|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |
| --- |
| **КУРСОВОЙ ПРОЕКТ** |
|  |
| **по междисциплинарному курсу МДК.01.01 Разработка программных модулей систем** |
|  |
| **на тему: Разработка информационно-обучающей системы «Блок-схемы для чайников»** |
|  |
| **студента группы 090207-9о-21/2**  **специальности 09.02.07 Информационные системы и программирование** |

Баязитова Павла Тамерлановича

|  |  |  |
| --- | --- | --- |
| **Студент** |  | **П.Т. Баязитов** |
| **Руководитель курсового проекта** |  | **Е.В. Гаиашвили** |
| **Председатель ПЦК специальности 09.02.07 Информационные системы и программирование** |  | **Е.В. Гаиашвили** |

|  |  |  |
| --- | --- | --- |
| **Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2023 г.** | | |
| **Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | |
| **Заведующий отделением №3** |  | **И.Г. Дзюба** |

Москва

2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc169437843)

[ОСНОВНАЯ ЧАСТЬ 4](#_Toc169437844)

[1 Анализ предметной области системы 4](#_Toc169437845)

[2 Работа программы 5](#_Toc169437846)

[2.1 Системные требования 5](#_Toc169437847)

[3 Разработка 6](#_Toc169437848)

[3.2 Окно входа 7](#_Toc169437849)

[3.3 Основное окно 7](#_Toc169437850)

[4 Тестирование 9](#_Toc169437851)

[ЗАКЛЮЧЕНИЕ 10](#_Toc169437852)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 11](#_Toc169437853)

[ПРИЛОЖЕНИЕ А 12](#_Toc169437854)

записка — копия.docx

# ВВЕДЕНИЕ

Информационно-обучающая система – способ обучения, обеспечивающий удалённый доступ к учебным материалам. Данный вид обучения имеет ряд преимуществ: доступность материала обучения и интерактивность. В данной курсовой работе была рассмотрена реализация информационно-обучающей системы «Блок-схемы для чайников», где будет приведён курс по блок-схемам для начинающих. Программа реализована на языке программирования C# с использованием среды разработки Sharp Develop. Будут рассмотрены основные элементы обучающей среды, такие как использование аккаунта, просмотр лекций, сдача тестов, стандартный курс. Также будет описан процесс создания информационно-обучающей системы в Sharp Develop и тестирование на наличие багов. В результате работы будет получена полноценная система, которая будет готова к использованию со стороны обучающегося.

Цель: разработка информационно-обучающей системы «Блок-схемы для чайников». Для достижения цели необходимо решить следующие задачи:

* Провести обзор аналогичных информационно-обучающих систем;
* Спроектировать работу системы;
* Реализовать программу;
* Провести тестирование на наличие багов.

Основная часть проекта состоит из 4 глав. В первой главе будут описаны предметная область и популярные решения при создании информационно-обучающей системы. Во второй главе будет разработана работа системы. В третей главе будет рассказано об описании проектирования и рассмотрены стадии разработки системы. В четвертой главе будет проведено тестирование программы.

# ОСНОВНАЯ ЧАСТЬ

## Анализ предметной области системы

Информационно-обучающая система должна иметь в себе авторизацию, выполнение заданий, изучение лекций, оценивающую систему. Как удобная программа, система должна иметь понятный пользовательский интерфейс. Если обучающая система не подразумевает наличие преподавателя, то она может иметь в себе функции, мотивирующие пользователя продолжать обучение.

«Блок-схемы для чайников» будет иметь авторизацию пользователя. Пользователь не зарегистрировавшийся в системе сможет создать аккаунт с неповторяющимся именем. Для проверки при входе в программу будет использоваться пароль.

Пользователь будет иметь доступ к стандартному курсу «Для чайников». Система записывает просмотренный материал.

Стандартный курс имеет 3 раздела. Каждый раздел имеет по 5 уроков относящихся к теме. В конце каждого раздела будет реализован экзамен.

Урок – отдельная страница с материалом, которая будет всегда доступна пользователям.

Экзамен – отдельная страница, предназначенная для проверки изученного материала в конце раздела. Экзамены будут иметь вопросы с ответами, где на каждый вопрос есть только один правильный ответ.

# Работа программы

В рамках этой задачи необходимо спланировать работу программы. Важным условием является быстрая и стабильная работа приложения, поддержка Windows 10 и 11 версий.

Для реализации данного проекта использовался язык C# и последняя версия фреймворка «.NET». Пользовательский интерфейс был реализован с использованием встроенного конструктора форм в Sharp Develop. Фреймворк позволяет разрабатывать оконные приложения быстро и учитывая все запросы конечного потребителя. «.NET» полностью совместим с Windows 10 и 11, что покрывает большинство операционных систем на домашнем компьютере.

Стандартный курс включает в себя такие разделы как «чайник», «основы» и «хакер». У каждого раздела есть экзамен и теория, для подготовки.

## 2.1 Системные требования

## 2.1.1 Минимальные требования

* ОС Windows 10;
* Процессор: AMD A10-4655M (2000 МГц, 4 ядра, 4 потока);
* Оперативная память: 4 GB ОЗУ;
* Свободное место на диске: 50 МБ.

## 2.1.2 Рекомендуемые требования

* ОС Windows 10 или Windows 11;
* Процессор: AMD A10-4655M (2000 МГц, 4 ядра, 4 потока);
* Оперативная память: 8 GB ОЗУ;

# Разработка

В данной задаче необходимо завершить разработку программы.

**3.1 База данных**

База данных должна хранить пользователей и курсы с уроками и экзаменами. Система должна безопасно хранить и изменять данные, поэтому в базе данных будут настроены зависимости. Финальная версия изображена на первом рисунке.

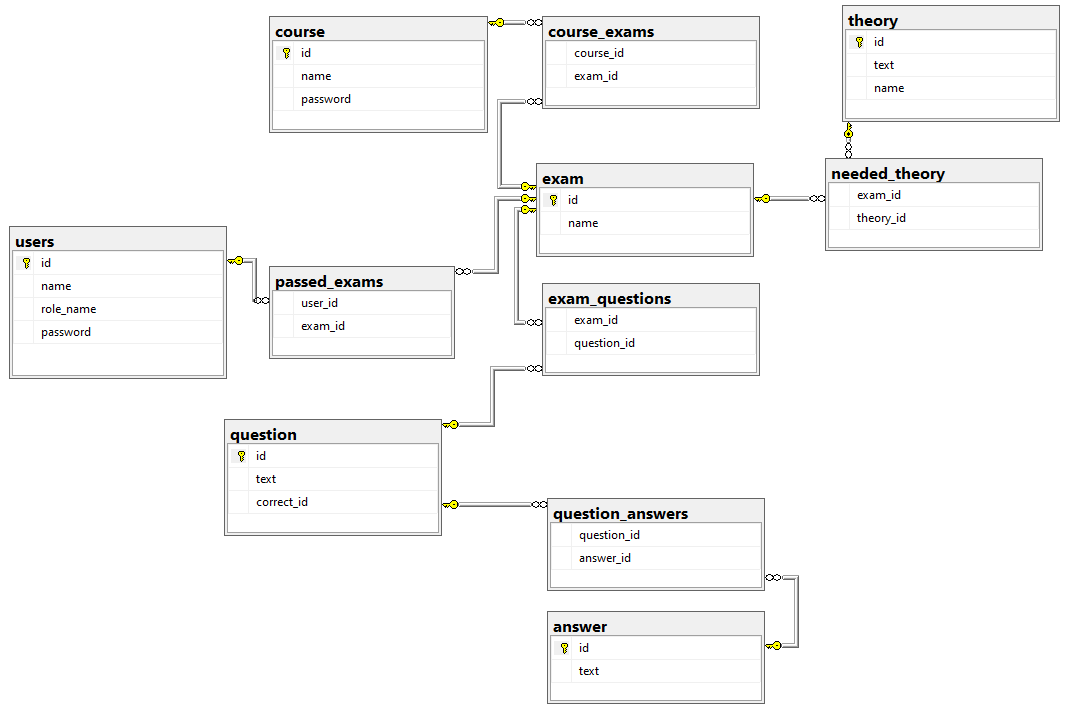


Рисунок 1 – База данных программы

# Окно входа

При запуске программы пользователь должен попадать в окно входа (Рисунок 2), где он может зарегистрироваться, либо войти в свой аккаунт. Для проверки используется база данных.

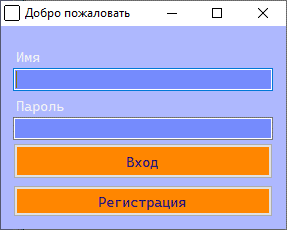


Рисунок 2 – Окно входа

При нажатии на кнопку «Регистрация» входные данные попадают в базу данных. Если имя уже занято, запрос прерывается и пользователю показывается ошибка (Рисунок 3).

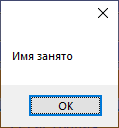


Рисунок 3 – Ошибка регистрации при повторном имени

Если пользователь не ввел одно из полей, то запрос прервется, указав конкретную ошибку. При пустом поле с поролем выводится ошибка из 4 рисунка, а при пустом поле с именем из 5 рисунка.

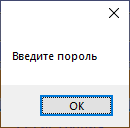


Рисунок 4 – Ошибка регистрации при отсутствии пороля

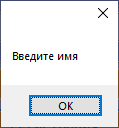


Рисунок 5 – Ошибка регистрации при отсутствии имени

Система регистрации имеет проверку на надежность пороля. При вводе, он сравнивается с другими поролями из списка частоиспользуеммых, как показано на рисунке 6. Также пороль должен содержать определенные символы, и не состоять только из пробелов (рисунки 7 и 8 соответственно). Система значительно уменьшает риски взлома аккаунта.

РИСУНОК 6 ЧАСТОИСПОЛЬЗУЕМЫЙ ПОРОЛЬ

РИСУНОК 7 ПОРОЛЬ ДОЛЖЕН БЫТЬ НАДЕЖНЫМ

РИСНОК 8 ПОРОЛЬ ДОЛЖЕН НЕ СОСТОЯТЬ ТОЛЬКО ИЗ ПРОБЕЛОВ

При нажатии на кнопку «Вход» входные данные попадают сравниваются в базе данных. Если имеется пользователь с введенных именем и поролем то программа открывает основное окно.

Если пользователь ввел неправильные имя или пороль, то выведется окно с ошибкой (Рисунок 6).

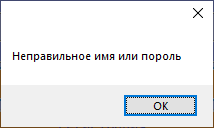


Рисунок 6 – Ошибка входа

При входе также учитывается что пользователь может не ввести имя или пороль. Тогда появится окна из рисунков 4 и 5.

# Основное окно

При входе в аккаунт пользователь имеет доступ к информационно-обучающей системе (Рисунок 7). Он должен видеть других пользователей и текущий курс.

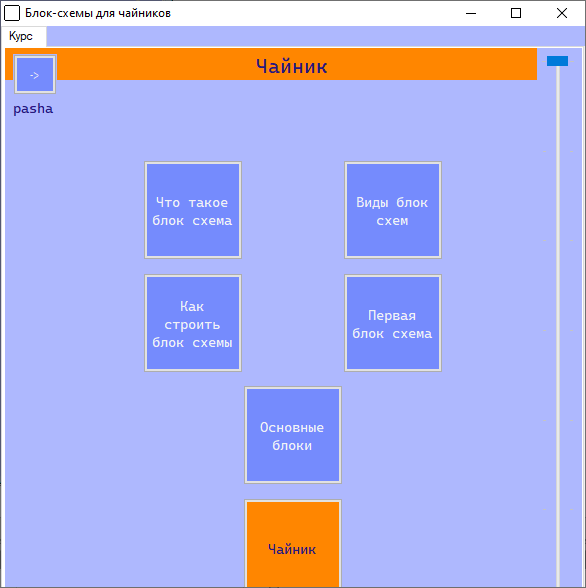


Рисунок 7 – Основное окно

# 3.4 Окно обучения

При выборе урока появляется окно с названием и содержанием (Рисунок 8). Окно работает отдельно от основной программы, может быть вызвано несколько разных окон.

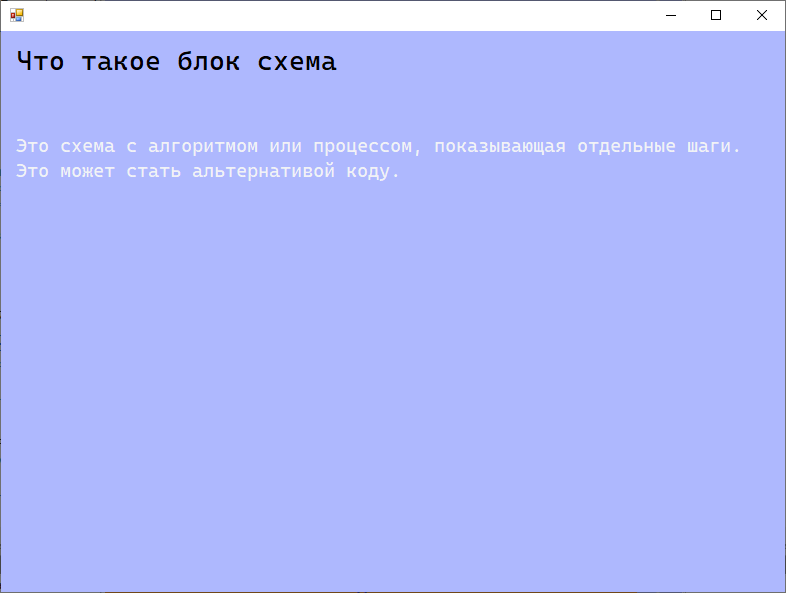


Рисунок 8 – Окно обучения

**3.5 Запоминание просмотренных материалов**

В системе имеется функция запоминания просмотренных лекций и пройденных тестов. Для запоминания используется файл «checked.txt». Приложение проверяет лекцию или тест, и если он пройден, то закрашивает в другой цвет. Таким образом пользователь может видеть пройденный материал. Пример работы системы запоминания просмотренных материалов изображен на рисунке 9. В примере пользователь просмотрел первые 3 урока.



Рисунок 9 – Запоминание просмотренных материалов

Файл «checked.txt» находится на компьютере пользователя, но меняется только программой. Пример этого файла изображен на рисунке 10.

РИСУНОК 10 ВСТАВИТЬ

# Тестирование

В данной задаче необходимо убедится в работоспособности и надёжности программы, она должна правильно функционировать и работать в соответствии с минимальными требованиями. Далее предоставлены виды тестирования, которые были проведены в процессе разработки и после её завершения.

Функциональное тестирование:

* Проверка корректного запуска программы;
* Проверка наличия всех необходимых элементов основной и формы входа;
* Проверка корректности работы управления, ответ программы на нажатие кнопок;
* Проверка корректности при вводе неверных данных (при логине и регистрации);
* Проверка корректности перехода между окнами;
* Проверка корректности отображения просмотренного материала;
* Проверка корректности выхода и программы.

Тестирование совместимости:

* Проверка работоспособности на операционных системах Windows 10 и 11.

# ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы была создана информационно-обучающая система «Блок-схемы для чайников» с использованием языка программирования C# и программы Sharp Develop. Разработка проекта позволила закрепить и применить полученные знания и навыки в области программирования и разработки программ с интерфейсом для пользователя.

В процессе работы изучены основы создания интерактивной информационно-обучающей среды и параллельное использование баз данных.

В итоге, выполнение данной курсовой работы расширило знания и навыки в области программирования и использования языка C#.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Microsoft Corporation. Принципы проектирования и разработки программного обеспечения. Учеб. курс MCSD. М.: Изд.-торг. дом «Русская редакция», 2000.
2. Windows Forms. Программирование на C# [Электронный ресурс]. — Режим доступа: http://csharpcoding.org/category/windows-forms/ (дата обращения: 15.03.2017).
3. Агуров П. В. C#. Сборник рецептов / Агуров П. В. – СПб.:БХВ-Петербург, 2007. – 412с.
4. Албахари Дж. C# 6.0. Справочник. Полное описание языка [Текст] / Дж . Албахари, Б. Албахари — 6-е изд. — Москва: Вильямc, 2016. — 1040 c.
5. Артамонов И.В. Разработка распределенных сервисно-ориентированных программных средств / И.В. Артамонов – Иркутск: Изд-во БГУЭП, 2012. – 130 с.
6. Биллиг В. А. Объектное программирование в классах на C# 3.0 [Электронный ресурс] / В.А. Биллиг. - 2-е изд., испр.. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 391 с. Полный текст находится в ЭБС "Университетская библиотека ONLINE".
7. Биллиг В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) [Текст]: учебное пособие / В. А. Биллиг. — Москва: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2016. — 584 c.
8. Биллиг В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В.A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c.

# ПРИЛОЖЕНИЕ А

С# код класса Program

using System;

using System.Windows.Forms;

using Block.form;

using Block.user.level;

namespace Block

{

internal sealed class Program

{

[STAThread]

private static void Main(string[] args)

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new LoginForm());

}

}

}

C# код класса LoginForm

using System;

using System.Drawing;

using System.Windows.Forms;

using Block.manage;

using System.Data;

using System.Data.SqlClient;

namespace Block.form

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

TopManager.instance.Update();

}

private void LoginFormFormClosed(object sender, FormClosedEventArgs e)

{

Hide();

}

private LoginInfo GetInput(Button pressed)

{

string name = nameTextBox.Text;

string password = passwordTextBox.Text;

if (pressed == loginButton)

return new LoginInfo(name, password, Input.Login);

if (pressed == registrationButton)

return new LoginInfo(name, password, Input.Registration);

return null;

}

private void ComeBack(LoginInfo info)

{

if (info.name.Equals("")) throw new PassException("Введите имя");

if (info.password.Equals("")) throw new PassException("Введите пороль");

if (info.input == Input.Login)

{

foreach (var curUser in TopManager.instance.Users)

if (curUser.Name.Equals(info.name) && curUser.Password.Equals(info.password))

{

Hide();

new BlockForm(TopManager.instance.GetUserByName(info.name)).Show();

return;

}

throw new PassException("Неправильное имя или пороль");

}

else

{

foreach (var curUser in TopManager.instance.Users)

if (curUser.Name.Equals(info.name))

throw new PassException("Имя занято");

TopManager.instance.RegisterUser(info.name, info.password);

Hide();

new BlockForm(TopManager.instance.GetUserByName(info.name)).Show();

return;

}

}

private void LoginButtonClick(object sender, EventArgs e)

{

try

{

ComeBack(GetInput((Button) sender));

}

catch (PassException ex)

{

MessageBox.Show(ex.Message);

}

}

private void RegistrationButtonClick(object sender, EventArgs e)

{

try

{

ComeBack(GetInput((Button) sender));

}

catch (PassException ex)

{

MessageBox.Show(ex.Message);

}

}

private void PassButtonClick(object sender, EventArgs e)

{

try

{

ComeBack(GetInput((Button) sender));

}

catch (PassException ex)

{

MessageBox.Show(ex.Message);

}

}

void LoginFormFormClosing(object sender, FormClosingEventArgs e)

{

if (e.CloseReason.Equals(CloseReason.UserClosing))

Application.Exit();

}

}

internal enum Input

{

Login,

Registration,

}

internal class LoginInfo

{

public string name;

public string password;

public Input input;

public LoginInfo(string name, string password, Input input)

{

this.name = name;

this.password = password;

this.input = input;

}

}

internal class PassException : Exception

{

public PassException(string msg) : base(msg) { }

}

}

C# код класса LoginForm.Designer

using System.Windows.Forms;

using Block.manage;

namespace Block.form

{

partial class BlockForm

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing) {

if (components != null) {

components.Dispose();

}

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(BlockForm));

this.pageControl = new System.Windows.Forms.TabControl();

this.coursePage = new System.Windows.Forms.TabPage();

this.trackBar = new System.Windows.Forms.TrackBar();

this.roundButton1 = new Block.form.RoundButton();

this.darkRadioButton = new System.Windows.Forms.RadioButton();

this.lightRadioButton = new System.Windows.Forms.RadioButton();

this.colorsLabel = new System.Windows.Forms.Label();

this.pageControl.SuspendLayout();

this.coursePage.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.trackBar)).BeginInit();

this.SuspendLayout();

//

// pageControl

//

this.pageControl.Controls.Add(this.coursePage);

this.pageControl.Location = new System.Drawing.Point(0, 0);

this.pageControl.Name = "pageControl";

this.pageControl.SelectedIndex = 0;

this.pageControl.Size = new System.Drawing.Size(585, 863);

this.pageControl.TabIndex = 1;

//

// coursePage

//

this.coursePage.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(174)))), ((int)(((byte)(184)))), ((int)(((byte)(254)))));

this.coursePage.Controls.Add(this.trackBar);

this.coursePage.Controls.Add(this.roundButton1);

this.coursePage.Location = new System.Drawing.Point(4, 22);

this.coursePage.Name = "coursePage";

this.coursePage.Padding = new System.Windows.Forms.Padding(3);

this.coursePage.Size = new System.Drawing.Size(577, 837);

this.coursePage.TabIndex = 1;

this.coursePage.Text = "Курс";

//

// trackBar

//

this.trackBar.LargeChange = 1;

this.trackBar.Location = new System.Drawing.Point(532, 0);

this.trackBar.Maximum = 9;

this.trackBar.Name = "trackBar";

this.trackBar.Orientation = System.Windows.Forms.Orientation.Vertical;

this.trackBar.Size = new System.Drawing.Size(45, 834);

this.trackBar.TabIndex = 2;

this.trackBar.TickStyle = System.Windows.Forms.TickStyle.Both;

this.trackBar.Value = 9;

this.trackBar.Scroll += new System.EventHandler(this.TrackBarScroll);

//

// roundButton1

//

this.roundButton1.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(117)))), ((int)(((byte)(139)))), ((int)(((byte)(253)))));

this.roundButton1.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(241)))), ((int)(((byte)(242)))), ((int)(((byte)(246)))));

this.roundButton1.Location = new System.Drawing.Point(8, 6);

this.roundButton1.Name = "roundButton1";

this.roundButton1.Size = new System.Drawing.Size(44, 41);

this.roundButton1.TabIndex = 0;

this.roundButton1.Text = "->";

this.roundButton1.UseVisualStyleBackColor = false;

this.roundButton1.Click += new System.EventHandler(this.SwitchCourseButtonClick);

//

// darkRadioButton

//

this.darkRadioButton.Location = new System.Drawing.Point(9, 60);

this.darkRadioButton.Name = "darkRadioButton";

this.darkRadioButton.Size = new System.Drawing.Size(104, 24);

this.darkRadioButton.TabIndex = 2;

this.darkRadioButton.Text = "Тёмное";

this.darkRadioButton.UseVisualStyleBackColor = true;

//

// lightRadioButton

//

this.lightRadioButton.Checked = true;

this.lightRadioButton.Location = new System.Drawing.Point(9, 30);

this.lightRadioButton.Name = "lightRadioButton";

this.lightRadioButton.Size = new System.Drawing.Size(104, 24);

this.lightRadioButton.TabIndex = 1;

this.lightRadioButton.TabStop = true;

this.lightRadioButton.Text = "Светлое";

this.lightRadioButton.UseVisualStyleBackColor = true;

//

// colorsLabel

//

this.colorsLabel.Location = new System.Drawing.Point(9, 4);

this.colorsLabel.Name = "colorsLabel";

this.colorsLabel.Size = new System.Drawing.Size(100, 23);

this.colorsLabel.TabIndex = 0;

this.colorsLabel.Text = "Оформление";

this.colorsLabel.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

//

// BlockForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(584, 861);

this.Controls.Add(this.pageControl);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));

this.Name = "BlockForm";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Блок-схемы для чайников";

this.FormClosed += new System.Windows.Forms.FormClosedEventHandler(this.BlockFormFormClosed);

this.pageControl.ResumeLayout(false);

this.coursePage.ResumeLayout(false);

this.coursePage.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.trackBar)).EndInit();

this.ResumeLayout(false);

}

private Block.form.RoundButton roundButton1;

private System.Windows.Forms.TabPage coursePage;

private System.Windows.Forms.TabControl pageControl;

private System.Windows.Forms.TrackBar trackBar;

private System.Windows.Forms.RadioButton darkRadioButton;

private System.Windows.Forms.RadioButton lightRadioButton;

private System.Windows.Forms.Label colorsLabel;

}

}

C# код класса BlockForm

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Collections.Generic;

using Block.user;

using Block.user.level;

using Block.manage;

using Block.Algorithm;

namespace Block.form

{

public partial class BlockForm : Form

{

private CourseVisualizer visualizer;

private int prevY = 9;

public BlockForm(User user)

{

InitializeComponent();

this.BackColor = Easel.BG\_COLOR;

TopManager.instance.activeUser = user;

visualizer = new CourseVisualizer(coursePage);

visualizer.PrintMembersList();

}

private void BlockFormFormClosed(object sender, FormClosedEventArgs e)

{

TopManager.instance.disconnectDatabase();

Application.ExitThread();

}

void SwitchCourseButtonClick(object sender, EventArgs e)

{

visualizer.Clear(new List<object> { sender, trackBar } );

var curCourse = TopManager.instance.NextCourse;

visualizer.Print(curCourse);

if (curCourse != null)

visualizer.PrintMembersList();

trackBar.Value = trackBar.Maximum;

prevY = trackBar.Maximum;

}

void TrackBarScroll(object sender, EventArgs e)

{

int offset = visualizer.GetMinMaxY().s / 10;

visualizer.MoveAllY((trackBar.Value - prevY) \* offset);

prevY = trackBar.Value;

}

}

internal sealed class CourseVisualizer

{

private static readonly int ELEMENT\_D = 70;

private static readonly int ELEMENT\_R = ELEMENT\_D / 2;

private readonly TabPage field;

private readonly int MIDDLE\_X;

private readonly int MIDDLE\_Y;

private readonly int OFFSET\_X;

private readonly int START\_Y;

private IntPairCache maxMinYCache;

public CourseVisualizer(TabPage page)

{

field = page;

MIDDLE\_X = page.Width / 2;

MIDDLE\_Y = page.Height / 2;

OFFSET\_X = MIDDLE\_X / 4 + ELEMENT\_D;

START\_Y = page.Height - (int) (ELEMENT\_D \* 1.5);

}

public IntPairCache GetMinMaxY()

{

if (maxMinYCache != null) return maxMinYCache;

maxMinYCache = new IntPairCache(field.Controls[0].Location.Y, field.Controls[0].Location.Y);

foreach(Control curControl in field.Controls)

{

if (curControl.Location.Y < maxMinYCache.f) maxMinYCache.f = curControl.Location.Y;

if (curControl.Location.Y > maxMinYCache.s) maxMinYCache.s = curControl.Location.Y;

}

return maxMinYCache;

}

public void MoveAllY(int yOffset)

{

foreach(Control curControl in field.Controls)

if (!(curControl.Name.Equals("trackBar")))

curControl.Top += yOffset;

}

public void Clear(List<object> except)

{

foreach (Control curEl in field.Controls)

{

if (!(except.Contains(curEl)))

field.Controls.Remove(curEl);

}

}

public void Print(Course course)

{

if (course == null) return;

List<RoundButton> visualized = new List<RoundButton>();

int startUnwrapPosition = 0;

foreach (Exam curExam in course.Exams)

{

List<Control> unwrapedTheory = UnwrapExam(curExam, startUnwrapPosition);

startUnwrapPosition += 2;

RoundButton examButton = GetExamRoundButton(

GetButtonMiddlePlacement(curExam.TheoryNeeded.Count + 2 + startUnwrapPosition, MIDDLE\_X),

curExam

);

foreach(Control curControl in unwrapedTheory)

field.Controls.Add(curControl);

field.Controls.Add(examButton);

startUnwrapPosition += curExam.TheoryNeeded.Count + 5;

startUnwrapPosition -= startUnwrapPosition % 2;

}

maxMinYCache = null;

}

private List<Control> UnwrapExam(Exam exam, int startUnwrapPosition)

{

List<Control> unwraped = new List<Control>();

Label label = new Label();

label.Text = exam.Name;

label.Font = new System.Drawing.Font("Cascadia Mono", 15F);

label.ForeColor = Easel.ATTENTION\_COLOR;

label.BackColor = Easel.ALT\_COLOR;

label.Location = GetButtonMiddlePlacement(startUnwrapPosition, MIDDLE\_X);

label.Location = new Point(0, label.Location.Y);

label.Size = new Size(field.Size.Width, 32);

label.TextAlign = ContentAlignment.MiddleCenter;

startUnwrapPosition += 2;

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

unwraped.Add(GetBasicRoundButton(

exam.TheoryNeeded[i].Name,

GetButtonPlacement(i + startUnwrapPosition, MIDDLE\_X, exam.TheoryNeeded.Count + startUnwrapPosition),

exam.TheoryNeeded[i].Text

));

unwraped.Add(label);

return unwraped;

}

private static Point GetButtonPlacement(int id, int middleX, int maxId)

{

if (id + 1 == maxId && id % 2 == 0)

return GetButtonMiddlePlacement(id, middleX);

return new Point(((id % 2) \* 2 - 1) \* ELEMENT\_D + middleX - ELEMENT\_R, id / 2 \* ELEMENT\_D + id / 2 \* ELEMENT\_R / 4);

}

private static Point GetButtonMiddlePlacement(int id, int middleX)

{

return new Point(middleX - ELEMENT\_R, id / 2 \* ELEMENT\_D + id / 2 \* ELEMENT\_R / 4);

}

private static RoundButtonContext GetBasicRoundButton(string name, Point location, string context)

{

var button = new RoundButtonContext(new StringContext(name, context));

button.Location = location;

button.Name = name;

button.Size = new System.Drawing.Size(ELEMENT\_D, ELEMENT\_D);

button.TabIndex = 0;

button.Text = name;

button.Font = new System.Drawing.Font("Cascadia Mono", 10F);

button.ForeColor = Easel.OUT\_COLOR;

button.BackColor = Easel.MAIN\_COLOR;

button.Visible = true;

if (button.IsChecked())

button.BackColor = Easel.DARK\_MAIN\_COLOR;

return button;

}

private static RoundButtonContext GetExamRoundButton(Point location, Exam exam)

{

var button = new RoundButtonContext(new FormContext(exam.Name, exam));

button.Location = location;

button.Name = exam.Name;

button.Size = new System.Drawing.Size(ELEMENT\_D, ELEMENT\_D);

button.TabIndex = 0;

button.Font = new System.Drawing.Font("Cascadia Mono", 10F);

button.ForeColor = Easel.ATTENTION\_COLOR;

button.BackColor = Easel.ALT\_COLOR;

button.Text = exam.Name;

button.Visible = true;

if (button.IsChecked())

button.BackColor = Easel.DARK\_ALT\_COLOR;

return button;

}

public void PrintMembersList()

{

for (int i = 0; i < TopManager.instance.Users.Count; i++)

{

var curUser = TopManager.instance.Users[i];

var curLabel = new Label();

curLabel.Text = curUser.Name;

curLabel.Location = new Point(5, 20 \* i + 50);

curLabel.ForeColor = ((i % 2 == 0) ? Easel.ATTENTION\_COLOR : Easel.DARK\_MAIN\_COLOR);

curLabel.Font = new System.Drawing.Font("Cascadia Mono", 10F);

curLabel.Visible = true;

field.Controls.Add(curLabel);

}

}

private Label GetLabelAtY(int y)

{

foreach (Control curControl in field.Controls)

if (curControl.GetType().Name.Equals("Label"))

if (curControl.Location.Y == y) return curControl as Label;

return null;

}

}

}

C# код класса BlockForm.Designer

using System.Windows.Forms;

using Block.manage;

namespace Block.form

{

partial class BlockForm

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing) {

if (components != null) {

components.Dispose();

}

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(BlockForm));

this.pageControl = new System.Windows.Forms.TabControl();

this.coursePage = new System.Windows.Forms.TabPage();

this.trackBar = new System.Windows.Forms.TrackBar();

this.roundButton1 = new Block.form.RoundButton();

this.darkRadioButton = new System.Windows.Forms.RadioButton();

this.lightRadioButton = new System.Windows.Forms.RadioButton();

this.colorsLabel = new System.Windows.Forms.Label();

this.pageControl.SuspendLayout();

this.coursePage.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.trackBar)).BeginInit();

this.SuspendLayout();

//

// pageControl

//

this.pageControl.Controls.Add(this.coursePage);

this.pageControl.Location = new System.Drawing.Point(0, 0);

this.pageControl.Name = "pageControl";

this.pageControl.SelectedIndex = 0;

this.pageControl.Size = new System.Drawing.Size(585, 863);

this.pageControl.TabIndex = 1;

//

// coursePage

//

this.coursePage.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(174)))), ((int)(((byte)(184)))), ((int)(((byte)(254)))));

this.coursePage.Controls.Add(this.trackBar);

this.coursePage.Controls.Add(this.roundButton1);

this.coursePage.Location = new System.Drawing.Point(4, 22);

this.coursePage.Name = "coursePage";

this.coursePage.Padding = new System.Windows.Forms.Padding(3);

this.coursePage.Size = new System.Drawing.Size(577, 837);

this.coursePage.TabIndex = 1;

this.coursePage.Text = "Курс";

//

// trackBar

//

this.trackBar.LargeChange = 1;

this.trackBar.Location = new System.Drawing.Point(532, 0);

this.trackBar.Maximum = 9;

this.trackBar.Name = "trackBar";

this.trackBar.Orientation = System.Windows.Forms.Orientation.Vertical;

this.trackBar.Size = new System.Drawing.Size(45, 834);

this.trackBar.TabIndex = 2;

this.trackBar.TickStyle = System.Windows.Forms.TickStyle.Both;

this.trackBar.Value = 9;

this.trackBar.Scroll += new System.EventHandler(this.TrackBarScroll);

//

// roundButton1

//

this.roundButton1.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(117)))), ((int)(((byte)(139)))), ((int)(((byte)(253)))));

this.roundButton1.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(241)))), ((int)(((byte)(242)))), ((int)(((byte)(246)))));

this.roundButton1.Location = new System.Drawing.Point(8, 6);

this.roundButton1.Name = "roundButton1";

this.roundButton1.Size = new System.Drawing.Size(44, 41);

this.roundButton1.TabIndex = 0;

this.roundButton1.Text = "->";

this.roundButton1.UseVisualStyleBackColor = false;

this.roundButton1.Click += new System.EventHandler(this.SwitchCourseButtonClick);

//

// darkRadioButton

//

this.darkRadioButton.Location = new System.Drawing.Point(9, 60);

this.darkRadioButton.Name = "darkRadioButton";

this.darkRadioButton.Size = new System.Drawing.Size(104, 24);

this.darkRadioButton.TabIndex = 2;

this.darkRadioButton.Text = "Тёмное";

this.darkRadioButton.UseVisualStyleBackColor = true;

//

// lightRadioButton

//

this.lightRadioButton.Checked = true;

this.lightRadioButton.Location = new System.Drawing.Point(9, 30);

this.lightRadioButton.Name = "lightRadioButton";

this.lightRadioButton.Size = new System.Drawing.Size(104, 24);

this.lightRadioButton.TabIndex = 1;

this.lightRadioButton.TabStop = true;

this.lightRadioButton.Text = "Светлое";

this.lightRadioButton.UseVisualStyleBackColor = true;

//

// colorsLabel

//

this.colorsLabel.Location = new System.Drawing.Point(9, 4);

this.colorsLabel.Name = "colorsLabel";

this.colorsLabel.Size = new System.Drawing.Size(100, 23);

this.colorsLabel.TabIndex = 0;

this.colorsLabel.Text = "Оформление";

this.colorsLabel.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

//

// BlockForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(584, 861);

this.Controls.Add(this.pageControl);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));

this.Name = "BlockForm";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Блок-схемы для чайников";

this.FormClosed += new System.Windows.Forms.FormClosedEventHandler(this.BlockFormFormClosed);

this.pageControl.ResumeLayout(false);

this.coursePage.ResumeLayout(false);

this.coursePage.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.trackBar)).EndInit();

this.ResumeLayout(false);

}

private Block.form.RoundButton roundButton1;

private System.Windows.Forms.TabPage coursePage;

private System.Windows.Forms.TabControl pageControl;

private System.Windows.Forms.TrackBar trackBar;

private System.Windows.Forms.RadioButton darkRadioButton;

private System.Windows.Forms.RadioButton lightRadioButton;

private System.Windows.Forms.Label colorsLabel;

}

}

C# код класса TopManager

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Text;

using System.Data;

using System.Data.SqlClient;

using Block.user;

using Block.user.level;

using Block.manage.SQL;

namespace Block.manage

{

public sealed class TopManager

{

public static TopManager instance = new TopManager();

//DESKTOP-DAQPCTH\SQLEXPRESS ноут

//DESKTOP-8UOLSK2\SQLEXPRESS комп

private string dbConnectionQuery = @"Data Source=DESKTOP-8UOLSK2\SQLEXPRESS;Initial Catalog=kurs3;Integrated Security=True";

private readonly string checkedCachePath = @"./checkedChache.txt";

private SqlTableManager SQLManager;

public User activeUser;

private List<User> users;

private List<Course> courses;

private int curCourseId;

private TopManager()

{

users = new List<User>();

courses = new List<Course>();

curCourseId = 0;

var connection = new SqlConnection(dbConnectionQuery);

connection.Open();

SQLManager = new SqlTableManager(connection);

}

public void disconnectDatabase()

{

SQLManager.CloseConnection();

}

public Course CurrentCourse

{

get

{

return (TopManager.instance.Courses.Count == 0) ? null : TopManager.instance.Courses[curCourseId % TopManager.instance.Courses.Count];

}

}

public Course NextCourse

{

get

{

curCourseId++;

return (TopManager.instance.Courses.Count == 0) ? null : TopManager.instance.Courses[curCourseId % TopManager.instance.Courses.Count];

}

}

public void RegisterUser(string name, string password)

{

SQLManager.AddRow(

SqlTableManager.USER\_TABLE,

new string[] { "id", "name", "role\_name", "password" },

new string[] { SQLManager.GetFreeId(SqlTableManager.USER\_TABLE).ToString(), name, "Student", password }

);

}

public void Update()

{

users = SQLManager.GetUsers();

courses = SQLManager.GetCourses(users);

}

public string DbConnectionQuery

{

get {return dbConnectionQuery;}

private set {dbConnectionQuery = value;}

}

public User GetUserByName(string name)

{

foreach (User curUser in users)

if (curUser.Name.Equals(name)) return curUser;

return null;

}

public List<User> Users

{

get {return users;}

private set {users = value;}

}

public List<Course> Courses

{

get {return courses;}

private set {courses = value;}

}

}

public enum AvaibleRoles

{

Student,

Tutor,

Admin,

}

}

C# код класса Easel

using System;

using System.Drawing;

using System.Linq;

namespace Block.manage

{

public class Easel

{

public static readonly Color ATTENTION\_COLOR = Color.FromArgb(255, 39, 24, 126);

public static readonly Color BG\_COLOR = Color.FromArgb(255, 174, 184, 254);

public static readonly Color MAIN\_COLOR = Color.FromArgb(255, 117, 139, 253);

public static readonly Color OUT\_COLOR = Color.FromArgb(255, 241, 242, 246);

public static readonly Color ALT\_COLOR = Color.FromArgb(255, 255, 134, 0);

public static readonly Color DARK\_MAIN\_COLOR = Color.FromArgb(255, 72, 79, 208);

public static readonly Color DARK\_ALT\_COLOR = Color.FromArgb(255, 205, 84, 0);

public static Color GetDarker(Color color)

{

if (color.Equals(MAIN\_COLOR))

return DARK\_MAIN\_COLOR;

if (color.Equals(ALT\_COLOR))

return DARK\_ALT\_COLOR;

return color;

}

}

}

C# код класса SqlTableManager

using System;

using System.Data;

using System.Data.SqlClient;

using System.Text;

using System.Collections.Generic;

using System.Collections;

using System.Windows.Forms;

using Block.user;

using Block.manage;

using Block.user.level;

using Block.Algorithm;

namespace Block.manage.SQL

{

public class SqlTableManager

{

private SqlConnection connection;

public static readonly string USER\_TABLE = "users";

public static readonly string USER\_PASSED\_EXAMS = "passed\_exams";

public static readonly string COURSE\_TABLE = "course";

public static readonly string COURSE\_EXAMS\_TABLE = "course\_exams";

public static readonly string EXAM\_TABLE = "exam";

public static readonly string NEEDED\_THEORY\_TABLE = "needed\_theory";

public static readonly string THEORY\_TABLE = "theory";

public static readonly string EXAM\_QUESTIONS\_TABLE = "exam\_questions";

public static readonly string QUESTION\_TABLE = "question";

public static readonly string QUESTION\_ANSWERS\_TABLE = "question\_answers";

public static readonly string ANSWER\_TABLE = "answer";

public List<User> GetUsers()

{

DataTable userTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + USER\_TABLE, connection).Fill(userTable);

List<User> users = new List<User>();

foreach (DataRow curRow in userTable.Rows)

users.Add(GetUser(curRow));

return users;

}

private User GetUser(DataRow row)

{

if (row["role\_name"].Equals("Student"))

return new Student((int) row["id"], row["name"].ToString(), row["password"].ToString());

if (row["role\_name"].Equals("Tutor"))

return new Tutor((int) row["id"], row["name"].ToString(), row["password"].ToString());

if (row["role\_name"].Equals("Admin"))

return new Admin((int) row["id"], row["name"].ToString(), row["password"].ToString());

return null;

}

public List<Course> GetCourses(List<User> users)

{

DataTable courseTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + COURSE\_TABLE, connection).Fill(courseTable);

List<Course> courses = new List<Course>();

DataTable courseExamsTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + COURSE\_EXAMS\_TABLE, connection).Fill(courseExamsTable);

DataTable examTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + EXAM\_TABLE, connection).Fill(examTable);

List<Exam> exams = new List<Exam>();

DataTable theoryTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + THEORY\_TABLE, connection).Fill(theoryTable);

List<Theory> theory= new List<Theory>();

DataTable neededTheoryTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + NEEDED\_THEORY\_TABLE, connection).Fill(neededTheoryTable);

DataTable questionTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + QUESTION\_TABLE, connection).Fill(questionTable);

List<Question> questions = new List<Question>();

DataTable examQuestionsTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + EXAM\_QUESTIONS\_TABLE, connection).Fill(examQuestionsTable);

DataTable answerTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + ANSWER\_TABLE, connection).Fill(answerTable);

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

DataTable questionAnswersTable = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + QUESTION\_ANSWERS\_TABLE, connection).Fill(questionAnswersTable);

DataTable userPassedExams = new DataTable();

new SqlDataAdapter("SELECT \* FROM " + USER\_PASSED\_EXAMS, connection).Fill(userPassedExams);

foreach (DataRow curCourse in courseTable.Rows)

courses.Add(new Course((int) curCourse["id"], curCourse["name"].ToString(), curCourse["password"].ToString(), new List<Exam>()));

foreach (DataRow curExam in examTable.Rows)

exams.Add(new Exam((int) curExam["id"], curExam["name"].ToString(), new List<Theory>(), new List<Question>(), new List<User>()));

foreach (DataRow curConnect in courseExamsTable.Rows)

{

Course curCourse = (Course) GetObjectById((int) curConnect["course\_id"], courses);

Exam curExam = (Exam) GetObjectById((int) curConnect["exam\_id"], exams);

curCourse.Exams.Add(curExam);

}

foreach (DataRow curTheory in theoryTable.Rows)

theory.Add(new Theory((int) curTheory["id"], curTheory["name"].ToString(), curTheory["text"].ToString()));

foreach (DataRow curConnect in neededTheoryTable.Rows)

{

Exam curExam = (Exam) GetObjectById((int) curConnect["exam\_id"], exams);

Theory curTheory = (Theory) GetObjectById((int) curConnect["theory\_id"], theory);

curExam.TheoryNeeded.Add(curTheory);

}

foreach (DataRow curQuestion in questionTable.Rows)

questions.Add(new Question((int) curQuestion["id"], curQuestion["text"].ToString(), new List<Answer>()));

foreach (DataRow curConnect in examQuestionsTable.Rows)

{

Exam curExam = (Exam) GetObjectById((int) curConnect["exam\_id"], exams);

Question curQuestion = (Question) GetObjectById((int) curConnect["question\_id"], questions);

curExam.Questions.Add(curQuestion);

}

foreach (DataRow curAnswer in answerTable.Rows)

answers.Add(new Answer((int) curAnswer["id"], curAnswer["text"].ToString()));

foreach (DataRow curConnect in questionAnswersTable.Rows)

{

Question curQuestion = (Question) GetObjectById((int) curConnect["question\_id"], questions);

Answer curAnswer = (Answer) GetObjectById((int) curConnect["answer\_id"], answers);

curQuestion.Answers.Add(curAnswer);

}

foreach (DataRow curConnect in userPassedExams.Rows)

{

User curUser = (User) GetObjectById((int) curConnect["user\_id"], users);

Exam curExam = (Exam) GetObjectById((int) curConnect["exam\_id"], exams);

curExam.Passed.Add(curUser);

}

return courses;

}

private static object GetObjectById(int id, List<Course> list) // List<ISQLData>

{

foreach (var el in list)

if (el.GetId() == id) return el;

return null;

}

private static object GetObjectById(int id, List<User> list) // List<ISQLData>

{

foreach (var el in list)

if (el.GetId() == id) return el;

return null;

}

private static object GetObjectById(int id, List<Question> list)

{

foreach (var el in list)

if (el.GetId() == id) return el;

return null;

}

private static object GetObjectById(int id, List<Exam> list) // List<ISQLData>

{

foreach (var el in list)

if (el.GetId() == id) return el;

return null;

}

private static object GetObjectById(int id, List<Theory> list) // List<ISQLData>

{

foreach (var el in list)

if (el.GetId() == id) return el;

return null;

}

private static object GetObjectById(int id, List<Answer> list) // List<ISQLData>

{

foreach (var el in list)

if (el.GetId() == id) return el;

return null;

}

public SqlTableManager(SqlConnection connection)

{

this.connection = connection;

}

public DataTable GetTable(string tableName)

{

string query = "SELECT \* FROM " + tableName;

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable table = new DataTable();

dataAdapter.Fill(table);

return table;

}

private Course GetCourseByName(string name)

{

foreach (var curCourse in TopManager.instance.Courses)

if (curCourse.Name.Equals(name))

return curCourse;

return null;

}

public int GetFreeId(string tableName)

{

List<int> invalids = new List<int>();

foreach (DataRow curRow in GetTable(tableName).Rows)

invalids.Add((int) curRow["id"]);

int lowest = 0;

while (invalids.Contains(lowest)) lowest++;

return lowest;

}

public void AddRow(string tableName, string[] columns, string[] values)

{

string query = "INSERT INTO " + tableName + " (";

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

query += columns[i] + ((i == columns.Length - 1) ? ") VALUES (" : ",");

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

query += "\'"+ values[i] + ((i == values.Length - 1) ? "\')" : "\',");

ExecuteCommand(query);

}

private void ExecuteCommand(string query)

{

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

TopManager.instance.Update();

}

public void CloseConnection()

{

connection.Close();

}

}

}

C# код класса RoundButton

using System;

using System.Linq;

using System.Windows.Forms;

using System.IO;

using Block.user.level;

using Block.manage;

namespace Block.form

{

public class RoundButton : Button

{

}

public class RoundButtonContext : RoundButton

{

public static readonly string checkedCachePath = "./checked.txt";

private Context context;

public RoundButtonContext(Context context)

{

this.context = context;

Click += OnClick;

}

public Context Context

{

get {return context;}

private set {context = value;}

}

public void OnClick(object sender, EventArgs e)

{

context.GetMessage();

Check();

BackColor = Easel.GetDarker(BackColor);

}

public void Check()

{

string content = File.ReadAllText(checkedCachePath);

if (!content.Contains(context.Title))

{

content += context.Title;

File.WriteAllText(checkedCachePath, content);

}

}

public bool IsChecked()

{

string content = File.ReadAllText(checkedCachePath);

return content.Contains(context.Title);

}

}

}

C# код класса User

using System;

using System.Linq;

using Block.Algorithm;

namespace Block.user

{

public abstract class User : ISQLData

{

protected int id;

protected string name;

protected string password;

public User(int id, string name, string password)

{

this.id = id;

this.name = name;

this.password = password;

}

public string Name

{

get { return name; }

protected set { name = value; }

}

public string Password

{

get { return password; }

protected set { password = value; }

}

public int GetId()

{

return id;

}

protected abstract bool IsAvaibleToChat();

protected abstract bool IsAvaibleToMakeGroups();

protected abstract bool IsAvaibleToKickMembers();

protected abstract bool IsAvaibleToAddMembers();

protected abstract bool IsAvaibleToCreateUsers();

protected abstract bool IsAvaibleToDeleteUsers();

}

}

C# код класса Tutor

using System;

namespace Block.user

{

public class Tutor : User

{

public Tutor(int id, string name, string password) : base(id, name, password) { }

protected sealed override bool IsAvaibleToChat()

{

return true;

}

protected sealed override bool IsAvaibleToMakeGroups()

{

return true;

}

protected sealed override bool IsAvaibleToKickMembers()

{

return true;

}

protected sealed override bool IsAvaibleToAddMembers()

{

return true;

}

protected sealed override bool IsAvaibleToCreateUsers()

{

return false;

}

protected sealed override bool IsAvaibleToDeleteUsers()

{

return false;

}

}

}

C# код класса Student

using System;

namespace Block.user

{

public class Student : User

{

public Student(int id, string name, string password) : base(id, name, password) { }

protected sealed override bool IsAvaibleToChat()

{

return true;

}

protected sealed override bool IsAvaibleToMakeGroups()

{

return false;

}

protected sealed override bool IsAvaibleToKickMembers()

{

return false;

}

protected sealed override bool IsAvaibleToAddMembers()

{

return true;

}

protected sealed override bool IsAvaibleToCreateUsers()

{

return false;

}

protected sealed override bool IsAvaibleToDeleteUsers()

{

return false;

}

}

}

C# код класса Admin

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Text;

namespace Block.user

{

public class Admin : User

{

public Admin(int id, string name, string password) : base(id, name, password) { }

protected sealed override bool IsAvaibleToChat()

{

return true;

}

protected sealed override bool IsAvaibleToMakeGroups()

{

return true;

}

protected sealed override bool IsAvaibleToKickMembers()

{

return true;

}

protected sealed override bool IsAvaibleToAddMembers()

{

return true;

}

protected sealed override bool IsAvaibleToCreateUsers()

{

return true;

}

protected sealed override bool IsAvaibleToDeleteUsers()

{

return true;

}

}

}

C# код класса Action

using System;

using System.Linq;

namespace Block.user

{

public enum Action

{

CreateUser,

DeleteCreatedUser,

DeleteUser,

CreateGroup,

DeleteCreatedGroup,

DeleteGroup,

Chat,

DeleteMessage,

}

}

C# код класса Theory

using System;

using Block.Algorithm;

namespace Block.user.level

{

public class Theory : ISQLData

{

private int id;

private string name;

private string text;

public Theory(int id, string name, string text)

{

this.id = id;

this.name = name;

this.text = text;

}

public string Name

{

get {return name;}

private set {name = value;}

}

public string Text

{

get {return text;}

private set {text = value;}

}

public int GetId()

{

return id;

}

}

}

C# код класса Question

using System;

using System.Collections.Generic;

using Block.Algorithm;

namespace Block.user.level

{

public class Question : ISQLData

{

private int id;

private string text;

private List<Answer> answers;

private Answer correctAnswer;

public Question(int id, string text, List<Answer> answers, int correctAnswerId)

{

this.id = id;

this.text = text;

this.answers = answers;

correctAnswer = null;

if (answers != null)

if (correctAnswerId < answers.Count) this.correctAnswer = answers[correctAnswerId];

}

public Question(int id, string question, List<Answer> answers) : this(id, question, answers, 0) { }

public string Text

{

get {return text;}

private set{text = value;}

}

public List<Answer> Answers

{

get {return answers;}

private set {answers = value;}

}

public int GetId()

{

return id;

}

}

}

C# код класса Exam

using System;

using System.Collections.Generic;

using Block.Algorithm;

namespace Block.user.level

{

public class Exam : ISQLData

{

private int id;

private string name;

private List<Theory> theoryNeeded;

private List<Question> questions;

private List<User> passed;

public Exam(List<Theory> theoryNeeded, List<Question> questions, List<User> passed) :

this (0, "Тест", theoryNeeded, questions, passed) { }

public Exam(int id, string name, List<Theory> theoryNeeded, List<Question> questions, List<User> passed)

{

this.id = id;

this.passed = passed;

this.name = name;

this.theoryNeeded = theoryNeeded;

this.questions = questions;

}

public string Name

{

get {return name;}

private set {name = value;}

}

public List<Theory> TheoryNeeded

{

get {return theoryNeeded;}

private set {theoryNeeded = value;}

}

public List<Question> Questions

{

get {return questions;}

private set {questions = value;}

}

public List<User> Passed

{

get {return passed;}

private set {passed = value;}

}

public int GetId()

{

return id;

}

}

}

C# код класса Course

using System;

using System.Collections.Generic;

using Block.Algorithm;

using Block.manage;

namespace Block.user.level

{

public class Course : ISQLData

{

protected int id;

protected string name;

protected string password;

protected List<Exam> exams;

public Course(int id, string name, string password, List<Exam> exams)

{

this.id = id;

this.name = name;

this.password = password;

this.exams = exams;

}

public List<Exam> Exams

{

get {return exams;}

set {exams = value;}

}

public string Name

{

get {return name;}

private set {name = value;}

}

public string Password

{

get {return password;}

private set {password = value;}

}

public int GetId()

{

return id;

}

}

}

C# код класса Context

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Collections.Generic;

namespace Block.user.level

{

public abstract class Context

{

protected static readonly Size BUTTON\_SIZE = new Size(40, 40);

protected static readonly Size ELEMENT\_SIZE = new Size(30, 30);

protected static readonly int SCREEN\_WIDTH = Screen.PrimaryScreen.Bounds.Width;

protected static readonly int SCREEN\_HEIGHT = Screen.PrimaryScreen.Bounds.Height;

protected string title;

protected Context(string title)

{

this.title = title;

}

public abstract void GetMessage();

public string Title

{

get {return title;}

private set{title = value;}

}

}

public class StringContext : Context

{

private string text;

public StringContext(string title, string text) : base(title)

{

this.text = text;

}

public override void GetMessage()

{

MessageBox.Show(text);

}

public string Text

{

get {return text;}

private set {text = value;}

}

}

public class FormContext : Context

{

private static readonly int FORM\_WIDTH = 500;

private static readonly int FORM\_HEIGHT = 500;

private Exam exam;

private int curQuestionId;

public FormContext(string title, Exam exam) : base(title)

{

this.exam = exam;

curQuestionId = 0;

}

public override void GetMessage()

{

List<Answer> answers = exam.Questions[curQuestionId].Answers;

Form questionForm = new Form();

questionForm.Size = new Size(FORM\_WIDTH, FORM\_HEIGHT);

questionForm.StartPosition = FormStartPosition.CenterScreen;

questionForm.Visible = true;

addQuestionLabel(questionForm.Controls, exam.Questions[curQuestionId].Text);

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

addAnswerLabel(questionForm.Controls, answers[i].Text, i);

addControlButtons(questionForm.Controls, this, questionForm, curQuestionId, exam.Questions.Count);

}

private static void addControlButtons(Control.ControlCollection controls, FormContext formHolder, Form form, int curIndex, int maxIndex)

{

Button[] controlButtons = new Button[2];

controlButtons[0] = new Button();

controlButtons[0].Text = ">";

controlButtons[0].Size = BUTTON\_SIZE;

controlButtons[0].Location = new Point(FORM\_WIDTH - (int)(BUTTON\_SIZE.Width \* 1.5), FORM\_HEIGHT - BUTTON\_SIZE.Height \* 2);

if (curIndex < maxIndex - 1)

controlButtons[0].Click += (sender, e) => {

formHolder.curQuestionId++;

form.Close();

formHolder.GetMessage();

};

controlButtons[1] = new Button();

controlButtons[1].Text = "<";

controlButtons[1].Size = BUTTON\_SIZE;

controlButtons[1].Location = new Point(0, FORM\_HEIGHT - BUTTON\_SIZE.Height \* 2);

if (curIndex > 0)

controlButtons[1].Click += (sender, e) => {

formHolder.curQuestionId--;

form.Close();

formHolder.GetMessage();

};

foreach(var curControl in controlButtons)

controls.Add(curControl);

}

private static void addQuestionLabel(Control.ControlCollection controls, string text)

{

Label questionLabel = new Label();

questionLabel.Text = text;

questionLabel.Location = new Point(ELEMENT\_SIZE.Width / 2, ELEMENT\_SIZE.Height / 2);

questionLabel.Size = new Size(FORM\_WIDTH, questionLabel.PreferredHeight);

controls.Add(questionLabel);

}

private static void addAnswerLabel(Control.ControlCollection controls, string text, int id)

{

RadioButton answerLabel = new RadioButton();

answerLabel.Text = text;

answerLabel.Location = new Point(ELEMENT\_SIZE.Width / 2, id \* ELEMENT\_SIZE.Height + ELEMENT\_SIZE.Height \* 2);

answerLabel.Size = new Size(FORM\_WIDTH, 20);

answerLabel.CheckedChanged += (sender, e) => {

};

controls.Add(answerLabel);

}

}

}

C# код класса Answer

using System;

using Block.Algorithm;

namespace Block.user.level

{

public class Answer : ISQLData

{

protected int id;

private string text;

public Answer(int id, string text)

{

this.id = id;

this.text = text;

}

public string Text

{

get {return text;}

private set {text = value;}

}

public int GetId()

{

return id;

}

}

}

C# код интерфейса ISQLData

using System;

namespace Block.Algorithm

{

public interface ISQLData

{

int GetId();

}

}

C# код класс IntPairCache

using System;

namespace Block.Algorithm

{

public class IntPairCache

{

public int f;

public int s;

public IntPairCache(int f, int s)

{

this.f = f;

this.s = s;

}

public IntPairCache() {}

}

}