**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

СОГЛАСОВАНО  
Старший преподаватель программной инженерии факультета компьютерных наук  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.В. Дворянский  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. УТВЕРЖДАЮ  
Академический руководитель образовательной программы «Программная инженерия»  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**Обучающий визуализатор с тестами: основные понятия теорий множеств и отношений.**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.15-01 12 01-1-ЛУ**

**Исполнитель**

Студент группы БПИ173

\_\_\_\_\_\_\_\_\_\_\_\_ /К.Г. Кожакин/

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**Москва 2018**

**УТВЕРЖДЕН**

**RU.17701729.04.15-01 12 01-1-ЛУ**

**Обучающий визуализатор с тестами: основные понятия теорий множеств и отношений.**

**Текст программы**

**RU.17701729.04.15-01 51 01-1**

**Листов 42**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**Москва 2018**

**СОДЕРЖАНИЕ**

1. ТЕКСТ ПРОГРАММЫ………………………………………………………………...3

1.1. Класс Login.cs…………….……………………………………………………….3

1.2. Класс ChangeInfo.cs...…….……………………………………………………….4

1.3. Класс DBMySQLUtils.cs....……………………………………………………….6

1.4. Класс DBUtils.cs………….……………………………………………………….7

1.5. Класс General.cs.………….……………………………………………………….8

1.6. Классы Graph.cs….……….……………………………………………………...12

1.6.1. Класс Graph………..……………………………………………………...12

1.6.2. Класс Vertice..……..……………………………………………………...12

1.6.3. Класс Edge..………..……………………………………………………...13

1.6.4. Класс Origin...……..……………………………………………………...13

1.6.5. Класс Destination.....……………………………………………………...13

1.7. Класс GraphLayout.xaml.cs……………………………………………………...14

1.8. Класс Program.cs………….……………………………………………………...15

1.9. Класс Reg.cs...…………….……………………………………………………...16

1.10. Класс Results.cs………….……………………………………………………...18

1.11. Классы Test.cs.….……….……………………………………………………...21

1.11.1. Класс Test………..……….……………………………………………...22

1.11.2. Класс Task..……..…………..…………………………………………...23

1.12. Класс TestForm.cs……….……………………………………………………...23

2. Текст скрипта базы данных MySQL………………………………………………..39

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 41

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 42

1. **ТЕКСТ ПРОГРАММЫ**
   1. **Класс Login.cs:**
2. **using** MySql.Data.MySqlClient;
3. **using** System;
4. **using** System.Data.Common;
5. **using** System.IO;
6. **using** System.Windows.Forms;
8. **namespace** Sets\_RelationsLessons
9. {
10. **public** partial **class** Login : Form
11. {
12. MySqlConnection conn;
14. **public** Login()
15. {
16. InitializeComponent();
17. conn = DBUtils.GetDBConnection();
18. **if** (Application.OpenForms.Count > 10)
19. {
20. Application.Restart();
21. }
22. }
23. /// <summary>
24. /// Обработчик события нажатия кнопки "Зарегистрироваться"(Перейти на форму регистрации)
25. /// </summary>
26. **private** **void** Registration\_Click(**object** sender, EventArgs e)
27. {
28. Reg \_RegForm = **new** Reg(**this**);
29. \_RegForm.Show();
30. **this**.Hide();
31. }
32. /// <summary>
33. /// Обработчик события нажатия кнопки "Вход"
34. /// </summary>
35. **private** **void** Start\_Click(**object** sender, EventArgs e)
36. {
37. **bool** flag = **false**;
38. **try**
39. {
40. conn.Open();
42. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idUsers, Login, Password from users", conn))
43. {
44. **using** (DbDataReader reader = cmd.ExecuteReader())
45. {
46. **if** (reader.HasRows)
47. {
48. **while** (reader.Read())
49. {
50. **if** (Login\_value.Text == reader.GetString(reader.GetOrdinal("Login")))
51. **if** (Password\_value.Text.GetHashCode().ToString() == reader.GetString(reader.GetOrdinal("Password")))
52. {
53. Properties.Settings.Default.login = Login\_value.Text;
54. Properties.Settings.Default.Save();
55. General \_GeneralForm = **new** General();
56. \_GeneralForm.Show();
57. **this**.Hide();
58. flag = **true**;
59. **break**;
60. }
61. **else**
62. **throw** **new** Exception("Неверный пароль.");
63. }
64. **if** (!flag) **throw** **new** Exception("Такой логин не существует.");
65. }
66. }
67. }
68. }
69. **catch** (Exception ex)
70. {
71. MessageBox.Show(ex.Message);
72. }
73. **finally**
74. {
75. conn.Close();
76. conn.Dispose();
77. }
78. }
79. /// <summary>
80. /// Обработчик события закрытия формы
81. /// </summary>
82. **private** **void** Login\_FormClosing(**object** sender, FormClosingEventArgs e)
83. {
84. **foreach**(**string** name **in** Directory.GetFiles(Directory.GetCurrentDirectory() + @"\Lessons\", "tmp\*"))
85. {
86. File.Delete(name);
87. }
88. }
89. }
90. }
    1. **Класс ChangeInfo.cs:**
91. **using** MySql.Data.MySqlClient;
92. **using** System;
93. **using** System.Data.Common;
94. **using** System.Windows.Forms;
96. **namespace** Sets\_RelationsLessons
97. {
98. **public** partial **class** ChangeInfo : Form
99. {
100. MySqlConnection conn;
101. General \_GenForm;
103. **public** ChangeInfo(General GenForm)
104. {
105. \_GenForm = GenForm;
106. InitializeComponent();
107. conn = DBUtils.GetDBConnection();
108. }
109. /// <summary>
110. /// Обработчик события нажатия на кнопку "Отмена"
111. /// </summary>
112. **private** **void** Cancel\_Click(**object** sender, EventArgs e)
113. {
114. **this**.Close();
115. }
116. /// <summary>
117. /// Обработчик события нажатия на кнопку "Изменить"
118. /// </summary>
119. **private** **void** Change\_Click(**object** sender, EventArgs e)
120. {
121. **int** id = 0;
122. **try**
123. {
124. conn.Open();
125. **if** (Login\_value.Text.Length < 3) **throw** **new** Exception("Слишком короткий логин(должен содержать 3 или больше символов).");
126. **if** (Login\_value.Text.Length > 15) **throw** **new** Exception("Слишком длинный логин(должен содержать 15 или меньше символов).");
127. **if** (Login\_value.Text.IndexOf(' ') > -1) **throw** **new** Exception("Логин не должен содержать пробелы.");
128. **if** (Password\_value2.Text.Length < 3) **throw** **new** Exception("Слишком короткий пароль(должен содержать 3 или больше символов).");
129. **if** (Password\_value2.Text.Length > 20) **throw** **new** Exception("Слишком длинный пароль(должен содержать 20 или меньше символов).");
130. **if** (Password\_value2.Text.IndexOf(' ') > -1) **throw** **new** Exception("Пароль не должен содержать пробелы.");
132. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idUsers, Login, Password from users", conn))
133. {
134. **using** (DbDataReader reader = cmd.ExecuteReader())
135. {
136. **if** (reader.HasRows)
137. {
138. **while** (reader.Read())
139. {
140. **if** (Properties.Settings.Default.login == reader.GetString(reader.GetOrdinal("Login")))
141. **if** (Password\_value.Text.GetHashCode().ToString() != reader.GetString(reader.GetOrdinal("Password")))
142. **throw** **new** Exception("Неверный пароль.");
143. **else** id = reader.GetInt32(reader.GetOrdinal("idUsers"));
144. **if** (Login\_value.Text == reader.GetString(reader.GetOrdinal("Login")))
145. **throw** **new** Exception("Этот логин уже занят.");
146. }
147. }
148. }
149. }
151. **using** (MySqlCommand cmd = **new** MySqlCommand("Update users set Login = @login, Password = @password where idUsers = @id", conn))
152. {
153. cmd.Parameters.AddWithValue("@login", Login\_value.Text);
154. cmd.Parameters.AddWithValue("@id", id);
155. cmd.Parameters.AddWithValue("@password", Password\_value2.Text.GetHashCode());
157. cmd.ExecuteNonQuery();
158. }
159. MessageBox.Show("Изменения успешно сохранены!");
161. Properties.Settings.Default.login = Login\_value.Text;
162. Properties.Settings.Default.Save();
163. \_GenForm.Text = "Добро пожаловать, " + Login\_value.Text;
164. \_GenForm.Show();
165. conn.Close();
166. **this**.Close();
167. }
168. **catch** (Exception ex)
169. {
170. MessageBox.Show(ex.Message);
171. }
172. **finally**
173. {
174. conn.Close();
175. }
176. }
177. /// <summary>
178. /// Обработчик события закрытия формы
179. /// </summary>
180. **private** **void** ChangeInfo\_FormClosing(**object** sender, FormClosingEventArgs e)
181. {
182. \_GenForm.Show();
183. }
184. }
185. }

**1.3 Класс DBMySQLUtils.cs:**

1. **using** System;
2. **using** MySql.Data.MySqlClient;
4. **namespace** Sets\_RelationsLessons
5. {
6. **class** DBMySQLUtils
7. {
8. /// <summary>
9. /// Получение соединения с базой данных
10. /// </summary>
11. /// <param name="host">хост</param>
12. /// <param name="port">порт</param>
13. /// <param name="database">название базы данных</param>
14. /// <param name="username">имя пользователя</param>
15. /// <param name="password">пароль</param>
16. /// <returns>соединение</returns>
17. **public** **static** MySqlConnection GetDBConnection(**string** host, **uint** port, **string** database, **string** username, **string** password)
18. {
19. var builder = **new** MySqlConnectionStringBuilder
20. {
21. Database = database,
22. Password = password,
23. Server = host,
24. UserID = username,
25. SslMode = MySqlSslMode.None,
26. Port = port
27. };
29. MySqlConnection conn = **new** MySqlConnection
30. {
31. ConnectionString = builder.ConnectionString
32. };
34. **return** conn;
35. }
36. }
37. }

**1.4 Класс DBUtils.cs:**

1. **using** MySql.Data.MySqlClient;
3. **namespace** Sets\_RelationsLessons
4. {
5. **class** DBUtils
6. {
7. /// <summary>
8. /// Получение соединения с базой данных
9. /// </summary>
10. /// <returns>содинение</returns>
11. **public** **static** MySqlConnection GetDBConnection()
12. {
13. **string** host = "IP\_Adress";
14. **uint** port = 3306;
15. **string** database = "DataBase\_name";
16. **string** username = "Username";
17. **string** password = "Password";
19. **return** DBMySQLUtils.GetDBConnection(host, port, database, username, password);
20. }
21. }
22. }

**1.5 Класс General.cs:**

1. **using** System;
2. **using** System.IO;
3. **using** System.Windows.Forms;
5. **namespace** Sets\_RelationsLessons
6. {
7. **public** partial **class** General : Form
8. {
9. **public** General()
10. {
11. InitializeComponent();
12. **this**.Text = "Добро пожаловать, " + Properties.Settings.Default.login;
13. TextArea.ScriptErrorsSuppressed = **true**;
14. **for** (**int** i = 1; i <= 10; i++)
15. {
16. **if** (File.Exists(Directory.GetCurrentDirectory() + @"\Lessons\lesson" + i + ".html"))
17. выборТемыToolStripMenuItem.DropDownItems[i - 1].Visible = **true**;
18. **else**
19. выборТемыToolStripMenuItem.DropDownItems[i - 1].Visible = **false**;
20. }
21. }
22. /// <summary>
23. /// Обработчик события нажатия на пункт меню "Изменить персонльные данные"
24. /// </summary>
25. **private** **void** изменитьДанныеToolStripMenuItem\_Click(**object** sender, EventArgs e)
26. {
27. **this**.Hide();
28. ChangeInfo form = **new** ChangeInfo(**this**);
29. form.Show();
30. }
31. /// <summary>
32. /// Обработчик события завершения загрузки html-документа
33. /// </summary>
34. **private** **void** TextArea\_DocumentCompleted(**object** sender, WebBrowserDocumentCompletedEventArgs e)
35. {
36. **foreach** (HtmlElement he **in** TextArea.Document.GetElementsByTagName("input"))
37. {
38. **switch** (he.GetAttribute("name"))
39. {
40. **case** "Next":
41. he.AttachEventHandler("onclick", Next\_Click);
42. **break**;
43. **case** "Previous":
44. he.AttachEventHandler("onclick", Previous\_Click);
45. **break**;
46. **case** "StartTest":
47. he.AttachEventHandler("onclick", StartTest\_Click);
48. **break**;
49. **case** "ContinueTest":
50. he.AttachEventHandler("onclick", ContinueTest\_Click);
51. **break**;
52. **default**:
53. **break**;
54. }
55. }
56. }
57. /// <summary>
58. /// Обработчик события нажатия кнопки "Следующий урок"
59. /// </summary>
60. **private** **void** Next\_Click(**object** sender, EventArgs e)
61. {
62. **if** (File.Exists(Directory.GetCurrentDirectory() + @"\Lessons\lesson" + (**int**.Parse(TextArea.DocumentTitle.Split()[1]) + 1) + ".html"))
63. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson" + (**int**.Parse(TextArea.DocumentTitle.Split()[1]) + 1) + ".html");
64. }
65. /// <summary>
66. /// Обработчик события нажатия кнопки "Предыдущий урок"
67. /// </summary>
68. **private** **void** Previous\_Click(**object** sender, EventArgs e)
69. {
70. **if** (File.Exists(Directory.GetCurrentDirectory() + @"\Lessons\lesson" + (**int**.Parse(TextArea.DocumentTitle.Split()[1]) - 1) + ".html"))
71. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson" + (**int**.Parse(TextArea.DocumentTitle.Split()[1]) - 1) + ".html");
72. }
73. /// <summary>
74. /// Обработчик события нажатия кнопки "Начать тест"
75. /// </summary>
76. **private** **void** StartTest\_Click(**object** sender, EventArgs e)
77. {
78. **if** (File.Exists(Directory.GetCurrentDirectory() + @"\Lessons\test" + **int**.Parse(TextArea.DocumentTitle.Split()[1]) + ".txt"))
79. {
80. **this**.Hide();
81. TestForm form = **new** TestForm(Directory.GetCurrentDirectory() + @"\Lessons\test" + **int**.Parse(TextArea.DocumentTitle.Split()[1]) + ".txt");
82. **if** (form.IsDisposed == **false**) form.Show();
83. }
84. }
85. /// <summary>
86. /// Обработчик события нажатия кнопки "Продолжить тест"
87. /// </summary>
88. **private** **void** ContinueTest\_Click(**object** sender, EventArgs e)
89. {
90. **if** (File.Exists(Directory.GetCurrentDirectory() + @"\Lessons\test" + **int**.Parse(TextArea.DocumentTitle.Split()[1]) + ".txt"))
91. {
92. **this**.Hide();
93. TestForm form = **new** TestForm(Directory.GetCurrentDirectory() + @"\Lessons\test" + **int**.Parse(TextArea.DocumentTitle.Split()[1]) + ".txt", 1);
94. **if** (form.IsDisposed == **false**) form.Show();
95. }
96. }
97. /// <summary>
98. /// Обработчик события нажатия на пункт меню "Сменить пользователя"
99. /// </summary>
100. **private** **void** сменитьПользователяToolStripMenuItem\_Click(**object** sender, EventArgs e)
101. {
102. DialogResult dialogResult = MessageBox.Show("Вы действительно хотите выйти?", "Смена пользователя", MessageBoxButtons.OKCancel);
103. **if** (dialogResult == DialogResult.OK)
104. {
105. Login log = **new** Login();
106. log.Show();
107. **this**.Hide();
108. Properties.Settings.Default.login = "";
109. Properties.Settings.Default.Save();
110. }
111. }
112. /// <summary>
113. /// Обработчик события нажатия на пункт меню "Результаты тестов"
114. /// </summary>
115. **private** **void** результатыТестовToolStripMenuItem\_Click(**object** sender, EventArgs e)
116. {
117. Results res = **new** Results();
118. res.Show();
119. }
120. /// <summary>
121. /// Обработчик события нажатия на пункт меню "Урок 1"
122. /// </summary>
123. **private** **void** урок1ToolStripMenuItem\_Click(**object** sender, EventArgs e)
124. {
125. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson1.html");
126. }
127. /// <summary>
128. /// Обработчик события нажатия на пункт меню "Урок 2"
129. /// </summary>
130. **private** **void** урок2ToolStripMenuItem\_Click(**object** sender, EventArgs e)
131. {
132. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson2.html");
133. }
134. /// <summary>
135. /// Обработчик события нажатия на пункт меню "Урок 3"
136. /// </summary>
137. **private** **void** урок3ToolStripMenuItem\_Click(**object** sender, EventArgs e)
138. {
139. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson3.html");
140. }
141. /// <summary>
142. /// Обработчик события нажатия на пункт меню "Урок 4"
143. /// </summary>
144. **private** **void** урок4ToolStripMenuItem\_Click(**object** sender, EventArgs e)
145. {
146. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson4.html");
147. }
148. /// <summary>
149. /// Обработчик события нажатия на пункт меню "Урок 5"
150. /// </summary>
151. **private** **void** урок5ToolStripMenuItem\_Click(**object** sender, EventArgs e)
152. {
153. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson5.html");
154. }
155. /// <summary>
156. /// Обработчик события нажатия на пункт меню "Урок 6"
157. /// </summary>
158. **private** **void** урок6ToolStripMenuItem\_Click(**object** sender, EventArgs e)
159. {
160. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson6.html");
161. }
162. /// <summary>
163. /// Обработчик события нажатия на пункт меню "Урок 7"
164. /// </summary>
165. **private** **void** урок7ToolStripMenuItem\_Click(**object** sender, EventArgs e)
166. {
167. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson7.html");
168. }
169. /// <summary>
170. /// Обработчик события нажатия на пункт меню "Урок 8"
171. /// </summary>
172. **private** **void** урок8ToolStripMenuItem\_Click(**object** sender, EventArgs e)
173. {
174. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson8.html");
175. }
176. /// <summary>
177. /// Обработчик события нажатия на пункт меню "Урок 9"
178. /// </summary>
179. **private** **void** урок9ToolStripMenuItem\_Click(**object** sender, EventArgs e)
180. {
181. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson9.html");
182. }
183. /// <summary>
184. /// Обработчик события нажатия на пункт меню "Урок 10"
185. /// </summary>
186. **private** **void** урок10ToolStripMenuItem\_Click(**object** sender, EventArgs e)
187. {
188. TextArea.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\lesson10.html");
189. }
190. /// <summary>
191. /// Обработчик события нажатия на пункт меню "Выход"
192. /// </summary>
193. **private** **void** выходToolStripMenuItem\_Click(**object** sender, EventArgs e)
194. {
195. **this**.Close();
196. }
197. /// <summary>
198. /// Обработчик события закрытия формы
199. /// </summary>
200. **private** **void** General\_FormClosing(**object** sender, FormClosingEventArgs e)
201. {
202. **if** (e.CloseReason.ToString() == "UserClosing")
203. {
204. DialogResult dialogResult = MessageBox.Show("Вы действительно хотите выйти?", "Выход", MessageBoxButtons.OKCancel);
205. **if** (dialogResult == DialogResult.OK)
206. {
207. **foreach** (**string** name **in** Directory.GetFiles(Directory.GetCurrentDirectory() + @"\Lessons\", "tmp\*"))
208. {
209. File.Delete(name);
210. }
211. }
212. **else**
213. {
214. e.Cancel = **true**;
215. }
216. }
217. }
218. }
219. }
220. **Классы Graph.cs**
     1. **Класс Graph:**
221. **using** System.Collections.Generic;
222. **using** System.Xml.Serialization;
224. **namespace** Sets\_RelationsLessons
225. {
226. /// <summary>
227. /// Класс графа
228. /// </summary>
229. [XmlRoot("Diagram")]
230. **public** **class** Graph
231. {
232. [XmlArray("Nodes")]
233. [XmlArrayItem("Node")]
234. **public** List<Vertice> vertices;
235. [XmlArray("Links")]
236. [XmlArrayItem("Link")]
237. **public** List<Edge> edges;
238. }
239. }
     1. **Класс Vertice:**
240. **using** System.Xml.Serialization;
242. **namespace** Sets\_RelationsLessons
243. {
244. /// <summary>
245. /// Класс вершины
246. /// </summary>
247. **public** **class** Vertice
248. {
249. [XmlAttribute("Id")]
250. **public** **string** Id;
251. }
252. }
     1. **Класс Edge:**
253. **using** System.Xml.Serialization;
255. **namespace** Sets\_RelationsLessons
256. {
257. /// <summary>
258. /// Класс ребра
259. /// </summary>
260. **public** **class** Edge
261. {
262. [XmlElement("Origin")]
263. **public** Origin OId;
264. [XmlElement("Destination")]
265. **public** Destination DId;
266. }
267. }
     1. **Класс Origin:**
268. **using** System.Xml.Serialization;
270. **namespace** Sets\_RelationsLessons
271. {
272. /// <summary>
273. /// Класс вершины, из которой выходит ребро
274. /// </summary>
275. **public** **class** Origin : Vertice
276. {
277. [XmlAttribute("Id")]
278. **public** **new** **string** Id;
279. }
280. /// <summary>
281. }
     1. **Класс Destination:**
282. **using** System.Xml.Serialization;
284. **namespace** Sets\_RelationsLessons
285. {
286. /// <summary>
287. /// Класс вершины, в которую входит ребро
288. /// </summary>
289. **public** **class** Destination : Vertice
290. {
291. [XmlAttribute("Id")]
292. **public** **new** **string** Id;
293. }
294. }
295. **Класс GraphLayout.xaml.cs:**
296. **using** System;
297. **using** System.IO;
298. **using** System.Windows;
299. **using** System.Windows.Controls;
300. **using** MindFusion.Diagramming.Wpf;
302. **namespace** Sets\_RelationsLessons
303. {
304. /// <summary>
305. /// Логика взаимодействия для GraphLayout.xaml
306. /// </summary>
307. **public** partial **class** GraphLayout : UserControl
308. {
309. Test test;
310. **int** num;
312. **public** GraphLayout(**ref** Test test, **int** num, **string** name, **bool** EdgeAuto)
313. {
314. InitializeComponent();
315. **this**.test = test;
316. **this**.num = num;
317. diagram.DefaultShape = Shapes.Ellipse;
318. **try**
319. {
320. **if** (name == "ans")
321. diagram.LoadFromXml(Directory.GetCurrentDirectory() + @"\Lessons\tmpGraph" + num);
322. **else** **if** (name == "true")
323. diagram.LoadFromXml(Directory.GetCurrentDirectory() + @"\Lessons\" + test.tasks[num].answer);
324. **else** **if** (test.tasks[num].graph\_name != "")
325. diagram.LoadFromXml(test.tasks[num].graph\_name);
326. **else**
327. diagram.LoadFromString("<diagram></diagram>");
328. }
329. **catch** (FileLoadException exp)
330. {
331. MessageBox.Show(exp.Message);
332. **return**;
333. }
334. **catch** (Exception exp)
335. {
336. MessageBox.Show(exp.Message);
337. }
339. **if** (name == "DrawingGraph")
340. {
341. **int** m = diagram.Links.Count;
342. **for** (**int** i = 0; i < m; i++)
343. {
344. diagram.Links.RemoveAt(0);
345. }
346. }
347. **if** (name == "PictureGraph") diagram.EnableLanes = **false**;
349. DiagramLinkCollection diag = **new** DiagramLinkCollection();
350. **int** n = diagram.Links.Count;
351. **for** (**int** i = 0; i < n; i++)
352. {
353. diag.Add(**new** DiagramLink(diagram.Links[0].Parent, diagram.Links[0].Origin, diagram.Links[0].Destination));
354. diag[i].AutoRoute = EdgeAuto;
355. diagram.Links.RemoveAt(0);
356. }
357. **foreach** (var item **in** diag)
358. {
359. diagram.Links.Add(item);
360. }
361. }
362. /// <summary>
363. /// Метод сохранения текущего ответа-графа во временный файл
364. /// </summary>
365. **private** **void** UserControl\_IsEnabledChanged(**object** sender, DependencyPropertyChangedEventArgs e)
366. {
367. **if** (test.tasks[num].choice\_ans == 0 || test.tasks[num].choice\_ans == 4) diagram.SaveToXml(Directory.GetCurrentDirectory() + @"\Lessons\tmpGraph" + num);
368. }
369. /// <summary>
370. /// Запрет на создание и удаление вершин
371. /// </summary>
372. **private** **void** diagram\_NodeChanging(**object** sender, NodeValidationEventArgs e)
373. {
374. e.Cancel = **true**;
375. }
376. /// <summary>
377. /// Позволяет удалять только петли
378. /// </summary>
379. **private** **void** diagram\_LinkDeleting(**object** sender, LinkValidationEventArgs e)
380. {
381. **if** (e.Origin != e.Destination) e.Cancel = **true**;
382. }
383. }
384. }
385. **Класс Program.cs:**
386. **using** System;
387. **using** System.Windows.Forms;
389. **namespace** Sets\_RelationsLessons
390. {
391. **static** **class** Program
392. {
393. /// <summary>
394. /// Главная точка входа для приложения.
395. /// </summary>
396. [STAThread]
397. **static** **void** Main()
398. {
399. Application.EnableVisualStyles();
400. Application.SetCompatibleTextRenderingDefault(**false**);
401. **if** (Properties.Settings.Default.login == "")
402. {
403. Application.Run(**new** Login());
404. }
405. **else**
406. {
407. Application.Run(**new** General());
408. }
409. }
410. }
411. }
     1. **Класс Reg.cs:**
412. **using** MySql.Data.MySqlClient;
413. **using** System;
414. **using** System.Data.Common;
415. **using** System.Windows.Forms;
417. **namespace** Sets\_RelationsLessons
418. {
419. **public** partial **class** Reg : Form
420. {
421. Login \_LogForm;
422. MySqlConnection conn;
424. **public** Reg(Login LogForm)
425. {
426. \_LogForm = LogForm;
427. InitializeComponent();
428. conn = DBUtils.GetDBConnection();
429. }
430. /// <summary>
431. /// Обработчик события нажатия кнопки "Отмена"
432. /// </summary>
433. **private** **void** Cancel\_Click(**object** sender, EventArgs e)
434. {
435. **this**.Close();
436. }
437. /// <summary>
438. /// Обработчик события нажатия кнопки "Зарегистрироваться"(Создать пользователя)
439. /// </summary>
440. **private** **void** Registration\_Click(**object** sender, EventArgs e)
441. {
442. **int** c = 1;
443. **try**
444. {
445. **if** (Login\_value.Text.Length < 3) **throw** **new** Exception("Слишком короткий логин(должен содержать 3 или больше символов).");
446. **if** (Login\_value.Text.Length > 15) **throw** **new** Exception("Слишком длинный логин(должен содержать 15 или меньше символов).");
447. **if** (Login\_value.Text.IndexOf(' ') > -1) **throw** **new** Exception("Логин не должен содержать пробелы.");
448. **if** (Password\_value.Text.Length < 3) **throw** **new** Exception("Слишком короткий пароль(должен содержать 3 или больше символов).");
449. **if** (Password\_value.Text.Length > 20) **throw** **new** Exception("Слишком длинный пароль(должен содержать 20 или меньше символов).");
450. **if** (Password\_value.Text.IndexOf(' ') > -1) **throw** **new** Exception("Пароль не должен содержать пробелы.");
451. **if** (Password\_value.Text.GetHashCode() != Password\_value2.Text.GetHashCode()) **throw** **new** Exception("Пароли не совпадают.");
453. conn.Open();
454. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idUsers, Login, Password from users", conn))
455. {
456. **using** (DbDataReader reader = cmd.ExecuteReader())
457. {
458. **if** (reader.HasRows)
459. {
460. **while** (reader.Read())
461. {
462. c++;
463. **if** (Login\_value.Text == reader.GetString(reader.GetOrdinal("Login")))
464. **throw** **new** Exception("Этот логин уже занят.");
465. }
466. }
467. }
468. }
470. **using** (MySqlCommand cmd = **new** MySqlCommand("Insert into users(idUsers, Login, Password)" + "values (@id, @login, @password)", conn))
471. {
472. MySqlParameter idParam = **new** MySqlParameter("@id", MySqlDbType.Int32)
473. {
474. Value = c
475. };
476. cmd.Parameters.Add(idParam);
478. MySqlParameter loginParam = **new** MySqlParameter("@login", MySqlDbType.VarChar)
479. {
480. Value = Login\_value.Text
481. };
482. cmd.Parameters.Add(loginParam);
484. MySqlParameter passwordParam = **new** MySqlParameter("@password", MySqlDbType.Int32)
485. {
486. Value = Password\_value.Text.GetHashCode()
487. };
488. cmd.Parameters.Add(passwordParam);
490. cmd.ExecuteNonQuery();
491. }
492. MessageBox.Show("Регистрация прошла успешно!");
494. \_LogForm.Show();
495. conn.Close();
496. **this**.Close();
497. }
498. **catch** (Exception ex)
499. {
500. MessageBox.Show(ex.Message);
501. }
502. **finally**
503. {
504. conn.Close();
505. }
506. }
507. /// <summary>
508. /// Обработчик события закрытия формы
509. /// </summary>
510. **private** **void** Reg\_FormClosing(**object** sender, FormClosingEventArgs e)
511. {
512. \_LogForm.Show();
513. }
514. }
515. }
516. **Класс Results.cs:**
517. **using** MySql.Data.MySqlClient;
518. **using** System;
519. **using** System.Data.Common;
520. **using** System.Drawing;
521. **using** System.Linq;
522. **using** System.Windows.Forms;
524. **namespace** Sets\_RelationsLessons
525. {
526. **public** partial **class** Results : Form
527. {
528. MySqlConnection conn;
530. **public** Results()
531. {
532. InitializeComponent();
534. conn = DBUtils.GetDBConnection();
535. ResultToTable();
536. }
537. /// <summary>
538. /// Добавление столбца в таблицу результатов
539. /// </summary>
540. /// <param name="name">название столбца</param>
541. **private** **void** AddColumn(**string** name)
542. {
543. var column = **new** DataGridViewColumn
544. {
545. HeaderText = name,
546. ReadOnly = **true**,
547. Frozen = **false**,
548. CellTemplate = **new** DataGridViewTextBoxCell()
549. };
550. ResultTable.Columns.Add(column);
551. }
552. /// <summary>
553. /// Внесение результатов в таблицу
554. /// </summary>
555. /// <param name="test">номер теста</param>
556. **private** **void** ResultToTable(**int** test = -1)
557. {
558. ResultTable.Rows.Clear();
559. ResultTable.Columns.Clear();
560. **int** id = 0;
561. **try**
562. {
563. conn.Open();
565. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idUsers, Login from users", conn))
566. {
567. **using** (DbDataReader reader = cmd.ExecuteReader())
568. {
569. **if** (reader.HasRows)
570. {
571. **while** (reader.Read())
572. {
573. **if** (Properties.Settings.Default.login == reader.GetString(reader.GetOrdinal("Login"))) id = reader.GetInt32(reader.GetOrdinal("idUsers"));
574. }
575. }
576. }
577. }
579. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idTests, Result, Users\_idUsers, Date, Time from tests", conn))
580. {
581. **if** (test == -1)
582. {
583. AddColumn("Тест");
584. AddColumn("Процент верных ответов (%)");
585. AddColumn("Оценка первой попытки");
587. ResultTable.AllowUserToAddRows = **false**;
588. **string** res;
589. **for** (**int** i = 0; i < 10; i++)
590. {
591. **using** (DbDataReader reader = cmd.ExecuteReader())
592. {
593. **if** (reader.HasRows)
594. {
595. **while** (reader.Read())
596. {
597. **if** (id == reader.GetInt32(reader.GetOrdinal("Users\_idUsers")) && i == reader.GetInt32(reader.GetOrdinal("idTests")))
598. {
599. res = reader.GetString(reader.GetOrdinal("Result"));
600. ResultTable.Rows.Add("Тест " + i, String.Format("{0:0.00}", 1.0 \* res.Count(f => f == 't') / res.Length \* 100), String.Format("{0:0}", 1.0 \* res.Count(f => f == 't') / res.Length \* 10));
601. **if** (**double**.Parse(ResultTable[2, ResultTable.Rows.Count - 1].Value.ToString()) >= 8) ResultTable[2, ResultTable.Rows.Count - 1].Style.BackColor = Color.Green;
602. **else** **if** (**double**.Parse(ResultTable[2, ResultTable.Rows.Count - 1].Value.ToString()) >= 6) ResultTable[2, ResultTable.Rows.Count - 1].Style.BackColor = Color.GreenYellow;
603. **else** **if** (**double**.Parse(ResultTable[2, ResultTable.Rows.Count - 1].Value.ToString()) >= 4) ResultTable[2, ResultTable.Rows.Count - 1].Style.BackColor = Color.Yellow;
604. **else** **if** (**double**.Parse(ResultTable[2, ResultTable.Rows.Count - 1].Value.ToString()) >= 1) ResultTable[2, ResultTable.Rows.Count - 1].Style.BackColor = Color.Orange;
605. **else** ResultTable[2, ResultTable.Rows.Count - 1].Style.BackColor = Color.Red;
606. **break**;
607. }
608. }
609. }
610. }
611. }
612. }
613. **else**
614. {
615. AddColumn("Тест " + (test + 1));
617. **string** res;
618. **bool** f = **true**;
619. **using** (DbDataReader reader = cmd.ExecuteReader())
620. {
621. **if** (reader.HasRows)
622. {
623. **while** (reader.Read())
624. {
625. **if** (id == reader.GetInt32(reader.GetOrdinal("Users\_idUsers")) && test + 1 == reader.GetInt32(reader.GetOrdinal("idTests")))
626. {
627. res = reader.GetString(reader.GetOrdinal("Result"));
628. **if** (f)
629. {
630. **for** (**int** j = 0; j < res.Length; j++)
631. {
632. AddColumn("Вопрос " + (j + 1));
633. }
634. f = **false**;
635. }
636. **if** (ResultTable.Rows.Count == 0)
637. {
638. AddColumn("Процент верных ответов (%)");
639. AddColumn("Оценка");
640. AddColumn("Время сдачи");
641. AddColumn("Время выполнения");
642. }
643. ResultTable.Rows.Add();
644. ResultTable[0, ResultTable.Rows.Count - 1].Value = "Попытка " + ResultTable.Rows.Count;
645. **for** (**int** i = 0; i < res.Length; i++)
646. {
647. **if** (res[i] == 't')
648. {
649. ResultTable[i + 1, ResultTable.Rows.Count - 1].Value = "+";
650. ResultTable[i + 1, ResultTable.Rows.Count - 1].Style.BackColor = Color.Green;
651. }
652. **else**
653. {
654. ResultTable[i + 1, ResultTable.Rows.Count - 1].Value = "-";
655. ResultTable[i + 1, ResultTable.Rows.Count - 1].Style.BackColor = Color.Red;
656. }
657. }
658. ResultTable[ResultTable.Columns.Count - 4, ResultTable.Rows.Count - 1].Value = String.Format("{0:0.00}", 1.0 \* res.Count(t => t == 't') / res.Length \* 100);
659. ResultTable[ResultTable.Columns.Count - 3, ResultTable.Rows.Count - 1].Value = String.Format("{0:0}", 1.0 \* res.Count(t => t == 't') / res.Length \* 10);
660. ResultTable[ResultTable.Columns.Count - 2, ResultTable.Rows.Count - 1].Value = reader.GetDateTime(reader.GetOrdinal("Date"));
661. ResultTable[ResultTable.Columns.Count - 1, ResultTable.Rows.Count - 1].Value = reader.GetValue(reader.GetOrdinal("Time"));
662. }
663. }
664. }
665. }
666. ResultTable.AllowUserToAddRows = **false**;
667. }
668. }
669. }
670. **catch** (Exception)
671. {
672. MessageBox.Show("Соединение отсутствует. Попробуйте переподключиться к Интернету.");
673. }
674. **finally**
675. {
676. conn.Close();
677. conn.Dispose();
678. }
679. }
680. /// <summary>
681. /// Метод, для осуществления перехода от таблицы с результатами всех тестов к определённому с результатами попыток его решения
682. /// </summary>
683. **private** **void** ResultTable\_CellDoubleClick(**object** sender, DataGridViewCellEventArgs e)
684. {
685. **if** (e.ColumnIndex == 0)
686. {
687. **try**
688. {
689. **if** (ResultTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString().Split()[0] == "Тест")
690. ResultToTable(**int**.Parse(ResultTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString().Split()[1]) - 1);
691. }
692. **catch**
693. {
694. ResultToTable();
695. }
696. }
697. }
698. }
699. }
700. **Классы Test.cs**
     * 1. **Класс Test:**
701. **using** System.Xml.Serialization;
703. **namespace** Sets\_RelationsLessons
704. {
705. /// <summary>
706. /// Класс теста
707. /// </summary>
708. [XmlRoot("Test")]
709. **public** **class** Test
710. {
711. [XmlArray("Tasks")]
712. [XmlArrayItem("Task")]
713. **public** Task[] tasks;
714. [XmlElement("Id")]
715. **public** **int** id;
716. }
717. }
     * 1. **Класс Task:**
718. **using** System.IO;
719. **using** System.Xml.Serialization;
721. **namespace** Sets\_RelationsLessons
722. {
723. /// <summary>
724. /// Класс задания(вопроса)
725. /// </summary>
726. **public** **class** Task
727. {
728. [XmlElement("Graph")]
729. **public** **string** graph\_name;
730. [XmlElement("Condition")]
731. **public** **string** condition;
732. [XmlElement("Choice")]
733. **public** **int** choice\_ans;
734. [XmlArray("Answers")]
735. [XmlArrayItem("Ans")]
736. **public** **string**[] choices;
737. [XmlElement("Answer")]
738. **public** **string** answer;
739. /// <summary>
740. /// Получение графа из xml файла
741. /// </summary>
742. /// <param name="graph\_name">название файла</param>
743. /// <returns>граф</returns>
744. **public** Graph GetGraph(**string** graph\_name)
745. {
746. **if** (File.Exists(graph\_name))
747. {
748. XmlSerializer serializer = **new** XmlSerializer(**typeof**(Graph));
749. FileStream fs = **new** FileStream(graph\_name, FileMode.Open);
750. Graph graph = (Graph)serializer.Deserialize(fs);
751. fs.Close();
752. **return** graph;
753. }
754. **else** **return** **new** Graph();
755. }
756. }
757. }
758. **Класс TestForm.cs:**
759. **using** MySql.Data.MySqlClient;
760. **using** System;
761. **using** System.Collections.Generic;
762. **using** System.Data;
763. **using** System.Data.Common;
764. **using** System.Drawing;
765. **using** System.IO;
766. **using** System.Windows.Forms;
767. **using** System.Xml.Serialization;
769. **namespace** Sets\_RelationsLessons
770. {
772. **public** partial **class** TestForm : Form
773. {
774. Test test;
775. **string**[] answers;
776. **string** ans;
777. **bool** EdgeAuto = **true**;
778. MySqlConnection conn;
779. DateTime time = **new** DateTime(0,0);
781. **public** TestForm(**string** testname, **int** t = 0)
782. {
783. InitializeComponent();
784. автоToolStripMenuItem.BackColor = Color.Aqua;
785. conn = DBUtils.GetDBConnection();
786. DrawingGraph.Location = **new** Point(**this**.Width / 2 + 1, 39);
788. **try**
789. {
790. XmlSerializer serializer = **new** XmlSerializer(**typeof**(Test));
791. **using** (FileStream fs = **new** FileStream(testname, FileMode.Open))
792. {
793. test = (Test)serializer.Deserialize(fs);
794. }
795. **for** (**int** i = 0; i < test.tasks.Length; i++)
796. {
797. **if** (File.Exists(Directory.GetCurrentDirectory() + @"\Lessons\" + test.tasks[i].graph\_name)) test.tasks[i].graph\_name = Directory.GetCurrentDirectory() + @"\Lessons\" + test.tasks[i].graph\_name;
798. **else**
799. {
800. MessageBox.Show("Файл " + test.tasks[i].graph\_name + " не найден!");
801. test.tasks[i].graph\_name = "";
802. }
803. }
804. }
805. **catch**(Exception ex)
806. {
807. MessageBox.Show(ex.Message);
808. **this**.Close();
809. **return**;
810. }
812. answers = **new** **string**[test.tasks.Length];
813. **if** (t == 1) LoadAns();
815. **int** id = 0;
816. **try**
817. {
818. conn.Open();
820. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idUsers, Login from users", conn))
821. {
822. **using** (DbDataReader reader = cmd.ExecuteReader())
823. {
824. **if** (reader.HasRows)
825. {
826. **while** (reader.Read())
827. {
828. **if** (Properties.Settings.Default.login == reader.GetString(reader.GetOrdinal("Login"))) id = reader.GetInt32(reader.GetOrdinal("idUsers"));
829. }
830. }
831. }
832. }
834. **using** (MySqlCommand cmd = **new** MySqlCommand("Delete from files where idTest = @test and Users\_idUsers = @id", conn))
835. {
836. cmd.Parameters.Add(**new** MySqlParameter { ParameterName = "@id", MySqlDbType = MySqlDbType.Int32, Value = id });
837. cmd.Parameters.Add(**new** MySqlParameter { ParameterName = "@test", MySqlDbType = MySqlDbType.Int32, Value = test.id });
839. cmd.ExecuteNonQuery();
840. }
841. }
842. **catch** (Exception e)
843. {
844. MessageBox.Show(e.Message);
845. }
846. **finally**
847. {
848. conn.Close();
849. conn.Dispose();
850. }
852. timer.Start();
853. StartTask(0);
854. }
855. /// <summary>
856. /// Метод инициализации задания
857. /// </summary>
858. /// <param name="num">номер задания</param>
859. /// <param name="status">текущий статус</param>
860. **private** **void** StartTask(**int** num, **int** status = 0)
861. {
862. **if** (num == 0) previous.Enabled = **false**;
863. **else** previous.Enabled = **true**;
864. **if** (num == test.tasks.Length - 1) next.Enabled = **false**;
865. **else** next.Enabled = **true**;
866. Condition.Text = test.tasks[num].condition;
867. **if** (status == 0) **this**.Text = "Вопрос " + (num + 1) + " Время: " + time.ToLongTimeString();
868. **else** **this**.Text = "Вопрос " + (num + 1);
869. **if** (status == 1)
870. **if** (ans != **null** && ans[num] == 't')
871. {
872. **this**.Text += "  -  Верно";
873. **this**.BackColor = Color.Green;
874. }
875. **else**
876. {
877. **this**.Text += "  -  Неверно";
878. **this**.BackColor = Color.Red;
879. }
881. **try** {
882. **switch** (test.tasks[num].choice\_ans)
883. {
884. **case** 1:
885. oneGraph.Child = **null**;
886. **if** (test.tasks[num].graph\_name != "") oneGraph.Child = **new** GraphLayout(**ref** test, num, "oneGraph", EdgeAuto);
887. pictureBox.Visible = **false**;
888. oneGraph.Visible = **true**;
889. PictureGraph.Visible = **false**;
890. DrawingGraph.Visible = **false**;
891. ChooseOneAnsPanel.Visible = **true**;
892. ChooseSomeAnsPanel.Visible = **false**;
893. WriteAnsPanel.Visible = **false**;
894. ChooseOneAnsPanel.Items.Clear();
895. **foreach** (**string** ans **in** test.tasks[num].choices)
896. {
897. ChooseOneAnsPanel.Items.Add(ans);
898. }
899. **if** (status == 1)
900. {
901. **for** (**int** i = 0; i < ChooseOneAnsPanel.Items.Count; i++)
902. {
903. **if** (ChooseOneAnsPanel.GetItemChecked(i)) ChooseOneAnsPanel.SetItemChecked(i, **false**);
904. }
905. **for** (**int** i = 0; i < test.tasks[num].answer.Length; i++)
906. {
907. ChooseOneAnsPanel.SetItemChecked(**int**.Parse(test.tasks[num].answer[i].ToString()) - 1, **true**);
908. }
909. } **else** **if** (answers[num] != **null**)
910. {
911. **for** (**int** i = 0; i < answers[num].Length; i++)
912. {
913. ChooseOneAnsPanel.SetItemChecked(**int**.Parse(answers[num][i].ToString()) - 1, **true**);
914. }
915. }
916. **break**;
917. **case** 2:
918. oneGraph.Child = **null**;
919. **if** (test.tasks[num].graph\_name != "") oneGraph.Child = **new** GraphLayout(**ref** test, num, "oneGraph", EdgeAuto);
920. pictureBox.Visible = **false**;
921. oneGraph.Visible = **true**;
922. PictureGraph.Visible = **false**;
923. DrawingGraph.Visible = **false**;
924. ChooseOneAnsPanel.Visible = **false**;
925. ChooseSomeAnsPanel.Visible = **true**;
926. WriteAnsPanel.Visible = **false**;
927. ChooseSomeAnsPanel.Items.Clear();
928. **foreach** (**string** ans **in** test.tasks[num].choices)
929. {
930. ChooseSomeAnsPanel.Items.Add(ans);
931. }
932. **if** (status == 1)
933. {
934. **for** (**int** i = 0; i < ChooseSomeAnsPanel.Items.Count; i++)
935. {
936. **if** (ChooseSomeAnsPanel.GetItemChecked(i)) ChooseSomeAnsPanel.SetItemChecked(i, **false**);
937. }
938. **for** (**int** i = 0; i < test.tasks[num].answer.Length; i++)
939. {
940. ChooseSomeAnsPanel.SetItemChecked(**int**.Parse(test.tasks[num].answer[i].ToString()) - 1, **true**);
941. }
942. }
943. **else** **if** (answers[num] != **null**)
944. {
945. **for** (**int** i = 0; i < answers[num].Length; i++)
946. {
947. ChooseSomeAnsPanel.SetItemChecked(**int**.Parse(answers[num][i].ToString()) - 1, **true**);
948. }
949. }
950. **break**;
951. **case** 3:
952. oneGraph.Child = **null**;
953. **if** (test.tasks[num].graph\_name != "") oneGraph.Child = **new** GraphLayout(**ref** test, num, "oneGraph", EdgeAuto);
954. pictureBox.Visible = **false**;
955. oneGraph.Visible = **true**;
956. PictureGraph.Visible = **false**;
957. DrawingGraph.Visible = **false**;
958. ChooseOneAnsPanel.Visible = **false**;
959. ChooseSomeAnsPanel.Visible = **false**;
960. WriteAnsPanel.Visible = **true**;
961. WriteAnsPanel.Clear();
962. WriteAnsPanel.Text = answers[num] ?? "";
963. **if** (status == 1) WriteAnsPanel.Text = test.tasks[num].answer;
964. **break**;
965. **case** 4:
966. DrawingGraph.Child = **null**;
967. **if** (test.tasks[num].graph\_name != "")
968. {
969. **if** (status == 1)
970. DrawingGraph.Child = **new** GraphLayout(**ref** test, num, "true", EdgeAuto);
971. **else** **if** (answers[num] != **null**)
972. DrawingGraph.Child = **new** GraphLayout(**ref** test, num, "ans", EdgeAuto);
973. **else**
974. DrawingGraph.Child = **new** GraphLayout(**ref** test, num, "DrawingGraph", EdgeAuto);
975. PictureGraph.Child = **new** GraphLayout(**ref** test, num, "PictureGraph", EdgeAuto);
976. }
977. pictureBox.Visible = **false**;
978. oneGraph.Visible = **false**;
979. PictureGraph.Visible = **true**;
980. DrawingGraph.Visible = **true**;
981. ChooseOneAnsPanel.Visible = **false**;
982. ChooseSomeAnsPanel.Visible = **false**;
983. WriteAnsPanel.Visible = **false**;
984. **break**;
985. **case** 5:
986. **if** (test.tasks[num].graph\_name != "") pictureBox.Image = Image.FromFile(test.tasks[num].graph\_name);
987. pictureBox.Visible = **true**;
988. oneGraph.Visible = **false**;
989. PictureGraph.Visible = **false**;
990. DrawingGraph.Visible = **false**;
991. ChooseOneAnsPanel.Visible = **true**;
992. ChooseSomeAnsPanel.Visible = **false**;
993. WriteAnsPanel.Visible = **false**;
994. ChooseOneAnsPanel.Items.Clear();
995. **foreach** (**string** ans **in** test.tasks[num].choices)
996. {
997. ChooseOneAnsPanel.Items.Add(ans);
998. }
999. **if** (status == 1)
1000. {
1001. **for** (**int** i = 0; i < ChooseOneAnsPanel.Items.Count; i++)
1002. {
1003. **if** (ChooseOneAnsPanel.GetItemChecked(i)) ChooseOneAnsPanel.SetItemChecked(i, **false**);
1004. }
1005. **for** (**int** i = 0; i < test.tasks[num].answer.Length; i++)
1006. {
1007. ChooseOneAnsPanel.SetItemChecked(**int**.Parse(test.tasks[num].answer[i].ToString()) - 1, **true**);
1008. }
1009. } **else** **if** (answers[num] != **null**)
1010. {
1011. **for** (**int** i = 0; i < answers[num].Length; i++)
1012. {
1013. ChooseOneAnsPanel.SetItemChecked(**int**.Parse(answers[num][i].ToString()) - 1, **true**);
1014. }
1015. }
1016. **break**;
1017. **case** 6:
1018. **if** (test.tasks[num].graph\_name != "") pictureBox.Image = Image.FromFile(test.tasks[num].graph\_name);
1019. pictureBox.Visible = **true**;
1020. oneGraph.Visible = **false**;
1021. PictureGraph.Visible = **false**;
1022. DrawingGraph.Visible = **false**;
1023. ChooseOneAnsPanel.Visible = **false**;
1024. ChooseSomeAnsPanel.Visible = **true**;
1025. WriteAnsPanel.Visible = **false**;
1026. ChooseSomeAnsPanel.Items.Clear();
1027. **foreach** (**string** ans **in** test.tasks[num].choices)
1028. {
1029. ChooseSomeAnsPanel.Items.Add(ans);
1030. }
1031. **if** (status == 1)
1032. {
1033. **for** (**int** i = 0; i < ChooseSomeAnsPanel.Items.Count; i++)
1034. {
1035. **if** (ChooseSomeAnsPanel.GetItemChecked(i)) ChooseSomeAnsPanel.SetItemChecked(i, **false**);
1036. }
1037. **for** (**int** i = 0; i < test.tasks[num].answer.Length; i++)
1038. {
1039. ChooseSomeAnsPanel.SetItemChecked(**int**.Parse(test.tasks[num].answer[i].ToString()) - 1, **true**);
1040. }
1041. }
1042. **else** **if** (answers[num] != **null**)
1043. {
1044. **for** (**int** i = 0; i < answers[num].Length; i++)
1045. {
1046. ChooseSomeAnsPanel.SetItemChecked(**int**.Parse(answers[num][i].ToString()) - 1, **true**);
1047. }
1048. }
1049. **break**;
1050. **case** 7:
1051. **if** (test.tasks[num].graph\_name != "") pictureBox.Image = Image.FromFile(test.tasks[num].graph\_name);
1052. pictureBox.Visible = **true**;
1053. oneGraph.Visible = **false**;
1054. PictureGraph.Visible = **false**;
1055. DrawingGraph.Visible = **false**;
1056. ChooseOneAnsPanel.Visible = **false**;
1057. ChooseSomeAnsPanel.Visible = **false**;
1058. WriteAnsPanel.Visible = **true**;
1059. WriteAnsPanel.Clear();
1060. WriteAnsPanel.Text = answers[num] ?? "";
1061. **if** (status == 1) WriteAnsPanel.Text = test.tasks[num].answer;
1062. **break**;
1063. **default**:
1064. **if** (test.tasks[num].graph\_name != "")
1065. {
1066. **if** (status == 1)
1067. oneGraph.Child = **new** GraphLayout(**ref** test, num, "true", EdgeAuto);
1068. **else** **if** (answers[num] != **null**)
1069. oneGraph.Child = **new** GraphLayout(**ref** test, num, "ans", EdgeAuto);
1070. **else**
1071. oneGraph.Child = **new** GraphLayout(**ref** test, num, "oneGraph", EdgeAuto);
1072. }
1073. pictureBox.Visible = **false**;
1074. ChooseOneAnsPanel.Visible = **false**;
1075. ChooseSomeAnsPanel.Visible = **false**;
1076. WriteAnsPanel.Visible = **false**;
1077. oneGraph.Visible = **true**;
1078. PictureGraph.Visible = **false**;
1079. DrawingGraph.Visible = **false**;
1080. **break**;
1081. }
1082. }
1083. **catch** (Exception ex)
1084. {
1085. MessageBox.Show(ex.Message);
1086. }
1087. }
1088. /// <summary>
1089. /// Обработчик события нажатия кнопки "Сохранить результат"
1090. /// </summary>
1091. **private** **void** SaveAns\_Click(**object** sender, EventArgs e)
1092. {
1093. timer.Stop();
1094. SaveThisAnswer();
1095. **int** id = 0;
1096. **try**
1097. {
1098. conn.Open();
1100. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idUsers, Login from users", conn))
1101. {
1102. **using** (DbDataReader reader = cmd.ExecuteReader())
1103. {
1104. **if** (reader.HasRows)
1105. {
1106. **while** (reader.Read())
1107. {
1108. **if** (Properties.Settings.Default.login == reader.GetString(reader.GetOrdinal("Login"))) id = reader.GetInt32(reader.GetOrdinal("idUsers"));
1109. }
1110. }
1111. }
1112. }
1114. **for** (**int** i = 0; i < answers.Length; i++)
1115. {
1116. **if** (answers[i] != **null**)
1117. {
1118. **using** (MySqlCommand cmd = **new** MySqlCommand("Insert into files(idTest, idTask, File, Users\_idUsers) " + " values(@idTest, @idTask, @file, @id)", conn))
1119. {
1120. cmd.Parameters.Add(**new** MySqlParameter("@idTest", SqlDbType.Int));
1121. cmd.Parameters["@idTest"].Value = test.id;
1122. cmd.Parameters.Add(**new** MySqlParameter("@idTask", SqlDbType.Int));
1123. cmd.Parameters["@idTask"].Value = i;
1124. cmd.Parameters.Add(**new** MySqlParameter("@file", MySqlDbType.LongText));
1125. **if** (answers[i] == (@"Lessons\tmpGraph" + i))
1126. {
1127. FileInfo file = **new** FileInfo(Directory.GetCurrentDirectory() + @"\Lessons\tmpGraph" + i);
1128. **if** (file.Length <= 1050000)
1129. **using** (StreamReader sr = **new** StreamReader(**new** FileStream(Directory.GetCurrentDirectory() + @"\Lessons\tmpGraph" + i, FileMode.Open, FileAccess.Read)))
1130. {
1131. cmd.Parameters["@file"].Value = sr.ReadToEnd();
1132. }
1133. File.Delete(Directory.GetCurrentDirectory() + @"\Lessons\tmpGraph" + i);
1134. }
1135. **else**
1136. {
1137. cmd.Parameters["@file"].Value = answers[i];
1138. }
1139. cmd.Parameters.Add(**new** MySqlParameter("@id", SqlDbType.Int));
1140. cmd.Parameters["@id"].Value = id;
1142. cmd.ExecuteNonQuery();
1143. }
1144. }
1145. }
1146. }
1147. **catch** (Exception ex)
1148. {
1149. MessageBox.Show(ex.Message);
1150. }
1151. **finally**
1152. {
1153. conn.Close();
1154. conn.Dispose();
1155. }
1156. }
1157. /// <summary>
1158. /// Загружает ответы пользователя из базы данных
1159. /// </summary>
1160. **public** **void** LoadAns()
1161. {
1162. **int** id = 0;
1163. **try**
1164. {
1165. conn.Open();
1167. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idUsers, Login from users", conn))
1168. {
1169. **using** (DbDataReader reader = cmd.ExecuteReader())
1170. {
1171. **if** (reader.HasRows)
1172. {
1173. **while** (reader.Read())
1174. {
1175. **if** (Properties.Settings.Default.login == reader.GetString(reader.GetOrdinal("Login"))) id = reader.GetInt32(reader.GetOrdinal("idUsers"));
1176. }
1177. }
1178. }
1179. }
1181. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idTest, idTask, File, Users\_idUsers from files", conn))
1182. {
1183. **using** (DbDataReader reader = cmd.ExecuteReader())
1184. {
1185. **if** (reader.HasRows)
1186. {
1187. **while** (reader.Read())
1188. {
1189. **if** (id == reader.GetInt32(reader.GetOrdinal("Users\_idUsers")) && test.id == reader.GetInt32(reader.GetOrdinal("idTest")))
1190. **if** (test.tasks[reader.GetInt32(reader.GetOrdinal("idTask"))].choice\_ans % 4 != 0) answers[reader.GetInt32(reader.GetOrdinal("idTask"))] = reader.GetString(reader.GetOrdinal("File"));
1191. **else**
1192. {
1193. answers[reader.GetInt32(reader.GetOrdinal("idTask"))] = @"Lessons\tmpGraph" + reader.GetInt32(reader.GetOrdinal("idTask"));
1194. **using** (FileStream fs = **new** FileStream(Directory.GetCurrentDirectory() + @"\Lessons\tmpGraph" + reader.GetInt32(reader.GetOrdinal("idTask")), FileMode.Create, FileAccess.Write))
1195. {
1196. **using** (StreamWriter sw = **new** StreamWriter(fs))
1197. {
1198. sw.WriteLine(reader.GetString(reader.GetOrdinal("File")));
1199. }
1200. }
1201. }
1202. }
1203. }
1204. }
1205. }
1207. }
1208. **catch** (Exception ex)
1209. {
1210. MessageBox.Show(ex.Message);
1211. }
1212. **finally**
1213. {
1214. conn.Close();
1215. conn.Dispose();
1216. }
1217. }
1218. /// <summary>
1219. /// Обработчик события нажатия кнопки ">" (следующий вопрос)
1220. /// </summary>
1221. **private** **void** next\_Click(**object** sender, EventArgs e)
1222. {
1223. **if** (ShowAns.Enabled == **true**)
1224. {
1225. **if** (ShowAns.Visible == **false**) SaveThisAnswer();
1226. StartTask(**int**.Parse(**this**.Text.Split()[1]));
1227. }
1228. **else**
1229. StartTask(**int**.Parse(**this**.Text.Split()[1]), 1);
1230. }
1231. /// <summary>
1232. /// Обработчик события нажатия кнопки "<" (предыдущий вопрос)
1233. /// </summary>
1234. **private** **void** previous\_Click(**object** sender, EventArgs e)
1235. {
1236. **if** (ShowAns.Enabled == **true**)
1237. {
1238. **if** (ShowAns.Visible == **false**) SaveThisAnswer();
1239. StartTask(**int**.Parse(**this**.Text.Split()[1]) - 2);
1240. }
1241. **else**
1242. StartTask(**int**.Parse(**this**.Text.Split()[1]) - 2, 1);
1243. }
1244. /// <summary>
1245. /// Обработчик события нажатия кнопки "Ответить"
1246. /// </summary>
1247. **private** **void** answer\_Click(**object** sender, EventArgs e)
1248. {
1249. timer.Stop();
1250. SaveThisAnswer();
1251. ans = "";
1252. Answer.Visible = **false**;
1253. SaveAns.Visible = **false**;
1254. Repeat.Visible = **true**;
1255. ShowAns.Visible = **true**;
1256. ChooseSomeAnsPanel.Enabled = **false**;
1257. ChooseOneAnsPanel.Enabled = **false**;
1258. WriteAnsPanel.Enabled = **false**;
1259. oneGraph.Enabled = **false**;
1260. DrawingGraph.Enabled = **false**;
1261. **for** (**int** i = 0; i < answers.Length; i++)
1262. {
1263. **if** (answers[i] == test.tasks[i].answer) ans += "t";
1264. **else** **if** (answers[i] == (@"Lessons\tmpGraph" + i))
1265. {
1266. **if** (Compare(test.tasks[i].GetGraph(Directory.GetCurrentDirectory() + @"\Lessons\" + test.tasks[i].answer), test.tasks[i].GetGraph(Directory.GetCurrentDirectory() + @"\Lessons\tmpGraph" + i)))
1267. ans += "t";
1268. **else** ans += "f";
1269. }
1270. **else** ans += "f";
1271. }
1273. **int** id = 0;
1274. **try**
1275. {
1276. conn.Open();
1278. **using** (MySqlCommand cmd = **new** MySqlCommand("Select idUsers, Login from users", conn))
1279. {
1280. **using** (DbDataReader reader = cmd.ExecuteReader())
1281. {
1282. **if** (reader.HasRows)
1283. {
1284. **while** (reader.Read())
1285. {
1286. **if** (Properties.Settings.Default.login == reader.GetString(reader.GetOrdinal("Login"))) id = reader.GetInt32(reader.GetOrdinal("idUsers"));
1287. }
1288. }
1289. }
1290. }
1291. **using** (MySqlCommand cmd = **new** MySqlCommand("Insert into tests(idTests, Result, Users\_idUsers, Date, Time) " + " values(@idTests, @result, @id, @date, @time)", conn))
1292. {
1293. cmd.Parameters.Add(**new** MySqlParameter("@idTests", SqlDbType.Int));
1294. cmd.Parameters["@idTests"].Value = test.id;
1295. cmd.Parameters.Add(**new** MySqlParameter("@result", MySqlDbType.VarChar));
1296. cmd.Parameters["@result"].Value = ans;
1297. cmd.Parameters.Add(**new** MySqlParameter("@id", SqlDbType.Int));
1298. cmd.Parameters["@id"].Value = id;
1299. cmd.Parameters.Add(**new** MySqlParameter("@date", MySqlDbType.DateTime));
1300. cmd.Parameters["@date"].Value = DateTime.Now;
1301. cmd.Parameters.Add(**new** MySqlParameter("@time", MySqlDbType.Time));
1302. cmd.Parameters["@time"].Value = time.TimeOfDay;
1304. cmd.ExecuteNonQuery();
1305. }
1306. }
1307. **catch** (Exception ex)
1308. {
1309. MessageBox.Show(ex.Message);
1310. MessageBox.Show("Результат не сохранен!");
1311. }
1312. **finally**
1313. {
1314. conn.Close();
1315. conn.Dispose();
1316. }
1317. }
1318. /// <summary>
1319. /// Очистка выбранных пунктов при выборе нового
1320. /// </summary>
1321. **private** **void** ChooseOneAnsPanel\_SelectedIndexChanged(**object** sender, EventArgs e)
1322. {
1323. **for** (**int** i = 0; i < ChooseOneAnsPanel.Items.Count; i++)
1324. **if** (i != ChooseOneAnsPanel.SelectedIndex) ChooseOneAnsPanel.SetItemChecked(i, **false**);
1325. }
1326. /// <summary>
1327. /// Сравнение двух графов
1328. /// </summary>
1329. /// <param name="x">первый граф</param>
1330. /// <param name="y">второй граф</param>
1331. /// <returns>равны ли графы</returns>
1332. **private** **bool** Compare(Graph x, Graph y)
1333. {
1334. **if** (x.edges.Count != y.edges.Count) **return** **false**;
1335. **else**
1336. {
1337. **bool** f;
1338. **int** j;
1339. List<**int**> ed = **new** List<**int**> { };
1340. **for** (**int** i = 0; i < x.edges.Count; i++)
1341. {
1342. f = **false**;
1343. j = 0;
1344. **while** (!f & j < y.edges.Count)
1345. {
1346. **if** (!ed.Contains(j) && x.edges[i].OId.Id == y.edges[j].OId.Id &&
1347. x.edges[i].DId.Id == y.edges[j].DId.Id) f = **true**;
1348. **else** j++;
1349. }
1350. **if** (j == y.edges.Count) **return** **false**;
1351. ed.Add(j);
1352. }
1353. **if** (ed.Count == x.edges.Count) **return** **true**;
1354. **else** **return** **false**;
1355. }
1356. }
1357. /// <summary>
1358. /// Сохраняет ответ на текущий вопрос
1359. /// </summary>
1360. **public** **void** SaveThisAnswer()
1361. {
1362. **int** num = **int**.Parse(**this**.Text.Split()[1]) - 1;
1363. **switch** (test.tasks[num].choice\_ans)
1364. {
1365. **case** 1:
1366. **string** ans = "";
1367. **foreach** (**int** i **in** ChooseOneAnsPanel.CheckedIndices) { ans += (i + 1); }
1368. **if** (ChooseOneAnsPanel.CheckedItems != **null**) answers[num] = ans;
1369. **break**;
1370. **case** 2:
1371. ans = "";
1372. **foreach** (**int** i **in** ChooseSomeAnsPanel.CheckedIndices) { ans += (i + 1); }
1373. **if** (ChooseSomeAnsPanel.CheckedItems != **null**) answers[num] = ans;
1374. **break**;
1375. **case** 3:
1376. **if** (WriteAnsPanel.Text != **null**) answers[num] = WriteAnsPanel.Text;
1377. **break**;
1378. **case** 4:
1379. DrawingGraph.Enabled = **false**;
1380. DrawingGraph.Enabled = **true**;
1381. answers[num] = @"Lessons\tmpGraph" + num;
1382. **break**;
1383. **case** 5:
1384. ans = "";
1385. **foreach** (**int** i **in** ChooseOneAnsPanel.CheckedIndices) { ans += (i + 1); }
1386. **if** (ChooseOneAnsPanel.CheckedItems != **null**) answers[num] = ans;
1387. **break**;
1388. **case** 6:
1389. ans = "";
1390. **foreach** (**int** i **in** ChooseSomeAnsPanel.CheckedIndices) { ans += (i + 1); }
1391. **if** (ChooseSomeAnsPanel.CheckedItems != **null**) answers[num] = ans;
1392. **break**;
1393. **case** 7:
1394. **if** (WriteAnsPanel.Text != **null**) answers[num] = WriteAnsPanel.Text;
1395. **break**;
1396. **default**:
1398. oneGraph.Enabled = **false**;
1399. oneGraph.Enabled = **true**;
1400. answers[num] = @"Lessons\tmpGraph" + num;
1401. **break**;
1402. }
1403. }
1404. /// <summary>
1405. /// Обработчик события нажатия кнопки "Повторить тест"
1406. /// </summary>
1407. **private** **void** Repeat\_Click(**object** sender, EventArgs e)
1408. {
1409. **this**.BackColor = DefaultBackColor;
1410. ChooseSomeAnsPanel.Enabled = **true**;
1411. ChooseOneAnsPanel.Enabled = **true**;
1412. WriteAnsPanel.Enabled = **true**;
1413. oneGraph.Enabled = **true**;
1414. DrawingGraph.Enabled = **true**;
1415. Repeat.Visible = **false**;
1416. ShowAns.Visible = **false**;
1417. answers = **new** **string**[test.tasks.Length];
1418. Answer.Visible = **true**;
1419. SaveAns.Visible = **true**;
1420. time = **new** DateTime(0, 0);
1421. timer.Start();
1422. StartTask(0);
1423. }
1424. /// <summary>
1425. /// Обработчик события нажатия кнопки "Показать ответы"
1426. /// </summary>
1427. **private** **void** ShowAns\_Click(**object** sender, EventArgs e)
1428. {
1429. ShowAns.Enabled = **false**;
1430. StartTask(0, 1);
1431. }
1432. /// <summary>
1433. /// Обработчик события закрытия формы
1434. /// </summary>
1435. **private** **void** TestForm\_FormClosing(**object** sender, FormClosingEventArgs e)
1436. {
1437. **if** (e.CloseReason.ToString() == "UserClosing")
1438. {
1439. DialogResult dialogResult = MessageBox.Show("Все несохраненные ответы не будут учтены! Вы точно хотите прервать тест?", "Выход", MessageBoxButtons.OKCancel);
1440. **if** (dialogResult == DialogResult.OK)
1441. {
1442. Application.Restart();
1443. }
1444. **else**
1445. {
1446. e.Cancel = **true**;
1447. }
1448. }
1449. }
1450. /// <summary>
1451. /// Обработчик события изменения размера формы
1452. /// </summary>
1453. **private** **void** TestForm\_SizeChanged(**object** sender, EventArgs e)
1454. {
1455. DrawingGraph.Location = **new** Point(**this**.Width / 2 + 1, 39);
1456. DrawingGraph.Size = **new** Size(5 \* **this**.Width / 14, **this**.Height / 2 - 20);
1457. PictureGraph.Size = **new** Size(5 \* **this**.Width / 14, **this**.Height / 2 - 20);
1458. }
1459. /// <summary>
1460. /// Такт таймера
1461. /// </summary>
1462. **private** **void** timer\_Tick(**object** sender, EventArgs e)
1463. {
1464. time = time.AddSeconds(1);
1465. **this**.Text = **this**.Text.Substring(0, **this**.Text.Length - 7) + time.ToLongTimeString();
1466. }
1467. /// <summary>
1468. /// Открывает окно помощи по тесту
1469. /// </summary>
1470. **private** **void** помощьToolStripMenuItem\_Click(**object** sender, EventArgs e)
1471. {
1472. timer.Stop();
1473. webBrowser.Navigate(Directory.GetCurrentDirectory() + @"\Lessons\help.html");
1474. webBrowser.Visible = **true**;
1475. вернутьсяКТестуToolStripMenuItem.Visible = **true**;
1476. помощьToolStripMenuItem.Visible = **false**;
1477. форматРёберToolStripMenuItem.Visible = **false**;
1478. next.Visible = **false**;
1479. previous.Visible = **false**;
1480. }
1481. /// <summary>
1482. /// Возобновляет тест
1483. /// </summary
1484. **private** **void** вернутьсяКТестуToolStripMenuItem\_Click(**object** sender, EventArgs e)
1485. {
1486. webBrowser.Visible = **false**;
1487. вернутьсяКТестуToolStripMenuItem.Visible = **false**;
1488. помощьToolStripMenuItem.Visible = **true**;
1489. форматРёберToolStripMenuItem.Visible = **true**;
1490. next.Visible = **true**;
1491. previous.Visible = **true**;
1492. timer.Start();
1493. }
1494. /// <summary>
1495. /// Делает ребра графа прямыми
1496. /// </summary>
1497. **private** **void** прямыеToolStripMenuItem\_Click(**object** sender, EventArgs e)
1498. {
1499. EdgeAuto = **false**;
1500. автоToolStripMenuItem.BackColor = прямыеToolStripMenuItem.BackColor;
1501. прямыеToolStripMenuItem.BackColor = Color.Aqua;
1502. }
1503. /// <summary>
1504. /// Делает ребра графа автовыстраиваемыми
1505. /// </summary>
1506. **private** **void** автоToolStripMenuItem\_Click(**object** sender, EventArgs e)
1507. {
1508. EdgeAuto = **true**;
1509. прямыеToolStripMenuItem.BackColor = автоToolStripMenuItem.BackColor;
1510. автоToolStripMenuItem.BackColor = Color.Aqua;
1511. }
1512. }
1513. }

**2. ТЕКСТ СКРИПТА БАЗЫ ДАННЫХ MySQL**

1. **CREATE** **DATABASE**  IF NOT EXISTS `mydb` /\*!40100 **DEFAULT** **CHARACTER** **SET** utf8 \*/;
2. USE `mydb`;
3. -- MySQL dump 10.13  Distrib 5.7.17, for Win64 (x86\_64)
4. --
5. -- Host: 127.0.0.1    Database: mydb
6. -- ------------------------------------------------------
7. -- Server version   5.6.25-log
9. /\*!40101 **SET** @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;
10. /\*!40101 **SET** @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;
11. /\*!40101 **SET** @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;
12. /\*!40101 **SET** NAMES utf8 \*/;
13. /\*!40103 **SET** @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;
14. /\*!40103 **SET** TIME\_ZONE='+00:00' \*/;
15. /\*!40014 **SET** @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;
16. /\*!40014 **SET** @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;
17. /\*!40101 **SET** @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;
18. /\*!40111 **SET** @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;
20. --
21. -- Table structure for table `files`
22. --
24. **DROP** **TABLE** IF EXISTS `files`;
25. /\*!40101 **SET** @saved\_cs\_client     = @@character\_set\_client \*/;
26. /\*!40101 **SET** character\_set\_client = utf8 \*/;
27. **CREATE** **TABLE** `files` (
28. `idTest` **int**(2) NOT NULL,
29. `idTask` **int**(2) NOT NULL,
30. `File` longtext NOT NULL,
31. `Users\_idUsers` **int**(11) NOT NULL,
32. **KEY** `fk\_Files\_Users1\_idx` (`Users\_idUsers`),
33. **CONSTRAINT** `fk\_Files\_Users1` **FOREIGN** **KEY** (`Users\_idUsers`) **REFERENCES** `users` (`idUsers`) **ON** **DELETE** **NO** **ACTION** **ON** **UPDATE** **NO** **ACTION**
34. ) ENGINE=InnoDB **DEFAULT** CHARSET=utf8;
35. /\*!40101 **SET** character\_set\_client = @saved\_cs\_client \*/;
37. --
38. -- Dumping data for table `files`
39. --
41. LOCK TABLES `files` WRITE;
42. /\*!40000 **ALTER** **TABLE** `files` DISABLE KEYS \*/;
43. /\*!40000 **ALTER** **TABLE** `files` ENABLE KEYS \*/;
44. UNLOCK TABLES;
46. --
47. -- Table structure for table `tests`
48. --
50. **DROP** **TABLE** IF EXISTS `tests`;
51. /\*!40101 **SET** @saved\_cs\_client     = @@character\_set\_client \*/;
52. /\*!40101 **SET** character\_set\_client = utf8 \*/;
53. **CREATE** **TABLE** `tests` (
54. `idTests` **int**(2) NOT NULL,
55. `Result` **varchar**(20) **DEFAULT** NULL,
56. `Users\_idUsers` **int**(11) NOT NULL,
57. `**Date**` datetime NOT NULL,
58. `**Time**` **time**(6) NOT NULL,
59. **KEY** `fk\_Tests\_Users\_idx` (`Users\_idUsers`),
60. **CONSTRAINT** `fk\_Tests\_Users` **FOREIGN** **KEY** (`Users\_idUsers`) **REFERENCES** `users` (`idUsers`) **ON** **DELETE** **NO** **ACTION** **ON** **UPDATE** **NO** **ACTION**
61. ) ENGINE=InnoDB **DEFAULT** CHARSET=utf8;
62. /\*!40101 **SET** character\_set\_client = @saved\_cs\_client \*/;
64. --
65. -- Dumping data for table `tests`
66. --
68. LOCK TABLES `tests` WRITE;
69. /\*!40000 **ALTER** **TABLE** `tests` DISABLE KEYS \*/;
70. /\*!40000 **ALTER** **TABLE** `tests` ENABLE KEYS \*/;
71. UNLOCK TABLES;
73. --
74. -- Table structure for table `users`
75. --
77. **DROP** **TABLE** IF EXISTS `users`;
78. /\*!40101 **SET** @saved\_cs\_client     = @@character\_set\_client \*/;
79. /\*!40101 **SET** character\_set\_client = utf8 \*/;
80. **CREATE** **TABLE** `users` (
81. `idUsers` **int**(11) NOT NULL AUTO\_INCREMENT,
82. `Login` **varchar**(15) NOT NULL,
83. `**Password**` **int**(11) NOT NULL,
84. **PRIMARY** **KEY** (`idUsers`)
85. ) ENGINE=InnoDB AUTO\_INCREMENT=4 **DEFAULT** CHARSET=utf8;
86. /\*!40101 **SET** character\_set\_client = @saved\_cs\_client \*/;
88. /\*!40101 **SET** SQL\_MODE=@OLD\_SQL\_MODE \*/;
89. /\*!40014 **SET** FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;
90. /\*!40014 **SET** UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;
91. /\*!40101 **SET** CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;
92. /\*!40101 **SET** CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;
93. /\*!40101 **SET** COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;
94. /\*!40111 **SET** SQL\_NOTES=@OLD\_SQL\_NOTES \*/;
96. -- Dump completed on 2018-05-13 13:08:14

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. ГОСТ 19.201-78. Техническое задание. Требование к содержанию и оформлению. // Единая система программной документации. – Москва: Издательство стандартов, 2005.
2. ГОСТ 19.404-79. Пояснительная записка. // Единая система программной документации. – Москва: Издательство стандартов, 2005.
3. ГОСТ 19.301-78. Программа и методика испытаний. // Единая система программной документации. – Москва: Издательство стандартов, 2005.
4. ГОСТ 19.505-79. Руководство оператора. // Единая система программной документации. – Москва: Издательство стандартов, 2005.
5. ГОСТ 19.401-78. Текст программы. // Единая система программной документации. – Москва: Издательство стандартов, 2005.
6. ГОСТ 19.106-78. Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – Москва: Издательство стандартов, 2005.
7. ГОСТ 19.101-77. Виды программ и программных документов. // Единая система программной документации. – Москва: Издательство стандартов, 2005.

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |