МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до курсової роботи

з дисципліни

«Програмування під платформу .Net Framework»

Виконали: студенти групи ІТ-91

Акименко В.В.

Шальнова Р.Ю.

Варіант: 1

Перевірив: Нагорний В.В

Суми

2021

**ЗМІСТ**

[ВСТУП 3](#_Toc72448728)

[1 ПОСТАНОВКА ЗАДАЧІ 4](#_Toc72448729)

[1.1 Мета роботи 4](#_Toc72448730)

[1.2 Методи та інструменти реалізації 4](#_Toc72448731)

[2 МОДЕЛЬ ПРОГРАМНОГО ПРОДУКТУ 6](#_Toc72448732)

[3 РЕАЛІЗАЦІЯ ПЗ 17](#_Toc72448733)

[3.1 Опис даних 20](#_Toc72448734)

[3.2 Опис файлів та їх призначення 20](#_Toc72448735)

[4 ОПИС ВИКОРИСТАНИХ ПРОЦЕДУР ТА ФУНКЦІЙ КОРИСТУВАЧА 22](#_Toc72448736)

[5 ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ 25](#_Toc72448737)

[1.1 Тестування програми 25](#_Toc72448738)

[1.2 Результат роботи програми 26](#_Toc72448739)

[ВИСНОВКИ 29](#_Toc72448740)

[СПИСОК ЛІТЕРАТУРИ 30](#_Toc72448741)

[ДОДАТОК А 31](#_Toc72448742)

[ДОДАТОК Б 34](#_Toc72448743)

[ДОДАТОК В 35](#_Toc72448744)

[ДОДАТОК Г 50](#_Toc72448745)

[ДОДАТОК Д 53](#_Toc72448746)

# ВСТУП

В роботі була поставлена задача розроблення застосунку із записами про погоду для подальшої роботи з ними, а також реалізації двох видів доступу до наявної інформації – форму співробітника компанії, яка дозволить редагувати список погодних записів, а саме редагувати окремі записи, видаляти та додавати нові записи, а також форму користувача, яка дозволить фільтрувати список записів, створювати звіти та будувати графіки.

Всі записи повинні зберігатися за допомогою системи управління базами даних MySQL, що буде інтегрована в програмний застосунок.

Дане завдання є наглядним та актуальним в сучасному світі, адже всі виробництва та центри обробки певної інформації з кожним роком все більше автоматизуються та потребують зручного та функціонального інтерфейсу для керування даними.

Навики та знання, що були отримані у ході виконання даного завдання будуть корисними в подальшій реалізації складних та практичних програмних застосунків, що будуть використовуватися для вирішення різноманітних задач, включаючи зберігання та обробку фіксованих даних.

# ПОСТАНОВКА ЗАДАЧІ

## Мета роботи

Загальною метою курсової роботи є створення програмного застосунку для роботи із записами про погоду, реалізувавши можливість редагування даних, створення звітів та побудову графіків.

Для реалізації поставленої мети необхідно створити відповідні форми доступу до програмного застосунку – форму співробітника, яка дозволить редагувати, видаляти, а також додавати інформацію до списку записів, та форму звичайного користувача, яка дозволить створювати звіт, сортувати та фільтрувати записи, а також будувати графік за містом.

В звіт, створений в MS Word необхідно записувати кількість та всі дати, коли температура була вище нуля і йшов дощ, а також середньомісячну температуру та середній тиск по місту XX та місяцю YY.

Для вирішення поставленої задачі вхідними даними є опис необхідної структури програмного застосунку, структура даних, а також файл шаблону звіту MS Word. При цьому вихідними даними є програмний застосунок, а також створений звіт за допомогою MS Word.

На основі вхідних даних було створене завдання на розробку програмного забезпечення (Додаток А).

## Методи та інструменти реалізації

Для реалізації поставленої задачі були використані такі методи та інструменти реалізації, як:

1. методологія Scrum, яка застосовувалась для гнучкої розробки програмного забезпечення та створення акценту на якісному контролі процесу розробки;
2. уніфікований процес розробки програмного забезпечення (Rational Unified Process) для чіткої структуризації та планування розробки програми шляхом поділу його на фази;
3. мова програмування C#, як одна із найбільш гнучких мов, що застосовується для вирішення безлічі сучасних задач програмування;
4. WPF – гнучка технологія створення графічного інтерфейсу користувача;
5. середовище розробки програмного забезпечення IDE Visual Studio;
6. система керування реляційними базами даних MySQL, як система, яка надає зручний та зрозумілий інтерфейс управління базою даних;
7. система контролю версій Git, яка є невід’ємним атрибутом розробки будь-якого програмного застосунку, адже дає можливість перевірки змін та повернення до попередньої стабільної версії.

# МОДЕЛЬ ПРОГРАМНОГО ПРОДУКТУ

На основі методології Уніфікованого процесу розроблення програмних застосунків (Rational Unified Process) була розроблена повна модель ПЗ. До основних елементів моделі відносяться моделі варіантів використання, до складу якої входять опис акторів (табл. 2.1), опис варіантів використання програмного продукту (табл. 2.2) та діаграма варіантів використання;

Таблиця 2.1 - Список акторів

|  |  |
| --- | --- |
| **Назва актора** | **Короткий опис** |
| Актори-користувачі | |
| Editor | робітник компанії, якій може за допомогою програми редагувати файл БД cloudy.sql із інформацією про погоду компанії «Cloudy» |
| Customer | клієнт, який бажає переглянути інформацію про погоду за допомогою програми |
| Актори - зовнішні системи | |
| MySQL | СКБД MySQL для керування БД із інформацією про погоду компанії «Cloudy» |
| MS Word | програма MS Word для збереження файлу із результатами вибору клієнтом інформації про погоду компанії «Cloudy» |

Таблиця 2.2 - Опис варіантів використання

|  |  |
| --- | --- |
| **Назва ВВ** | **Сценарій виконання ВВ** |
| UC01 Log In | Editor зацікавлений в авторизації.  UC01.1 Програма працює, відкрите головне вікно програми "Інформація про погоду" із інтерфейсом Customer.  UC01.2 Editor у головному меню натискає кнопку «Вхід». Відкривається форма "Авторизація". Editor вводить свій логін і пароль та запускає перевірку даних (субпотоки UC01.3-1 та UC01.3-2). Після успішної авторизації форма "Авторизація" закривається, інтерфейс головної форми змінюється на інтерфейс Editor.  UC01.3 Запуск перевірки введених логіну і паролю: UC01.3-1 Натиснувши кнопку "Enter" на клавіатурі UC01.3-2 Натиснувши кнопку "Вхід" у формі "Авторизація".  UC01.4 Якщо дані авторизації введені не вірно, користувач отримує відповідне повідомлення. Щоб закрити вікно авторизації потрібно натиснути кнопку "Скасувати" в формі "Авторизація".  UC01.5 Відкрита головна форма програми "Інформація про погоду". Якщо авторизація пройшла успішно показаний інтерфейс Editor, інакше інтерфейс Customer. |
| UC02 Load Data | Editor, Customer зацікавлені у перегляді даних, котрі зберігаються в БД cloudy.sql.  UC02.1 БД cloudy.sql повинна бути розміщена на сервері БД MySQL. Editor або Customer запустив виконуваний файл програми "Записи".  UC02.2 Програма зчитує із БД cloudy.sql дані про записи, які вона показує у табл.ичній формі на головній формі програми "Інформація про погоду".  UC02.3 Відсутні.  UC02.4 Якщо підключитись до БД не вдалось або дані не можуть бути зчитані, користувач отримує повідомлення про це  UC02.5 Відкрита головна форма програми "Інформація про погоду" із інтерфейсом Customer (див. Прототип інтерфейсу), де список записів заповнений даними із БД cloudy.sql у формі списку. |
| UC03 Edit Data | Editor зацікавлений в редагуванні інформації про записи в БД cloudy.sql.  UC03.1 Програма працює. UC01 виконаний. Відкрите головне вікно програми "Інформація про погоду" із інтерфейсом Editor. Editor має дані для оновлення інформації про певний запис у БД.  UC03.2 Editor:  або у головному меню на формі "Інформація про погоду" обирає у списку записів на формі "Інформація про погоду" рядок для редагування. Ці записи вносяться у поля блоку для редагування на формі "Інформація про погоду". Editor редагує інформацію та натискає кнопку "Зберегти". Відредаговані дані про відповідний запис переносяться у список записів форми "Інформація про погоду" та записуються у БД cloudy.sql.  UC03.3 Відсутні  UC03.4 Якщо змінені дані не можуть бути записані, користувач повідомляється про це.  UC03.5 Список записів (на формі "Інформація про погоду" та у БД cloudy.sql) містить оновлені дані про відповідний запис. |
| **Назва ВВ** | **Сценарій виконання ВВ** |
| UC04 Add Data | Editor зацікавлений у додаванні запису даних про новий запис у список записів БД cloudy.sql.  UC04.1 Програма працює. UC01 виконаний. Відкрите головне вікно програми "Інформація про погоду" із інтерфейсом Editor. Editor має дані про новий запис. Жоден запис у списку записів на формі "Інформація про погоду" не вибраний.  UC04.2 Editor у головному меню на формі "Інформація про погоду" виконує команду "Додати". Editor вносить дані про запис у поля блоку для редагування на формі "Інформація про погоду" та натискає кнопку Зберегти. Дані про новий запис записуються у кінець списку записів форми "Інформація про погоду" та БД cloudy.sql (альтернативний потік UC04.4)  UC04.3 Відсутні  UC04.4 Якщо запис не може бути доданий, користувач повідомляється про це.  UC04.5 Список записів (на формі "Інформація про погоду" та у БД cloudy.sql) останнім рядком містить дані про новий запис. |
| UC05 Select XY Data | Customer зацікавлений у перегляді записів по місту Х за місяць Y.  UC05.1 Програма працює. Відкрите головне вікно програми "Інформація про погоду" із інтерфейсом  UC05.2 Customer обирає місто Х та місяць Y з випадаючих списків, у формі "Інформація про погоду". Customer натискає кнопку Пошук. Програма відбирає записи по місту Х за місяць Y (альтернативний потік UC05.4). Оновлений список записів, виводиться у формі "Інформація про погоду".  UC05.3 Відсутні  UC05.4 Якщо у списку по місту Х немає записів за місяць Y, користувач про це повідомляється, список записів на формі "Інформація про погоду" не оновлюється.  UC05.5 Список записів по місту Х за заданим Customer місяцем Y виведений на формі "Інформація про погоду" замість списку записів по місту Х. Список записів до заданого Customer міста Х збережений у пам'яті програми. |
| UC06 Select Rain Data | Customer зацікавлений у перегляді записів, коли температура була більша за 0 та йшов дощ.  UC06.1 Програма працює. UC02 виконаний. Відкрите головне вікно програми "Інформація про погоду" із інтерфейсом Customer.  UC06.2 Customer у головному меню на формі "Інформація про погоду" натискає кнопку "Пошук дощових днів". Програма відбирає список записів, коли температура була більша за 0 та йшов дощ. Оновлений список записів, виводиться у формі "Інформація про погоду".  UC06.3 Відсутні  UC06.4 Якщо у списку записів немає записів, коли температура була більша за 0 та йшов дощ, користувач про це повідомляється, список записів на формі "Інформація про погоду" не оновлюється.  UC06.5 Список записів, коли температура була більша за 0 та йшов дощ виведений на формі "Інформація про погоду" замість списку всіх записів із БД. Список записів, коли температура була більша за 0, збережений у пам'яті програми. |
| **Назва ВВ** | **Сценарій виконання ВВ** |
| UC07 Write Data | Customer зацікавлений у запису інформації, яка була отримана при відборі записів (UC05 та UC06), у звіт, сформований на основі шаблону Template MS Word (Шаблон\_Пошуку.dotx).  UC07.1 Програма працює. Відкрите головне вікно програми "Інформація про погоду" із інтерфейсом Customer. UC05 або UC06 виконаний. Список записів до заданого Customer міста Х та місяцем Y, а також (якщо був виконаний UC06) список записів з дощовими днями збережені у пам'яті програми.  UC07.2 Customer натискає кнопку "Створити звіт". Списки записів із пам'яті програми записуються у файл формату Word, сформований на основі шаблону Template MS Word (Шаблон\_Пошуку.dotx) (альтернативні потоки UC07.4-1 та UC07.4-2).  UC07.3 Відсутні  UC07.4 UC07.4-1 Якщо файл для запису списків не може бути створений, користувач повідомляється про це. UC07.4-2 Якщо файл із записаними списками не може бути збережений, користувач повідомляється про це.  UC07.5 Список записів по місту Х за місяцем Y збережені у звіті на основі шаблону Template MS Word. |
| UC08 Create Chart | Customer зацікавлений у створенні візуального графіку за даними, що зберігаються на сервері  UC08.1 Програма працює. Відкрите головне вікно програми "Інформація про погоду" із інтерфейсом Customer. UC02 виконаний.  UC08.2 Customer натискає кнопку "Графік". Відкривається вікно форми «Графік», в якому користувач обирає необхідне місто та тип даних, за якими необхідно створити графік. Графік створюється автоматично  UC08.3 Відсутні  UC08.4 Відсутні  UC08.5 Графік відображений у формі. |

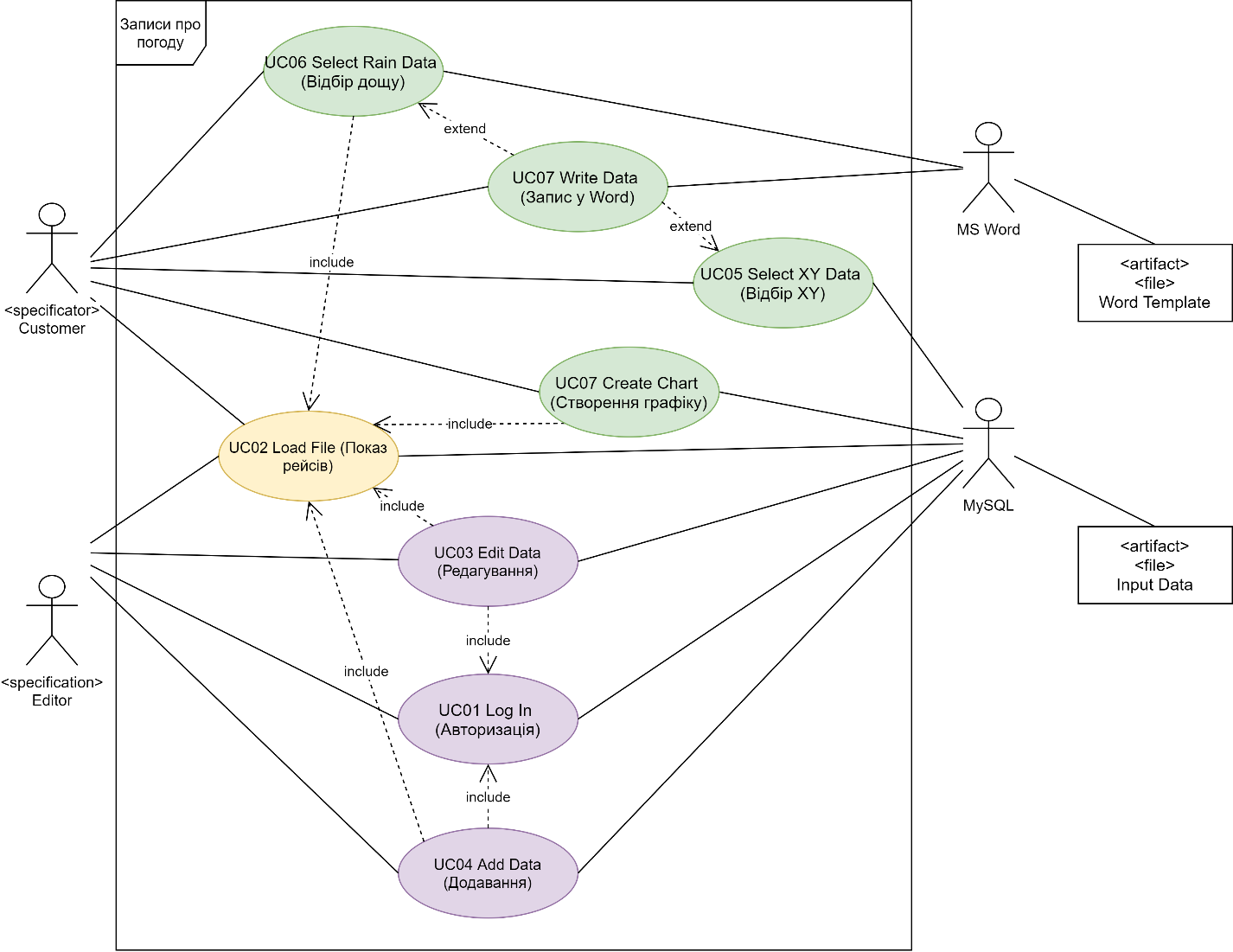


Рисунок . - Діаграма варіантів використання

Також до моделі додається список вимог до програмного забезпечення (табл. 2.3).

Таблиця 2.3 - Вимоги до ПЗ

| **ID** | **SR** | **Опис** | **Метод верифікації** |
| --- | --- | --- | --- |
| SR-SYS-0100 | Аутентифікація користувача-робітника компанії | Визначення прав доступу користувача до додавання та редагування записів у БД | D |
| SR-SRS-0200 | Завантаження та відображення записів про погоду | Завантаження записів із БД cloudy.sql і відображення списку записів у головній формі програми | D |
| SR-SRS-0300 | Додавання записів | Можливість користувачу-робітнику додавати до БД cloudy.sql нових записів про погоду із зазначенням температури повітря, тиску, наявності опадів, дати, міста і т.д. | T |
| SR-SRS-0400 | Редагування записів | Можливість користувачу-робітнику змінити інформацію обраного запису у формі програми із записом (або видаленням) у БД cloudy.sql | T |
| SR-SRS-0500 | Вибір записів за місяцем і містом | Можливість користувачу-клієнту переглянути у формі програми список записів за відповідним містом у відповідному місяці | T |
| SR-SRS-0600 | Вибір записів, коли температура повітря була більша за 0 та йшов дощ. | Можливість користувачу-клієнту переглянути у формі програми список записів коли температура повітря була більша за 0 та йшов дощ | T |
| SR-SRS-0700 | Формування звіту | Можливість користувачу-клієнту зберегти результати відбору інформації та записів у файл MS Word на основі шаблону Шаблон\_Пошуку.dotx | D |
| SR-SRS-0800 | Створення графіку | Можливість користувачу-клієнту графічно відобразити дані в виді графіку | D |

Графічно програмне забезпечення було зображене за допомогою ескізу інтерфейсу, який складається з форми «Авторизація» (рис. 2.2) та головної форми «Інформація про погоду» (рис 2.3).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок . - Вікно авторизації

Изображение выглядит как стол

Автоматически созданное описание

Рисунок . - Інтерфейс співробітника

Головна форма також має інтерфейс для користувача (рис. 2.4), який надає можливість переглядати записи, що існують в базі, а також створювати графік та звіт.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок . - Інтерфейс користувача

Для роботи з графіком наявна форма «Графік» (рис. 2.5).

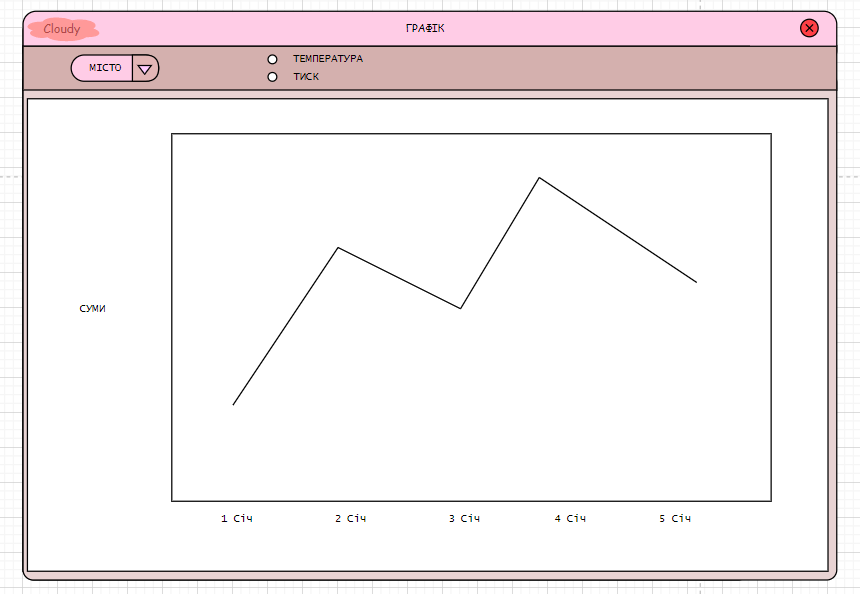


Рисунок . - Форма "Графік"

Також до моделі ПЗ були включені моделі проектування із описом класів (табл. 2.4), інтерфейсів (табл. 2.5).

Таблиця 2.4 - Класи програми

|  |  |  |  |
| --- | --- | --- | --- |
| **Клас** | **Короткий опис** | **Атрибути** | **Операції** |
| Weather | Головний клас програми. Усі атрибути класу знаходяться в зоні видимості private. | string city, string month, int day, int temperature, string precipitation, uint pressure | Weather(string city, date month, date day, short temperature, bool precipitation, ushort pressure) – ініціалізація полів запису; |
| EditData | Клас редагування (додавання, зміна інформації, видалення) інформації про записи в БД MySQL | - | bool Add() – додавання запису до БД;  bool Delete() – видалення обраного запису з БД;  Weather Edit(Weather selected) – редагування обраного запису |
| Authorization | Клас, який дозволяє Editor авторизуватися для доступу до розширеного функціоналу | bool loged\_in | bool Check\_LogIn() |
| DataAccess | Клас завантаження даних с серверу. Включає в себе методи SelectXY для відбору записів по заданому місту у обраний місяць (month). | string connectionString | List<Weather> Get\_Weathers ()  List<Weather> SelectXY– відбір записів по заданому місту у обраний місяць  bool WriteData(Weather weather) - операція запису/дозапису інформації про запис |
| SelectData | Клас запису відібраної інформації в файл MS Word | - | bool WriteData(List<Weather> selectXY, List<Weather> rainData) – метод запису до файлу MS Word інформації |
| Chart | Клас побудови графіків за відповідними параметрами, заданими користувачем |  | void Info\_Changed(string city, string data) – метод зміни графіку залежно від вибору користувача |

Таблиця 2.5 - Інтерфейси програми

|  |  |  |
| --- | --- | --- |
| **Інтерфейс** | **Короткий опис** | **Опис інтерфейсу** |
| InfoWeatherForm | Компонент-головна форма програми | Обробка подій елементів користувацького інтерфейсу |
| LogInForm | Компонент-форма для авторизації | Обробка подій авторизації |
| Chart\_Form | Компонент-форма для побудови графіків | Обробка подій побудови графіків |

Для графічного зображення класів та інтерфейсів були побудовані діаграми класів (рис. 2.6 – 2.8)

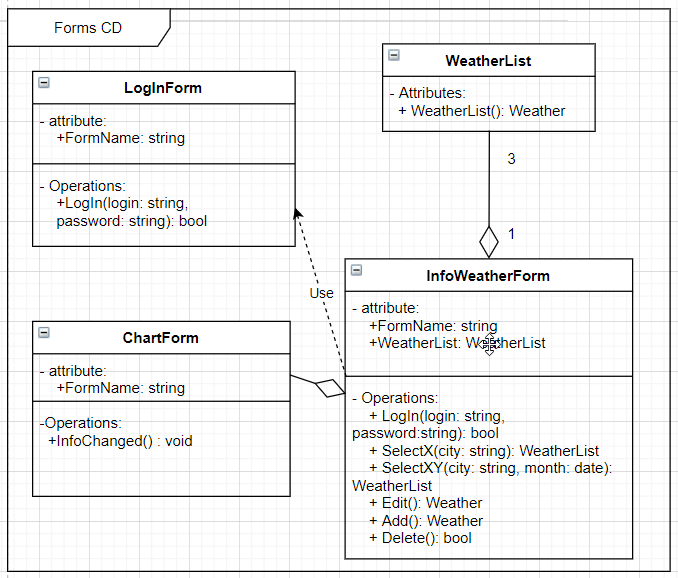


Рисунок . - Діаграма класів із формами програми

Изображение выглядит как стол

Автоматически созданное описание

Рисунок . - Діаграма класів із меню користувача

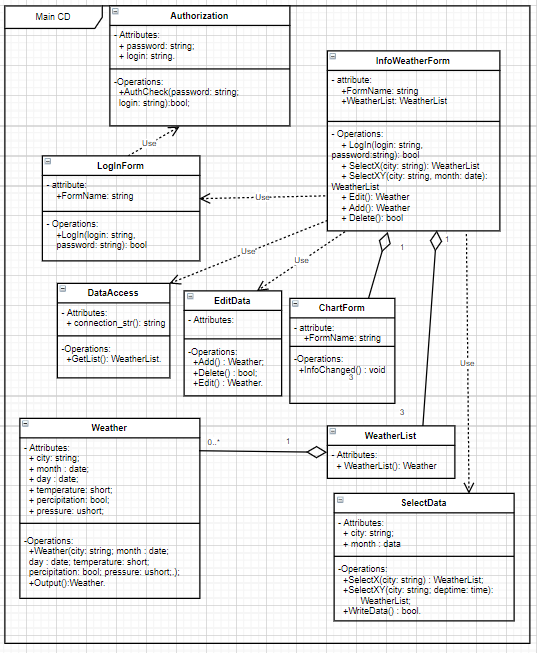


Рисунок . - Діаграма класів Main

# РЕАЛІЗАЦІЯ ПЗ

Розробка програмного забезпечення проводилася в середовищі MS Visual Studio на мові програмування C# з використанням WPF – гнучкої технології створення графічного інтерфейсу користувача (рис. 3.2).

Сама розробка почалася зі створення графічних інтерфейсів головної форми програми (рис. 3.1), а також форми авторизації (рис. 3.2).

Изображение выглядит как текст, компьютер, внутренний, снимок экрана

Автоматически созданное описание

Рисунок . - Головна форма

Изображение выглядит как текст, монитор, внутренний, компьютер

Автоматически созданное описание

Рисунок . - Форма авторизації

Наступним кроком був написаний код реалізації програмного застосунку з прив’язкою до вже створених форм (рис. 3.3 – 3.4).

Изображение выглядит как текст, монитор, снимок экрана, компьютер

Автоматически созданное описание

Рисунок . – Головна логіка програмного забезпечення

Изображение выглядит как текст, монитор, снимок экрана, внутренний

Автоматически созданное описание

Рисунок . - Логіка авторизації

Після цього була створена форма та написаний код для неї для додаткового створення графіку за обраними користувачем вхідними даними (

Изображение выглядит как текст, компьютер, монитор, внутренний

Автоматически созданное описание

Рисунок . - Форма створення графіків

## Опис даних

Для зберігання та обробки вхідних даних використовувалися змінні, наведені в таблиці 3.1.

Таблиця 3.1 - Змінні та їх характеристики

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип** | **Діапазон** |
| City | String | 0..50 символів |
| Month | String | 0..10 символів |
| Date | Int | 0..31 |
| Temperature | Int | -60..50 |
| Pressure | Int | 700..800 |
| Precipitation | String | «Є», «Нема» |
| Weathers | List<Weather> | 0..∞ елементів |
| selectedWeather | Weather | - |
| CityList | List<string> | 0..∞ елементів |
| Login | String | 0..50 символів |
| Password | String | 0..50 символів |
| SelectedCity | String | 0..50 символів |
| SelectedMonth | String | 0..50 символів |

## Опис файлів та їх призначення

До списку необхідних для роботи файлів програми належать 13 файлів (табл. 3.2).

Таблиця 3.2 - Файли, необхідні для роботи програми

|  |  |
| --- | --- |
| **Файл** | **Призначення** |
| Authorization.cs | Авторизація користувача |
| Chart.xaml | Інтерфейс форми «Графік» |
| Chart.xaml.cs | Реалізація побудови графіку |
| DataAccess.cs | Доступ до бази даних |
| Globals.cs | Клас, який зберігає інформацію про запис |
| Icon.ico | Іконка програми |
| Image.png | Зображення, що виводиться в формі користувача |
| InfoWeatherForm.xaml | Інтерфейс головної форми програми |
| InfoWeatherForm.xaml.cs | Реалізація головної форми програми |
| LogInForm.xaml | Інтерфейс авторизації |
| LogInForm.xaml.cs | Реалізація авторизації |
| SelectData.cs | Запис обраної інформації до файлу MS Word |
| Шаблон\_Пошуку.dotx | Файл-шаблон для створення звітів MS Word |

# ОПИС ВИКОРИСТАНИХ ПРОЦЕДУР ТА ФУНКЦІЙ КОРИСТУВАЧА

В процесі розроблення програмного застосунку були створені процедури та функції (Додаток В), що направлені на відслідковування дій користувача, а також завантаження та редагування даних БД MySQL (табл. 4.1 – 4.2).

Таблиця 4.1 - Використані процедури та функції користувача

|  |  |
| --- | --- |
| **Процедура або функція** | **Призначення** |
| CheckLogIn | Перевірка введеної користувачем інформації для авторизації |
| ToggleDisplay | Зміна відображення головної форми |
| ErrorShow | Виведення повідомлення |
| ClearList | Очищення табл.иці з записами |
| ClearEditForm | Очищення форми редагування |
| UserInputOk | Перевірка вводу користувача на коректність |
| DisplayList | Відобразити певний список в формі |
| MainWindow\_Loaded | Заповнення форми після завантаження головного вікна |
| LoadBase | Завантаження всіх записів з БД |
| AddToList | Додавання запису до загального списку записів |
| DeleteWeather | Видалення обраного користувачем запису зі списку |
| AddWeather | Процес додавання запису до БД |
| SaveButton\_Click | Зберігання запису після натискання кнопки користувачем |
| LogInButton\_Click | Відкриття форми авторизації після натискання кнопки авторизації |
| DeleteButton\_Click | Видалення запису після натискання користувачем кнопки видалення |
| SearchButton\_Click | Пошук записів за введеною користувачем інформацією після натискання кнопки пошуку |
| SearchRainButton\_Click | Пошук записів з дощем після натискання користувачем кнопки пошуку дощових днів |
| ReportButton\_Click | Створення звіту за знайденою інформацією для користувача після натискання кнопки створення звіту |
| RefreshButton\_Click | Відображення початкової бази усіх записів |
| WeatherTable\_SelectionChanged | Заповнення форми редагування даними обраного запису |
| ChartButton\_Click | Запуск вікна для побудови графіку |
| cityList\_SelectionChanged | Зміна виду графіку в залежності від вибору користувача |
| radioButton\_Checked | Зміна виду графіку в залежності від вибору користувача |
| ChangeChart | Зміна даних графіку |
| **Процедура або функція** | **Призначення** |
| GetList | Отримання списку з БД за командою |
| GetWeathers | Отримання всіх записів з БД |
| AddToBase | Додавання запису до БД |
| DeleteFromBase | Видалення запису з БД |
| SelectXY | Отримання записів за містом та місяцем |
| GetRainData | Отримання записів з дощем |
| WriteData | Запис отриманої інформації в файл MS Word |
| ReplaceText | Заміна тексту відповідними значеннями |
| AverageTemp | Середнє значення температури в записах |
| AveragePres | Середнє значення тиску в отриманих записах |

Таблиця 4.2 - Вхідні та вихідні параметри процедур та функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Процедура або функція** | **Вхідні параметри** | | **Вихідні параметри** | | |
| **Позначення** | **Призначення** | **Тип** | **Призначення** | | |
| CheckLogIn | login (string) | Логін | bool | Успішність авторизації | | |
| password (string) | Пароль |
| ToggleDisplay | - | | - | | |
| ErrorShow | message (string) | Повідомлення | - | | | |
| ClearList | - | | - | | |
| ClearEditForm | - | | - | | |
| UserInputOk | - | | - | | |
| DisplayList | List (List<Weathers>) | Список записів | - | | | |
| MainWindow\_Loaded | sender (object) | Об’єкт, що викликав метод | - | | | |
| LoadBase | - | | bool | Успішність завантаження | |
| AddToList | weather (Weather) | Запис, що додається | bool | Успішність додавання | | |
| DeleteWeather | weather (Weather) | Запис, що видаляється | bool | Успішність видалення | | |
| AddWeather | - | | - | | |
| SaveButton\_Click | sender (object) | Об’єкт, що викликав метод | - | | | |
| LogInButton\_Click | sender (object) | Об’єкт, що викликав метод | - | | | |
| DeleteButton\_Click | sender (object) | Об’єкт, що викликав метод | - | | | |
| SearchButton\_Click | sender (object) | Об’єкт, що викликав метод | - | | | |
| SearchRainButton\_Click | sender (object) | Об’єкт, що викликав метод | - | | | |
| ReportButton\_Click | sender (object) | Об’єкт, що викликав метод | - | | | |
| **Процедура або функція** | **Вхідні параметри** | | **Вихідні параметри** | | | |
| **Позначення** | **Призначення** | **Тип** | | **Призначення** | |
| RefreshButton\_Click | sender (object) | Об’єкт, що викликав метод | - | | | |
| WeatherTable\_SelectionChanged | sender (object) | Об’єкт, що викликав метод | - | | | |
| ChartButton\_Click | sender (object) | Об’єкт, що викликав метод | - | | | |
| cityList\_SelectionChanged | sender (object) | Об’єкт, що викликав метод | - | | | |
| radioButton\_Checked | sender (object) | Об’єкт, що викликав метод | - | | | |
| ChangeChart | - | | - | | |
| GetList | command (string) | Команда | List <Weathers> | Записи | | |
| GetWeathers | - | | List <Weathers> | Записи | |
| AddToBase | weather (Weather) | Запис | Int | Результат додавання | | |
| DeleteFromBase | weather (Weather) | Запис | bool | Результат видалення | | |
| SelectXY | City (string) | Місто | List | Записи | | |
| Month (string) | Місяць |
| GetRainData | - | | List <Weathers | Список записів | |
| WriteData | selectXY (List) | Обрані записи по місту та місяцю | - | | | |
| rainData (List) | Записи з дощем |
| ReplaceText | textToReplace (string) | Текст, що заміняється | - |  | | |
| replacedText (string) | Текст, яким заміняємо |
| ReplaceText | rainData (List<Weathers>) | Записи з дощем | - |  | | |
| AverageTemp | weathers (List<Weathers>) | Записи | double | Середнє значення температури | | |
| AveragePres | weathers (List<Weathers>) | Записи | double | Середнє значення тиску | | |

Для правильного встановлення програмного забезпечення на пристрій необхідно скористатися інструкцією системного адміністратора (Додаток Г) [3].

Для розуміння можливостей програмного забезпечення необхідно скористатися інструкцією користувача (Додаток Д) [4].

# ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ

## Тестування програми

Для тестування роботи програмного забезпечення було використане функціональне тестування, а саме написання модулів для тестування (табл. 5.1), а також тестування інтерфейсу користувача.

Таблиця 5.1 - Тест-кейси

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер кроку | Ім’я кроку | Дії | ER – що очікується | Статус  (пройдений/не пройдений) |
| 1 | Тест методу SearchXY(…) | Запуск тесту методу SearchXY(…) при значенні параметрів  city = "Суми"  month = "Січень" | Список в якому 2 елементи типу Weather:  expected[0] = new Weather("Суми", 4, "Січень", -8, "Є", 915)  expected[1] new Weather("Суми", 1, "Січень", -10, "Є", 920) | Пройдений |
| 2 | Тест методу SearchRainData(…) | Запуск тесту методу SearchRainData(…) | Список в якому 2 елементи типу Weather:  expected[0] = new Weather("Краснопілля", 8, "Січень", 1, "Є", 900),  expected[1] = new Weather("Львів", 5, "Лютий", 1, "Є", 900) | Пройдений |
| 3 | Тест методу CityListFill () | Запуск тесту методу CityListFill () | Список назв міст:  expected[0] = "Краснопілля",  expected[1] = "Львів",  expected[2] = "Суми",  expected[3] = "Київ" | Пройдений |

Тестування за допомогою написання модулів було пройдено успішно (рис. 5.1).

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание

Рисунок . - Результат тестування

## Результат роботи програми

Робота з програмним забезпеченням починається з завантаження головної форми користувача (рис. 5.2).

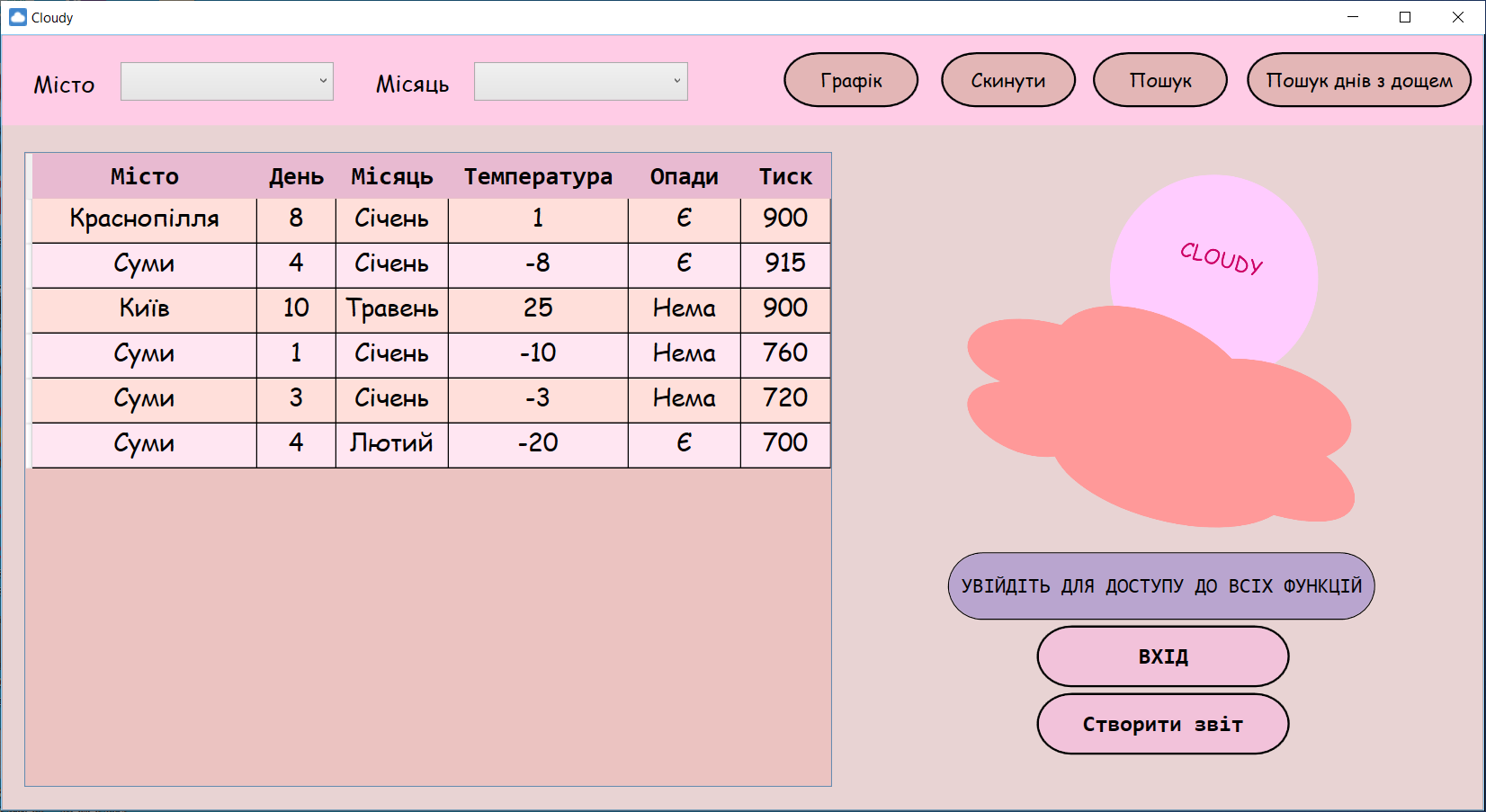


Рисунок . - Головна форма користувача

Для доступу до редагування бази необхідно авторизуватися (рис. 5.3).

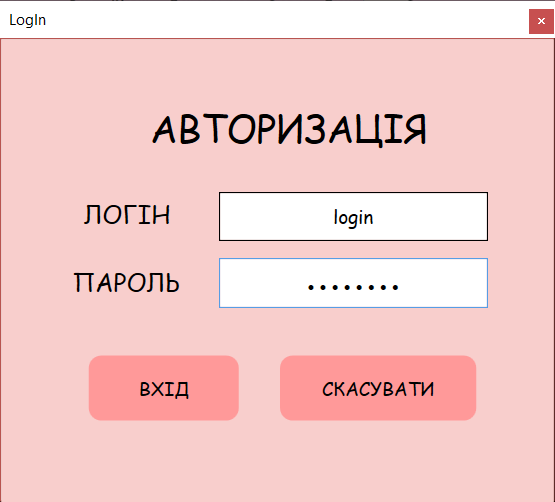


Рисунок . - Авторизація

У разі вдалої авторизації вид головної форми зміниться (рис. 5.4).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок . - Головна форма співробітника

При натисканні на кнопці «Графік» відкриється вікно, в якому можна створити графік за відповідними даними по місту (рис. 5.5).

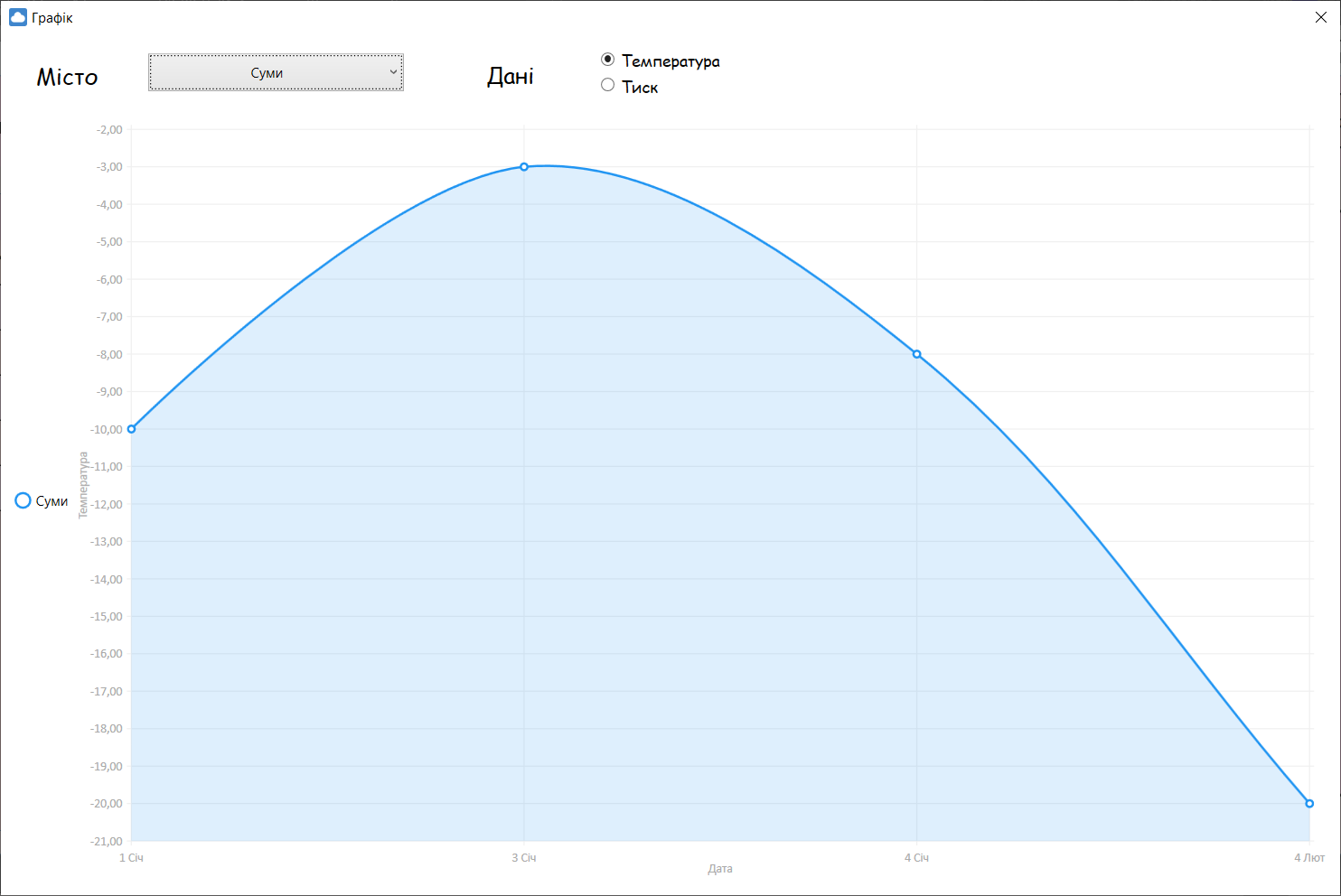


Рисунок . - Побудова графіку

# ВИСНОВКИ

Виконуючи дану курсову роботу ми дослідили та застосували на практиці методи створення програмних застосунків із інтерфейсом користувача, а також роботу з базами даних за допомогою системи управління базами даних MySQL, що є практичними знаннями в сучасному світі та може бути використане при розробці будь-якого програмного застосунку, не залежно від його складності.

В роботі була реалізована задача розроблення застосунку із записами про погоду для подальшої роботи з ними.

Весь цей функціонал був реалізований за допомогою додавання можливості авторизації користувача, яка дозволяє розділити стиль роботи з базою даних на два види – для співробітника, який отримує можливість повністю редагувати базу, та звичайного користувача, який має доступ до корисних функцій, таких як створення звітів та побудова графіків.

Всі записи зберігаються на локальному сервері, який в будь-який момент може бути перенесений на звичайний сервер, що полегшує роботу і доступ до бази даних та робить можливим роботу програми без наявності самої бази даних на пристрої користувача.

# СПИСОК ЛІТЕРАТУРИ

1. Энрю Троелсен. Язык программирования C# 5.0 и платформа .Net 4.5, 6-е изд: Пер. с англ. –М.: ООО “И.Д. Вильямс", 2013. – 1312 с.
2. C# 2005 и платформа .Net 3.0 для профессионалов. Нейгел К., Ивьен Б. и др. –М.: ООО “И.Д. Вильямс", 2008. –1376 с.
3. Deployment Package Needs and Requirements Engineering – Systems Engineering Basic Profile. 2014.
4. Сэм Канер. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бізнес-приложений/ Сэм Канер, Джек Фолк, Енг Кек Нгуен; Пер. с англ. – К.: Изд-во «ДиаСофт», 2001.
5. Техническое задание. Требования к содержанию и оформлению. ГОСТ 19.201-78.\
6. Руководство системного программиста. Требования к содержанию и оформлению. ГОСТ 19.503-79
7. Руководство оператора. Требования к содержанию и оформлению. ГОСТ 19.505-79
8. Дж. Раскин. Интерфейс новые направления в проектировании компьютерных систем. Влад Головач. Дизайн пользовательского интерфейса. Искусство мыть слона. - [Електронний ресурс]. Режим доступу: <http://uibook2.usethics.ru>.
9. API - [Електронний ресурс] // Вікіпедія – URL: <https://ru.wikipedia.org/wiki/API>
10. Тестирование программного обеспечения - [Електронний ресурс] // Вікіпедія – URL: <https://ru.wikipedia.org/wiki/Тестирование_программного_обеспечения>

# ДОДАТОК А

**ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Розробити програму, яка читає із файлу БД MySQL довідки про погодні умови (табл. А.1)

Таблиця А.1 - Структура запису про погоду

|  |  |  |
| --- | --- | --- |
| **Дані** | **Назва поля у БД** | **Формат поля у БД** |
| Місто | City | Text(50) |
| Місяць | Month | Text(10) |
| День | Day | Int |
| Температура | Temperature | Int |
| Наявність опадів | precipitation | Byte(1) |
| Тиск | Pressure | Int unsigned |

Користувач – робітник метеостанції може додавати записи у файл про погоду та редагувати наявну інформацію у формі програми. Для доступу до функцій роботи із записами робітник комнанії має авторизуватись (пункт головного меню)

Користувач – клієнт може передивлятись в головній формі програми перелік усіх записів, будувати графіки, а також виконувати пошук інформації:

- середньомісячна температура та середній тиск за місяць Х в місті Y,

- к-ть днів, коли температура була >0 і йшов дощ,

- список дат, коли температура була >0 і йшов дощ.

Значення X, Y вводяться користувачем у відповідні поля головної форми.

Результати відбору інформації користувач – клієнт може зберегти у файл MS Word, який формується на основі шаблону Шаблон\_Пошуку.dotx.

Обмеження проекту

- Тривалість проекту із розроблення ПП – з 26.01.2021р. до 27.04.2021р.

- Програма має бути написана мовою C# у IDE Visual Studio із використанням фреймворку UI [Windows Presentation Foundation (WPF)].

- Проектування має спиратися на моделі процесів компанії у нотації IDEF0+BPMN та моделі програмного продукту відповідно до RUP у нотації UML.

- Реалізація програми буде вестись методом Scrum. Product Backlog буде поділений на 2 спринта тривалістю по 2 тижні.

- По завершенні проекту програма повинна виконувати всі передбачені функції, мати документацію, перелік якої наведений у розділі \*\*Склад програмної документації\*\*.

Склад програмної документації.

1. \*\*Специфікація вимог\*\* (за шаблоном [Requirement Document])

2. \*\*Технічний проект ПЗ\*\*, який складається із:

- структурно-функціональної моделі процесу у нотації IDEF0, до складу якої мають входити:

+ контекстна діаграма

+ діаграма декомпозиції 1-го рівня

- моделі процесу у нотації BPMN

- моделей RUP у нотації UML:

+ модель варіантів використання

+ модель аналізу

+ модель проектування

+ модель реалізації ПЗ

3. \*\*Інструкції адміністратора\*\* (щодо встановлення програми) та \*\*інструкції користувача\*\*.

Обмеження проектної команди

У проекті приймають участь 2 спеціалісти: усі учасники мають досвід написання коду, із них 1 - має досвід проектування ПЗ, 1 – має досвід створення документації до програми та раніше приймав участь у проектах як спеціаліст із визначення якості ПЗ.

Робочі місця повністю обладнані.

# ДОДАТОК Б

**ПРИКЛАД ФАЙЛУ ВХІДНИХ ДАНИХ**

Таблиця Б.1 – Кількість днів, коли температура була >0 і йшов дощ

|  |  |
| --- | --- |
| К-ть днів, коли температура була >0 і йшов дощ |  |

Таблиця Б.2 – Список дат, коли температура була >0 і йшов дощ

|  |  |
| --- | --- |
| № | Дати, коли температура була >0 і йшов дощ |

Таблиця Б.3 - Середньомісячні показники в місті [X] за місяць [Y]

|  |  |  |  |
| --- | --- | --- | --- |
| Місто | Місяць | Середньомісячні показники | |
| Температура | Тиск |
| [X] | [Y] | [T] | [P] |

# ДОДАТОК В

**ПРОГРАМНИЙ КОД**

InfoWeatherForm.xaml.cs

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

namespace cloudy

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public static Authorization authorization = new Authorization();

private List<Weather> weathers { get; set; }

private List<Weather> selectXYList { get; set; }

private List<Weather> rainDataList { get; set; }

private SelectData selectData = new SelectData();

private Weather selectedWeather { get; set; }

private DataAccess dataAccess = new DataAccess();

public static List<string> cityList = new List<string>();

public static string selectedCity = String.Empty;

public static string selectedMonth = String.Empty;

public MainWindow()

{

InitializeComponent();

}

public static void ErrorShow(string Message, string Header)

{

MessageBox.Show(Message, Header, MessageBoxButton.OK, MessageBoxImage.Warning);

}

public void ToggleDisplay()

{

EditForm.Visibility = (EditForm.Visibility == Visibility.Visible) ? Visibility.Hidden : Visibility.Visible;

UserForm.Visibility = (UserForm.Visibility == Visibility.Visible) ? Visibility.Hidden : Visibility.Visible;

}

private void ClearList()

{

weathers = new List<Weather> { };

WeatherTable.Items.Clear();

}

private void ClearEditForm()

{

city\_box.Text = String.Empty;

day\_box.Text = String.Empty;

month\_box.Text = String.Empty;

precip\_box.Text = String.Empty;

pressure\_box.Text = String.Empty;

temp\_box.Text = String.Empty;

}

private bool UserInputOk()

{

return !String.IsNullOrWhiteSpace(city\_box.Text) && !String.IsNullOrWhiteSpace(day\_box.Text) &&

!String.IsNullOrWhiteSpace(month\_box.Text) && !String.IsNullOrWhiteSpace(temp\_box.Text) &&

!String.IsNullOrWhiteSpace(precip\_box.Text) && !String.IsNullOrWhiteSpace(pressure\_box.Text) &&

Convert.ToInt32(pressure\_box.Text) > 650 && Convert.ToInt32(pressure\_box.Text) < 800 &&

Convert.ToInt32(temp\_box.Text) > -60 && Convert.ToInt32(temp\_box.Text) < 50;

}

public void MainWindow\_Loaded(object sender, RoutedEventArgs e)

{

this.Show();

LoadBase();

for (int i = 1; i <= 31; i++)

{

day\_box.Items.Add(Convert.ToString(i));

}

}

private void DisplayList(List<Weather> list)

{

WeatherTable.Items.Clear();

foreach(Weather element in list)

{

WeatherTable.Items.Add(element);

}

}

public bool LoadBase()

{

try

{

List<Weather> result = dataAccess.GetWeathers();

ClearList();

ClearEditForm();

city\_x.Items.Clear();

cityList.Clear();

if (result == null)

{

ErrorShow("База даних порожня", "Увага!");

}

else

{

foreach (Weather element in result)

{

AddToList(element);

if (!city\_x.Items.Contains(element.city))

{

city\_x.Items.Add(element.city);

cityList.Add(element.city);

}

}

}

WeatherTable.SelectedItem = null;

return true;

}

catch

{

ErrorShow("Не вдалось прочитати базу даних", "Помилка!");

return false;

}

}

private bool AddToList(Weather weather)

{

try

{

WeatherTable.Items.Add(weather);

weathers.Add(weather);

return true;

}

catch

{

ErrorShow("Не вдалось додати запис", "Помилка!");

return false;

}

}

private bool DeleteWeather(Weather weather)

{

try

{

if (dataAccess.DeleteFromBase(weather))

{

return true;

}

else

{

ErrorShow("Не вдалось видалити запис з бази даних", "Помилка!");

return false;

}

}

catch

{

ErrorShow("Не вдалось видалити запис", "Помилка!");

return false;

}

}

private void AddWeather()

{

try

{

Weather weather = new Weather(city\_box.Text, Convert.ToInt16(day\_box.Text), month\_box.Text, Convert.ToInt16(temp\_box.Text), precip\_box.Text, Convert.ToUInt32(pressure\_box.Text));

int result = dataAccess.AddToBase(weather);

if (result == 1)

{

LoadBase();

ClearEditForm();

}

else if(result == 0)

{

throw new Exception("Не вдалось додати запис");

}

else

{

throw new Exception("Записи на цю дату в даному місті вже існують");

}

}

catch(Exception ex)

{

ErrorShow(ex.Message, "Помилка");

}

}

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

{

try

{

if (!UserInputOk())

{

throw new Exception("Неправильні вхідні дані");

}

if(WeatherTable.SelectedItem == null)

{

AddWeather();

}

else

{

if(DeleteWeather(selectedWeather))

{

AddWeather();

}

else

{

throw new Exception("Не вдалось відредагувати запис");

}

}

ClearEditForm();

WeatherTable.SelectedItem = null;

}

catch (Exception ex)

{

ErrorShow(ex.Message, "Помилка");

}

}

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

{

LogInForm logInForm = new LogInForm();

logInForm.ShowDialog();

if(MainWindow.authorization.loged\_in)

{

ToggleDisplay();

}

}

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

{

try

{

selectedWeather = (Weather)WeatherTable.SelectedItem;

if (selectedWeather == null)

{

throw new Exception();

}

DeleteWeather(selectedWeather);

LoadBase();

ClearEditForm();

WeatherTable.SelectedItem = null;

}

catch

{

ErrorShow("Ви не обрали запис", "Помилка!");

}

}

public void SearchButton\_Click(object sender, RoutedEventArgs e)

{

try

{

selectedCity = city\_x.Text;

selectedMonth = month\_y.Text;

if(String.IsNullOrEmpty(selectedCity) || String.IsNullOrEmpty(selectedMonth))

{

throw new Exception("Неправильні вхідні дані");

}

selectXYList = dataAccess.SelectXY(selectedCity, selectedMonth);

if(selectXYList == null)

{

throw new Exception("Жодного запису не знайдено");

}

else

{

ClearList();

city\_x.Text = String.Empty;

month\_y.Text = String.Empty;

DisplayList(selectXYList);

}

}

catch(Exception ex)

{

ErrorShow(ex.Message, "Помилка");

}

}

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

{

try

{

rainDataList = dataAccess.GetRainData();

if (rainDataList == null)

{

throw new Exception("Жодного запису не знайдено");

}

else

{

DisplayList(rainDataList);

}

}

catch (Exception ex)

{

ErrorShow(ex.Message, "Помилка");

}

}

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

{

try

{

if (selectXYList == null)

{

throw new Exception("Здійсніть спочатку пошук записів за містом та місяцем, або оберіть потрібні місто та місяць");

}

if (rainDataList == null)

{

rainDataList = dataAccess.GetRainData();

}

selectData.WriteData(selectXYList, rainDataList);

MessageBox.Show("Інформація була успішно записана");

}

catch (Exception ex)

{

ErrorShow(ex.Message, "Помилка");

}

}

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

{

LoadBase();

}

private void WeatherTable\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

try

{

selectedWeather = (Weather)WeatherTable.SelectedItem;

if (selectedWeather == null)

{

ClearEditForm();

return;

}

city\_box.Text = selectedWeather.city.Trim();

day\_box.Text = Convert.ToString(selectedWeather.day).Trim();

month\_box.Text = selectedWeather.month;

temp\_box.Text = Convert.ToString(selectedWeather.temperature).Trim();

precip\_box.Text = selectedWeather.precipitation.Trim();

pressure\_box.Text = Convert.ToString(selectedWeather.pressure).Trim();

}

catch

{

ErrorShow("Ви не обрали запис", "Помилка!");

}

}

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

{

Chart chart = new Chart(weathers);

chart.Show();

}

}

}

LogInForm.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace cloudy

{

/// <summary>

/// Логика взаимодействия для LogInForm.xaml

/// </summary>

public partial class LogInForm : Window

{

public LogInForm()

{

InitializeComponent();

}

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

{

this.Close();

}

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

{

MainWindow.authorization.loged\_in = MainWindow.authorization.CheckLogIn(login.Text.Trim(), password.Password.Trim());

if (MainWindow.authorization.loged\_in)

{

this.Close();

}

else

{

MainWindow.ErrorShow("Неправильний логін або пароль", "Помилка!");

}

}

}

}

Chart.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using LiveCharts;

using LiveCharts.Wpf;

namespace cloudy

{

/// <summary>

/// Логика взаимодействия для Chart.xaml

/// </summary>

public partial class Chart : Window

{

public List<Weather> weathers { get; set; } = new List<Weather>();

public List<Weather> current { get; set; } = new List<Weather>();

public SeriesCollection SeriesCollection { get; set; } = new SeriesCollection() { };

public List<string> Labels { get; set; } = new List<string>();

public Func<double, string> Formatter { get; set; }

public ChartValues<int> values { get; set; } = new ChartValues<int>();

public Chart(List<Weather> weathers)

{

InitializeComponent();

this.weathers = weathers;

SeriesCollection.Add(new LineSeries() { Title = "", Values = values });

Formatter = value => value.ToString("N");

DataContext = this;

}

private void ChartForm\_Loaded(object sender, RoutedEventArgs e)

{

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

{

if (!cityList.Items.Contains(weathers[i].city))

{

cityList.Items.Add(weathers[i].city);

}

}

}

private void cityList\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

current = weathers.FindAll(x => x.city == cityList.SelectedItem.ToString());

current.Sort();

ChangeChart();

}

private void radioButton\_Checked(object sender, RoutedEventArgs e)

{

if(cityList.SelectedItem != null)

{

ChangeChart();

}

}

private void ChangeChart()

{

SeriesCollection.Clear();

Labels.Clear();

values.Clear();

foreach (Weather element in current)

{

values.Add(temperature.IsChecked == true ? element.temperature : Convert.ToInt32(element.pressure));

Labels.Add(Convert.ToString(element.day) + " " + element.month.Substring(0, 3));

}

SeriesCollection.Add(new LineSeries() { Title = current[0].city, Values = values });

Formatter = value => value.ToString("N");

DataContext = this;

}

}

}

Authorization.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MySql.Data;

using MySql.Data.MySqlClient;

namespace cloudy

{

public class Authorization

{

public bool loged\_in { get; set; }

MySqlConnection conn;

MySqlCommand command;

MySqlDataReader reader;

public Authorization()

{

string connStr = "server=localhost; user=admin; password=; database=cloudy; port=3306;";

conn = new MySqlConnection(connStr);

command = new MySqlCommand();

command.Connection = conn;

command.Connection.Open();

}

public bool CheckLogIn(string login, string password)

{

try

{

command.CommandText = $"SELECT \* FROM editors WHERE login = '{login}' AND password = '{password}';";

reader = command.ExecuteReader();

if(reader.HasRows)

{

return true;

}

else

{

return false;

}

}

catch

{

return false;

}

}

}

}

DataAccess.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MySql.Data;

using MySql.Data.MySqlClient;

using System.Windows;

namespace cloudy

{

public class DataAccess

{

MySqlConnection conn;

MySqlCommand command;

MySqlDataReader reader;

public DataAccess()

{

string connStr = "server=localhost; user=admin; password=; database=cloudy; port=3306;";

conn = new MySqlConnection(connStr);

command = new MySqlCommand();

command.Connection = conn;

command.Connection.Open();

}

private List<Weather> GetList(string commandText)

{

try

{

command.CommandText = commandText;

reader = command.ExecuteReader();

List<Weather> result = new List<Weather> { };

while (reader.Read())

{

result.Add(new Weather(reader.GetString(0), reader.GetInt16(1), reader.GetString(2), reader.GetInt32(3), reader.GetBoolean(4) == false ? "Нема" : "Є", reader.GetUInt32(5)));

}

reader.Close();

if (result.Count == 0)

{

return null;

}

else

{

return result;

}

}

catch

{

return null;

}

}

public List<Weather> GetWeathers()

{

return GetList("SELECT \* FROM cloudy;");

}

public int AddToBase(Weather weather)

{

try

{

command.CommandText = $"SELECT \* FROM cloudy WHERE city = '{weather.city}' AND date = {weather.day} " +

$"AND month = '{weather.month}';";

reader = command.ExecuteReader();

if (!reader.HasRows)

{

reader.Close();

command.CommandText = "INSERT INTO cloudy(city,date,month,temperature,precipitation,pressure) VALUES(@city, @date, @month, @temperature, @precipitation, @pressure)";

command.Parameters.Clear();

command.Parameters.AddWithValue("@city", weather.city);

command.Parameters.AddWithValue("@date", weather.day);

command.Parameters.AddWithValue("@month", weather.month);

command.Parameters.AddWithValue("@temperature", weather.temperature);

command.Parameters.AddWithValue("@precipitation", weather.precipitation == "Є" ? true : false);

command.Parameters.AddWithValue("@pressure", weather.pressure);

command.ExecuteNonQuery();

return 1;

}

else

{

reader.Close();

return -1;

}

}

catch

{

reader.Close();

return 0;

}

}

public bool DeleteFromBase(Weather weather)

{

try

{

command.CommandText = $"DELETE FROM cloudy WHERE city = '{weather.city}' AND date = {weather.day} " +

$"AND month = '{weather.month}' AND temperature = {weather.temperature} AND precipitation = {(weather.precipitation == "Є" ? true : false)} " +

$"AND pressure = {weather.pressure};";

if (command.ExecuteNonQuery() == 0)

{

throw new Exception();

}

return true;

}

catch

{

return false;

}

}

public List<Weather> SelectXY(string city, string month)

{

try

{

string commandText;

if (city != String.Empty && month != String.Empty)

{

commandText = $"SELECT \* FROM cloudy WHERE city = '{city}' AND month = '{month}';";

}

else if (city != String.Empty)

{

commandText = $"SELECT \* FROM cloudy WHERE city = '{city}';";

}

else

{

throw new Exception();

}

return GetList(commandText);

}

catch

{

return null;

}

}

public List<Weather> GetRainData()

{

return GetList($"SELECT \* FROM cloudy WHERE temperature > 0 AND precipitation = 1;");

}

}

}

Globals.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace cloudy

{

public class Weather : IComparable<Weather>

{

public string city { get; set; }

public short day { get; set; }

public string month { get; set; }

public int temperature { get; set; }

public string precipitation { get; set; }

public uint pressure { get; set; }

Dictionary<string, int> months = new Dictionary<string, int>()

{

{ "", 0 },

{ "Січень", 1 },

{"Лютий", 2 },

{ "Березень", 3 },

{ "Квітень", 4 },

{ "Травень", 5 },

{ "Червень", 6 },

{ "Липень", 7 },

{ "Серпень", 8 },

{ "Вересень", 9 },

{ "Жовтень", 10 },

{ "Листопад", 11 },

{ "Грудень", 12 }

};

public Weather(string city, short day, string month, int temp, string prec, uint pres)

{

this.city = city;

this.day = day;

this.month = month;

this.temperature = temp;

this.precipitation = prec;

this.pressure = pres;

}

public int CompareTo(Weather other)

{

months.TryGetValue(this.month, out int f);

months.TryGetValue(other.month, out int s);

return (f - s) \* 30 + (this.day - other.day);

}

}

}

SelectData.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Globalization;

namespace cloudy

{

class SelectData

{

Microsoft.Office.Interop.Word.Application wordApp;

Microsoft.Office.Interop.Word.Document wordDoc;

string filePath;

public void WriteData(List<Weather> selectXY, List<Weather> rainData)

{

try

{

filePath = Environment.CurrentDirectory.ToString();

wordApp = new Microsoft.Office.Interop.Word.Application();

wordDoc = wordApp.Documents.Add(filePath + "\\Шаблон\_Пошуку.dotx");

}

catch (Exception ex)

{

MainWindow.ErrorShow(ex.Message + char.ConvertFromUtf32(13) +

"Помістіть файл Шаблон\_Пошуку.dotx" + char.ConvertFromUtf32(13) +

"у каталог із ехе-файлом програми і повторіть збереження", "Помилка");

}

try

{

ReplaceText(MainWindow.selectedCity, "[X]");

ReplaceText(Convert.ToString(AverageTemp(selectXY)), "[T]");

ReplaceText(MainWindow.selectedMonth, "[Y]");

ReplaceText(Convert.ToString(AveragePres(selectXY)), "[P]");

ReplaceText(rainData);

wordDoc.Save();

}

catch (Exception ex)

{

MainWindow.ErrorShow(ex.Message + char.ConvertFromUtf32(13) +

"Помилка збереження відібраних даних", "Помилка");

}

}

private void ReplaceText(string textToReplace, string replacedText)

{

Object missing = Type.Missing;

Object wrap = Microsoft.Office.Interop.Word.WdFindWrap.wdFindContinue;

Object replace = Microsoft.Office.Interop.Word.WdReplace.wdReplaceAll;

Microsoft.Office.Interop.Word.Range content;

content = wordDoc.Range(wordDoc.Content.Start, wordDoc.Content.End);

Microsoft.Office.Interop.Word.Find find = wordApp.Selection.Find;

find.Text = replacedText;

find.Replacement.Text = textToReplace;

find.Execute(

FindText: Type.Missing,

MatchCase: false,

MatchWholeWord: false,

MatchWildcards: false,

MatchAllWordForms: false,

Forward: true,

Wrap: wrap,

Format: false,

ReplaceWith: missing,

Replace: replace

);

}

private void ReplaceText(List<Weather> rainData)

{

if (rainData == null)

{

wordDoc.Tables[1].Cell(1, 2).Range.Text = Convert.ToString(0);

return;

}

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

{

string month = rainData[i].month;

month = month.Replace("ень", "ня");

month = month.Replace("пад", "пада");

month = month.Replace("ий", "ого");

wordDoc.Tables[2].Rows.Add();

wordDoc.Tables[2].Cell(2 + i, 1).Range.Text = Convert.ToString(i + 1);

wordDoc.Tables[2].Cell(2 + i, 2).Range.Text = rainData[i].day + " " + month;

}

wordDoc.Tables[1].Cell(1, 2).Range.Text = Convert.ToString(rainData.Count);

}

private double AverageTemp(List<Weather> weathers)

{

if(weathers == null)

{

return 0;

}

double result = 0;

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

{

result += weathers[i].temperature;

}

return result / weathers.Count;

}

private double AveragePres(List<Weather> weathers)

{

if (weathers == null)

{

return 0;

}

double result = 0;

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

{

result += weathers[i].pressure;

}

return result / weathers.Count;

}

}

}

# ДОДАТОК Г

**ІНСТРУКЦІЯ З ІНСТАЛЯЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Встановлення програмного забезпечення на пристрій користувача відбувається за допомогою стандартного інсталятору, робота з яким починається з вікна привітання (рис. Г.1). Для продовження встановлення потрібно натиснути кнопку «Далее»., для виходу з інсталятора – «Отмена».

Изображение выглядит как текст

Автоматически созданное описание

Рисунок Г.1 – Початкове вікно інсталятора

Наступним кроком користувачу необхідно обрати папку, до якої буде завантажено та встановлено програмний застосунок, та натиснути кнопку «Далее» (рис. Г.2).

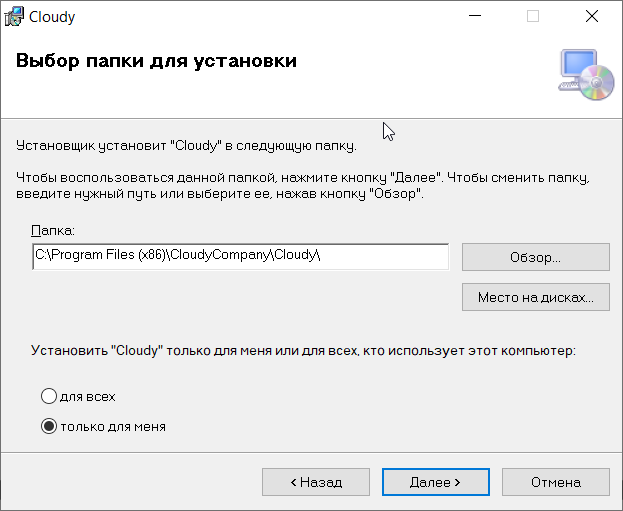


Рисунок Г.2 – Вибір папки

Для переходу до безпосереднього встановлення програмного застосунку після вибору необхідного кореню для інсталяції програми необхідно підтвердити встановлення, натиснувши кнопку «Далее» (рис. Г.3).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок Г.3 – Підтвердження встановлення

Чекаємо завершення інсталяції (рис. Г.4).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок Г.4 – Процес інсталяції

Після пророблених дій програма буде успішно встановлена на пристрій користувача (рис. Г.5).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Завершення інсталяції

# ДОДАТОК Д

**ІНСТРУКЦІЯ КОРИСТУВАЧА**

Після запуску програми перед користувачем відкривається головна форма програми, в які вже завантажені всі записи з бази даних (рис. Д.1).



Рисунок Д.1 – Початок роботи з програмою

Для пошуку інформації за відповідним містом та місяцем користувачу необхідно обрати з відповідних випадаючих списків необхідні вхідні дані та натиснути кнопку «Пошук» (рис. Д.2).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок Д.2 – Виконання пошуку за містом та місяцем

Для повернення до початкового відображення форми та скинення всіх фільтрів пошуку необхідно натиснути кнопку «Скинути» (рис. Д.3)



Рисунок Д.3 – Виконання скидання фільтрів

У разі необхідності виконання пошуку днів, коли температура повітря була більша за 0 та йшов дощ, необхідно натиснути відповідну кнопку «Пошук днів з дощем» (рис. Д.4).



Рисунок Д.4 – Пошук дощових днів

Наступним функціоналом, який доступний користувачу, є створення звіту з отриманої під час пошуку інформації. Обов’язковим є виконання даного пошуку, інакше користувач буде повідомлений про відсутність необхідних даних. У разі наявності інформації відкриється форма збереження файлу (рис. Д.5).

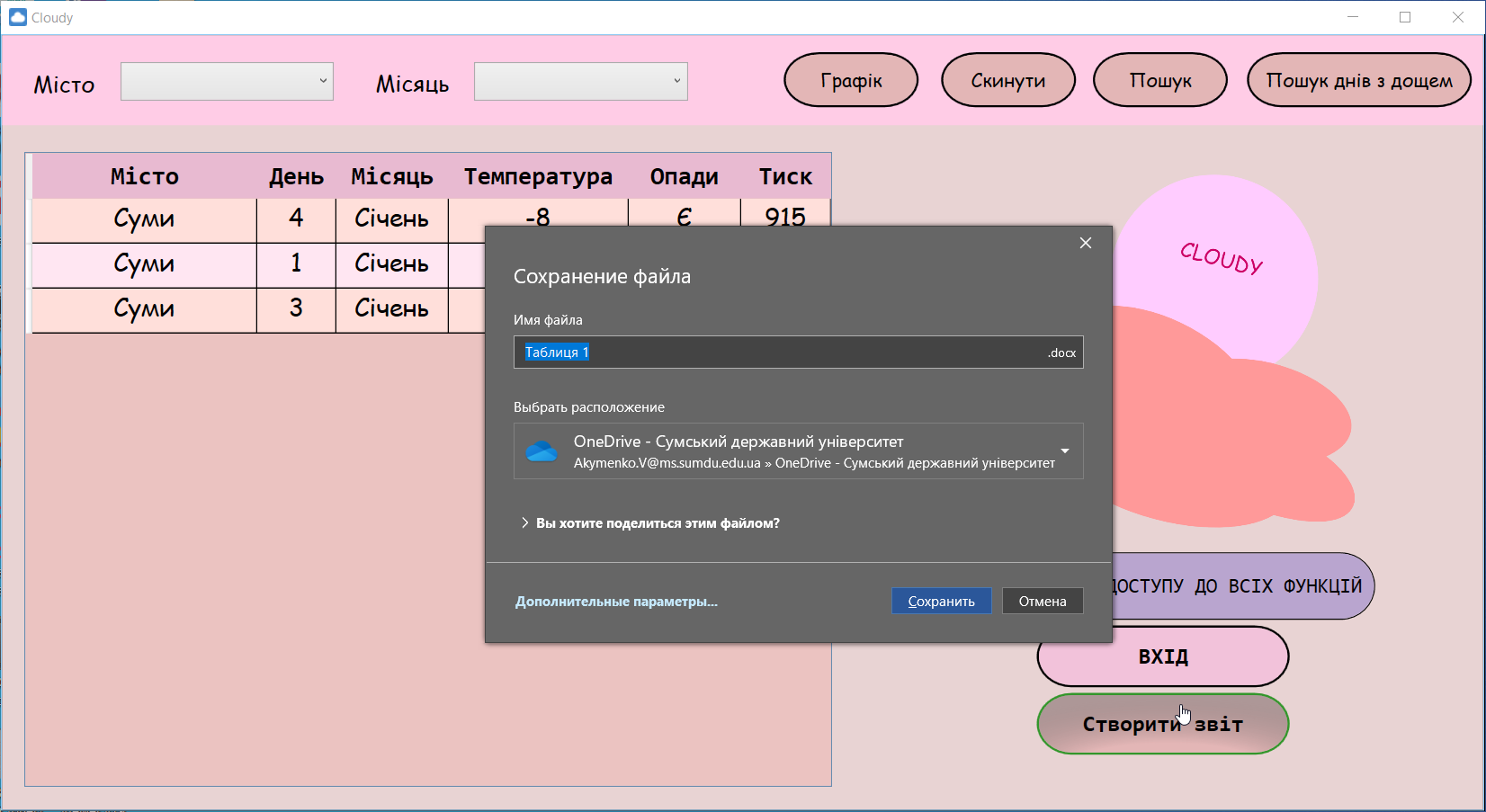


Рисунок Д.5 – Збереження звіту з пошуку

У обраній директорії буде створено файл за шаблоном MS Word, загальна структура якого зображена на рисунку Д.6.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок Д.6 – Структура звіту

Для переходу до функцій співробітника та отримання можливості редагувати базу даних необхідно натиснути кнопку «Вхід» на головній формі програми, після чого ввести дані авторизації у відповідні поля та запустити перевірку за допомогою натискання кнопки «Вхід» або натиснувши клавішу Enter на клавіатурі (рис. Д.7).

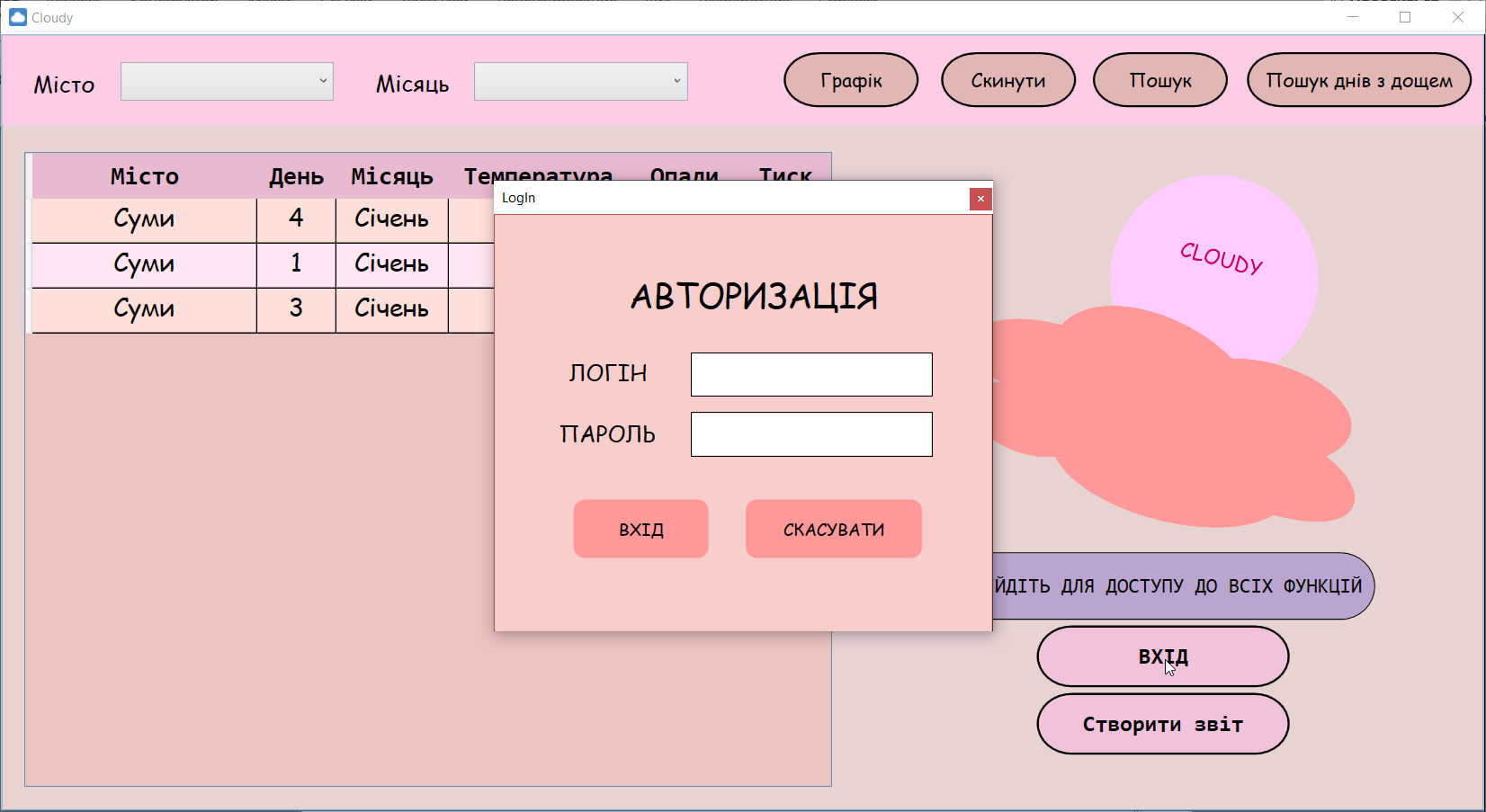


Рисунок Д.7 – Авторизація співробітника компаніїї

У разі коректності введених даних головна форма програми змінить свій вид на інтерфейс співробітника (рис. Д.8).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок Д.8 – Форма співробітника

Для доступу до редагування окремого запису необхідно обрати даний запис із загального списку записів (рис. Д.9).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок Д.9 – Доступ до редагування

. У разі видалення необхідно натиснути кнопку «Видалити» , після чого запис буде видалений із бази даних (рис. Д.10).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок Д.10 – Успішне видалення запису

Для редагування даних необхідно змінити відповідну інформацію, враховуючи область допустимих значень та натиснути кнопку «Зберегти» (рис. Д.11).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок Д.11 – Відредаговані дані в формі редагування

У разі коректності введених даних запис буде змінено (рис. Д.12).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок Д.12 – Відредагований запис успішно збережений

Також користувач має можливість створення графіків по даним, що зберігаються в базі даних. Для доступу до даного функціоналу необхідно натиснути кнопку «Графік», після чого відкриється відповідна форма. Обравши місто із випадаючого списку, користувач помітить зміни в графіку (рис. Д.13).

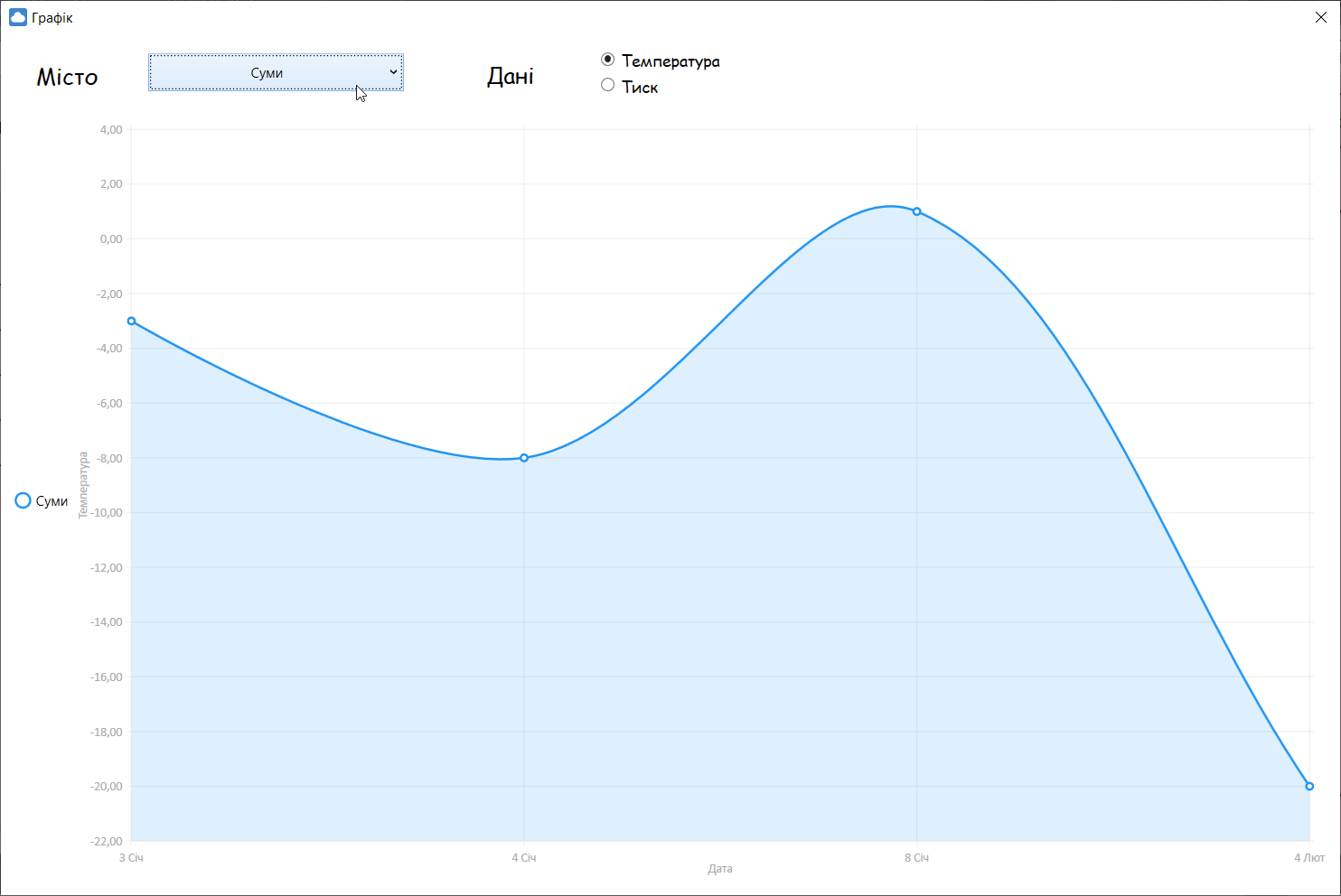


Рисунок Д.13 – Побудова графіку за вхідними даними