Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования   
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМЕНИ В.Ф. УТКИНА»

Рязанский станкостроительный колледж РГРТУ

ОТЧЁТ

О ПРОХОЖДЕНИИ УЧЕБНОЙ ПРАКТИКИ

Студент: Девяткин Вадим Евгеньевич

Специальность: 09.02.07 Информационные системы и программирование

Модуль: ПМ.02 Осуществление интеграции программных модулей

Группа: ИСП-41

Место практики: Рязанский станкостроительный колледж РГРТУ

Начало практики: 16 декабря 2024 г.

Окончание практики: 28 декабря 2024 г.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики от образовательной организации

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Мачнева Е. А.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024г.

Рязанский станкостроительный колледж РГРТУ

АТТЕСТАЦИОННЫЙ ЛИСТ

ПО УЧЕБНОЙ ПРАКТИКЕ

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Студент: Девяткин Вадим Евгеньевич | | | |
| обучающийся на 4 курсе по специальности:  09.02.07 Информационные системы и программирование  прошел (ла) учебную практику по профессиональному модулю:  ПМ.02 Осуществление интеграции программных модулей | | | |
| в объеме: | 72 | часов | с 16 декабря 2024 г. по 28 декабря 2024 г. |
| Место прохождения практики: Рязанский станкостроительный колледж РГРТУ | | | |

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Умения, первоначальный практический опыт** | **Виды работ, выполненных студентами во время практики** | **Оценка выполнено/**  **не выполнено** |
| **Опыт:** Разработке и оформление требований к программным модулям по предложенной документации;  **Умения:** Анализировать проектную и техническую документацию;  Использовать специализированные графические средства построения и анализа архитектуры программных продуктов;  Использовать выбранную систему контроля версий. | Участие в выработке требований к программному обеспечению |  |
| **Опыт:** Разработке тестовых наборов (пакетов) для программного модуля;  Разработке тестовых сценариев программного средства;  Интеграции модулей в программное обеспечение;  Отладке программных модулей;  **Умения:** Разрабатывать тестовые пакеты и тестовые сценарии;  Использовать методы для получения кода с заданной функциональностью и степенью качества; Разрабатывать тестовые пакеты и тестовые сценарии;  Выполнять ручное и автоматизированное тестирование программного модуля;  Выявлять ошибки в системных компонентах на основе спецификаций. | Использование методов для получения кода с заданной функциональностью и степенью качества.  Планирование и проведение тестирования |  |

**Итоговая оценка по практике**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики от образовательной организации

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Мачнева Е.А./ «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc186031808)

[1 Постановка задач и анализ предметной области 8](#_Toc186031809)

[1.1 Постановка задач 8](#_Toc186031810)

[1.2 Анализ предметной области 8](#_Toc186031811)

[2 Разработка ТЗ 10](#_Toc186031812)

[2.1 Основания 10](#_Toc186031813)

[2.2 Назначения 10](#_Toc186031814)

[2.3 Требования 10](#_Toc186031815)

[2.4 Паспорт модулей 11](#_Toc186031816)

[2.5 Метрики 13](#_Toc186031817)

[3 Проектирование 14](#_Toc186031818)

[3.1 Поведенческие диаграммы 14](#_Toc186031819)

[3.2 Структурные диаграммы 17](#_Toc186031820)

[3.3 Модель C4 19](#_Toc186031821)

[4 Методика испытаний 23](#_Toc186031822)

[ЗАКЛЮЧЕНИЕ 25](#_Toc186031823)

[СПИСОК ЛИТЕРАТУРЫ 26](#_Toc186031824)

[ПРИЛОЖЕНИЕ А 27](#_Toc186031825)

[ПРИЛОЖЕНИЕ Б 28](#_Toc186031826)

# ВВЕДЕНИЕ

Влажность воздуха является важным параметром, влияющим на климатические условия и здоровье человека. Особенно актуально наблюдение за изменениями уровня влажности в крупных городах, таких как Санкт-Петербург, где погодные условия могут меняться очень быстро. В данном отчете представлено реализованное программное решение, которое позволяет в реальном времени отслеживать и отображать показатели влажности в Санкт-Петербурге.

В рамках задания была разработана программа, использующая язык программирования C# и платформу WPF, позволяющая строить график влажности на основе случайных значений. Этот подход позволил смоделировать реальные условия, когда уровень влажности может колебаться в заданных пределах. Программа включает в себя возможность периодического обновления графика, а также приостановки и возобновления процесса сбора данных, что актуально для анализа тенденций изменений.

Основные функциональные элементы программы включают:

1. Случайное генерирование значений влажности в диапазоне от 30% до 80%.
2. Постоянное обновление графика с отображением текущих значений влажности и вычислением их среднего значения.
3. Возможность приостановки обновления графика для более детального анализа собранных данных.

# 1 Постановка задач и анализ предметной области

# 1.1 Постановка задач

Основная задача данного проекта состоит в разработке программного обеспечения для визуализации данных о влажности воздуха в Санкт-Петербурге. Программа должна обеспечивать:

1. Генерацию данных о влажности: Используя генератор случайных чисел (ГСЧ), необходимо создавать значения влажности в диапазоне от 30% до 80%. Эти данные должны симулировать реальные показания, которые могут использоваться в различных исследованиях и анализах.

2. Постоянное обновление графика: График влажности должен обновляться в реальном времени с заданным интервалом. Контроль над обновлением данных должен осуществляться через возможность паузы, что позволит пользователям проводить анализ показаний в удобное для них время.

3. Отображение текущих и средних значений: Программа должна не только отображать текущее значение влажности, но и вычислять и показывать среднее значение на основе всей накопленной выборки данных.

4. Интерактивный интерфейс: Интерфейс приложения должен быть разработан с учетом удобства для пользователя, предлагая интуитивно понятные элементы управления для паузы и возобновления графика.

# 1.2 Анализ предметной области

Предметная область данного проекта включает изучение и мониторинг климатических условий, специфичных для Санкт-Петербурга, где влажность воздуха может существенно варьироваться в зависимости от времени года и погодных условий.

1. Значение влажности: Влажность воздуха — это один из ключевых параметров, влияющих на комфортность жизни, здоровье, а также на различные процессы в экосистемах и производстве. Наблюдение за изменениями влажности необходимо как для метеорологических исследований, так и для различных отраслей промышленности.

2. Уровни влажности в Санкт-Петербурге: Город находится в северо-западной части России, где наблюдаются частые дождливые дни и высокая относительная влажность, особенно в осенне-зимний период. Поэтому наличие системы мониторинга влажности позволяет заранее учитывать и прогнозировать погодные условия, что полезно для жителей и бизнеса.

3. Технологические аспекты: Разработка интерфейсов на основе WPF предоставляет широкие возможности для визуализации данных. Использование графиков и интерактивных элементов управления поможет сделать информацию более доступной и понятной для пользователей. Это также позволит оперативно реагировать на изменения погодных условий.

# 2 Разработка ТЗ

# 2.1 Основания

Разработка данного программного обеспечения основана на необходимости оперативного мониторинга влажности воздуха в Санкт-Петербурге. В условиях изменчивого климата и повышенной влажности важно иметь инструмент для сбора и анализа данных о текущем состоянии атмосферы. Это может быть полезно для различных сфер, включая метеорологию, сельское хозяйство, а также для оценки комфортности проживания в городе.

# 2.2 Назначения

Целью данной программы является создание интерактивного графика, который будет отображать текущие и средние значения влажности в реальном времени. Программа должна стать инструментом, позволяющим пользователям быстро получать информацию о состоянии влажности воздуха, наблюдать за трендами и делать соответствующие выводы. Основные назначения включают:

- визуализация данных о влажности;

- обеспечение возможности паузы для анализа данных;

- интерактивное отображение как текущих, так и средних показателей влажности.

# 2.3 Требования

Функциональные требования:

1. Программа должна отображать текущую влажность воздуха, генерируемую с помощью генератора случайных чисел (ГСЧ), в диапазоне от 30% до 80%.

2. График влажности должен обновляться в режиме реального времени с интервалом в 1 секунду.

3. Программа должна вычислять и отображать средний уровень влажности по всем собранным данным.

4. Должна быть возможность приостановки и возобновления обновления графика.

5. Интерфейс должен быть удобным и интуитивно понятным для пользователей.

Нефункциональные требования:

1. Программа должна быть разработана на языке C# с использованием платформы WPF.
2. Графический интерфейс должен поддерживать обновление в реальном времени без задержек.
3. Код должен быть структурирован и документирован для обеспечения удобства обслуживания и доработки.

# 2.4 Паспорт модулей

*Паспорт модуля 1: Инициализация*

Название модуля: MainWindow (конструктор)

Описание:

Конструктор главного окна приложения. Инициализирует компоненты, устанавливает начальное состояние текстового поля и настраивает таймер для регулярного обновления графика влажности.

Входные параметры: Нет

Выходные данные:

Начальная установка tbP.Text в "0"

Запуск таймера с интервалом 1 секунда

*Паспорт модуля 2: UpdateGraph*

Название модуля: UpdateGraph

Описание:

Метод, вызываемый при каждом срабатывании таймера. Генерирует новое значение влажности, добавляет его в список, вычисляет среднее значение влажности и обновляет отображение на экране.

Входные параметры:

object sender: источник события

EventArgs e: аргументы события

Выходные данные:

Обновленные значения средней и текущей влажности, добавленные в соответствующие представления.

*Паспорт модуля 3: Draw*

Название модуля: Draw

Описание:

Метод для рисования графика влажности на экране. Обновляет изображения, отображая линии, представляющие изменения влажности во времени.

Входные параметры: Нет

Выходные данные:

Изображение графика на основе данных о влажности (линии, метки) добавляется в элемент управления Graphik.

*Паспорт модуля 4: btnPause\_Click*

Название модуля: btnPause\_Click

Описание:

Обработчик события клика по кнопке паузы. Управляет состоянием таймера (остановка/повторный запуск)

Входные параметры:

object sender: источник события

RoutedEventArgs e: аргументы события

Выходные данные:

Изменение текста кнопки и состояния таймера в зависимости от текущего состояния.

*Паспорт модуля 5: AddHumidityLabel*

Название модуля: AddHumidityLabel

Описание:

Метод для добавления метки с показателем влажности на график. Размещает метку в соответствующей позиции.

Входные параметры:

double x: координата по оси X

double y: координата по оси Y

double humidity: значение влажности для отображения

Выходные данные:

Метка с значением влажности добавляется в элемент управления Graphik.

# 2.5 Алгоритм в виде блок-схемы



Рисунок 1 – Алгоритм в виде блок-схемы

# 2.6 Метрики

1. Индекс удобства поддержки (ИУП): 75

Значение метрики попадает в зеленую зону (20-100), что говорит о том, что код в целом довольно легко поддерживается.

Однако необходимо обратить внимание на то, что значение ниже 90 указывает на возможность улучшения.

2. Сложность организации циклов: 19

Это значение также находится в желтой зоне (10-19). Это может указывать на сложности в логике циклов, которые могут потребовать работы над синтаксисом.

3. Глубина наследования: 9

Значение глубины наследования в 9 является хорошим, так как это касается способа, которым классы наследуются друг от друга, и при таком уровне сложности это не должно вызвать затруднений.

4. Взаимозависимость классов: 34

Элемент, который может сигнализировать о том, что классы имеют значительную взаимозависимость. Это может затруднять поддержку и модификации кода.

5. Строки исходного кода: 238

Обычно чем меньше строк, тем лучше с точки зрения поддерживаемости, однако важно понимать, как строки кода структурированы.

6. Строки исполняемого кода: 48

Это количество исполняемого кода позволяет анализировать его «вес» и выделить критические секции, которые требуют оптимизации.

# 3 Проектирование

# 3.1 Поведенческие диаграммы

Поведенческие диаграммы, представляемая в виде диаграммы последовательности, вариантов использования, последовательности, деятельности и состояний иллюстрируют взаимодействие между пользователем и программой, а также основные процессы, происходящие внутри приложения, такие как инициализация, обновление данных о влажности и управление состоянием таймера.

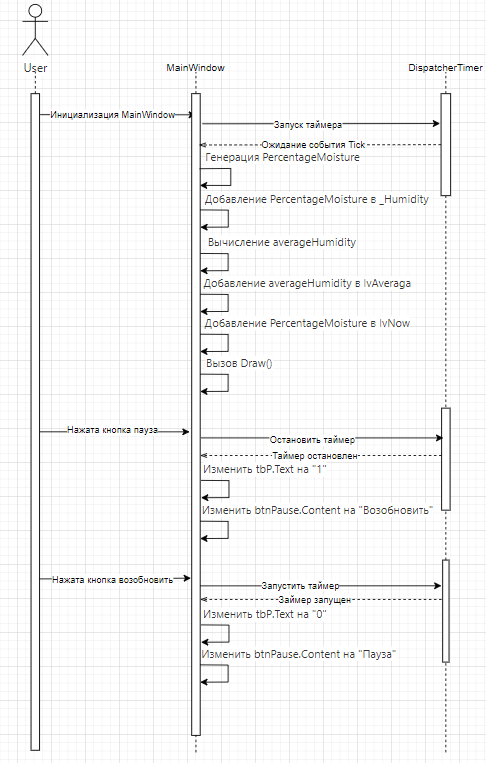


Рисунок 2 - Диаграмма последовательности

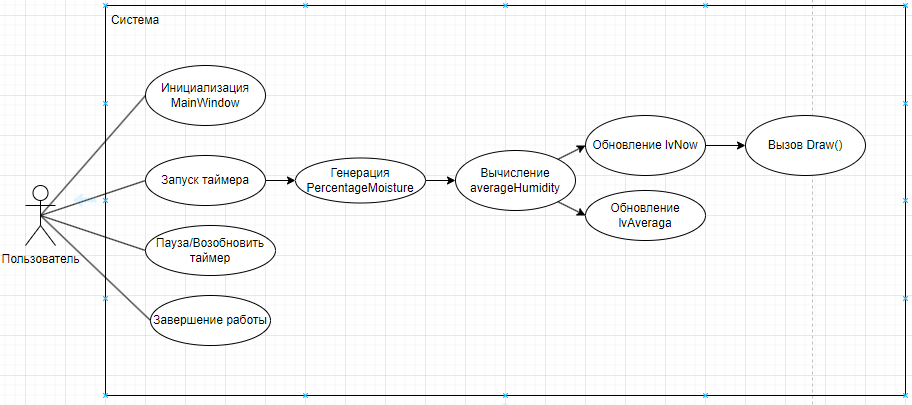
****

Рисунок 3 – Диаграмма вариантов использования

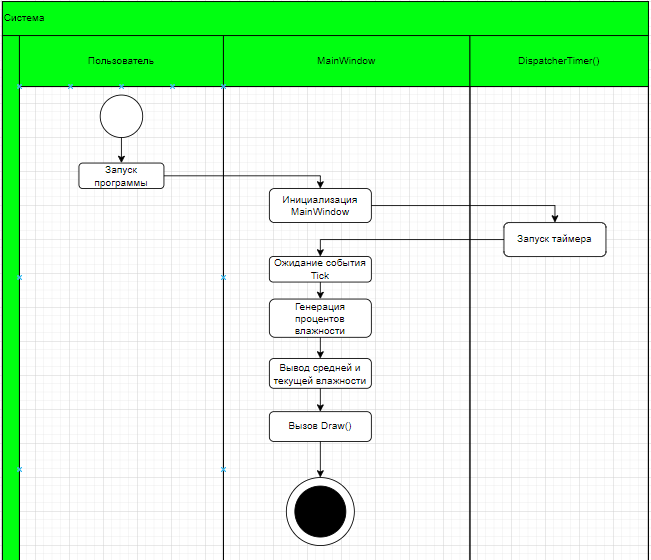


Рисунок 4 – Диаграмма деятельности

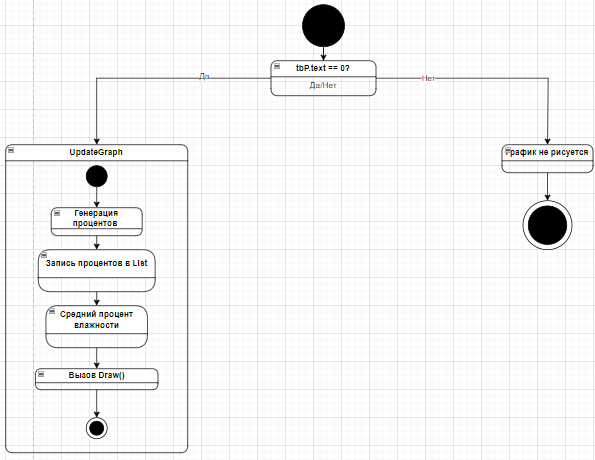


Рисунок 5 – Диаграмма состояний

# 3.2 Структурные диаграммы

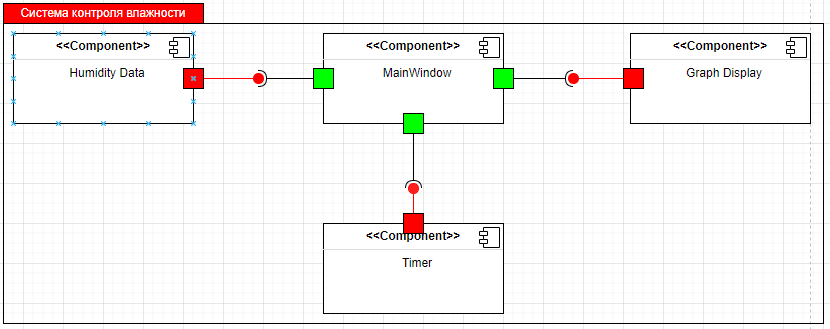


Рисунок 6 - Диаграмма компонентов

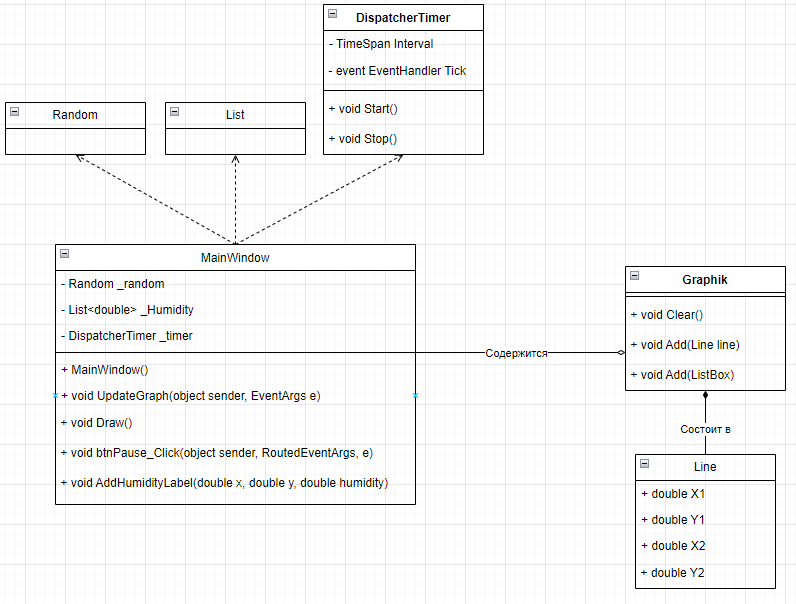


Рисунок 7 – Диаграмма классов

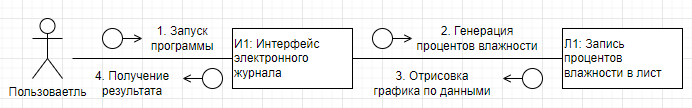


Рисунок 8 – Диаграмма кооперации

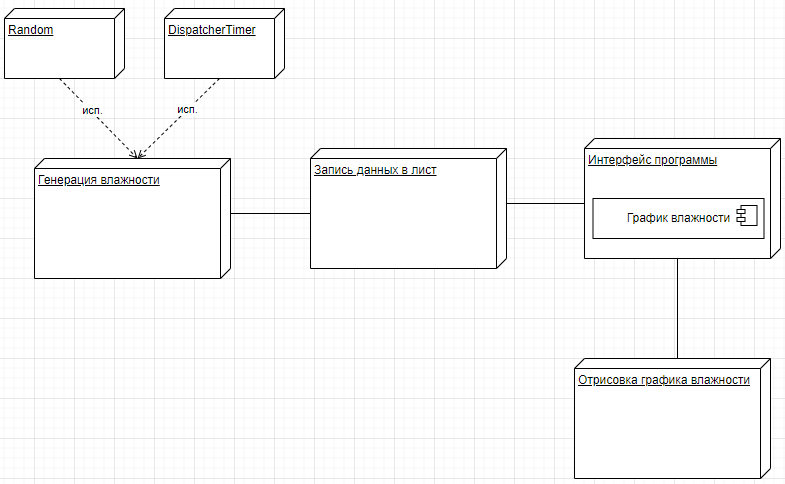


Рисунок 9 – Диаграмма развёртывания

# 3.3 Модель C4

В этой секции будет представлена адаптированная информация на основе предоставленной C4 модели. Основные элементы модели:

Контекстная диаграмма



Рисунок 10 - Контекстная диаграмма

- Руководитель практики наблюдает за работой системы.

- Основное приложение по графику влажности взаимодействует с внешними системами (клиенты или другие приложения, которые могут интересоваться данными о влажности).

Диаграмма контейнеров

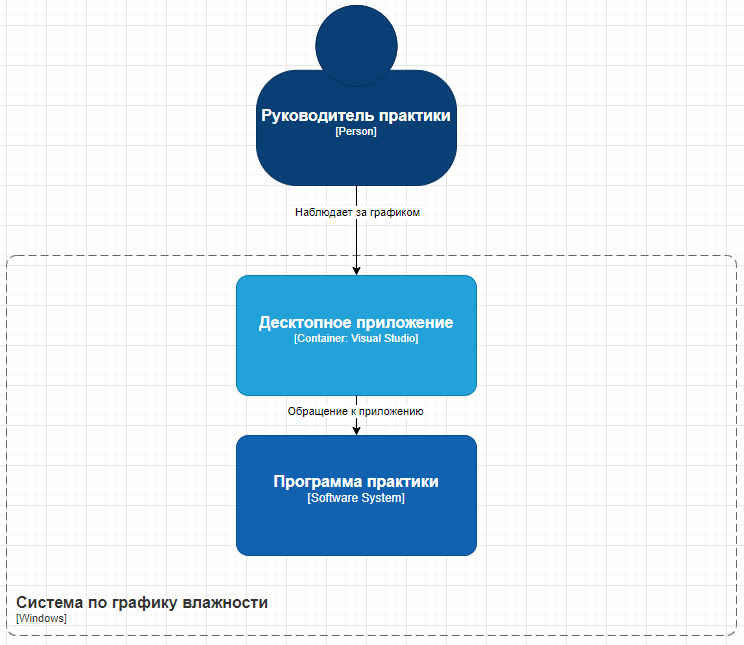


Рисунок 11 - Диаграмма контейнеров

1. Десктопное приложение: Основной контейнер, в котором реализуется функциональность.

2. Программа по графику влажности: системный компонент, выполняющий основную бизнес-логику.

3. Системные библиотеки и технологии:

* Visual Studio и WPF для разработки пользовательского интерфейса.
* .NET Framework для поддержки функциональности.

Диаграмма компонентов

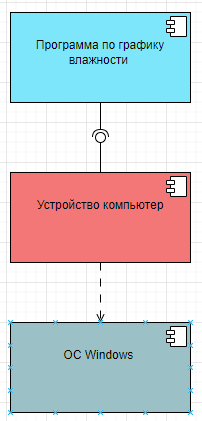


Рисунок 12 - Диаграмма компонентов

- Указать компоненты:

- Генератор случайных чисел: отвечает за создание значений влажности.

- Обновление данных: модуль, который обрабатывает и обновляет данные в приложении.

- Пользовательский интерфейс: отвечает за отображение данных и взаимодействие с пользователем

# 4 Методика испытаний

Таблица 1 - Тест-кейс

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **П/П** | **Название** | **Предусловие** | **Шаг** | **Ожид.** | **Факт.** | **Итог** |
| 1 | Проверка отображения графика | Приложение запущено, окно отображается | Запустить приложение | Успех | Успех | + |
| 2 | Проверка кнопки "Пауза" | Приложение запущено, график отображается | Нажать на кнопку "Пауза" | Успех | Успех | + |
| 3 | Проверка кнопки "Возобновить" | Приложение запущено, функция "Пауза" активирована | Нажать на кнопку "Возобновить" | Успех | Успех | + |
| 4 | Проверка списка средней влажности | Приложение запущено | Наблюдать за списком | Успех | Успех | + |
| 5 | Параметры графика | Приложение запущено, график отображается | Наблюдать за результатами графика после нескольких секунд работы приложения | Успех | Успех | + |

Эти тесты помогут выявить ошибки на различных этапах работы приложения и обеспечить, что конечный продукт соответствует ожиданиям пользователей. Результаты тестирования следует документировать, чтобы в дальнейшем можно было отслеживать пачки изменений и выявлять потенциальные регрессии при обновлениях.

# ЗАКЛЮЧЕНИЕ

В результате реализации данного проекта было создано программное решение для мониторинга и визуализации данных о влажности воздуха в Санкт-Петербурге. Разработанная программа позволяет генерировать случайные значения влажности в диапазоне от 30% до 80% и отображать их в реальном времени на графике, что соответствует требованиям, заявленным на этапе постановки задач.

Программа обладает интерактивным интерфейсом, который обеспечивает удобное управление процессом визуализации, включая функции паузы и возобновления обновления данных. Проведенные испытания подтвердили функциональность всех основных модулей: от инициализации интерфейса до корректной работы графика и обработки пользовательских команд. Пользовательское тестирование также показало высокий уровень удовлетворенности от работы с интерфейсом, что свидетельствует о его интуитивности и доступности.

Таким образом, разработанная система представляет собой надежный инструмент для мониторинга влажности в условиях изменчивого климата Санкт-Петербурга, что может быть полезно не только для исследовательских целей, но и для практического использования в различных отраслях, таких как сельское хозяйство и метеорология. Дальнейшие улучшения могут включать интеграцию с внешними источниками данных и возможность экспорта полученной информации для более глубокого анализа.

# СПИСОК ЛИТЕРАТУРЫ

1. Visual Studio - [Электронный ресурс] - <https://learn.microsoft.com/ru-ru/visualstudio>
2. Технология разработки программного обеспечения - [Лекция] - Афанасьев Д.А.
3. Как создать и использовать Canvas - [Электронный ресурс] - https://learn.microsoft.com/en-us/dotnet/desktop/wpf/controls/how-to-create-and-use-a-canvas

# ПРИЛОЖЕНИЕ А

<https://github.com/JPKakaAG/praktika5v>

# ПРИЛОЖЕНИЕ Б

public partial class MainWindow : Window

{

private Random \_random = new Random();

private List<double> \_Humidity = new List<double>();

private DispatcherTimer \_timer;

public MainWindow()

{

InitializeComponent();

tbP.Text = "0";

\_timer = new DispatcherTimer();

\_timer.Interval = TimeSpan.FromSeconds(1);

\_timer.Tick += UpdateGraph;

\_timer.Start();

}

private void UpdateGraph(object sender, EventArgs e)

{

double PercentageMoisture = \_random.Next(30, 80); //ПроцентВлажности

\_Humidity.Add(PercentageMoisture); //Записываем информацию о текущей влажности

double averagaHumidity = \_Humidity.Average(); //Средний процент влажности

lvAveraga.Items.Add($"Средняя влажность(%): {averagaHumidity}");

lvNow.Items.Add($"Текущая влажность(%): {PercentageMoisture}");

Draw();

}

private void Draw()

{

Graphik.Children.Clear();

double width = Graphik.ActualWidth;

double height = Graphik.ActualHeight;

if (\_Humidity.Count == 0) return;

double maxHumidity = 80;

double scale = height / maxHumidity;

for (int i = 1; i < \_Humidity.Count; i++)

{

double x1 = (width / (\_Humidity.Count - 1)) \* (i - 1);

double y1 = height - (\_Humidity[i - 1] \* scale);

double x2 = (width / (\_Humidity.Count - 1)) \* i;

double y2 = height - (\_Humidity[i] \* scale);

Line line = new Line

{

X1 = x1,

Y1 = y1,

X2 = x2,

Y2 = y2,

Stroke = System.Windows.Media.Brushes.Blue,

StrokeThickness = 2

};

Graphik.Children.Add(line);

AddHumidityLabel(x2, y2, \_Humidity[i]);

}

}

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

{

if (tbP.Text == "0")

{

\_timer.Stop();

tbP.Text = "1";

btnPause.Content = "Возобновить";

}

else if (tbP.Text == "1")

{

\_timer.Start();

tbP.Text = "0";

btnPause.Content = "Пауза";

}

}

private void AddHumidityLabel(double x, double y, double humidity)

{

TextBlock label = new TextBlock

{

Text = $"{humidity:F1}%", // Форматируем текст до одного знака после запятой

Foreground = Brushes.Black,

FontSize = 12,

Margin = new Thickness(x + 5, y - 10, 0, 0) // Сдвигаем метку немного вправо и вверх

};

Graphik.Children.Add(label);

}

}