**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФГАОУ ВО «Пермский государственный национальный исследовательский университет»**

Физический факультет

Кафедра радиоэлектроники и защиты информации

Компонентное программирование

Лабораторная работа №4 на тему   
**«Разработка настольного приложения»**

Выполнил:

студент 4 курса физического факультета

группа ФЗ/О ФЗ-11-2020 НБ

направление «01.03.02 Прикладная математика и информатика»

профиль «Инженерия программного обеспечения»

Кириллов Данил Алексеевич

Пермь, 2023 г.

Оглавление

[1. Цель работы 3](#_Toc148535349)

[2. Постановка задач 3](#_Toc148535350)

[3. Реализация 4](#_Toc148535351)

[3.1 Общее описание приложения 4](#_Toc148535352)

[3.2 Модель данных 5](#_Toc148535353)

[3.3 Компоненты 7](#_Toc148535354)

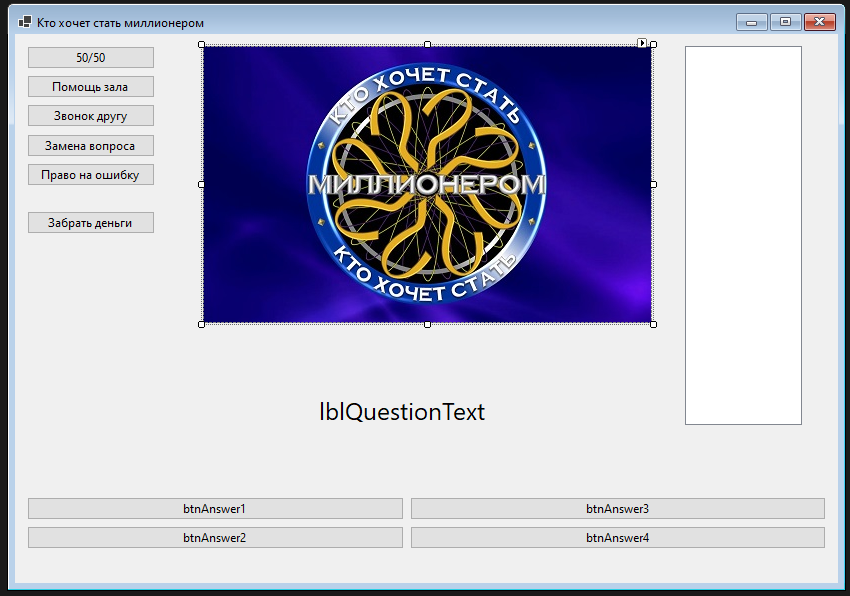
[4. Заключение 8](#_Toc148535355)

1. Цель работы

Познакомиться с разработкой настольного приложения и взаимодействию с СУБД.

1. Постановка задач

Реализовать настольное приложение по мотивам телевизионного шоу «Кто хочет стать миллионером?».

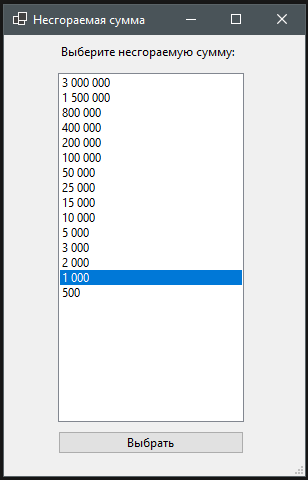


1. Реализация

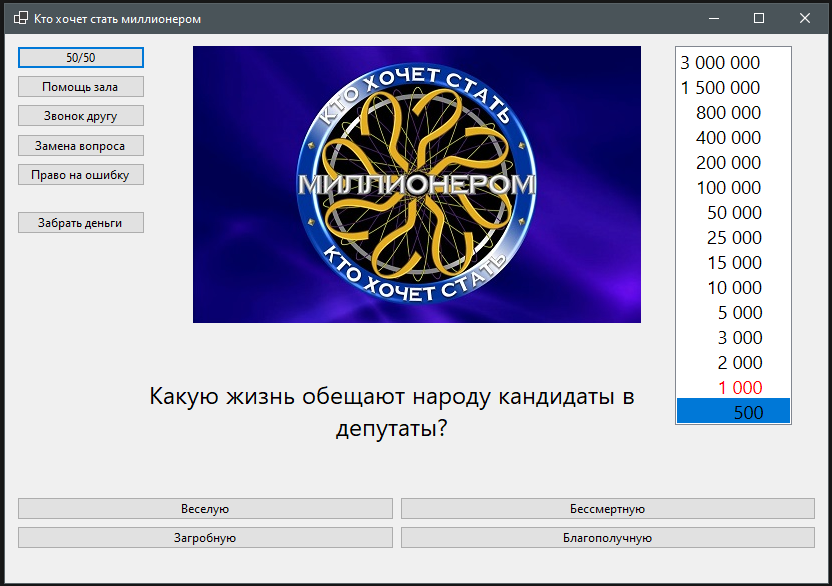
В методичке к лабораторной работе настольное приложение разрабатывалось на языке программирования Java, а для хранения данных была использована встраиваемая СУБД SQLite.  
В моем случае приложение написано на языке программирования C#, а для хранения вопросов использована СУБД Postgres.

## 3.1 Общее описание приложения

При запуске приложения нас встречает окно выбора «несгораемой суммы»



После этого открывается основное окно приложения, и игра начинается.



Как и в оригинальном телевизионном шоу, в разработанном аналоге есть 5 подсказок:

1. «Помощь зала» – каждый зритель в студии голосует за правильный, на его взгляд, ответ, а игроку предоставляется статистика голосования.
2. «50 на 50» – компьютер убирает два неправильных ответа.
3. «Звонок другу» – в течение 30 секунд игрок может посоветоваться с одним из пяти друзей, заявленных заранее.
4. «Право на ошибку» – Игрок может дать два ответа на заданный вопрос. Если первый ответ игрока оказался правильным, подсказка всё равно считается использованной.
5. «Замена вопроса» – игрок может заменить вопрос на другой того же уровня и той же стоимости.

Также, в любой момент пользователь может забрать деньги и закончить игру.

## 3.2 Модель данных

Для удобного взаимодействия с данными был разработан класс Question.

|  |
| --- |
| public class Question  {  public string Text { get; set; }  public string Answer1 { get; set; }  public string Answer2 { get; set; }  public string Answer3 { get; set; }  public string Answer4 { get; set; }  public string RightAnswer { get; set; }  public int Level { get; set; }  public Question(string Text, string Answer1, string Answer2, string Answer3, string Answer4, string RightAnswer, int Level)  {  this.Text = Text;  this.Answer1 = Answer1;  this.Answer2 = Answer2;  this.Answer3 = Answer3;  this.Answer4 = Answer4;  this.RightAnswer = RightAnswer;  this.Level = Level;  }  } |
| *Листинг 1* – Класс Question для удобной работы с вопросами |

Для хранения вопросов была создана база данных who\_wants\_to\_be\_a\_millionaire с одной таблицей questions.

|  |
| --- |
|  |
| *Приложение 1* – Таблица questions в БД |

Для чтения вопросов из базы данных был реализован метод, который тянет все вопросы из таблицы с вопросами.

|  |
| --- |
| private void ReadQuestionsFromDatabase()  {  string connectionString = "Host=localhost;Port=5432;Database=who\_wants\_to\_be\_a\_millionaire;Username=postgres;Password=post!2439!gres;";  using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))  {  connection.Open();  using (NpgsqlCommand qsql = new NpgsqlCommand("SELECT \* FROM Questions", connection))  {  using (NpgsqlDataReader reader = qsql.ExecuteReader())  {  while (reader.Read())  {  string questionText = reader["QuestionText"].ToString();  string answer1 = reader["Answer1"].ToString();  string answer2 = reader["Answer2"].ToString();  string answer3 = reader["Answer3"].ToString();  string answer4 = reader["Answer4"].ToString();  string correctAnswerIndex = reader["CorrectAnswerIndex"].ToString();  int weight = Convert.ToInt32(reader["Weight"]);  // Создаем объект Question и добавляем его в коллекцию  Question question = new Question(questionText, answer1, answer2, answer3, answer4, correctAnswerIndex, weight);  questions.Add(question);  }  }  }  }  } |
| *Листинг 2* – Метод для чтения вопроссов из бд |

## 3.3 Компоненты

Во время разработки настольного приложения, помимо использования стандартных компонентов, был реализован элемент управления, наследуемый от класса Form, для вывода оставшегося времени при выборе подсказки «звонок другу».

|  |
| --- |
|  |
| *Приложение 2* – Окно при выборе подсказки «звонок другу» |

|  |
| --- |
| public partial class CustomMessageBox : Form  {  private int countdown = 30;  private Timer timer;  public CustomMessageBox()  {  InitializeComponent();  timer = new Timer();  timer.Interval = 1000;  timer.Tick += Timer\_Tick;  timer.Start();  }  private void Timer\_Tick(object sender, EventArgs e)  {  if (countdown > 0)  {  countdown--;  labelCountdown.Text = "Обратный отсчет: " + countdown.ToString();  }  else  {  timer.Stop();  DialogResult = DialogResult.OK;  Close();  }  }  private void CustomMessageBox\_Load(object sender, EventArgs e)  {  labelCountdown.Text = "Обратный отсчет: " + countdown.ToString();  }  } |
| *Листинг 3* – Код пользовательской реализации формы |

Остальные используемые компоненты стандартные.

1. Заключение

В ходе выполнения лабораторной работы был получен опыт разработки настольного приложения, взаимодействия с базой данных из кода.

С подробным кодом можно познакомиться в репозитории <https://github.com/danon21/ComponentProgramming>.