLeftButtonUp">

<Image.Stretch>Fill</Image.Stretch>

<Grid.Row>6</Grid.Row>

<Image.Cursor>Hand</Image.Cursor>

</Image>

<ScrollViewer Style="{StaticResource ScrollViewerStyle}">

<Grid.Column>1</Grid.Column>

<Grid.Row>6</Grid.Row>

<TextBlock x:Name="tbSeventh" Style="{StaticResource AnswersTextBlockStyle}">

<TextBlock.Text>Седьмой</TextBlock.Text>

</TextBlock>

</ScrollViewer>

<RadioButton Style="{StaticResource RadioButtonStyle}" x:Name="rbEighth">

<Grid.Row>7</Grid.Row>

<RadioButton.Visibility>Hidden</RadioButton.Visibility>

</RadioButton>

<Image x:Name="iEighth" Source="Images/TestWindow/Checker/Unchecked.png" MouseLeftButtonUp="Image\_MouseLeftButtonUp">

<Image.Stretch>Fill</Image.Stretch>

<Grid.Row>7</Grid.Row>

<Image.Cursor>Hand</Image.Cursor>

</Image>

<ScrollViewer Style="{StaticResource ScrollViewerStyle}">

<Grid.Column>1</Grid.Column>

<Grid.Row>7</Grid.Row>

<TextBlock x:Name="tbEighth" Style="{StaticResource AnswersTextBlockStyle}">

<TextBlock.Text>Восьмой</TextBlock.Text>

</TextBlock>

</ScrollViewer>

<!-- Вопрос - ответ -->

<TextBox x:Name="tbAnswer" AcceptsReturn="True">

<Grid.Column>1</Grid.Column>

<TextBox.FontSize>20</TextBox.FontSize>

</TextBox>

<!-- Изменение порядка -->

<Canvas x:Name="gOrder" MouseLeftButtonUp="Order\_MouseLeftButtonUp">

<Grid.Column>1</Grid.Column>

<TextBlock x:Name="tbOrderFirst" Style="{StaticResource OrderAnswersBlockStyle}">

<TextBlock.Text>0</TextBlock.Text>

<TextBlock.Padding>5,15</TextBlock.Padding>

</TextBlock>

<TextBlock x:Name="tbOrderSecond" Style="{StaticResource OrderAnswersBlockStyle}">

<TextBlock.Text>1</TextBlock.Text>

<TextBlock.Padding>5,15</TextBlock.Padding>

</TextBlock>

<TextBlock x:Name="tbOrderThird" Style="{StaticResource OrderAnswersBlockStyle}">

<TextBlock.Text>2</TextBlock.Text>

<TextBlock.Padding>5,15</TextBlock.Padding>

</TextBlock>

<TextBlock x:Name="tbOrderFourth" Style="{StaticResource OrderAnswersBlockStyle}">

<TextBlock.Text>3</TextBlock.Text>

<TextBlock.Padding>5,15</TextBlock.Padding>

<TextBlock.TextWrapping>Wrap</TextBlock.TextWrapping>

</TextBlock>

</Canvas>

</Grid>

<Grid Grid.Column="1" Background="#7FDDDDDD">

<Border Grid.Column="1" Grid.Row="0" Style="{StaticResource BorderStyle}"/>

<Image x:Name="iPicture">

<Grid.Column>1</Grid.Column>

<Image.HorizontalAlignment>Center</Image.HorizontalAlignment>

<Image.VerticalAlignment>Center</Image.VerticalAlignment>

</Image>

</Grid>

</Grid>

<Grid>

<Grid.Row>3</Grid.Row>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="bBack" Style="{StaticResource ButtonStyle}" Click="Back\_Click">

<Button.IsEnabled>False</Button.IsEnabled>

<Button.Content>Назад</Button.Content>

<Grid.Column>0</Grid.Column>

</Button>

<Button x:Name="bDraft" Style="{StaticResource ButtonStyle}" Click="Draft\_Click">

<Button.Content>Черновик</Button.Content>

<Grid.Column>1</Grid.Column>

</Button>

<Button x:Name="bCalc" Style="{StaticResource ButtonStyle}" Click="Calc\_Click">

<Button.Content>Калькулятор</Button.Content>

<Grid.Column>2</Grid.Column>

</Button>

<Button x:Name="bInc" Style="{StaticResource ButtonStyle}" Click="Inc\_Click">

<Button.Content>Холст</Button.Content>

<Grid.Column>3</Grid.Column>

</Button>

<Button x:Name="bHint" Style="{StaticResource ButtonStyle}" Click="Hint\_Click">

<Button.Content>Подсказка</Button.Content>

<Grid.Column>4</Grid.Column>

</Button>

<Button x:Name="bNext" Style="{StaticResource ButtonStyle}" Click="Next\_Click">

<Button.Content>Далее</Button.Content>

<Grid.Column>5</Grid.Column>

</Button>

</Grid>

</Grid>

</Window>

Листинг модуля Visual.Core.CustomMessageBox.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace TestPro.Visual.Core

{

/// <summary>

/// Информационное окно

/// </summary>

public static class CustomMessageBox

{

/// <summary>

/// Отображает окно сообщения с сообщением, которое возвращает результат.

/// </summary>

/// <param name="messageBoxText">Сообщение</param>

/// <returns>Значение кнопки нажатой пользователем</returns>

public static MessageBoxResult Show(string messageBoxText)

{

var custom = new CustomMessageBoxWindow(messageBoxText);

MessageBoxResult result = MessageBoxResult.Cancel;

custom.MessageBoxResult += (s, e) => result = e;

custom.ShowDialog();

return result;

}

/// <summary>

/// Отображает окно сообщения с сообщением, которое возвращает результат.

/// </summary>

/// <param name="messageBoxText">Сообщение пользователю</param>

/// <param name="caption">Заголовок окна</param>

/// <returns>Значение кнопки нажатой пользователем</returns>

public static MessageBoxResult Show(string messageBoxText, string caption)

{

var custom = new CustomMessageBoxWindow(messageBoxText, caption);

MessageBoxResult result = MessageBoxResult.Cancel;

custom.MessageBoxResult += (s, e) => result = e;

custom.ShowDialog();

return result;

}

/// <summary>

/// Отображает окно сообщения с сообщением, которое возвращает результат.

/// </summary>

/// <param name="messageBoxText">Сообщение пользователю</param>

/// <param name="caption">Заголовок окна</param>

/// <param name="button">Набор кнопок окна</param>

/// <returns>Значение кнопки нажатой пользователем</returns>

public static MessageBoxResult Show(string messageBoxText, string caption, MessageBoxButton button)

{

var custom = new CustomMessageBoxWindow(messageBoxText, caption, button);

MessageBoxResult result = MessageBoxResult.Cancel;

custom.MessageBoxResult += (s, e) => result = e;

custom.ShowDialog();

return result;

}

/// <summary>

/// Отображает окно сообщения с сообщением, которое возвращает результат.

/// </summary>

/// <param name="messageBoxText">Сообщение пользователю</param>

/// <param name="caption">Заголовок окна</param>

/// <param name="button">Набор кнопок окна</param>

/// <param name="icon">Изображение окна</param>

/// <returns>Значение кнопки нажатой пользователем</returns>

public static MessageBoxResult Show(string messageBoxText, string caption, MessageBoxButton button, MessageBoxImage icon)

{

var custom = new CustomMessageBoxWindow(messageBoxText, caption, button, icon);

MessageBoxResult result = MessageBoxResult.Cancel;

custom.MessageBoxResult += (s, e) => result = e;

custom.ShowDialog();

return result;

}

/// <summary>

/// Отображает окно сообщения с сообщением, которое возвращает результат.

/// </summary>

/// <param name="messageBoxText">Сообщение пользователю</param>

/// <param name="caption">Заголовок окна</param>

/// <param name="button">Набор кнопок окна</param>

/// <param name="icon">Изображение окна</param>

/// <param name="defaultResult">Результат по-умолчанию</param>

/// <returns>Значение кнопки нажатой пользователем</returns>

public static MessageBoxResult Show(string messageBoxText, string caption, MessageBoxButton button, MessageBoxImage icon, MessageBoxResult defaultResult)

{

var custom = new CustomMessageBoxWindow(messageBoxText, caption, button, icon, defaultResult);

MessageBoxResult result = MessageBoxResult.Cancel;

custom.MessageBoxResult += (s, e) => result = e;

custom.ShowDialog();

return result;

}

}

}

Листинг модуля Visual.Core.CustomMessageBoxWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace TestPro.Visual.Core

{

/// <summary>

/// Делегат возврата результата окна

/// </summary>

/// <param name="sender">Элемент вызвавший событие</param>

/// <param name="messageBoxResult">Результат</param>

public delegate void MessageBoxResultHandler(object sender, MessageBoxResult messageBoxResult);

/// <summary>

/// Логика взаимодействия для CustomMessageBoxWindow.xaml

/// </summary>

partial class CustomMessageBoxWindow : Window

{

/// <summary>

/// Событие возврата результата окна

/// </summary>

public event MessageBoxResultHandler MessageBoxResult;

/// <summary>

/// Базовый конструктор окна

/// </summary>

private CustomMessageBoxWindow()

{

InitializeComponent();

LinearGradientBrush newGradient = new LinearGradientBrush();

newGradient.GradientStops.Add(new GradientStop(new Color() { A = 191, R = 0xff, G = 0x76, B = 0x11 }, 0.0));

newGradient.GradientStops.Add(new GradientStop(new Color() { A = 191, R = 0xff, G = 0xff, B = 0xff }, 0.33));

newGradient.GradientStops.Add(new GradientStop(new Color() { A = 191, R = 0x42, G = 0xff, B = 0xae }, 0.66));

newGradient.GradientStops.Add(new GradientStop(new Color() { A = 191, R = 0x51, G = 0x43, B = 0xff }, 1.0));

newGradient.StartPoint = Point.Parse("0,0");

newGradient.EndPoint = Point.Parse("1,1");

Background = newGradient;

}

/// <summary>

/// Конструктор окна

/// </summary>

/// <param name="messageBoxText">Сообщение окна</param>

public CustomMessageBoxWindow(string messageBoxText) : this()

{

Message = messageBoxText;

cdIcon.Width = new GridLength(0);

cdYes.Width = new GridLength(0);

cdNo.Width = new GridLength(0);

cdCancel.Width = new GridLength(0);

cdPrevious.Width = new GridLength(2, GridUnitType.Star);

cdPost.Width = new GridLength(1, GridUnitType.Star);

result = System.Windows.MessageBoxResult.None;

}

/// <summary>

/// Конструктор окна

/// </summary>

/// <param name="messageBoxText">Сообщение окна</param>

/// <param name="caption">Заголовок окна</param>

public CustomMessageBoxWindow(string messageBoxText, string caption) : this(messageBoxText)

{

Caption = caption;

Title = caption;

}

/// <summary>

/// Конструктор окна

/// </summary>

/// <param name="messageBoxText">Сообщение окна</param>

/// <param name="caption">Заголовок окна</param>

/// <param name="button">Набор кнопок окна</param>

public CustomMessageBoxWindow(string messageBoxText, string caption, MessageBoxButton button)

: this(messageBoxText, caption)

{

switch (button)

{

case MessageBoxButton.OK:

break;

case MessageBoxButton.OKCancel:

cdPrevious.Width = new GridLength(2, GridUnitType.Star);

cdCancel.Width = new GridLength(1, GridUnitType.Star);

cdPost.Width = new GridLength(0, GridUnitType.Star);

result = System.Windows.MessageBoxResult.Cancel;

break;

case MessageBoxButton.YesNo:

cdPrevious.Width = new GridLength(2, GridUnitType.Star);

cdYes.Width = new GridLength(1, GridUnitType.Star);

cdNo.Width = new GridLength(1, GridUnitType.Star);

cdOk.Width = new GridLength(0, GridUnitType.Star);

cdCancel.Width = new GridLength(0, GridUnitType.Star);

cdPost.Width = new GridLength(0, GridUnitType.Star);

result = System.Windows.MessageBoxResult.No;

break;

case MessageBoxButton.YesNoCancel:

cdPrevious.Width = new GridLength(1, GridUnitType.Star);

cdYes.Width = new GridLength(1, GridUnitType.Star);

cdNo.Width = new GridLength(1, GridUnitType.Star);

cdOk.Width = new GridLength(0, GridUnitType.Star);

cdCancel.Width = new GridLength(1, GridUnitType.Star);

cdPost.Width = new GridLength(0, GridUnitType.Star);

result = System.Windows.MessageBoxResult.Cancel;

break;

}

}

/// <summary>

/// Конструктор окна

/// </summary>

/// <param name="messageBoxText">Сообщение окна</param>

/// <param name="caption">Заголовок окна</param>

/// <param name="button">Набор кнопок окна</param>

/// <param name="icon">Изображение окна</param>

public CustomMessageBoxWindow(string messageBoxText, string caption, MessageBoxButton button, MessageBoxImage icon)

: this(messageBoxText, caption, button)

{

string imageName = "";

switch (icon)

{

case MessageBoxImage.Error:

imageName = "Error.png";

break;

case MessageBoxImage.Information:

imageName = "Information.png";

break;

case MessageBoxImage.Question:

imageName = "Question.png";

break;

case MessageBoxImage.Warning:

imageName = "Warning.png";

break;

case MessageBoxImage.None:

return;

}

cdIcon.Width = new GridLength(1, GridUnitType.Star);

IconImage.Source = new BitmapImage(new Uri($"Images/MessageBox/{imageName}", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

}

/// <summary>

/// Конструктор окна

/// </summary>

/// <param name="messageBoxText">Сообщение окна</param>

/// <param name="caption">Заголовок окна</param>

/// <param name="button">Набор кнопок окна</param>

/// <param name="icon">Изображение окна</param>

/// <param name="defaultResult">Результат по-умолчанию</param>

public CustomMessageBoxWindow(string messageBoxText, string caption, MessageBoxButton button, MessageBoxImage icon, MessageBoxResult defaultResult)

: this(messageBoxText, caption, button, icon)

{

result = defaultResult;

}

private MessageBoxResult result;

/// <summary>

/// Заголовок окна

/// </summary>

public string Caption { get => tbCaption.Text; set => tbCaption.Text = value; }

/// <summary>

/// Сообщение окна

/// </summary>

public string Message { get => tbMessage.Text; set => tbMessage.Text = value; }

/// <summary>

/// Закрытие окна по нажатию на кнопку закрытия окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Escape\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

Close();

}

/// <summary>

/// Закрытие окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_Closed(object sender, EventArgs e)

{

MessageBoxResult?.Invoke(this, result);

}

/// <summary>

/// Нажатие на кнопку ОК

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Ok\_Click(object sender, RoutedEventArgs e)

{

result = System.Windows.MessageBoxResult.OK;

Close();

}

/// <summary>

/// Нажатие на кнопку Да

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Yes\_Click(object sender, RoutedEventArgs e)

{

result = System.Windows.MessageBoxResult.Yes;

Close();

}

/// <summary>

/// Нажатие на кнопку Нет

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void No\_Click(object sender, RoutedEventArgs e)

{

result = System.Windows.MessageBoxResult.No;

Close();

}

/// <summary>

/// Нажатие на кнопку Отмена

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Cancel\_Click(object sender, RoutedEventArgs e)

{

result = System.Windows.MessageBoxResult.Cancel;

Close();

}

}

}

Листинг модуля Visual.Core.CustomMessageBoxWindow.xaml

<Window x:Class="TestPro.Visual.Core.CustomMessageBoxWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:TestPro.Visual.Core;assembly=TestPro.Visual.Core"

mc:Ignorable="d" Height="175" Width="400" ResizeMode="NoResize" WindowStyle="None"

AllowsTransparency="True" WindowStartupLocation="CenterScreen" ShowInTaskbar="False" Closed="Window\_Closed" Topmost="True">

<WindowChrome.WindowChrome>

<WindowChrome CaptionHeight="21"/>

</WindowChrome.WindowChrome>

<Window.Resources>

<Style x:Key="Escape">

<Setter Property="Control.Margin" Value="0,0,0,0" />

<Setter Property="Image.Source" Value="Images/Escape/Escape.png" />

<Setter Property="Image.Stretch" Value="Fill" />

<EventSetter Event="Image.MouseLeftButtonUp" Handler="Escape\_MouseLeftButtonUp" />

<Style.Triggers>

<Trigger Property="Image.IsMouseOver" Value="True">

<Setter Property="Image.Source" Value="Images/Escape/EscapeActive.png" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="ButtonStyle">

<Setter Property="Control.Background" Value="#7FEAEAEA"/>

<Setter Property="Control.BorderBrush" Value="#BF707070"/>

<Setter Property="Control.MaxWidth" Value="50"/>

<Setter Property="Control.MaxHeight" Value="20"/>

</Style>

<Style x:Key="im">

<Setter Property="Image.Source" Value="Images/MessageBox/Information.png" />

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="21"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Background="#CCFFA500">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="21"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="35"/>

</Grid.ColumnDefinitions>

<Image x:Name="TitleLogo" Source="Images/TestProLogo.ico"/>

<TextBlock x:Name="tbCaption" Grid.Column="1" TextWrapping="Wrap" Foreground="Black" FontWeight="Bold" FontStyle="Italic" TextDecorations="Underline" Padding="5,2,0,5"/>

<Image x:Name="Escape" Style="{StaticResource Escape}" Grid.Column="2" WindowChrome.IsHitTestVisibleInChrome="True"/>

</Grid>

<Grid Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition Height="15"/>

<RowDefinition Height="11\*"/>

<RowDefinition Height="5\*"/>

<RowDefinition Height="15"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="25"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="5\*"/>

<ColumnDefinition Width="25"/>

</Grid.ColumnDefinitions>

<Border Background="#19000000" Grid.Column="1" Grid.ColumnSpan="2" Grid.Row="1" Grid.RowSpan="2" BorderBrush="#5f000000" BorderThickness="2"/>

<Grid Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2">

<Grid.ColumnDefinitions>

<ColumnDefinition x:Name="cdIcon" Width="\*"/>

<ColumnDefinition Width="5\*"/>

</Grid.ColumnDefinitions>

<Image x:Name="IconImage" Margin="3,0,0,0" Stretch="Uniform" Panel.ZIndex="5" Style="{StaticResource im}"/>

<ScrollViewer Grid.Column="1" Margin="0,0,2,0" VerticalScrollBarVisibility="Auto">

<TextBlock x:Name="tbMessage" Padding="15,10" TextWrapping="Wrap" ScrollViewer.CanContentScroll="True" ScrollViewer.VerticalScrollBarVisibility="Auto"/>

</ScrollViewer>

</Grid>

<Grid Grid.Row="2" Grid.Column="2">

<Grid.ColumnDefinitions>

<ColumnDefinition x:Name="cdPrevious" Width="0"/>

<ColumnDefinition x:Name="cdYes" Width="\*"/>

<ColumnDefinition x:Name="cdNo" Width="\*"/>

<ColumnDefinition x:Name="cdOk" Width="\*"/>

<ColumnDefinition x:Name="cdCancel" Width="\*"/>

<ColumnDefinition x:Name="cdPost" Width="0"/>

</Grid.ColumnDefinitions>

<Button x:Name="bYes" Content="Да" Grid.Column="1" Style="{StaticResource ButtonStyle}" Click="Yes\_Click" TabIndex="2"/>

<Button x:Name="bNo" Content="Нет" Grid.Column="2" Style="{StaticResource ButtonStyle}" Click="No\_Click" TabIndex="3"/>

<Button x:Name="bOk" Content="ОК" Grid.Column="3" Style="{StaticResource ButtonStyle}" Click="Ok\_Click" TabIndex="0"/>

<Button x:Name="bCancel" Content="Отмена" Grid.Column="4" Style="{StaticResource ButtonStyle}" Click="Cancel\_Click" TabIndex="1"/>

</Grid>

</Grid>

</Grid>

</Window>

Листинг модуля Visual.Core.Gradient.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Media;

using TestPro.Utils;

namespace TestPro.Visual.Core

{

/// <summary>

/// Активирован градиент

/// </summary>

public delegate void GradientOnHandler();

/// <summary>

/// Деактивирован градиент

/// </summary>

public delegate void GradientOffHandler();

/// <summary>

/// Изменение градиента

/// </summary>

public delegate void GradientChangeHandler();

/// <summary>

/// Класс градиента фона окна

/// </summary>

public class Gradient: ICloneable

{

/// <summary>

/// Включение градиента фона

/// </summary>

public event GradientOnHandler GradientOn;

/// <summary>

/// Выключение градиента фона

/// </summary>

public event GradientOffHandler GradientOff;

/// <summary>

/// Изменение градиента фона

/// </summary>

public event GradientChangeHandler GradientChange;

private void Event() { return; }

private bool active;

/// <summary>

/// Состояние использования фона

/// </summary>

public bool Active

{

get

{

return active;

}

set

{

active = value;

if (value == true)

GradientOn();

else

GradientOff();

}

}

GradientCorner leftTop, rightTop, rightBottom, leftBottom;

/// <summary>

/// Цвет левого-верхнего угла

/// </summary>

public GradientCorner LeftTop

{

get

{

return leftTop;

}

set

{

leftTop = value;

GradientChange();

}

}

/// <summary>

/// Цвет правого-верхнего угла

/// </summary>

public GradientCorner RightTop

{

get

{

return rightTop;

}

set

{

rightTop = value;

GradientChange();

}

}

/// <summary>

/// Цвет правого-нижнего угла

/// </summary>

public GradientCorner RightBottom

{

get

{

return rightBottom;

}

set

{

rightBottom = value;

GradientChange();

}

}

/// <summary>

/// Цвет левого-нижнего угла

/// </summary>

public GradientCorner LeftBottom

{

get

{

return leftBottom;

}

set

{

leftBottom = value;

GradientChange();

}

}

/// <summary>

/// Базовый конструктор класса

/// </summary>

public Gradient()

{

GradientChange += Event;

GradientOn += Event;

GradientOff += Event;

string color = "ffffff";

LeftTop = new GradientCorner(color);

RightTop = new GradientCorner(color);

RightBottom = new GradientCorner(color);

LeftBottom = new GradientCorner(color);

}

/// <summary>

/// Конструктор класса по цветам

/// </summary>

/// <param name="leftTop">Левый-верхний угол</param>

/// <param name="rightTop">Правый-верхний угол</param>

/// <param name="rightBottom">Правый-нижний угол</param>

/// <param name="leftBottom">Левый-нижний угол</param>

public Gradient(GradientCorner leftTop, GradientCorner rightTop, GradientCorner rightBottom, GradientCorner leftBottom) :this()

{

LeftTop = leftTop;

RightTop = rightTop;

RightBottom = rightBottom;

LeftBottom = leftBottom;

}

/// <summary>

/// Конструктор класса по цветам

/// </summary>

/// <param name="leftTop">Левый-верхний угол</param>

/// <param name="rightTop">Правый-верхний угол</param>

/// <param name="rightBottom">Правый-нижний угол</param>

/// <param name="leftBottom">Левый-нижний угол</param>

public Gradient(string leftTop, string rightTop, string rightBottom, string leftBottom) : this()

{

LeftTop = new GradientCorner(leftTop);

RightTop = new GradientCorner(rightTop);

RightBottom = new GradientCorner(rightBottom);

LeftBottom = new GradientCorner(leftBottom);

}

/// <summary>

/// Получение готового градиента

/// </summary>

/// <returns>Градиент</returns>

public LinearGradientBrush GetGradient()

{

LinearGradientBrush newGradient = new LinearGradientBrush();

GradientStop newGradientStop;

newGradientStop = new GradientStop(LeftTop.GetColor(), 0.0);

newGradient.GradientStops.Add(newGradientStop);

newGradientStop = new GradientStop(RightTop.GetColor(), 0.33);

newGradient.GradientStops.Add(newGradientStop);

newGradientStop = new GradientStop(RightBottom.GetColor(), 0.66);

newGradient.GradientStops.Add(newGradientStop);

newGradientStop = new GradientStop(LeftBottom.GetColor(), 1.0);

newGradient.GradientStops.Add(newGradientStop);

newGradient.StartPoint = Point.Parse("0,0");

newGradient.EndPoint = Point.Parse("1,1");

return newGradient;

}

/// <summary>

/// Получение готового градиента с маской по прозрачности

/// </summary>

/// <param name="transparency"></param>

/// <returns>Градиент</returns>

public LinearGradientBrush GetGradient(byte transparency)

{

LinearGradientBrush newGradient = new LinearGradientBrush();

GradientStop newGradientStop;

GradientCorner newCorner = LeftTop.Clone();

newCorner.Alpha = transparency;

newGradientStop = new GradientStop(newCorner.GetColor(), 0.0);

newGradient.GradientStops.Add(newGradientStop);

newCorner = RightTop.Clone();

newCorner.Alpha = transparency;

newGradientStop = new GradientStop(newCorner.GetColor(), 0.33);

newGradient.GradientStops.Add(newGradientStop);

newCorner = RightBottom.Clone();

newCorner.Alpha = transparency;

newGradientStop = new GradientStop(newCorner.GetColor(), 0.66);

newGradient.GradientStops.Add(newGradientStop);

newCorner = LeftBottom.Clone();

newCorner.Alpha = transparency;

newGradientStop = new GradientStop(newCorner.GetColor(), 1.0);

newGradient.GradientStops.Add(newGradientStop);

newGradient.StartPoint = Point.Parse("0,0");

newGradient.EndPoint = Point.Parse("1,1");

return newGradient;

}

/// <summary>

/// Получение готового градиента

/// </summary>

/// <param name="leftTop">Левый-верхний угол</param>

/// <param name="rightTop">Правый-верхний угол</param>

/// <param name="rightBottom">Правый-нижний угол</param>

/// <param name="leftBottom">Левый-нижний угол</param>

/// <returns>Градиент</returns>

public static LinearGradientBrush GetGradient(string leftTop, string rightTop, string rightBottom, string leftBottom)

{

LinearGradientBrush newGradient = new LinearGradientBrush();

GradientStop newGradientStop;

GradientCorner newCorner = new GradientCorner(leftTop);

newGradientStop = new GradientStop(newCorner.GetColor(), 0.0);

newGradient.GradientStops.Add(newGradientStop);

newCorner = new GradientCorner(rightTop);

newGradientStop = new GradientStop(newCorner.GetColor(), 0.33);

newGradient.GradientStops.Add(newGradientStop);

newCorner = new GradientCorner(rightBottom);

newGradientStop = new GradientStop(newCorner.GetColor(), 0.66);

newGradient.GradientStops.Add(newGradientStop);

newCorner = new GradientCorner(leftBottom);

newGradientStop = new GradientStop(newCorner.GetColor(), 1.0);

newGradient.GradientStops.Add(newGradientStop);

newGradient.StartPoint = Point.Parse("0,0");

newGradient.EndPoint = Point.Parse("1,1");

return newGradient;

}

/// <summary>

/// Получение готового градиента

/// </summary>

/// <param name="leftTop">Левый-верхний угол</param>

/// <param name="rightTop">Правый-верхний угол</param>

/// <param name="rightBottom">Правый-нижний угол</param>

/// <param name="leftBottom">Левый-нижний угол</param>

/// <returns>Градиент</returns>

public static LinearGradientBrush GetGradient(GradientCorner leftTop, GradientCorner rightTop, GradientCorner rightBottom, GradientCorner leftBottom)

{

LinearGradientBrush newGradient = new LinearGradientBrush();

GradientStop newGradientStop;

newGradientStop = new GradientStop(leftTop.GetColor(), 0.0);

newGradient.GradientStops.Add(newGradientStop);

newGradientStop = new GradientStop(rightTop.GetColor(), 0.33);

newGradient.GradientStops.Add(newGradientStop);

newGradientStop = new GradientStop(rightBottom.GetColor(), 0.66);

newGradient.GradientStops.Add(newGradientStop);

newGradientStop = new GradientStop(leftBottom.GetColor(), 1.0);

newGradient.GradientStops.Add(newGradientStop);

newGradient.StartPoint = Point.Parse("0,0");

newGradient.EndPoint = Point.Parse("1,1");

return newGradient;

}

/// <summary>

/// Клонирование экземпляра класса

/// </summary>

/// <returns>Упакованная копия текущего экземпляра класса</returns>

object ICloneable.Clone()

{

Gradient Clone = new Gradient();

Clone.LeftTop = LeftTop.Clone();

Clone.RightTop = RightTop.Clone();

Clone.RightBottom = RightBottom.Clone();

Clone.LeftBottom = LeftBottom.Clone();

Clone.Active = Active;

return Clone;

}

/// <summary>

/// Клонирование экземпляра класса

/// </summary>

/// <returns>Копия текущего экземпляра класса</returns>

public Gradient Clone()

{

return (Gradient)(this as ICloneable).Clone();

}

/// <summary>

/// Сравнение двух экземпляров класса

/// </summary>

/// <param name="value">Сравниваемый экземпляр класса</param>

/// <returns>Равенство экземпляров класса</returns>

public bool IsIdentiacal(Gradient value)

{

if (!LeftTop.IsIdentical(value.LeftTop)) return false;

if (!RightTop.IsIdentical(value.RightTop)) return false;

if (!RightBottom.IsIdentical(value.RightBottom)) return false;

if (!LeftBottom.IsIdentical(value.LeftBottom)) return false;

if (Active != value.Active) return false;

return true;

}

}

}

Листинг модуля Visual.Core.GradientCorner.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Media;

using TestPro.Utils;

namespace TestPro.Visual.Core

{

/// <summary>

/// Класс цвета угла градиента

/// </summary>

public class GradientCorner : ICloneable

{

private byte red, green, blue, alpha;

private string sred, sgreen, sblue, salpha;

/// <summary>

/// Знацение по каналу Red

/// </summary>

public byte Red

{

get

{

return red;

}

set

{

red = value;

sred = ByteColorToStringColor(value);

}

}

/// <summary>

/// Знацение по каналу Green

/// </summary>

public byte Green

{

get

{

return green;

}

set

{

green = value;

sgreen = ByteColorToStringColor(value);

}

}

/// <summary>

/// Знацение по каналу Blue

/// </summary>

public byte Blue

{

get

{

return blue;

}

set

{

blue = value;

sblue = ByteColorToStringColor(value);

}

}

/// <summary>

/// Знацение по каналу Alpha

/// </summary>

public byte Alpha

{

get

{

return alpha;

}

set

{

alpha = value;

salpha = ByteColorToStringColor(value);

}

}

/// <summary>

/// Знацение по каналу sRed

/// </summary>

public string sRed

{

get

{

return sred;

}

set

{

if (value.Length == 1)

value = "0" + value;

sred = value;

red = StringColorToByteColor(value);

}

}

/// <summary>

/// Знацение по каналу sGreen

/// </summary>

public string sGreen

{

get

{

return sgreen;

}

set

{

if (value.Length == 1)

value = "0" + value;

sgreen = value;

green = StringColorToByteColor(value);

}

}

/// <summary>

/// Знацение по каналу sBlue

/// </summary>

public string sBlue

{

get

{

return sblue;

}

set

{

if (value.Length == 1)

value = "0" + value;

sblue = value;

blue = StringColorToByteColor(value);

}

}

/// <summary>

/// Знацение по каналу sAlpha

/// </summary>

public string sAlpha

{

get

{

return salpha;

}

set

{

if (value.Length == 1)

value = "0" + value;

salpha = value;

alpha = StringColorToByteColor(value);

}

}

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="color">Цвет RGB(A)</param>

/// <param name="inverted">Цвет в (А)RGB</param>

public GradientCorner(string color, bool inverted = false)

{

if (inverted)

{

if (color.Length == 3)

{

sRed = color.Substring(0, 1);

sGreen = color.Substring(1, 1);

sBlue = color.Substring(2, 1);

sAlpha = "ff";

}

else if (color.Length == 4)

{

sRed = color.Substring(1, 1);

sGreen = color.Substring(2, 1);

sBlue = color.Substring(3, 1);

sAlpha = color.Substring(0, 1);

}

else if (color.Length == 6)

{

sRed = color.Substring(0, 2);

sGreen = color.Substring(2, 2);

sBlue = color.Substring(4, 2);

sAlpha = "ff";

}

else if (color.Length == 8)

{

sRed = color.Substring(2, 2);

sGreen = color.Substring(3, 2);

sBlue = color.Substring(6, 2);

sAlpha = color.Substring(0, 2);

}

else

{

Red = 255;

Green = 255;

Blue = 255;

Alpha = 255;

}

}

else

{

if (color.Length == 3)

{

sRed = color.Substring(0, 1);

sGreen = color.Substring(1, 1);

sBlue = color.Substring(2, 1);

sAlpha = "ff";

}

else if (color.Length == 4)

{

sRed = color.Substring(0, 1);

sGreen = color.Substring(1, 1);

sBlue = color.Substring(2, 1);

sAlpha = color.Substring(3, 1);

}

else if (color.Length == 6)

{

sRed = color.Substring(0, 2);

sGreen = color.Substring(2, 2);

sBlue = color.Substring(4, 2);

sAlpha = "ff";

}

else if (color.Length == 8)

{

sRed = color.Substring(0, 2);

sGreen = color.Substring(2, 2);

sBlue = color.Substring(4, 2);

sAlpha = color.Substring(6, 2);

}

else

{

Red = 255;

Green = 255;

Blue = 255;

Alpha = 0;

}

}

}

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="color">Цвет</param>

public GradientCorner(Color color)

{

Red = color.R;

Green = color.G;

Blue = color.B;

Alpha = color.A;

}

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="red">Значение канала Red</param>

/// <param name="green">Значение канала Green</param>

/// <param name="blue">Значение канала Blue</param>

/// <param name="alpha">Значение канала Alpha</param>

public GradientCorner(byte red, byte green, byte blue, byte alpha)

{

Red = red;

Green = green;

Blue = blue;

Alpha = alpha;

}

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="red">Значение канала Red</param>

/// <param name="green">Значение канала Green</param>

/// <param name="blue">Значение канала Blue</param>

/// <param name="alpha">Значение канала Alpha</param>

public GradientCorner(string red, string green, string blue, string alpha)

{

sRed = red;

sGreen = green;

sBlue = blue;

sAlpha = alpha;

}

/// <summary>

/// Возвращает цвет

/// </summary>

/// <returns>Цвет в RGBA</returns>

public string GetStringColor()

{

return sred + sgreen + sblue + salpha;

}

/// <summary>

/// Возвращает инвертированный цвет

/// </summary>

/// <returns>Цвет в ARGB</returns>

public string GetInvertedStringColor()

{

return salpha + sred + sgreen + sblue;

}

/// <summary>

/// Возвращает кортеж цвета

/// </summary>

/// <returns>Цвет</returns>

public Tuple<byte, byte, byte, byte> GetByteColor()

{

return Tuple.Create(red, green, blue, alpha);

}

/// <summary>

/// Возвращает цвет

/// </summary>

/// <returns>Цвет</returns>

public Color GetColor()

{

Color color = new Color();

color.R = red;

color.G = green;

color.B = blue;

color.A = alpha;

return color;

}

/// <summary>

/// Преобразует десятичное представление цвета в шестнадцатиричное

/// </summary>

/// <param name="color"></param>

/// <returns>Шестнадцатиричный цвет</returns>

private string ByteColorToStringColor(int color)

{

return Convert.ToString(color, 16);

}

/// <summary>

/// Преобразует шестнадцатиричное представление цвета в десятичное

/// </summary>

/// <param name="color"></param>

/// <returns>Десятичный цвет</returns>

private byte StringColorToByteColor(string color)

{

return Convert.ToByte(color, 16);

}

/// <summary>

/// Клонирование экземпляра класса

/// </summary>

/// <returns>Упакованная копия текущего экземпляра класса</returns>

object ICloneable.Clone()

{

return new GradientCorner(GetColor());

}

/// <summary>

/// Клонирование экземпляра класса

/// </summary>

/// <returns>Копия текущего экземпляра класса</returns>

public GradientCorner Clone()

{

return (GradientCorner)(this as ICloneable).Clone();

}

/// <summary>

/// Сравнение двух экземпляров класса

/// </summary>

/// <param name="value">Сравниваемый экземпляр класса</param>

/// <returns>Равенство экземпляров класса</returns>

public bool IsIdentical(GradientCorner value)

{

if (Red != value.Red) return false;

if (Green != value.Green) return false;

if (Blue != value.Blue) return false;

if (Alpha != value.Alpha) return false;

if (sRed != value.sRed) return false;

if (sGreen != value.sGreen) return false;

if (sBlue != value.sBlue) return false;

if (sAlpha != value.sAlpha) return false;

return true;

}

/// <summary>

/// Возвращает цвет из десятичного представления цвета

/// </summary>

/// <param name="red">Значение канала Red</param>

/// <param name="green">Значение канала Green</param>

/// <param name="blue">Значение канала Blue</param>

/// <param name="alpha">Значение канала Alpha</param>

/// <returns>Цвет</returns>

public static Color GetColor(byte red, byte green, byte blue, byte alpha)

{

Color color = new Color();

color.R = red;

color.G = green;

color.B = blue;

color.A = alpha;

return color;

}

/// <summary>

/// Возвращает цвет из шестнадцатиричного представления цвета

/// </summary>

/// <param name="red">Значение канала Red</param>

/// <param name="green">Значение канала Green</param>

/// <param name="blue">Значение канала Blue</param>

/// <param name="alpha">Значение канала Alpha</param>

/// <returns>Цвет</returns>

public static Color GetColor(string red, string green, string blue, string alpha)

{

if (String.IsNullOrWhiteSpace(red) || String.IsNullOrWhiteSpace(green) || String.IsNullOrWhiteSpace(blue) || String.IsNullOrWhiteSpace(alpha))

throw new NullStringException();

Color color = new Color();

color.R = Convert.ToByte(red, 16);

color.G = Convert.ToByte(green, 16);

color.B = Convert.ToByte(blue, 16);

color.A = Convert.ToByte(alpha, 16);

return color;

}

}

}

Листинг модуля TestKinds.AnswerTest.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using TestPro.Utils;

namespace TestPro.TestKinds

{

/// <summary>

/// Тест с письменным ответом на заданный вопрос

/// </summary>

public class AnswerTest: BasicTest

{

/// <summary>

/// Базовый конструктор класса

/// </summary>

public AnswerTest() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="fillString">Строка для расшифровки</param>

public AnswerTest(string fillString) : this()

{

Fill(fillString);

}

/// <summary>

/// Заполнение значений экземпляря класса из входной строки

/// </summary>

/// <param name="fillString">Входная строка</param>

public override void Fill(string fillString)

{

if (fillString == null) throw new ArgumentNullException();

Question = Utils.System.ExecTestValue(ref fillString, "Текст");

RightAnswer = Utils.System.ExecTestValue(ref fillString, "Ответ");

try

{

AnotherAnswer = Convert.ToBoolean(Utils.System.ExecTestValue(ref fillString, "Другой ответ"));

}

catch

{

AnotherAnswer = false;

}

Hint = Utils.System.ExecTestValue(ref fillString, "Подсказка");

Picture.Fill(ref fillString);

Solution = Utils.System.ExecTestValue(ref fillString, "Решение");

if (Picture.Status)

{

Question = Picture.Description;

}

}

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public override string AsString()

{

string QString = QuestionString();

string RaString = RightAnswerString();

string AAString = AnotherAnswerString();

string HString = HintString();

string SString = SolutionString();

string S = Environment.NewLine;

return QString + S + RaString + S + AAString + S + HString + S + SString;

}

}

}

Листинг модуля TestKinds.BasicTest.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestPro.TestKinds

{

/// <summary>

/// Базовый абстрактный класс для создания видов тестов

/// </summary>

public abstract class BasicTest

{

/// <summary>

/// Базовый конструктор

/// </summary>

public BasicTest()

{

Question = "";

RightAnswer = "";

AnotherAnswer = false;

Hint = "";

Solution = "";

Picture = new TestPicture();

}

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="fillString">Строка для расшифровки</param>

public BasicTest(string fillString) : this()

{

Fill(fillString);

}

private string question;

private string rightAnswer;

private bool anotherAnswer;

private string hint;

private string solution;

private TestPicture picture;

/// <summary>

/// Вопрос теста

/// </summary>

public string Question { get => question; set => question = value; }

/// <summary>

/// Правильный ответ

/// </summary>

public string RightAnswer { get => rightAnswer; set => rightAnswer = value; }

/// <summary>

/// Подсказка

/// </summary>

public string Hint { get => hint; set => hint = value; }

/// <summary>

/// Решение

/// </summary>

public string Solution { get => solution; set => solution = value; }

/// <summary>

/// Возможность самостоятельного ввода ответа

/// </summary>

public bool AnotherAnswer { get => anotherAnswer; set => anotherAnswer = value; }

/// <summary>

/// Иллюстрация к вопросу

/// </summary>

public TestPicture Picture { get => picture; set => picture = value; }

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public string QuestionString()

{

return "<Текст>" + Question + "</Текст>";

}

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public string RightAnswerString()

{

return "<Ответ>" + RightAnswer + "</Ответ>";

}

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public string HintString()

{

return "<Подсказка>" + Hint + "</Подсказка>";

}

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public string SolutionString()

{

return "<Решение>" + Solution + "</Решение>";

}

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public string AnotherAnswerString()

{

return $"<Другой ответ>{AnotherAnswer}</Другой ответ>";

}

/// <summary>

/// Заполнение значений экземпляря класса из входной строки

/// </summary>

/// <param name="fillString">Входная строка</param>

public abstract void Fill(string fillString);

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public abstract string AsString();

}

}

Листинг модуля TestKinds.BigTest.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using TestPro.Utils;

namespace TestPro.TestKinds

{

/// <summary>

/// Тест с заданными ответами на вопрос

/// </summary>

public class BigTest : BasicTest

{

/// <summary>

/// Базовый конструктор класса

/// </summary>

public BigTest() : base()

{

Answers = new List<string>();

}

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="fillString">Строка для расшифровки</param>

public BigTest(string fillString) : this()

{

Fill(fillString);

}

private List<string> answers;

/// <summary>

/// Список ответов на вопрос

/// </summary>

public List<string> Answers { get => answers; set => answers = value; }

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public string AnswersString()

{

string result = "";

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

result += $"<Ответ\_{i + 1}>{Answers[i]}</Ответ\_{i + 1}>{((i != Answers.Count - 1)?$"{Environment.NewLine}":"")}";

return result;

}

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public override string AsString()

{

string QString = QuestionString();

string AString = AnswersString();

string RaString = RightAnswerString();

string AAString = AnotherAnswerString();

string HString = HintString();

string SString = SolutionString();

string S = Environment.NewLine;

return QString + S + AString + S + RaString + S + AAString + S + HString + S + SString;

}

/// <summary>

/// Заполнение значений экземпляря класса из входной строки

/// </summary>

/// <param name="fillString">Входная строка</param>

public override void Fill(string fillString)

{

if (String.IsNullOrWhiteSpace(fillString))

throw new NullStringException();

try

{

Question = Utils.System.ExecTestValue(ref fillString, "Текст");

}

catch

{

Question = "";

}

int i = fillString.Split(new string[] { "<Ответ\_" }, StringSplitOptions.None).Count() - 1;

int j = fillString.Split(new string[] { "</Ответ\_" }, StringSplitOptions.None).Count() - 1;

int QuestionsCount = (i <= j) ? i : j;

for (int k = 1; k <= QuestionsCount; k++)

{

i = fillString.IndexOf($"<Ответ\_{k}>");

j = fillString.IndexOf($"</Ответ\_{k}>");

Answers.Add(fillString.Substring(i + 8 + $"{k}".Length, j - i - 8 - $"{k}".Length));

}

RightAnswer = Utils.System.ExecTestValue(ref fillString, "Ответ");

try

{

AnotherAnswer = Convert.ToBoolean(Utils.System.ExecTestValue(ref fillString, "Другой ответ"));

}

catch

{

AnotherAnswer = false;

}

Hint = Utils.System.ExecTestValue(ref fillString, "Подсказка");

Solution = Utils.System.ExecTestValue(ref fillString, "Решение");

Picture.Fill(ref fillString);

if (Picture.Status)

{

Question = Picture.Description;

}

}

/// <summary>

/// Перемешивание порядка вопросов

/// </summary>

/// <param name="random">Генератор значений (используется внешний для исключения повторений)</param>

public void Mix(Random random = null)

{

if (Answers != null)

{

string result = answers[Convert.ToInt32(RightAnswer) - 1];

Dictionary<int, string> dictionary = new Dictionary<int, string>();

if (random is null) random = new Random();

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

dictionary.Add(random.Next(int.MaxValue), Answers[i]);

Answers = dictionary.OrderBy(x => x.Key).Select(x => x.Value).ToList();

RightAnswer = (answers.IndexOf(result) + 1).ToString();

}

}

}

}

Листинг модуля TestKinds.OrderTest.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestPro.TestKinds

{

/// <summary>

/// Тест с изменением порядка ответов на вопрос

/// </summary>

public class OrderTest: BigTest

{

/// <summary>

/// Базовый конструктор класса

/// </summary>

public OrderTest() : base() { }

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="fillString">Строка для расшифровки</param>

public OrderTest(string fillString) : this()

{

Fill(fillString);

}

/// <summary>

/// Перемешивание порядка вопросов

/// </summary>

/// <param name="random">Генератор значений (используется внешний для исключения повторений)</param>

public new void Mix(Random random = null)

{

}

}

}

Листинг модуля TestKinds.TestMedia.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestPro.TestKinds

{

/// <summary>

/// Базовый абстрактный класс для создания видов медиа в тесте

/// </summary>

public abstract class TestMedia

{

string path;

string description;

bool status;

byte[] media;

/// <summary>

/// Базовый конструктор

/// </summary>

protected TestMedia()

{

path = "";

description = "";

status = false;

}

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="fillString">Строка для расшифровки</param>

protected TestMedia(string fillString) : this()

{

Fill(ref fillString);

}

/// <summary>

/// Путь к медиа-вложению

/// </summary>

public string Path { get => path; set => path = value; }

/// <summary>

/// Описание медиа-вложения

/// </summary>

public string Description { get => description; set => description = value; }

/// <summary>

/// Наличие медиа-вложения

/// </summary>

public bool Status { get => status; set => status = value; }

/// <summary>

/// Вложение

/// </summary>

public byte[] Media { get => media; set => media = value; }

/// <summary>

/// Заполнение значений экземпляря класса из входной строки

/// </summary>

/// <param name="fillString">Входная строка</param>

public abstract void Fill(ref string fillString);

}

}

Листинг модуля TestKinds.TestPicture.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using TestPro.Utils;

namespace TestPro.TestKinds

{

/// <summary>

/// Класс вложения-фотографии

/// </summary>

public class TestPicture : TestMedia

{

/// <summary>

/// Базовый конструктор класса

/// </summary>

public TestPicture()

{

Path = "";

Description = "";

Status = false;

}

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="fillString">Строка для расшифровки</param>

public TestPicture(string fillString) : this()

{

Fill(ref fillString);

}

/// <summary>

/// Заполнение значений экземпляря класса из входной строки

/// </summary>

/// <param name="fillString">Входная строка</param>

public override void Fill(ref string fillString)

{

if (String.IsNullOrWhiteSpace(fillString))

throw new NullStringException();

int i = fillString.IndexOf("[Фото]");

if (!Utils.System.NotExistsCheck(i)) return;

int j = fillString.IndexOf("[/Фото]");

if (!Utils.System.NotExistsCheck(j)) return;

string PictureString = fillString.Substring(i + 6, j - i - 6);

if (PictureString.Length == 0)

return;

Path = Utils.System.ExecTestValue(ref PictureString, "Путь");

Description = Utils.System.ExecTestValue(ref PictureString, "Описание");

var path = new Uri($@"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\TestPro\Test\Tests\".Replace('\\', '/') + Path.TrimStart(new char[] { '/' }), UriKind.Absolute).AbsolutePath;

if (File.Exists(path))

{

using (var fs = new FileStream(path, FileMode.Open))

{

Media = new byte[fs.Length];

fs.Position = 0;

fs.Read(Media, 0, Media.Length);

}

Status = true;

}

}

}

}

Листинг модуля TestKinds.YesNoTest.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using TestPro.Utils;

namespace TestPro.TestKinds

{

/// <summary>

/// Бинарный тест

/// </summary>

public class YesNoTest : BasicTest

{

/// <summary>

/// Базовый конструктор

/// </summary>

public YesNoTest()

{

Answer1 = "Да";

Answer2 = "Нет";

}

/// <summary>

/// Конструктор с параметром

/// </summary>

/// <param name="fillString">Строка для расшифровки</param>

public YesNoTest(string fillString) : this()

{

Fill(fillString);

}

private string answer1, answer2;

/// <summary>

/// Первый вриант ответа

/// </summary>

public string Answer1 { get => answer1; set => answer1 = value; }

/// <summary>

/// Второй вариант ответа

/// </summary>

public string Answer2 { get => answer2; set => answer2 = value; }

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public string Answer1String()

{

return "<Ответ1>" + Answer1 + "</Ответ1>";

}

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public string Answer2String()

{

return "<Ответ1>" + Answer2 + "</Ответ1>";

}

/// <summary>

/// Возвращает значения экземпляра класса в готовом к записи в файл виде

/// </summary>

/// <returns>Строка для записи в файл</returns>

public override string AsString()

{

string QString = QuestionString();

string AString = $"{Answer1String()}/n{Answer2String()}";

string RaString = RightAnswerString();

string AAString = AnotherAnswerString();

string HString = HintString();

string SString = SolutionString();

string S = Environment.NewLine;

return QString + S + AString + S + RaString + S + AAString + S + HString + S + SString;

}

/// <summary>

/// Заполнение значений экземпляря класса из входной строки

/// </summary>

/// <param name="fillString">Входная строка</param>

public override void Fill(string fillString)

{

if (fillString == null)

throw new ArgumentNullException();

Question = Utils.System.ExecTestValue(ref fillString, "Текст");

RightAnswer = Utils.System.ExecTestValue(ref fillString, "Ответ");

try

{

AnotherAnswer = Convert.ToBoolean(Utils.System.ExecTestValue(ref fillString, "Другой ответ"));

}

catch

{

AnotherAnswer = false;

}

Hint = Utils.System.ExecTestValue(ref fillString, "Подсказка");

Solution = Utils.System.ExecTestValue(ref fillString, "Решение");

Picture.Fill(ref fillString);

if (Picture.Status)

{

Question = Picture.Description;

}

}

}

}

ПРИЛОЖЕНИЕ 12

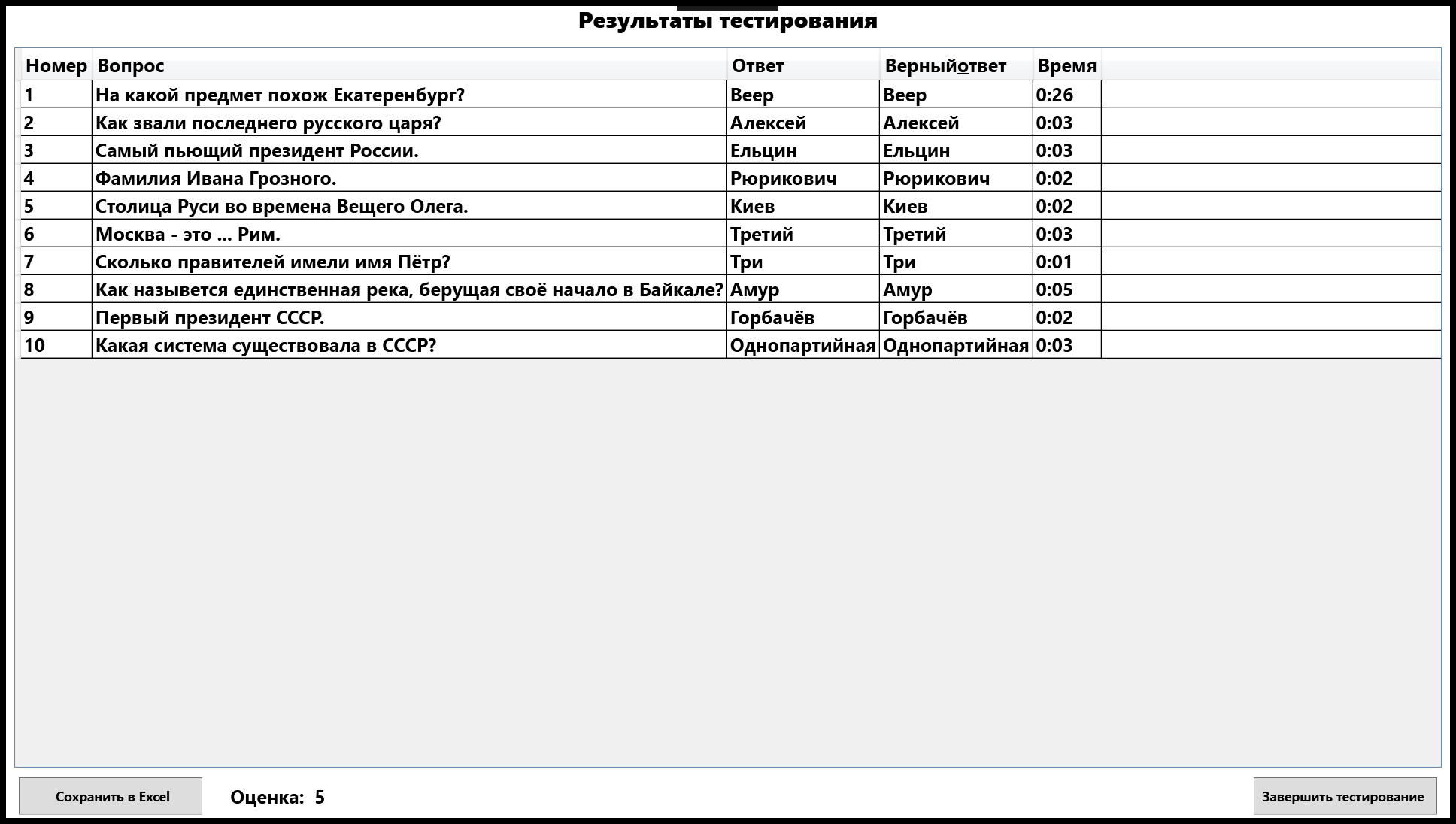


Рис. 12.1

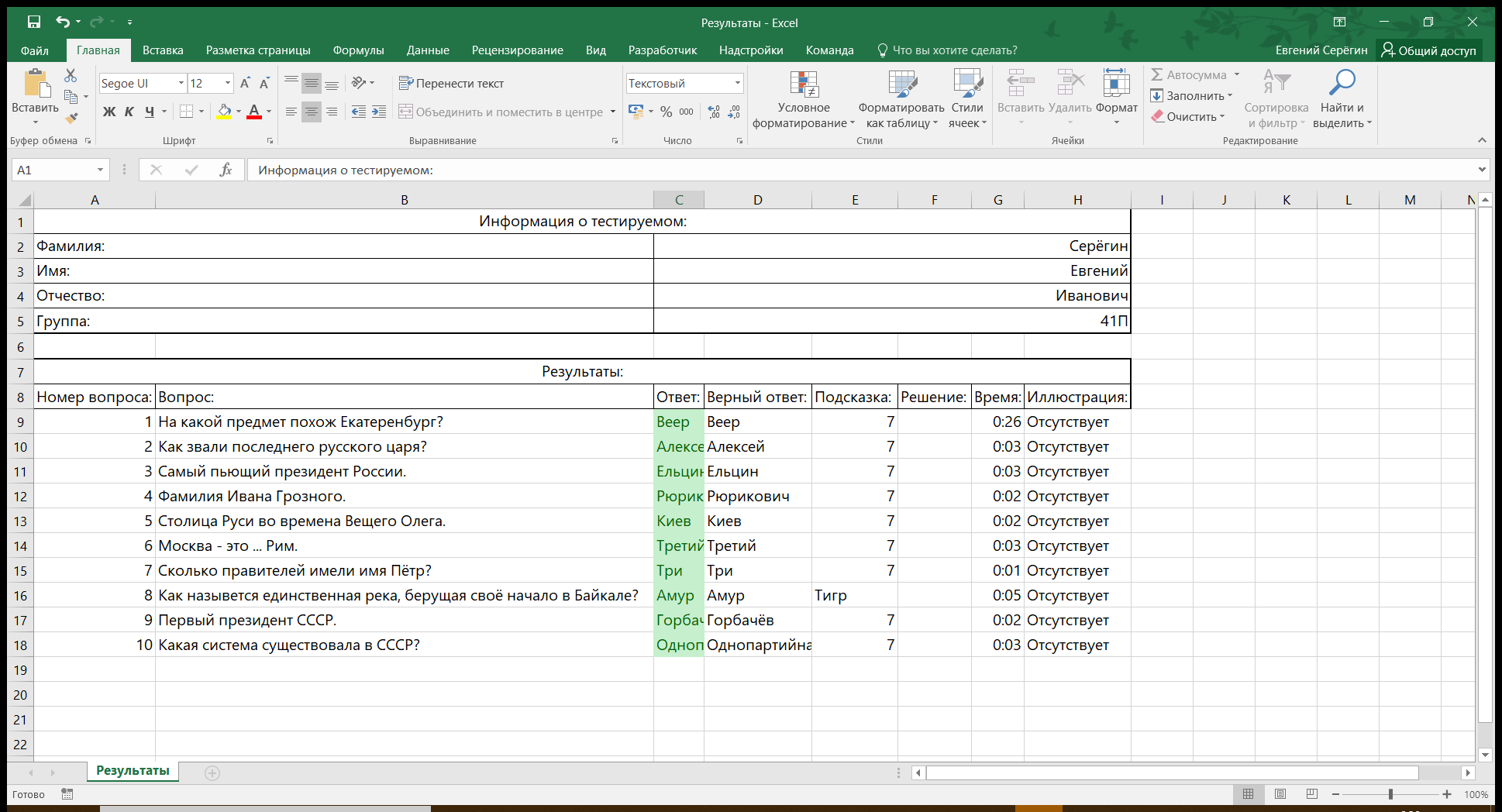


Рис. 12.2