Учреждение образования
«Гродненский государственный политехнический колледж»

ОТЧЁТ
ПО УЧЕБНОЙ ПРАКТИКЕ ПО ПРОГРАММИРОВАНИЮ

Учащейся 3 курса, группы ПЗТ-40
специальности 2 - 40 01 01 «Программное обеспечение информационных технологий»
Место прохождения практики УО «Гродненский государственный политехнический колледж»

Тема проекта: «Разработка программного продукта «StressLess»
Ссылка на проект: https://github.com/DanNikonovich/StressLess.git

Выполнил _____ Д.А. Никонович
(инициалы, фамилия)
Руководитель практики от колледжа _____ А.Г. Бабуль
(инициалы, фамилия)

2025

Содержание

Введение. Описание структуры предприятия

На данной учебной практике была поставлена задача разработать научно познавательное приложение «StressLess».

Цель курсового проекта заключается в разработке программного продукта, который позволит массовому пользователю улучшить возможность самообладания и способность бороться со стрессовым накоплением.

Создаваемое приложение будет рассчитано для любого рода пользователей. Применять его смогут как дети, так и взрослые.

Далее приведем краткое описание разделов пояснительной записки.

Первый раздел носит название «Анализ предметной области и формулировка требований к программе». В нем можно ознакомиться с постановкой задачи, которая включает в себя исследование предметной области поставленной задачи. В подразделе «Инструменты разработки» рассмотрена среда, в которой создается данный программный продукт.

В разделе «Проектирование» рассмотрены основные аспекты разработки программного продукта. Здесь можно узнать об организации данных в контексте среды разработки. В данном разделе будут представлены диаграмма вариантов использования и диаграмма деятельности.

«Построение программы» – это третий раздел отчета, в котором описываются все элементы и объекты, которые будут использованы при реализации данного приложения. В этом разделе представлена диаграмма компонентов.

Четвёртый раздел – «Тестирование». В нем будет описано полное и функциональное тестирование данной программы. Будут смоделированы все возможные действия пользователя при работе с приложением, начиная от запуска и заканчивая закрытием приложения.

В разделе «Применение» будет описано назначение программы, инсталляция. Будет указана последовательность действий пользователя, обеспечивающих загрузку, запуск, выполнение и завершение программы.

«Заключение» содержит краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В разделе «Список использованных источников» приведен список используемых при разработке источников.

В приложении А будет приведен листинг программы.

В приложении Б будут представлены тест-кейсы.

1 Анализ предметной области и формулировка требований к программе

1.1 Исследование предметной области

Наименование задачи – «Разработка программного продукта «StressLess».

Цель разработки: создание программного продукта, который рассчитан на людей, заинтересованных в развитии различных качеств, таких как: улучшение стрессоустойчивости, изучение способов самоконтроля, а также снижение уровня стресса. Приложение может быть использована в образовательных и развивающих целях, как инструмент для повышения стрессоустойчивости у детей и взрослых. Она также может быть применена в реабилитационных программах для людей, восстанавливающихся после травм нервной системы.

Назначение: для развития способности к фокусировке внимания на задаче и увеличения продуктивности своей деятельности, для улучшение стрессоустойчивости, реабилитации нервной системы, изучение способов самоконтроля, а также снижения уровня стресса.

Периодичность использования данного программного продукта зависит от нужд потребителя, неограниченно

1.2 Инструменты разработки

Для разработки данного проекта была использована среда Visual Studio и язык программирования C#, это обусловлено рядом факторов:

– широкая распространенность платформы .NET среди разработчиков;

– простота освоения языка C# благодаря ясному синтаксису и большому количеству документации;

– наличие встроенных библиотек (.NET Framework), обеспечивающих удобные инструменты для взаимодействия с графикой и мультимедиа;

– высокая производительность компилятора и оптимизированная виртуальная машина CLR обеспечивают быстрое выполнение приложений.

Выбор Visual Studio для реализации проекта обусловлен её возможностями как мощной интегрированной среды разработки, которая обеспечивает удобную и эффективную работу с языком программирования C# и платформой .NET. Среда предоставляет визуальный редактор для быстрого создания пользовательского интерфейса, интуитивно понятную структуру проекта и продвинутые инструменты для написания и редактирования кода.

Кроме того, Visual Studio обладает мощной системой отладки и тестирования, позволяющей быстро выявлять и устранять ошибки. Поддержка различных форматов проекта, встроенные шаблоны, широкая база знаний и

активное сообщество разработчиков делают эту среду идеальным выбором для создания учебных, развивающих и игровых программ. Всё это позволяет создать стабильный, удобный и функциональный продукт с минимальными затратами времени и усилий.

Иные инструменты, используемые при разработке и написании сопутствующей документации:

– WEB-ресурс DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм;

– Microsoft Office Word 2019 – для написания документации к программному продукту;

– Dr.Explain – инструмент разработки пользовательской документации;

– GitHub – веб-сервис для хостинга IT-проектов;

– Figma – инструмент для разработки иллюстраций;

2 Проектирование

2.1 Диаграмма вариантов использования

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования.

Актером или действующим лицом является любая сущность, взаимодействующая с системой извне. Это может быть как живое существо, так и любая друга система, которая может служить источником воздействия на моделируемую систему так, как определяет сам разработчик. На рисунке 1 представлено графическое обозначение актера.

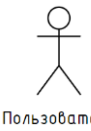

Рисунок 1 – Графическое обозначение актера

Вариант использования является стандартным языком UML и применяется для спецификаций общих особенностей системы и любой другой сущности. Отдельные варианты использования обозначаются на диаграмме эллипсом, в котором содержится его краткое название. Пример представлен на рисунке 2.

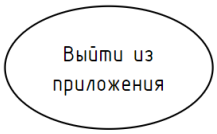

Рисунок 2 – Графическое обозначение вариантов использования

Отношение ассоциации является главным понятием языка UML и используется при построении всех графических моделей. Оно служит для обозначения роли актера и отдельном варианте использования. На диаграмме отношение ассоциации обозначается сплошной линией между актером и вариантом использования. Пример отношения ассоциации представлен на рисунке 3.

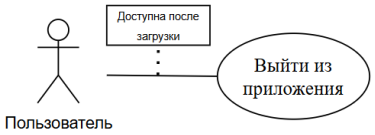

Рисунок 3 – Графическое обозначение отношения ассоциации

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данная линия помечается ключевым словом <extend>. Пример изображен на рисунке 4.

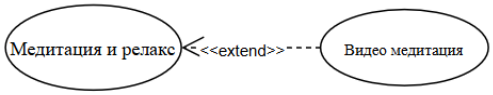

Рисунок 4 – Графическ › обозначение отношения расширения

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данная линия помечается ключевым словом <include> Пример изображен на рисунке 5.

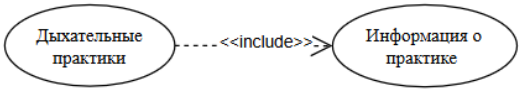

Рисунок 5 – Графическое обозначение отношения включения

Определяя для выбранного актера варианты использования и устанавливая отношения между вариантами использования, получим полную диаграмму вариантов использования, она представлена на рисунке 6.

Рисунок 6 – Диаграмма вариантов использования

2.2 Диаграмма деятельности

Диаграмма деятельности – поведенческая диаграмма, показывающая поток работы или действий в рамках системы или процесса. Она иллюстрирует последовательность шагов и возможные варианты выполнения работы, включая параллельные процессы и ветвления. Диаграмма деятельности включает в себя такие элементы, как начальные и конечные узлы, узлы действий, узлы решений, вилки и слияния, а также потоки управления, которые связывают эти узлы.

На диаграмме отображен процесс прохождение дыхательных практик.

Диаграмма деятельности представлена на рисунке 7.

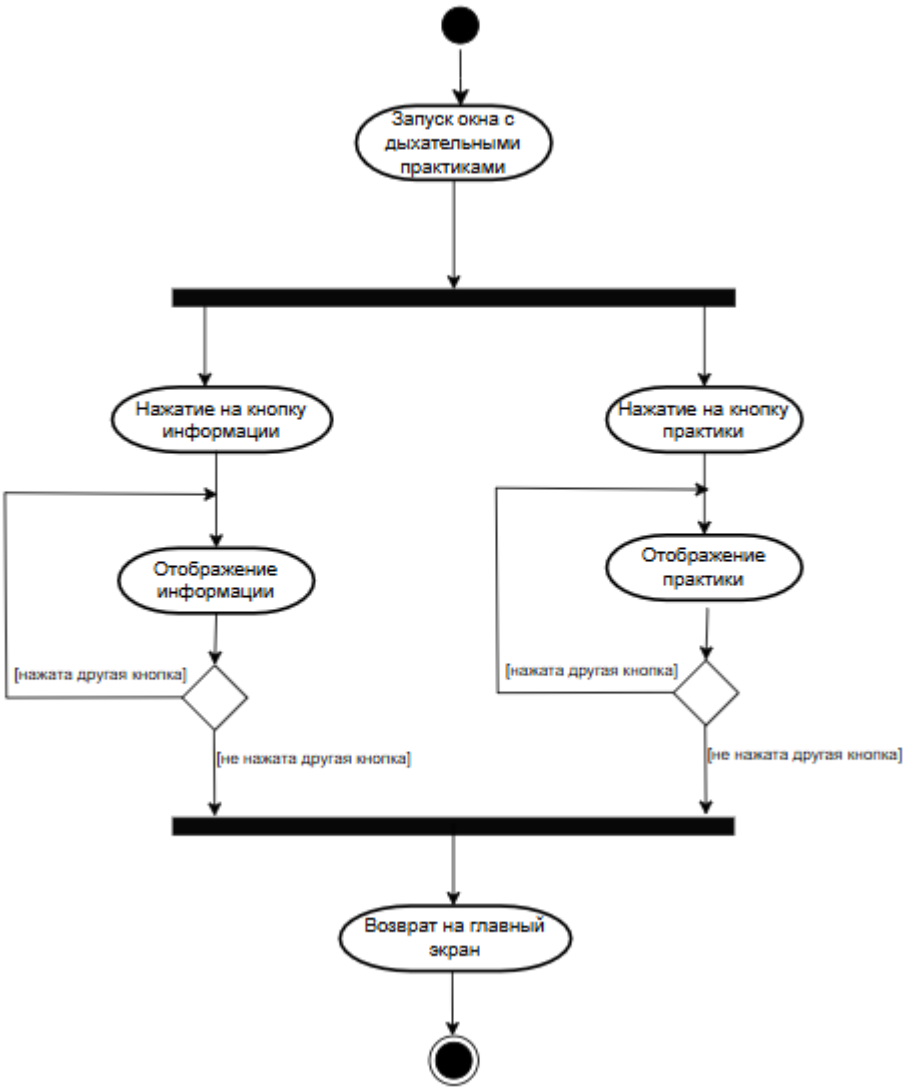

Рисунок 7 – Диаграмма деятельности

3 Построение программы

Диаграмма компонентов – статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи между компонентами.

Диаграмма компонентов представлена на рисунке 8.

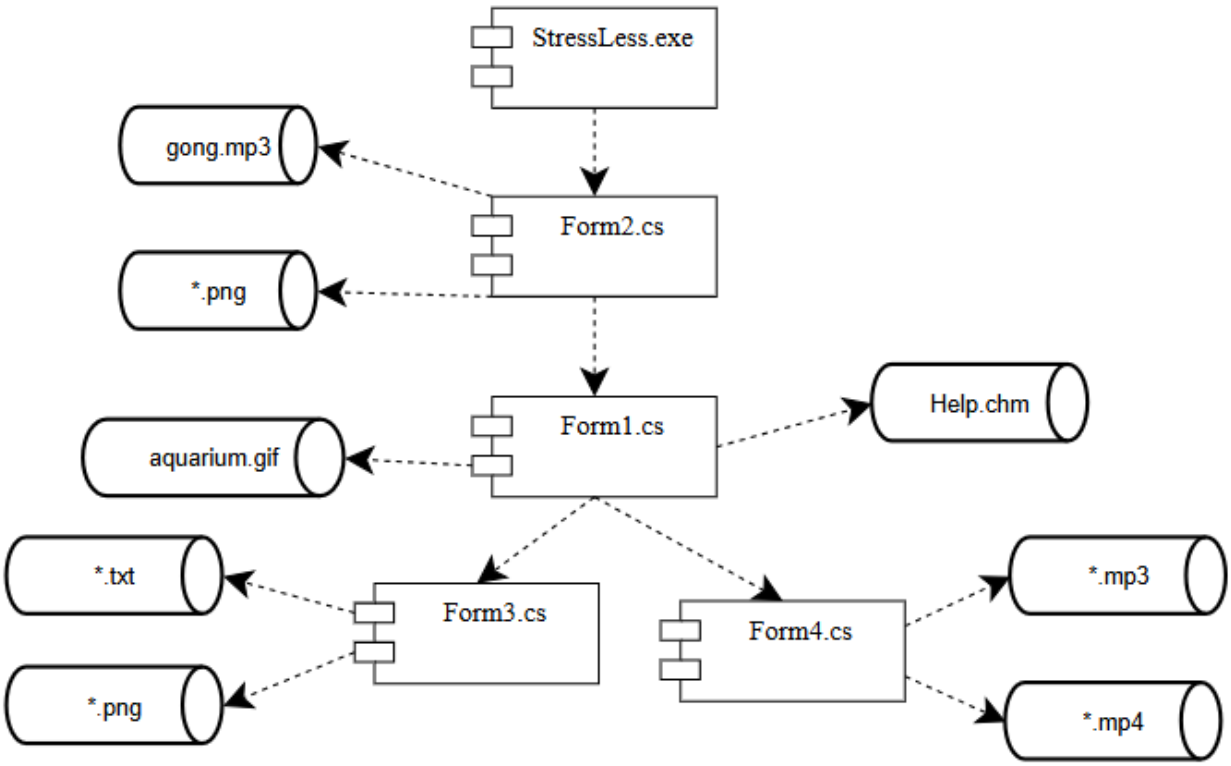

Рисунок 8 – Диаграмма компонентов

4 Тестирование

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения испытания реализации программы было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме.

Разработанные тест-кейсы и статус их выполнения представлены в приложение Б.

Расписание проведения и время, затраченное на тестирование, описано в таблице 1.

Таблица 1 – Расписание работ над проектом

Имя	Дата	Деятельность	Продолжительность, ч
Никонович Даниил	04.05.2025	Разработка тестов	1
Никонович Даниил	05.05.2025	Тестирование познавательного приложения	1,5
Никонович Даниил	05.05.2025	Исправление найденных ошибок	1,5
Никонович Даниил	06.05.2025	Проведение регрессионного тестирования	2
Никонович Даниил	06.05.2025	Составление отчета о результатах тестирования	2

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в процедурах.

Статистика по всем дефектам представлена в таблице 2.

Таблица 2 – Статистика по всем дефектам

Статус	Количество	Важность			
		Низкая	Средняя	Высокая	Критическая
Найдено	1	1	0	0	0
Исправлено		1	0	0	0
Проверено	0	0	0	0	0
Открыто заново	0	0	0	0	0
Отклонено	0	0	0	0	0

5 Применение

5.1 Назначение и условия применения программы

Цель данного проекта заключается в разработке программного продукта, который рассчитан на людей, заинтересованных в развитии различных качеств, таких как: улучшение стрессоустойчивости, реабилитации нервной системы, изучение способов самоконтроля, а также снижения уровня стресса. Приложение может быть использована в образовательных и развивающих целях, как инструмент для восстановления нервной системы у детей и взрослых.

Качество и скорость работы приложения всегда зависит от самих характеристик персонального компьютера. Поэтому приложение должно было быть протестировано на разных машинах. Тестирование проводилось на разных персональных компьютерах и результаты были удовлетворительные.

Сама программа была разработана на программном устройстве со следующими характеристиками:

– процессор: 12th Gen Intel(R) Core(TM) i5-12450H, 2000 МГц, ядер: 8, логических процессоров: 12

– объем оперативной памяти: 16.00 GB;

– ОС: Майкрософт Windows 10 Корпоративная LTSC

5.2 Инсталляция

Для того, чтобы установить программу необходимо запустить файл Setup.exe. Появится мастер установки игрового приложения «StressLess», представленный на рисунке 9.

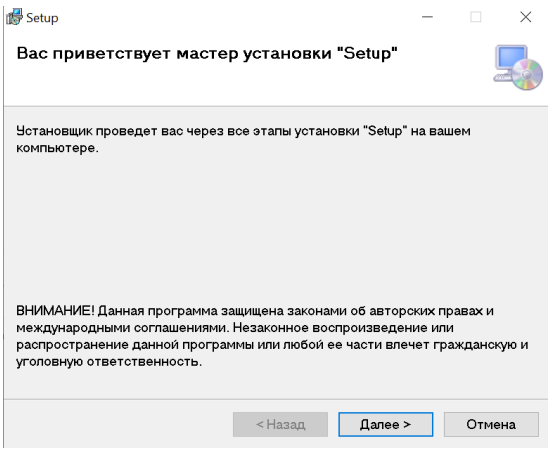

Рисунок 9 – Мастер установки игрового приложения «StressLess»

После нажатия кнопки «Далее» появляется возможность выбора места для установки программного продукта, представленное на рисунке 10.

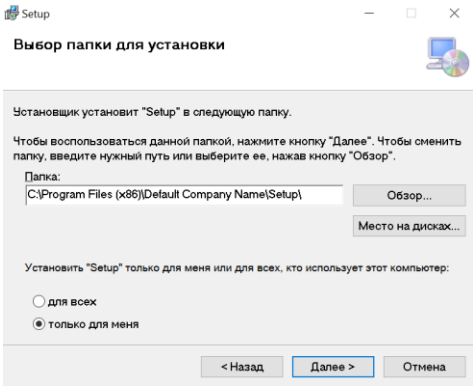

Рисунок 10 – Выбор папки установки

После нажатия на кнопку «Далее» появляется окно подтверждения установки, представленное на рисунке 11.

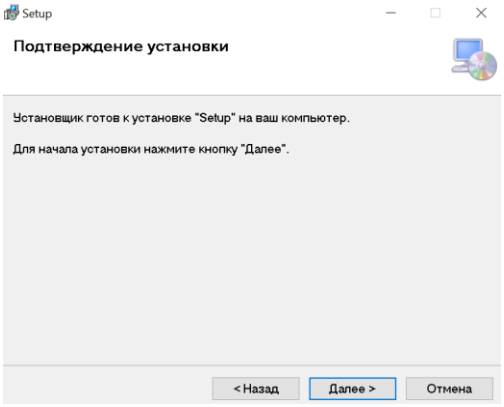

Рисунок 11 – Окно подтверждения установки

После успешной установки на рабочем столе появится окно «Установка завершена», представленное на рисунке 12 и ярлык для запуска приложения.

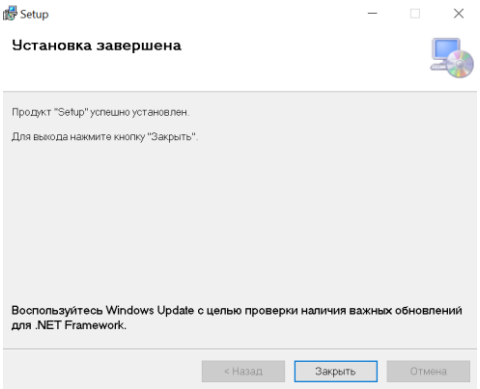

Рисунок 12 – Окно «Установка завершена»

5.3 Выполнение программы

После запуска приложения открывается форма «Загрузка», представленная на рисунке 13.

Рисунок 13 – Загрузка приложения

После загрузки открывается главное меню приложения, представленный на рисунке 14.

Рисунок 14 – Главное меню

У пользователя есть возможность нажать на кнопки «Дыхательные практики», «Медитация и релакс», «Выйти», «Справка».

При нажатии на кнопку «Справка» будет открыта справка.

При нажатии на кнопку «Выйти» программа закончит своё выполнение.

При нажатии на кнопку «Дыхательные практики» появляется новое окно с информацией, которое расположено на рисунке 15.

Рисунок 15 – Новое окно «Дыхательные практики»

После чего пользователю предоставится выбор ознакомления с материалом по практикам дыхания с помощью кнопки «Информация», представленной на рисунке 16, либо же пользователь может выбрать сразу интересующую его практику и приступить к выполнению по инструкции, представленной на рисунке 17.

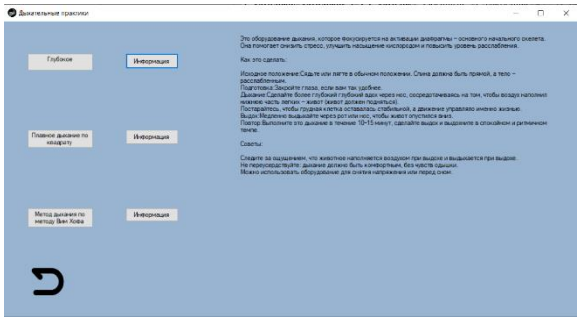

Рисунок 16 – Материал кнопки «Информация»

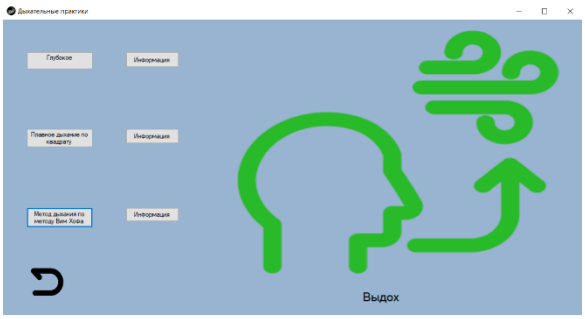

Рисунок 17 – Инструкция по технике дыхания

При нажатии на стрелку, изображённую на рисунке 18, происходит возврат на главное окно.

Рисунок 18 – Кнопка возврата на главную

При нажатии на кнопку «Медитация и релакс» появляется новое окно с информацией, а после выбора, пользователю отображается медитационная поза под успокаивающие звуки, которые расположена на рисунке 19.

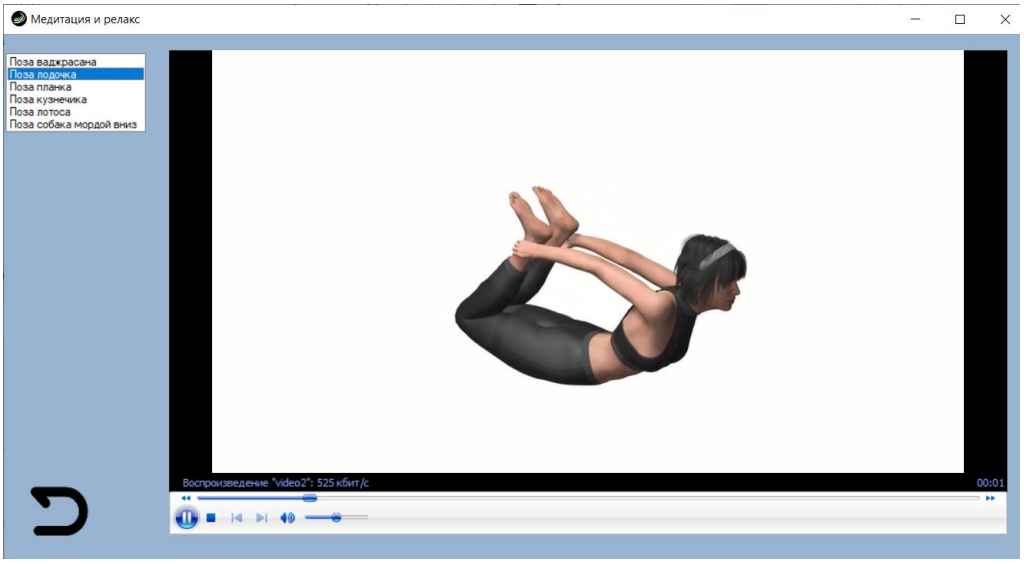

Рисунок 19 – Новое окно «Медитация и релакс»

По мимо этого есть кнопка возврата, которая изображена на рисунке 18.

Также, на главной форме на главном меню реализована кнопка «Справка», нажав на которую открывается справочная система. Она представлена на рисунке 20.

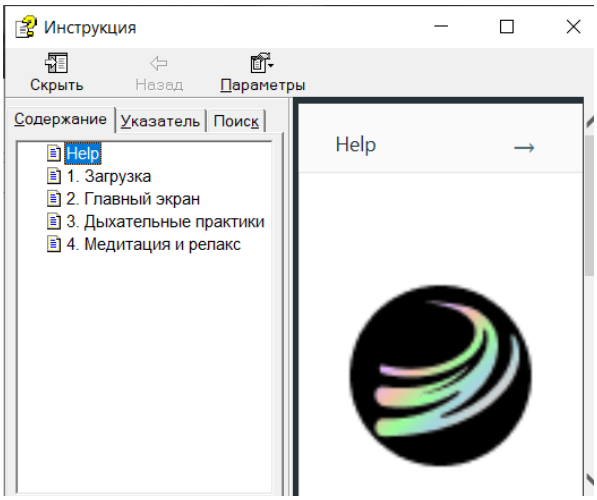

Рисунок 20 – Справка

Заключение

Цель данного проекта заключается в разработке программного продукта, который рассчитан на людей, заинтересованных в развитии различных качеств, таких как: улучшение стрессоустойчивости, реабилитации нервной системы, изучение способов самоконтроля, а также снижения уровня стресса. Приложение может быть использована в образовательных и развивающих целях, как инструмент для восстановления нервной системы у детей и взрослых.

В ходе выполнения данной курсовой работы было разработано и протестировано приложение «StressLess» с графическим интерфейсом. Приложение содержит дыхательные практики трёх видов с визуальным отображением их исполнение и имеет дополнительную текстовую формулировку правил, также имеются медитационные движения, которые можно повторять по предоставленному примеру под успокаивающую музыку.

Для разработки данного проекта была использована среда Visual Studio и язык программирования C#, так как Платформа Windows Forms предлагает интуитивно понятный визуальный конструктор форм, позволяющий быстро создавать интерфейсы и взаимодействовать с элементами управления, а обширная библиотека стандартных компонентов облегчает создание различных элементов UI (меню, кнопки, поля ввода), значительно ускоряя процесс проектирования графического интерфейса. Наличие множества сторонних компонентных библиотек расширяет возможности разработчиков и позволяет добавлять новые функциональные элементы без необходимости писать весь код вручную.

Таким образом, выбор Windows Forms и C# становится оптимальным решением для быстрого запуска качественного продукта с низкой стоимостью разработки и высоким уровнем производительности, ориентированного преимущественно на пользователей операционных систем семейства Windows.

В заключении можно сказать, что данный программный продукт (научно познавательное приложение) является законченной, полнофункциональной программой.

Поставленная задача выполнена в соответствии со всеми ранее задуманными требованиями, созданы и протестированы все необходимые компоненты проекта.

Исходя из этого, можно сделать вывод, что программа реализована успешно.

Список использованных источников

1. Руководство. Создание приложения Windows Forms в Visual Studio с помощью C# [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/visualstudio/ide/create-csharp-winform-visual-studio/ – Дата доступа: 15.04.2025.

2. Полное руководство по языку программирования C# 13 и платформе .NET 9 [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 15.04.2025.

3. Руководство по C# [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/ – Дата доступа: 16.04.2025.

Приложение А

Листинг программы

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace StressLess
{public partial class Form1 : Form
  {  Form mainForm;
     public Form1(Form mainForm)
     {   InitializeComponent();
        this.mainForm = mainForm;}
     private void Form1_Load(object sender,
EventArgs e)
     {pictureBox1.Image =
Image.FromFile("Video\\aquarium.gif");
        pictureBox1.SizeMode =
PictureBoxSizeMode.StretchImage;}
     private void pictureBox2_Click(object sender,
EventArgs e)
     {   Form3 form3 = new Form3(this);
        form3.Show();      // Показать Form3
        this.Hide();       // Скрыть текущую форму
(если нужно)}
     private void pictureBox4_Click(object sender,
EventArgs e)
     {   mainForm.Close(); }
     private void Form1_Shown(object sender,
EventArgs e)
     {  Form2 Load = new Form2();
        Load.Close();  }
     private void pictureBox3_Click(object sender,
EventArgs e)
     {Form4 form4 = new Form4(this);
        form4.Show();      // Показать Form4
        this.Hide();}
     private void pictureBox5_Click(object sender,
EventArgs e)
     {   try{
        string helpFilePath = @"Help.chm"; // путь к
файлу справки
        if (System.IO.File.Exists(helpFilePath))
        { Help.ShowHelp(this,
System.IO.Path.GetFullPath(helpFilePath)); }
        else
        {MessageBox.Show("Файл справки не
найден.", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);}  }
        catch (Exception ex)
        {MessageBox.Show("Ошибка при открытии
справки: " + ex.Message, "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);}}}}
using System;
using System.Windows.Forms;
namespace StressLess
{public partial class Form2 : Form
  {private AxWMPLib.AxWindowsMediaPlayer
audioPlayer;
     private int currentIndex = 0;
     private Timer timer;
     public Form2()
     {  InitializeComponent();
        timer = new Timer();
        timer.Interval = 500;
        timer.Tick += Timer_Tick;
        audioPlayer = new
AxWMPLib.AxWindowsMediaPlayer();
((System.ComponentModel.ISupportInitialize)(audioPl
ayer)).BeginInit(); audioPlayer.Visible = false; //
чтобы не мешал интерфейсу
this.Controls.Add(audioPlayer);
((System.ComponentModel.ISupportInitialize)(audioPl
ayer)).EndInit();} private void Form2_Load(object
sender, EventArgs e) {audioPlayer.URL =
@"Audio\\gong.mp3"; // путь к файлу
```

```
        audioPlayer.settings.setMode("loop", true); // если нужно зациклить
        audioPlayer.Ctlcontrols.play();timer.Start();}
    private void Timer_Tick(object sender, EventArgs e)
        {switch (currentIndex)
            {case 1: pictureBox1.Visible = true; break;
              case 2: pictureBox2.Visible = true; break;
              case 3: pictureBox4.Visible = true; break;
              case 4: pictureBox3.Visible = true; break;
              case 5: pictureBox5.Visible = true; break;
              case 6: pictureBox6.Visible = true; break;
              case 7: pictureBox7.Visible = true; break;}
            currentIndex++;
            if (currentIndex > 8)
            {timer.Stop(); ShowMainForm();  }}
        private void ShowMainForm()
        {  audioPlayer.Ctlcontrols.stop();
            Form1 mainForm = new Form1(this);
            mainForm.Show(); this.Hide();}}}
using System;
using System.Drawing;
using System.IO;
using System.Windows.Forms;
namespace StressLess
{public partial class Form3 : Form
    {   private Timer timer;
        private int imageIndex = 0;
        private string[] imagePaths;
        private Timer breathingTimer;
        private int breathingIndex = 0;
        private string[] breathingCycle;
        private Timer customCycleTimer;
        private int customCycleStep = 0;
        private int elapsedTime = 0;
        Form mainForm;
        public Form3(Form mainForm)
        {   InitializeComponent(); this.mainForm = mainForm;
            // Загружаем пути изображений (убираем двойной слеш)
            imagePaths = new string[]
            {   Path.Combine("Breath", "Inhale.png"), Path.Combine("Breath", "Outhale.png"),};
            // Инициализируем и настраиваем таймер
            timer = new Timer(); timer.Interval = 7000; // 7 секунд
            timer.Tick += Timer_Tick;}
        private void button1_Click(object sender, EventArgs e)
        {   // Останавливаем таймер дыхания, если он запущен
            if (breathingTimer != null && breathingTimer.Enabled) {   breathingTimer.Stop();}
            // Очищаем панель и запускаем основной таймер
            panel1.Controls.Clear();timer.Start();}
        private void pictureBox1_Click(object sender, EventArgs e)
        {   mainForm.Show();      // Показать Form3
            this.Close();}
        private void button2_Click(object sender, EventArgs e)
        {   if (timer != null && timer.Enabled) timer.Stop();
            if (breathingTimer != null && breathingTimer.Enabled) breathingTimer.Stop();
            if (customCycleTimer != null && customCycleTimer.Enabled) customCycleTimer.Stop();
            panel1.Controls.Clear(); panel1.BackgroundImage = null; label1.Text = "";
string filePath = @"Rules\\1.txt";
```

```
        if (File.Exists(filePath)) {   string content =
File.ReadAllText(filePath);
            Label textLabel = new Label {   Text =
content, AutoSize = true, MaximumSize = new
Size(panel1.Width - 10, 0), Location = new Point(5,
5)};
            panel1.Controls.Add(textLabel);}
        else { MessageBox.Show("Файл не
найден.");}}
    private void button4_Click(object sender,
EventArgs e)
    {   if (timer != null && timer.Enabled)
timer.Stop();
        if (breathingTimer != null &&
breathingTimer.Enabled) breathingTimer.Stop();
        if (customCycleTimer != null &&
customCycleTimer.Enabled)
customCycleTimer.Stop();
        panel1.Controls.Clear();
panel1.BackgroundImage = null; label1.Text = "";
string filePath = @"Rules\\2.txt";
        if (File.Exists(filePath)) {   string content =
File.ReadAllText(filePath);
            Label textLabel = new Label {    Text =
content, AutoSize = true, MaximumSize = new
Size(panel1.Width - 10, 0), Location = new Point(5,
5)};
            panel1.Controls.Add(textLabel); }
        else {MessageBox.Show("Файл не
найден.");}}
    private void button6_Click(object sender,
EventArgs e)
    {   if (timer != null && timer.Enabled)
timer.Stop();
        if (breathingTimer != null &&
breathingTimer.Enabled) breathingTimer.Stop();
        if (customCycleTimer != null &&
customCycleTimer.Enabled)
customCycleTimer.Stop();
        panel1.Controls.Clear();
panel1.BackgroundImage = null; label1.Text = "";
string filePath = @"Rules\\3.txt";
        if (File.Exists(filePath)) {   string content =
File.ReadAllText(filePath);  Label textLabel = new
Label {   Text = content, AutoSize = true,
MaximumSize = new Size(panel1.Width - 10, 0),
Location = new Point(5, 5)};
panel1.Controls.Add(textLabel);}
        else { MessageBox.Show("Файл не
найден.");}}
    private void Timer_Tick(object sender, EventArgs
e)
    {   if (imagePaths.Length == 0) return; try {string
imagePath = imagePaths[imageIndex];
            if (File.Exists(imagePath)) {
panel1.BackgroundImage =
Image.FromFile(imagePath);
SetBreathingText(imagePath);
panel1.BackgroundImageLayout =
ImageLayout.Stretch; imageIndex = (imageIndex + 1)
% imagePaths.Length;}
            else{ MessageBox.Show($"Файл не найден:
{imagePath}"); timer.Stop();}}
        catch (Exception ex) {
MessageBox.Show("Ошибка при загрузке
изображения: " + ex.Message);timer.Stop();}}
    private void button3_Click(object sender,
EventArgs e)
    {   // Останавливаем основной таймер, если он
активен
        if (timer != null && timer.Enabled) {
timer.Stop(); } // Очищаем панель
        panel1.Controls.Clear(); breathingIndex = 0;
```

```
        // Инициализируем массив, если ещё не инициализирован
        if (breathingCycle == null || breathingCycle.Length == 0) { breathingCycle = new string[]{ Path.Combine("Breath", "Inhale.png"), Path.Combine("Breath", "Vector.png"), Path.Combine("Breath", "Outhale.png"), Path.Combine("Breath", "Vector.png")};}
        // Создаем таймер, если он ещё не создан
        if (breathingTimer == null) {    breathingTimer = new Timer(); breathingTimer.Interval = 4000; breathingTimer.Tick += BreathingTimer_Tick;} breathingTimer.Start();}
    private void BreathingTimer_Tick(object sender, EventArgs e)
    { try {  if (breathingCycle.Length == 0) return; string imagePath = breathingCycle[breathingIndex];
            if (File.Exists(imagePath)) { panel1.BackgroundImage = Image.FromFile(imagePath); SetBreathingText(imagePath); panel1.BackgroundImageLayout = ImageLayout.Stretch; breathingIndex = (breathingIndex + 1) % breathingCycle.Length;}
            else {   MessageBox.Show($"Файл не найден: {imagePath}"); breathingTimer.Stop();}}
        catch (Exception ex) { MessageBox.Show("Ошибка при загрузке изображения: " + ex.Message); breathingTimer.Stop();}}
    private void button5_Click(object sender, EventArgs e)
    {   // Останавливаем все предыдущие таймеры
        if (timer != null && timer.Enabled) timer.Stop();
        if (breathingTimer != null && breathingTimer.Enabled) breathingTimer.Stop();
        // Сброс переменных
        panel1.Controls.Clear(); elapsedTime = 0; customCycleStep = 0;
        if (customCycleTimer == null) { customCycleTimer = new Timer(); customCycleTimer.Tick += CustomCycleTimer_Tick;}
        // Старт с 1-секундного интервала
        customCycleTimer.Interval = 1000; customCycleTimer.Start();}
    private void CustomCycleTimer_Tick(object sender, EventArgs e)
    {   string inhale = Path.Combine("Breath", "Inhale.png");
        string outhale = Path.Combine("Breath", "Outhale.png");
        string vector = Path.Combine("Breath", "Vector.png");
        try {   // Первая фаза — 30 секунд, переключение каждые 1 сек
            if (elapsedTime < 30) {   string imagePath = (elapsedTime % 2 == 0) ? inhale : outhale;
                if (File.Exists(imagePath)) { panel1.BackgroundImage = Image.FromFile(imagePath); SetBreathingText(imagePath); panel1.BackgroundImageLayout = ImageLayout.Stretch;} elapsedTime++;}
            else {   // Вторая фаза — 5 + 10 + 5 + 10 сек
                string imagePath = ""; int interval = 1000;
                switch (customCycleStep)
                {   case 0: imagePath = inhale; interval = 5000; break;
                    case 1: imagePath = vector; interval = 10000; break;
                    case 2: imagePath = outhale; interval = 5000; break;
```

```
                case 3: imagePath = vector; interval =
10000; break;}
            if (File.Exists(imagePath))
{panel1.BackgroundImage =
Image.FromFile(imagePath);
panel1.BackgroundImageLayout =
ImageLayout.Stretch;} customCycleStep++;
            // После 4 шагов возвращаемся к началу
using System;
using System.Windows.Forms;
using WMPLib;
namespace StressLess
{public partial class Form4 : Form
  {  private AxWMPLib.AxWindowsMediaPlayer
audioPlayer;
    Form mainForm;
    public Form4(Form mainForm)
    { InitializeComponent();
      this.mainForm = mainForm;
      listBox1.SelectedIndexChanged +=
listBox1_SelectedIndexChanged;audioPlayer = new
AxWMPLib.AxWindowsMediaPlayer();
((System.ComponentModel.ISupportInitialize)(audioPl
ayer)).BeginInit();
      this.Controls.Add(audioPlayer);
((System.ComponentModel.ISupportInitialize)(audioPl
ayer)).EndInit();audioPlayer.Visible = false;  }
    private void pictureBox1_Click(object sender,
EventArgs e)  {
axWindowsMediaPlayer1.Ctlcontrols.stop();
      mainForm.Show();      // Показать Form1
      this.Close();}private void
listBox1_SelectedIndexChanged(object sender,
EventArgs e){   string selected =
listBox1.SelectedItem.ToString();string videoPath = "";
      string audioPath = "";
      switch (selected)
      {    case "Поза ваджрасана":
            videoPath = @"Video\\video1.mp4";
            audioPath = @"Audio\\meditation-
spiritual-music.mp3"; break;
          case "Поза лодочка":
            videoPath = @"Video\\video2.mp4";
            audioPath = @"Audio\\meditation-
relaxing-music.mp3";break;
          case "Поза планка":
            videoPath = @"Video\\video3.mp4";
            audioPath = @"Audio\\calm-soul-
meditation.mp3";break;
          case "Поза кузнечика":
            videoPath = @"Video\\video4.mp4";
            audioPath = @"Audio\\clearing-negative-
energy.mp3";break;
          case "Поза лотоса":
            videoPath = @"Video\\video5.mp4";
            audioPath = @"Audio\\serenity-waves-
zen-meditation.mp3";break;
          case "Поза собака мордой вниз":
            videoPath = @"Video\\video6.mp4";
            audioPath = @"Audio\\deep-meditation-
vibes.mp3";  break; }
        // Запуск видео
        if (!string.IsNullOrEmpty(videoPath))
        {   axWindowsMediaPlayer1.URL = videoPath;
     axWindowsMediaPlayer1.settings.setMode("loop",
true);axWindowsMediaPlayer1.Ctlcontrols.play(); }
        // Запуск звука
        if (!string.IsNullOrEmpty(audioPath))
        {    audioPlayer.URL = audioPath;
          audioPlayer.settings.setMode("loop", true);
          audioPlayer.Ctlcontrols.play();} }}}
            if (customCycleStep >= 4) {
customCycleStep = 0; elapsedTime = 0; interval =
1000; // Снова 1 секунда
```

```
                } customCycleTimer.Interval = interval;}}
            catch (Exception ex)
{MessageBox.Show("Ошибка при показе
изображения: " + ex.Message);
customCycleTimer.Stop();}}
        private void SetBreathingText(string imagePath) {
if (imagePath.EndsWith("Inhale.png")) {label1.Text =
"Вдох";} else if (imagePath.EndsWith("Outhale.png"))
{label1.Text = "Выдох";} else if
(imagePath.EndsWith("Vector.png")) {label1.Text =
"Задержите дыхание";} else {label1.Text = "";}}}}
```