

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ

ЗАВАНТАЖЕННЯ МЕТЕОРОЛОГІЧНИХ ДАНИХ І ПРОГНОЗІВ

НА МОБІЛЬНИЙ ПРИСТРІЙ КОРИСТУВАЧА

SOFTWARE FOR DOWNLOADING

METEOROLOGICAL DATA AND FORECASTS

TO THE USER'S MOBILE DEVICE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-132

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ВОРОНА М.О.

(ПРІЗВИЩЕ та ініціали)

Керівник КОЛПАКОВА Т.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ВОРОНИ Михайла Олександровича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для завантаження метеорологічних даних і прогнозів на мобільний пристрій користувача. Software for Downloading Meteorological Data and Forecasts to the User's Mobile Device

керівник проєкту (роботи) к.т.н., доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “07” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЛПАКОВА Т.О., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 14 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Михайло ВОРОНА
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна КОЛПАКОВА
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
82 с., 3 табл., 23 рис., 3 дод., 17 джерел.

ANDROID STUDIO, JAVA, МЕТЕОРОЛОГІЧНІ ДАНІ, МОВА ПРОГРАМУВАННЯ, ПРОГНОЗ ПОГОДИ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПРОЄКТ.

Об'єкт дослідження – процес розробки програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Предмет дослідження – програмні засоби для завантаження метеорологічних даних і прогнозів.

Мета роботи – розробка програмного забезпечення для завантаження метеорологічних даних і прогнозів для підвищення рівнів зручності та оперативності прийняття рішень користувачами.

Матеріали, методи та технічні засоби: мова програмування Java, середовище розробки Android Studio.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів. Створено програмне забезпечення для завантаження метеорологічних даних і прогнозів. Проведено тестування розробленої програмної системи для завантаження метеорологічних даних і прогнозів.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для завантаження метеорологічних даних і прогнозів за допомогою мови програмування Java та середовища розробки Android Studio.

Галузь використання – прогнозування погоди.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 82 pages, 3 tables, 23 figures, 3 appendixes, 17 sources.

ANDROID STUDIO, JAVA, METEOROLOGICAL DATA, PROGRAMMING LANGUAGE, WEATHER FORECAST, SOFTWARE, PROJECT.

The object of research is the process of developing software for downloading meteorological data and forecasts.

The subject of the research is software for supporting the downloading meteorological data and forecasts.

The purpose of this work is to develop the software to support the downloading meteorological data and forecasts process.

Materials, methods and technical tools: Java programming language, Android Studio development environment.

Results. Project solutions for the creation of software for downloading meteorological data and forecasts has been designed. The software to support the process of downloading meteorological data and forecasts has been developed. The developed software system for downloading meteorological data and forecasts has been tested.

Conclusions. The goal of the work has been achieved. The software tools of for downloading meteorological data and forecasts using the Java programming language and the Android Studio development environment have been developed.

Scope of use – weather forecasting.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Програмне забезпечення для завантаження метеорологічних даних і прогнозів.....	11
1.2 Аналіз програмного забезпечення для завантаження метеорологічних даних і прогнозів	21
1.3 Висновки за розділом 1	29
2 Матеріали і методи	31
2.1 Вибір мови програмування	31
2.2 Вибір середовища розробки для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів.....	33
2.3 Висновки за розділом 2	36
3 Опис програми	37
3.1 Структура програмного забезпечення для завантаження метеорологічних даних і прогнозів	37
3.2 Функціонування програмного забезпечення для завантаження метеорологічних даних і прогнозів	39
3.3 Розробка бази даних програмного забезпечення для завантаження метеорологічних даних і прогнозів	40
3.4 Проєктування інтерфейсу програмного забезпечення для завантаження метеорологічних даних і прогнозів	42
3.5 Висновки за розділом 3	44
4 Експлуатація, тестування та експериментальне дослідження програми.....	46
4.1 Призначення й умови застосування програми	46
4.2 Характеристики програми для завантаження метеорологічних даних і прогнозів.....	47

4.3 Інструкція по експлуатації програми.....	48
4.3.1 Звернення до програми.....	48
4.3.2 Вхідні й вихідні дані.....	48
4.3.3 Повідомлення	49
4.4 Виконання програмного забезпечення для завантаження метеорологічних даних і прогнозів	49
4.5 Тестування програмного забезпечення для завантаження метеорологічних даних і прогнозів	52
4.6 Висновки за розділом 4	53
Висновки	54
Перелік джерел посилання	57
Додаток А Технічне завдання	59
Додаток Б Фрагмент тексту програми	64
Додаток В Слайди презентації	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

БД – база даних;

МД – метеорологічні дані;

ПЗ – програмне забезпечення;

ПП – прогнозування погоди.

ВСТУП

Висока динаміка змін погоди та значущість метеорологічної інформації для різних сфер діяльності зумовлюють потребу у спеціалізованих програмних інструментах, здатних автоматично отримувати, обробляти та оновлювати дані про стан атмосфери. Використання таких програмних рішень забезпечує безперервний доступ до актуальних прогнозів, що є критичним для управління транспортом, аграрними процесами, енергетичними ресурсами, туристичною діяльністю, системами безпеки та інфраструктурним плануванням [1]-[2].

Постійне зростання обсягів кліматичної інформації та розширення джерел її отримання роблять автоматизоване програмне завантаження метеоданих необхідним засобом для підвищення точності прогнозування, оперативного реагування на зміну погодних умов і побудови моделей впливу клімату на виробничі та соціальні процеси. Розроблення та впровадження таких програмних засобів сприяє ефективному використанню ресурсів, покращенню якості рішень та мінімізації ризиків, пов'язаних із природними явищами [3]-[4].

Проте деякі програми для завантаження метеорологічних даних і прогнозів можуть працювати з певними обмеженнями, що можуть впливати на точність та ефективність їх застосування. Отримання прогнозів залежить від надійності зовнішніх джерел і стабільності Інтернет з'єднання, тому у випадку технічних збоїв або затримок доступ до інформації може погіршуватися. Похибки у первинних вимірюваннях та різний рівень деталізації даних, які надаються різними метеосервісами, створюють ризик отримання неточних прогнозів, що ускладнює прийняття рішень у відповідальних сферах. Чимало програмних систем потребують значних обчислювальних ресурсів для оброблення даних великого обсягу, що підвищує вимоги до обладнання та енергоспоживання. Використання програм може обмежуватися платними ліцензіями, закритими протоколами або обмеженим доступом до повного набору метеоданих. Крім того, необхідність регулярного оновлення

програмного забезпечення та адаптації до нових форматів даних спричинює додаткові витрати часу та ресурсів, що знижує зручність експлуатації [1]-[6].

Тому актуальною є розробка програмного забезпечення для завантаження метеорологічних даних і прогнозів. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для завантаження метеорологічних даних і прогнозів;
- здійснити проектування програмного забезпечення для завантаження метеорологічних даних і прогнозів;
- створити програмне забезпечення для завантаження метеорологічних даних і прогнозів;
- виконати тестування розробленого програмного забезпечення для завантаження метеорологічних даних і прогнозів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для завантаження метеорологічних даних і прогнозів

Програмне забезпечення для завантаження метеорологічних даних і прогнозів являє собою як спеціалізований інструмент, що забезпечує подання інформації про погодні умови та їх очікувану динаміку. Його призначення полягає у створенні каналу між джерелами метеорологічних вимірювань, серверами прогнозних моделей та кінцевими програмними системами, які мають потребу в регулярному отриманні й оновленні таких відомостей. Функціонування подібних програмних рішень ґрунтується на автоматизованому зборі та подальшій трансформації даних у зручні для аналізу або відображення форми. Використання такого програмного забезпечення дозволяє забезпечити стабільний обмін погодною інформацією, що є важливим для подальших розрахунків, планування або попередження ризиків [1].

Особливістю таких програмних систем є здатність обробляти великі обсяги метеорологічної інформації, що надходить з різномірних джерел. Таке програмне забезпечення передбачає роботу з даними з наземних станцій, супутникових платформ, океанських буїв, радіолокаційних установок, а також із серверів прогнозних моделей, які формують результати на основі чисельних методів. Структура подібних систем зазвичай формується так, щоб забезпечити стійке підключення до зовнішніх інтерфейсів прикладного програмування (application programming interface, API), інтерпретацію отриманих масивів інформації та подальшу передачу їх у внутрішні модулі для збереження або подальшої візуалізації. Велика увага приділяється синхронізації та коректній обробці часових рядів, оскільки інформація про погоду змінюється безперервно, а відхилення у даних можуть призвести до помилкових висновків у залежних підсистемах [2].

Використання подібного програмного забезпечення дає можливість створити надійне середовище, здатне відслідковувати зміни у погодних процесах у майже реальному часі. Завдяки цьому забезпечується змога реагувати на зміни атмосферних умов у ті сфери, де точність і своєчасність метеорологічних даних має ключове значення. У результаті стає доступною підвищена інформативність, що дозволяє точніше оцінювати ризики, прогнозувати потреби в ресурсах чи організовувати технологічні процеси відповідно до очікуваних змін довкілля. Такий підхід також сприяє зменшенню невизначеності, оскільки зібрана та синтезована інформація надходить у більш упорядкованому вигляді, ніж у разі ручного відстеження джерел [3].

Певною перевагою таких програмних рішень вважається можливість безперервної автоматизації. Завдяки цьому знімається потреба в постійному людському контролі, оскільки завантаження, обробка та оновлення інформації здійснюються у фоновому режимі. Додатковою перевагою є здатність інтегруватися з іншими системами, що дає змогу включати метеорологічні дані до процесів планування, аналітики або візуалізації у більш складних технологічних комплексах. Гнучкість подібного програмного забезпечення дає можливість адаптувати його до різноманітних галузей, починаючи від сільського господарства, транспорту, енергетики і закінчуючи науковими дослідженнями або мобільними застосунками для широкого кола користувачів [4].

Одночасно подібні програмні системи мають і свої слабкі сторони. Більша частина з них пов'язана з залежністю від зовнішніх джерел інформації, які можуть бути недоступними або працювати з обмеженнями. У разі виникнення затримок або переривань у каналах передавання відбувається порушення циклу оновлення, що може вплинути на коректність прогнозів або точність оперативних даних. Крім того, складні метеорологічні моделі використовують значні обчислювальні ресурси, що може уповільнити роботу системи, якщо вона виконує ще інші функції. Суттєвою проблемою таких

програмних систем стає і неоднорідність форматів погодних даних, що змушує такі системи включати модулі нормалізації, перетворення та узгодження різних структур даних [4]-[5].

Ще однією особливістю вважається чутливість до якості вимірювань. Навіть найкраще програмне забезпечення не може гарантувати точність, якщо одержувані значення мають похибки, спричинені технічними обмеженнями сенсорів або збоями в мережах. Унаслідок цього іноді виникає потреба в додатковій фільтрації, усередненні або компенсації неприйнятних показників. У складних прогнозних сценаріях часто виникає необхідність у математичному згладжуванні, що змушує систему використовувати додаткові алгоритми для стабілізації результатів [6].

Загалом подібне програмне забезпечення є важливим елементом сучасної цифрової інфраструктури, що забезпечує постійний доступ до погодної інформації у зручному й структурованому вигляді. Його роль полягає у створенні надійного інструменту для опрацювання атмосферних даних та прогнозів, здатного інтегруватися в різні сфери діяльності та підтримувати прийняття рішень, заснованих на актуальних показниках навколишнього середовища. На практиці така технологія сприяє підвищенню ефективності в багатьох галузях, забезпечуючи стабільне підґрунтя для планування, аналізу і моніторингу процесів, що залежать від погоди [6].

Програмні рішення, призначені для відтворення погодних процесів, формують основу сучасного прогнозування атмосферних явищ і забезпечують доступ до результатів моделювання, без яких неможливо досягти високої точності у визначенні подальшої поведінки атмосфери. Таке програмне забезпечення використовується як у професійних метеорологічних системах, так і в тих сферах, де необхідно передбачати зміни погоди заздалегідь. Подібні інструменти підтримують можливість підготовки до будь-яких нестабільних умов та знижують ризик раптових технологічних або організаційних збоїв. Застосування методів моделювання впливає на функціонування багатьох галузей, оскільки дозволяє своєчасно прогнозувати критичні зміни

середовища. Такий підхід забезпечує можливість підготувати сільськогосподарські процеси відповідно до очікуваних коливань вологості й температури, організовувати логістичні операції з урахуванням потенційних небезпек та уникати порушень у транспортних мережах. Підрозділи, що відповідають за реагування на надзвичайні ситуації, завдяки моделюванню отримують основу для завчасної координації дій, що дозволяє зменшити негативний вплив небезпечних явищ на населення. Передбачення атмосферних процесів підвищує рівень індивідуальної та колективної безпеки й формує підґрунтя для прийняття виважених рішень [2]-[3].

Під час вибору програмних систем, що здійснюють моделювання погоди, важливими вважаються ті властивості, які забезпечують точність і зручність роботи. Швидке оновлення даних дозволяє без затримок отримувати поточну картину атмосферних процесів, а графічні засоби подання інформації допомагають швидко інтерпретувати складні структури погодних полів. Можливість змінювати параметри середовища моделювання сприяє адаптації результатів під конкретні завдання. Інтуїтивно зрозумілий інтерфейс полегшує роботу користувача, а підтримка мобільних платформ забезпечує доступ до даних незалежно від місця перебування [2]-[3].

Серед програмних рішень для моделювання погоди існують різні підходи, кожен з яких виконує свою роль у створенні прогнозів. Одним із напрямів є застосування чисельних методів, через які відтворюються складні атмосферні процеси на основі законів фізики. Такі моделі здатні описувати розвиток температурних полів, розподіл тиску, зміни у хмарності та інші параметри, використовуючи великі масиви даних зі сенсорних систем, супутників та радарів. Інший підхід полягає в аналізі історичних значень погодних показників, де за допомогою статистичних залежностей встановлюються ймовірні варіанти зміни умов у майбутньому. Подібні алгоритми особливо корисні у разі необхідності виявлення довготривалих трендів або визначення закономірностей у стабільних регіонах [2]-[3].

Серед програмних рішень у галузі моделювання використовуються системи, які дозволяють створювати високо деталізовані симуляції атмосфери та надають інструменти для дослідження різноманітних погодних сценаріїв. Подібні платформи застосовуються як у науковій діяльності, так і в оперативній роботі метеорологів. Інші моделі забезпечують глобальне охоплення та дозволяють отримувати прогнози на триваліший період, надаючи інформацію, що використовується у багатьох секторах для забезпечення стійкої роботи систем, які залежать від кліматичних факторів. Окреме місце займають рішення, орієнтовані на міжнародну співпрацю, де обробка даних здійснюється за допомогою складних алгоритмів, а результати характеризуються високим ступенем надійності [3]-[4].

Використання подібних програмних інструментів дає змогу підвищити ефективність прогнозування, оскільки поєднання реальних даних з алгоритмами моделювання забезпечує глибше розуміння структури атмосферних явищ. Отримані результати формують основу для точного передбачення подій, які можуть вплинути на економічні процеси, безпеку населення або функціонування критичних систем. Таким чином створюється більш передбачуване середовище, де рішення приймаються на основі достовірних обчислень і актуальних відомостей про атмосферу [3]-[4].

Точність прогнозування істотно підвищується тоді, коли атмосферні процеси відтворюються за допомогою спеціалізованих цифрових систем. Подібні інструменти працюють із величезними масивами метеорологічної інформації та застосовують математичні схеми, здатні відтворювати перебіг фізичних процесів у повітряних масах. Завдяки такому підходу досягається більш передбачувана поведінка атмосфери, а отримані результати вирізняються стабільністю та високою деталізацією. Це створює умови для раціонального планування в аграрному секторі, де можливість завчасно зрозуміти майбутні зміни стає основою оптимального циклу вирощування культур. У містобудівному середовищі надійні метеорологічні дані допомагають коригувати проєктні рішення так, щоб інфраструктура

витримувала несприятливі погодні впливи. Професійні прогнозисти, отримуючи точнішу інформацію, мають змогу оперативніше інформувати населення та підтримувати безпеку громад [3]-[4].

Разом із перевагами виникають і певні труднощі, оскільки будь-яке моделювання залежить від якості вихідних даних. Неповнота або неточність метеорологічної інформації ускладнює отримання реалістичного результату й може породжувати похибки, що негативно відображаються на подальших рішеннях. Рівень точності прогнозу значною мірою визначається тим, наскільки повно та узгоджено представлено дані, що надходять зі спостережних систем. Якщо інформація з віддалених або недостатньо обладнаних територій надходить із затримками чи має фрагментарний характер, моделі втрачають свою ефективність. Особливо складною стає ситуація під час екстремальних явищ, коли історичні дані не в змозі забезпечити достатній орієнтир для майбутнього. Відсутність доступу до даних високої просторової роздільності створює додаткові перепони, оскільки детальність таких даних є ключовою для точного аналізу регіональних особливостей. В умовах різноманіття кліматичних зон отримання персоналізованих наборів значень стає непростим завданням, що знижує можливість відтворити реальну картину атмосферних процесів [3]-[4].

Для роботи з моделями прогнозування погоди високої точності потрібні значні обчислювальні ресурси, і ця вимога зростає разом із ускладненням алгоритмів. Подібні системи потребують потужної інфраструктури, здатної обробляти великі обсяги даних у стислі терміни. Не всі організації можуть забезпечити таку технічну базу, тому виконання моделювання інколи обмежується менш деталізованими сценаріями. Зі збільшенням потреби у швидших обчисленнях постає питання масштабування, адже недостатня продуктивність комп'ютерних систем уповільнює отримання прогнозів та ускладнює своєчасне оновлення результатів. Необхідність поєднання високої точності з доступною обчислювальною вартістю залишається постійною проблемою для багатьох установ [3]-[4].

Попри обмеження, вплив сучасних програмних систем моделювання на якість прогнозування та його роль у керуванні ризиками є вкрай значущим. Завдяки удосконаленню алгоритмів і використанню оперативних даних стає можливим більш відповідальне планування та завчасна підготовка до складних атмосферних умов. Поступовий розвиток таких технологій відкриває нові можливості для різних галузей, сприяючи формуванню стійкіших підходів до взаємодії з природними процесами. У перспективі зростає ймовірність того, що поєднання потужніших моделей і якісніших даних дозволить ще точніше відтворювати погодні сценарії, забезпечуючи вищу захищеність суспільства перед природною мінливістю [3]-[4].

Моделі прогнозування погоди розглядаються як цифрові реконструкції атмосферних процесів, створені для наукового аналізу та передбачення майбутніх змін у стані повітряних мас. Уся процедура передбачення атмосферної еволюції спирається на опрацювання великого обсягу даних, що відображають теперішні та попередні погодні умови, аби сформувати уявлення про подальший перебіг атмосферних явищ. Для цього застосовуються відомості про температуру, тиск, вологість, типи та інтенсивність опадів, швидкість руху повітряних потоків та інші параметри, отримані з різних джерел спостереження, включаючи наземні станції, супутникові платформи та зонди. Постійна мінливість атмосфери призводить до накопичення величезного масиву інформації, яку необхідно впорядкувати та математично інтерпретувати [3]-[4].

Перетворення зібраних значень на прогноз погоди передбачає моделювання поведінки велетенської кількості змінних, що безперервно змінюються. Це вимагає розв'язання складних фізико-математичних рівнянь, відомих у науці як рівняння гідродинамічного типу. Їхня складність настільки висока, що для оброблення таких обчислень застосовуються спеціалізовані обчислювальні системи з надзвичайною продуктивністю. Метод передбачення, побудований на цих рівняннях, отримав назву числового

прогнозування, а програми, які здійснюють відповідні обчислення, стали відомі як атмосферні моделі [3]-[4].

Такі моделі фактично відтворюють поведінку атмосфери у вигляді цифрового простору. Повітряний шар, що оточує Землю, являє собою складну систему, де газові маси реагують на вплив різних фізичних законів – від хімічних процесів до термодинамічних ефектів. Якщо надати моделі достатню кількість початкових даних, потужність для оброблення та математичний апарат, здатний адекватно описати взаємодію цих явищ, стає можливим обчислення подальшого руху повітряних потоків [3]-[4].

Будь-яка модель прогнозування базується на трьох ключових складових: актуальних даних, потужній обчислювальній інфраструктурі та системі рівнянь, яка описує зміну атмосферних параметрів. Щоб ініціювати роботу моделі, у неї повинні бути подані початкові характеристики погоди для певної території. Подальший вибір залежить від того, чи спрямована модель на невелику ділянку, чи вона охоплює всю планету. Незалежно від типу, обидва підходи використовують аналогічні методи збирання інформації, але різняться масштабом [3]-[4].

Після отримання вихідних значень дані переносяться в спеціально створену сіткову структуру, що складається з великої кількості просторово впорядкованих точок. Вони утворюють тривимірний каркас, який охоплює територію моделювання і простягається вгору крізь шари атмосфери. Реальні вимірювання не прив'язані до конкретних точок сітки, оскільки вона створюється виключно для числових розрахунків. Кожна точка стає місцем, для якого алгоритм поетапно обчислює зміну характеристик повітря у часі. Поступове просування в часі дає змогу моделі відтворити можливі траєкторії руху атмосферних мас і прогнозованих погодних явищ [3]-[4].

Збільшення кількості точок підвищує деталізацію результатів, але потребує колосальних обчислювальних ресурсів. Найдетальніші системи світу виконують неймовірну кількість операцій за одиницю часу, тож для їхньої роботи задіяні високопродуктивні обчислювальні комплекси. Простіші моделі,

що мають менш розгалужену сітку, здатні функціонувати на помірно потужних пристроях, але їхня деталізація є нижчою. Таким чином, точність прогнозу залежить від якості початкових даних, складності математичного апарату та доступної обчислювальної сили, яка визначає здатність моделі опрацьовувати атмосферні зміни з потрібним рівнем деталізації [3]-[4].

Існування великої кількості моделей прогнозування пояснюється тим, що кожна з них створюється під конкретні завдання та типи атмосферних процесів, які необхідно відтворити з максимальною точністю. Різні масштабні рівні, різні набори вхідних даних і різні методи математичного опису призводять до того, що локальні моделі забезпечують одні типи результатів, а глобальні – зовсім інші. У процесі побудови моделі завжди здійснюється вибір: які параметри включати, яким рівнянням довірити опис фізичних процесів та які аспекти прогнозу вважати пріоритетними. Кожне таке рішення впливає на підсумковий результат і визначає, які види явищ відтворюватимуться точніше, а які менш детально [3]-[4].

Повністю універсальна модель, здатна з високою точністю охопити всі типи погодних ситуацій, не існує. Тому прийнято орієнтуватися на конкретну задачу і налаштовувати алгоритм саме під неї. Точність у певному прогнозному інтервалі завжди досягається ціною зниження точності в іншому. Моделі, орієнтовані на невелику часову перспективу, потребують особливо детальних даних про процеси у нижніх шарах атмосфери, тоді як у випадку довших прогнозів важливішими стають інші типи умов та інший спосіб математичного узагальнення. Для короткострокових оцінок часто застосовуються системи, що використовують дані з різних висот, оскільки така інформація дозволяє точніше відобразити швидкоплинні процеси. Для триваліших передбачень перевага надається моделям, які не акцентують увагу на дрібномасштабних явищах, а зосереджуються на загальних атмосферних тенденціях [3]-[4].

Постійне вдосконалення методів прогнозування є невід'ємною частиною роботи метеорологів. Алгоритми тестуються, порівнюються між

собою, коригуються й розширюються, щоб визначити, які математичні конструкції краще відтворюють реальну поведінку атмосфери. Частина таких розробок є відкритою для широкого використання, інші залишаються закритими й застосовуються лише в межах окремих наукових або державних установ [3]-[4].

У регіональному вимірі вагоме місце посідають моделі, орієнтовані на окремі частини світу. Крім цих систем, існують інші моделі, розроблені національними метеорологічними службами різних країн. Кожна з них створюється з урахуванням власних завдань, особливостей території та доступних даних. Їхні підходи різняться, як і їхня точність у певних ситуаціях, однак усі вони спрямовані на відтворення максимально достовірної картини майбутнього стану атмосфери там, де це є необхідним [5]-[6].

Програмне забезпечення для роботи з метеорологічними даними і прогнозами орієнтовано на виконання щоденних робочих потреб. Такими програмами активно користуються державні служби спостережень, приватні компанії, установи у сфері авіації та оборони, оскільки вони дають змогу об'єднувати різні джерела інформації, здійснювати багаторівневе відображення даних, формувати графічні та текстові продукти й підтримувати широкий спектр операційних процесів. Такі програмні системи сприяють ефективній організації роботи, формуванню попереджень, контролю над інформаційними потоками [5]-[6].

Визначено, що програмне забезпечення для завантаження метеорологічних даних і прогнозів являє собою як спеціалізований інструмент, що забезпечує подання інформації про погодні умови та їх очікувану динаміку. Його призначення полягає у створенні каналу між джерелами метеорологічних вимірювань, серверами прогнозних моделей та кінцевими програмними системами, які мають потребу в регулярному отриманні й оновленні таких відомостей. Функціонування подібних програмних рішень ґрунтується на автоматизованому зборі та подальшій трансформації даних у зручні для аналізу або відображення форми.

1.2 Аналіз програмного забезпечення для завантаження метеорологічних даних і прогнозів

Проаналізуємо програмне забезпечення для завантаження метеорологічних даних і прогнозів [7]-[13].

Програмне забезпечення Tomorrow.io (рис. 1.1) позиціонується як комплексне рішення, здатне трансформувати метеорологічні дані у корисні для бізнесу дії. Його концепція ґрунтується на тому, що погодні умови мають значний вплив на різні сфери діяльності, а тому потрібен інструмент, який не просто подає інформацію про атмосферні явища, а й допомагає зрозуміти їхній вплив на робочі процеси. Платформа побудована таким чином, щоб організації могли отримувати детальну картину змін у навколишньому середовищі та на її основі коригувати свою щоденну діяльність, підвищуючи злагодженість внутрішніх процесів. Замість подачі статичних параметрів вона структурує погодні показники у вигляді аналітичних висновків, що полегшує прийняття рішень та підсилює ефективність роботи [7], [9], [11].

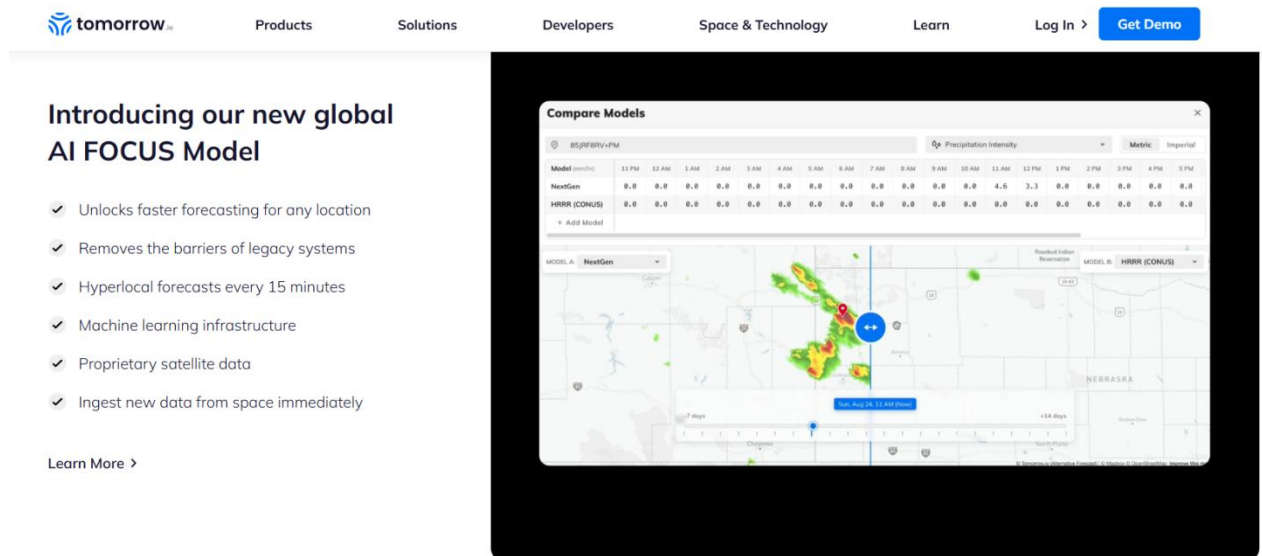


Рисунок 1.1 – Програмне забезпечення Tomorrow.io [11]

Програмне забезпечення Tomorrow.io дозволяє об'єднувати інформацію про ключові зони інтересу, важливі погодні межі, встановлені

правила реагування та внутрішні процедури у єдиному середовищі, забезпечуючи зрозумілий та узгоджений спосіб управління ризиками. У платформі поєднано дані про минулі явища, актуальний стан атмосфери та прогнозовані зміни, що створює широкі можливості для моделювання ситуацій та оптимізації ресурсів. Додатковою особливістю є інтерактивна карта, яка відображає численні погодні та екологічні показники, дозволяючи користувачам спостерігати за тенденціями у режимі реального часу за допомогою різнокольорових шарів та різних режимів прогнозування. Такий підхід допомагає компаніям адаптуватися до природної мінливості, зменшувати наслідки несприятливих умов і забезпечувати більшу стійкість у своїх операціях [7], [9], [11].

Основними характеристиками програмного забезпечення Tomorrow.io є такі [7], [9], [11]:

- всеосяжна аналітика погоди: програмне забезпечення Tomorrow.io перетворює метеорологічні показники на практичні поради для прийняття рішень, що дає змогу організаціям оцінювати ризики та адаптувати операції відповідно до атмосферних змін [7], [9], [11];

- інтеграція історичних та поточних даних: програмне забезпечення Tomorrow.io поєднує інформацію про минулі погодні явища, актуальний стан атмосфери та прогнозні моделі, формуючи повну картину впливу природних умов на діяльність [7], [9], [11];

- інтуїтивний інтерфейс: структура програмного забезпечення Tomorrow.io дозволяє легко керувати ключовими параметрами, зонами інтересу та пороговими значеннями погоди, забезпечуючи швидкий доступ до важливих налаштувань та сценаріїв реагування [7], [9], [11];

- інтерактивна карта: графічне середовище відображає погодні та екологічні індикатори у вигляді кольорових шарів, що спрощує спостереження за тенденціями та прогнозами в реальному часі [7], [9], [11];

- підтримка галузевих сценаріїв: програмне забезпечення Tomorrow.io враховує специфіку різних сфер діяльності та пропонує адаптовані

рекомендації, які підвищують операційну стійкість і мінімізують вплив несприятливих умов [7], [9], [11];

- можливість централізації даних: програмне забезпечення Tomorrow.io дозволяє зводити в одному місці протоколи, стратегії реагування та ключові об'єкти інфраструктури, забезпечуючи узгодженість роботи команд під час планування та реагування [7], [9], [11];

- адаптивність рішень: програмне забезпечення Tomorrow.io здатне підлаштовуватися під зміну погодних умов та оперативні потреби організації, що сприяє більш гнучкому управлінню процесами та підвищенню ефективності [7], [9], [11].

Серед недоліків програмного забезпечення Tomorrow.io можна відзначити обмеження повноцінного доступу до всіх функцій у безкоштовних версіях, що ускладнює використання сервісу малими організаціями або окремими користувачами; залежність точності прогнозів від зовнішніх джерел даних, через що в окремих регіонах можливі похибки; складність інтеграції з деякими нестандартними корпоративними системами, що потребує додаткових налаштувань; а також значну вартість розширених пакетів, яка може бути неприйнятною для невеликих бізнесів [7], [9], [11].

Програмне забезпечення Weatherstack (рис. 1.2) подає погодні відомості через спеціальний інтерфейс, що дає змогу отримувати актуальні та попередні параметри стану атмосфери для широкого кола географічних точок. Розв'язання орієнтоване на використання у програмних рішеннях, онлайн ресурсах та аналітичних системах, де є потреба в оперативних і стабільних даних про погодні умови. Запит до цього сервісу повертає структуровану відповідь, що містить відомості про теперішній стан погоди, прогнозні показники та накопичену інформацію про вже зафіксовані атмосферні явища. Формат подання спрощує включення таких даних у зовнішні проєкти, забезпечуючи швидке та безпроблемне підключення [7], [8], [12].

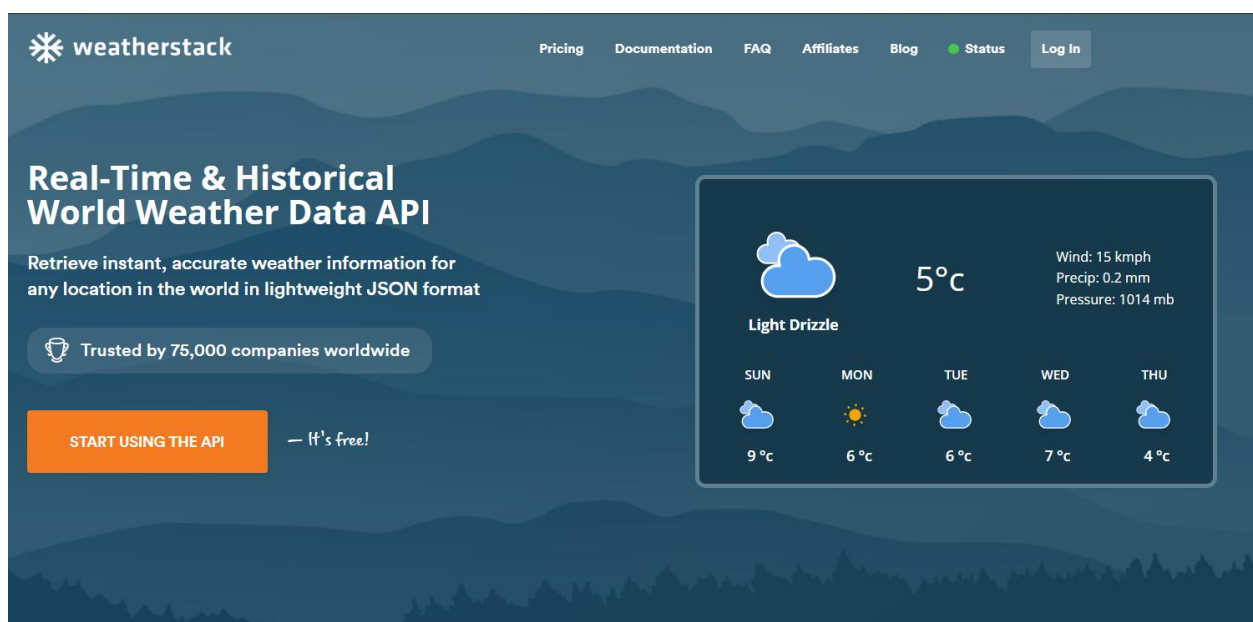


Рисунок 1.2 – Програмне забезпечення Weatherstack [12]

Завдяки надійності роботи платформи можна розраховувати на своєчасні оновлення, придатні для різних задач, пов'язаних із моніторингом і аналізом погодних змін [7], [8], [12].

Програмне забезпечення Weatherstack має такі особливості [7], [8], [12]:

- актуальні погодні дані: програмне забезпечення Weatherstack забезпечує доступ до поточної інформації про стан атмосфери для будь-якої зазначеної локації [7], [8], [12];
- історичні відомості: програмне забезпечення Weatherstack надає доступ до раніше зафіксованих показників, що дає змогу аналізувати попередні погодні умови [7], [8], [12];
- прогнозні параметри: програмне забезпечення Weatherstack дозволяє отримувати очікувані зміни погоди на майбутній період для різних сценаріїв використання [7], [8], [12];
- зручний формат відповіді: дані передаються у структурованому вигляді, що спрощує інтеграцію в програмні продукти та аналітичні платформи [7], [8], [12];
- орієнтованість на розробників: програмне забезпечення Weatherstack створено для використання в застосунках, вебсистемах та автоматизованих

програмних рішеннях, де важлива швидкість і надійність отримання погодних відомостей [7], [8], [12].

Серед недоліків програмного забезпечення Weatherstack можна відзначити обмежені можливості безкоштовного доступу, що ускладнює повноцінне використання сервісу без переходу на розширені плани. Також спостерігається залежність від стабільності зовнішніх джерел даних, через що точність окремих показників може варіюватися залежно від регіону. У певних випадках відзначається недостатня деталізація прогнозів для віддалених або малонаселених територій, що знижує придатність інструмента для специфічних потреб. Крім того, інтеграція у складні аналітичні системи інколи потребує додаткового налаштування, оскільки не всі функції доступні у базовій конфігурації [7], [8], [12].

Програмне забезпечення Flash Weather AI (рис. 1.3) подається як аналітичне рішення, створене для точного відстеження атмосферних процесів і оперативного реагування на зміни погодних умов [8], [13].

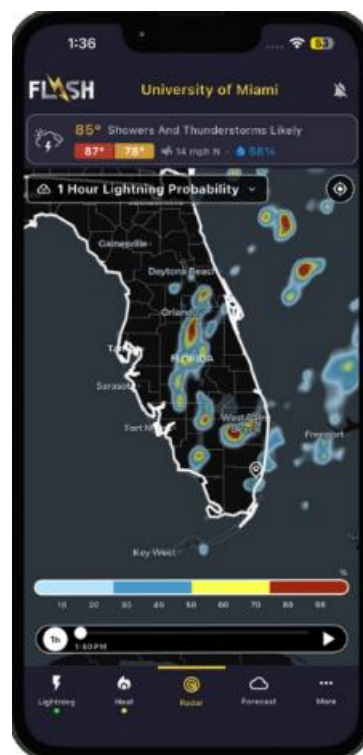


Рисунок 1.3 – Програмне забезпечення Flash Weather AI [13]

Робота програмного забезпечення Flash Weather AI ґрунтується на використанні власних алгоритмів штучного інтелекту, що забезпечують максимально локалізовані описи ситуації та здатність миттєво фіксувати появу небезпечних явищ. Програмне забезпечення Flash Weather AI орієнтована на оперативне інформування користувачів про різкі зміни погоди, зокрема про виникнення активності блискавок та інші критичні події, а також на підтримку тих сфер діяльності, де погодні ризики можуть порушити робочі процеси. Програмне забезпечення Flash Weather AI дозволяє одночасно контролювати кілька територій, налаштовувати індивідуальні сповіщення та отримувати текстові повідомлення, що відповідають конкретним потребам. Завдяки такому підходу забезпечується можливість швидкої реакції на потенційні загрози та створюються умови для більш впевненого планування в середовищах, чутливих до змін атмосферної ситуації. [8], [13].

Програмне забезпечення Flash Weather AI має такі особливості [8], [13]:

- гіперлокальна аналітика: програмне забезпечення Flash Weather AI формує детальні описи погодних умов для точкових територій, забезпечуючи високу точність під час оцінювання ризиків [8], [13];

- миттєві попередження: алгоритми забезпечують оперативну генерацію сповіщень про небезпечні явища, що дозволяє заздалегідь реагувати на зміни атмосферної ситуації [8], [13];

- прогнозування блискавок: технологія штучного інтелекту використовується для визначення появи блискавкової активності в реальному часі, що підвищує ефективність заходів безпеки [8], [13];

- гнучкі текстові повідомлення: програмне забезпечення Flash Weather AI дає можливість налаштовувати індивідуальні формати інформування відповідно до конкретних завдань та потреб користувача [8], [13];

- одночасний моніторинг територій: програмне забезпечення Flash Weather AI забезпечує контроль декількох об'єктів, що важливо для галузей із розгалуженою інфраструктурою [8], [13];

– підтримка чутливих галузей: рішення орієнтоване на сфери діяльності, для яких погодні коливання мають визначальний вплив, що робить його корисним інструментом для управління ризиками [8], [13].

Серед недоліків програмного забезпечення Flash Weather AI можна відзначити можливу залежність точності розрахунків від якості вхідних даних, що інколи призводить до появи обмежень у формуванні локальних прогнозів. Також помітною є потреба у стабільному підключенні до мережі для коректного отримання оперативної інформації, що може ускладнювати використання платформи в умовах обмеженого доступу до Інтернету. Окремі функції можуть вимагати додаткових витрат, що підвищує загальну вартість впровадження для невеликих організацій. Певні труднощі можуть виникати і через складність початкового налаштування, оскільки система орієнтується на специфічні потреби користувачів та потребує ретельної конфігурації для повноцінного використання всіх можливостей [8], [13].

Порівняльну характеристику програмного забезпечення для завантаження метеорологічних даних і прогнозів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для завантаження метеорологічних даних і прогнозів

Критерій порівняння	Tomorrow.io [11]	Weatherstack [12]	Flash Weather AI [13]
Наявність інтерактивних карт	+	—	—
Зручність інтеграції через API	+	+	+—
Гіперлокальні прогнози	+	—	+
Невисокі вимоги до ресурсів	+—	+	+—
Гнучкість налаштування під різноманітні завдання бізнесу	+	+—	+
Низька вартість використання	+—	+	+—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для завантаження метеорологічних даних і прогнозів. Проте деякі програми для завантаження метеорологічних даних і прогнозів можуть працювати з певними обмеженнями, що можуть впливати на точність та ефективність їх застосування. Отримання прогнозів залежить від надійності зовнішніх джерел і стабільності Інтернет з'єднання, тому у випадку технічних збоїв або затримок доступ до інформації може погіршуватися. Похибки у первинних вимірюваннях та різний рівень деталізації даних, які надаються різними метеосервісами, створюють ризик отримання неточних прогнозів, що ускладнює прийняття рішень у відповідальних сферах. Чимало програмних систем потребують значних обчислювальних ресурсів для оброблення даних великого обсягу, що підвищує вимоги до обладнання та енергоспоживання. Використання програм може обмежуватися платними ліцензіями, закритими протоколами або обмеженим доступом до повного набору метеоданих. Крім того, необхідність регулярного оновлення програмного забезпечення та адаптації до нових форматів даних спричинює додаткові витрати часу та ресурсів, що знижує зручність експлуатації. Тому актуальною є розробка програмного забезпечення для завантаження метеорологічних даних і прогнозів.

При розробці програмного забезпечення для завантаження метеорологічних даних і прогнозів необхідно забезпечити такі функціональні вимоги:

- підтримка можливості подання метеорологічних даних і прогнозів;
- підтримка можливості обрання міста для завантаження інформації щодо прогнозу погоди;
- можливість виведення прогнозів погоди на сьогодні та найближчі два дні;
- можливість подання інформації стосовно температури, вологості, швидкості вітру та хмарності;

– підтримка можливості виведення інформації про помилки у процесі роботи з програмою.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для завантаження метеорологічних даних і прогнозів являє собою як спеціалізований інструмент, що забезпечує подання інформації про погодні умови та їх очікувану динаміку. Його призначення полягає у створенні каналу між джерелами метеорологічних вимірювань, серверами прогнозних моделей та кінцевими програмними системами, які мають потребу в регулярному отриманні й оновленні таких відомостей. Функціонування подібних програмних рішень ґрунтується на автоматизованому зборі та подальшій трансформації даних у зручні для аналізу або відображення форми.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для завантаження метеорологічних даних і прогнозів. Проте деякі програми для завантаження метеорологічних даних і прогнозів можуть працювати з певними обмеженнями, що можуть впливати на точність та ефективність їх застосування. Отримання прогнозів залежить від надійності зовнішніх джерел і стабільності Інтернет з'єднання, тому у випадку технічних збоїв або затримок доступ до інформації може погіршуватися. Похибки у первинних вимірюваннях та різний рівень деталізації даних, які надаються різними метеосервісами, створюють ризик отримання неточних прогнозів, що ускладнює прийняття рішень у відповідальних сферах. Чимало програмних систем потребують значних обчислювальних ресурсів для оброблення даних великого обсягу, що підвищує вимоги до обладнання та енергоспоживання. Використання програм може обмежуватися платними ліцензіями, закритими протоколами або обмеженим доступом до повного набору метеоданих. Крім того, необхідність регулярного оновлення програмного забезпечення та адаптації до нових

форматів даних спричинює додаткові витрати часу та ресурсів, що знижує зручність експлуатації. Тому актуальною є розробка програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Сформульовано функціональні вимоги до програмного забезпечення для завантаження метеорологічних даних і прогнозів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для завантаження метеорологічних даних і прогнозів було використано мову програмування Java [14]-[15].

Java поєднує універсальність з можливістю подальшого розширення функціоналу. Ця мова сформувала потужну професійну спільноту, яка активно підтримує розвиток інструментів із відкритим кодом, що значно полегшує процес створення нових програмних рішень. Розроблення на Java дозволяє спиратися на величезний досвід попередніх проєктів, оскільки мова вже багато років використовується у сферах, де критично важливими є точність, надійність і безпека обчислень [14]-[15].

Важливою перевагою мови програмування Java є її здатність інтегруватися з великою кількістю зовнішніх сервісів і технологій, включно з корпоративними системами, мобільними платформами, аналітичними інструментами та серверними рішеннями. Завдяки таким особливостям мови програмування Java можливість розширення функціональності метеорологічного програмного забезпечення не обмежується лише отриманням даних, а може охоплювати прогнозування на основі моделей машинного навчання, побудову інтерактивних візуалізацій і створення систем оповіщення про небезпечні погодні явища [14]-[15].

Не менш суттєвим аспектом мови програмування Java є стабільність Java-платформи, що забезпечує тривалу підтримку програмних продуктів без необхідності повного переписування коду. Також наявність віртуальної машини Java дає змогу уникнути залежності від конкретного обладнання, а система автоматичного керування пам'яттю спрощує реалізацію складних алгоритмів, звільняючи розробника від рутинних низькорівневих операцій. Сукупність цих властивостей робить мову програмування Java сприятливим

середовищем для створення гнучких, довговічних і продуктивних метеорологічних інформаційних систем [14]-[15].

Використання мови програмування Java під час створення програмного забезпечення для завантаження метеорологічних даних є доцільним завдяки поєднанню стабільності, продуктивності та широких комунікаційних можливостей цієї мови. Підтримка великої кількості мережевих протоколів дає змогу безперешкодно взаємодіяти з вебсервісами, інтерфейсами прикладного програмування та хмарними платформами, що є ключовим для отримання метеорологічних даних у режимі реального часу. Наявність багатопотокових механізмів мови програмування Java робить можливим паралельне завантаження, оновлення та оброблення великих обсягів інформації, не знижуючи швидкодії системи [14]-[15].

Кросплатформність мови програмування Java забезпечує роботу програм на різних операційних системах без потреби адаптації коду, що полегшує впровадження рішень у державних установах, приватних компаніях і дослідницьких організаціях із різноманітною технічною інфраструктурою. Розвинена екосистема бібліотек і фреймворків мови програмування Java прискорює розробку інструментів для оброблення даних, візуалізації прогнозів, інтеграції з базами даних та аналітичними системами. Високий рівень безпеки, властивий Java, сприяє захисту даних від несанкціонованого доступу та втручань [14]-[15].

Завдяки таким властивостям Java здатна забезпечити надійну, масштабовану й продуктивну основу для побудови програмних засобів, що працюють з метеорологічними даними, забезпечуючи точність, доступність та безперервність їх використання [14]-[15].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для завантаження метеорологічних даних і прогнозів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для завантаження метеорологічних даних і прогнозів

Критерій порівняння мов програмування	Мова програмування		
	Java	JavaScript	Python
Кросплатформність	+	+—	+
Висока стабільність та надійність	+	—	+—
Багатопотоковість та ефективна паралельність обчислень	+	+—	+—
Безпека та захист даних	+	—	+—
Оптимальний баланс швидкодії та зручності розроблення	+	+—	+—

Отже, для реалізації програмного забезпечення для завантаження метеорологічних даних і прогнозів обрано мову програмування Java завдяки поєднанню стабільності, продуктивності та широких комунікаційних можливостей цієї мови. Крім того, підтримка великої кількості мережевих протоколів дає змогу безперешкодно взаємодіяти з вебсервісами, API та хмарними платформами, що є ключовим для отримання метеорологічних даних у режимі реального часу.

2.2 Вибір середовища розробки для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів

В якості середовища розробки для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів було обрано середовище Android Studio [16]-[17].

Середовище розробки Android Studio є засобом, здатним забезпечувати тривалу підтримку та масштабування мобільних застосунків, що особливо

важливо в умовах постійного оновлення джерел метеоданих і розвитку сервісів прогнозування. Середовище розробки Android Studio регулярно оновлюється самим Google, що гарантує сумісність застосунків із новими версіями операційної системи Android та актуальними апаратними можливостями мобільних пристроїв [16]-[17].

Особливу цінність має інтелектуальний редактор коду, який пропонує підказки, автодоповнення, вбудовані інструменти перевірки помилок і оптимізації продуктивності. Завдяки цьому розроблення навіть складних мережових рішень стає менш трудомістким і менш залежним від людських факторів. Інтеграція з системами контролю версій, зокрема Git, спрощує колективну роботу над проєктами, що може бути актуальним для наукових або корпоративних команд, які створюють та вдосконалюють програмне забезпечення для роботи з даними погоди [16]-[17].

Android Studio також пропонує можливість гнучкого керування ресурсами застосунку, включаючи адаптацію інтерфейсу до різних типів екранів, автоматичне перемикання тем, оптимізацію анімацій і роботу з графічними елементами. Це дозволяє створювати професійні інтерфейси, здатні відображати великі масиви погодної інформації в зручній для користувача формі. Усе це робить Android Studio не лише інструментом для програмування, а й середовищем комплексної побудови надійних, масштабованих і зручних мобільних рішень, здатних працювати із сучасними метеорологічними сервісами [16]-[17].

Використання Android Studio для створення програмного забезпечення, орієнтованого на завантаження метеорологічних даних і прогнозів, є доцільним, оскільки це середовище забезпечує повний цикл розроблення мобільних рішень, які можуть оперативно отримувати актуальну інформацію незалежно від місця перебування користувача. Можливість роботи безпосередньо з Java та Kotlin спрощує реалізацію мережових запитів, багатопотокових обчислень і взаємодії з API метеосервісів. Android Studio підтримує інструменти для доступу до хмарних сервісів, локального

кешування даних, офлайн режиму та інтеграції з картографічними платформами, що є важливим для візуалізації погодних даних [16]-[17].

Таким чином, Android Studio забезпечує оптимальне поєднання інструментів для створення мобільних систем, здатних у реальному часі отримувати, аналізувати та відображати метеорологічну інформацію, що робить її ефективним середовищем для розроблення таких програмних продуктів [16]-[17].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів

Критерій порівняння середовищ розробки	Середовища розробки		
	Android Studio	IntelliJ IDEA	NetBeans
Оптимізація для мобільної розробки	+	+—	—
Інструменти для роботи з API метеосервісів та хмарними платформами	+	+	+—
Вбудований емулятор мобільних пристроїв	+	—	—
Засоби візуального проєктування інтерфейсу для мобільних застосунків	+	+—	+—
Інтеграція з сервісами Google	+	+	—

Середовище розробки Android Studio має розвинені засоби тестування та емуляції, що дає змогу перевіряти роботу застосунку на різних пристроях і за різних умов отримання Інтернет сигналу. Наявність офіційних бібліотек Google дає можливість використовувати надійні механізми оброблення геолокації, оповіщення про небезпечні метеоявища та планування автоматичних оновлень прогнозів. Гнучкі інструменти інтерфейсної розробки забезпечують зручну подачу складних метеорологічних даних у вигляді графіків, діаграм, інтерактивних карт та сповіщень [16]-[17].

Таким чином, для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів обрано середовище розробки Android Studio, оскільки воно забезпечує спеціалізовану підтримку мобільних технологій, прямий доступ до інструментів Google, потужні засоби тестування та повну інтеграцію платформ, необхідних для роботи з погодними сервісами.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для завантаження метеорологічних даних і прогнозів обрано мову програмування Java завдяки поєднанню стабільності, продуктивності та широких комунікаційних можливостей цієї мови. Крім того, підтримка великої кількості мережевих протоколів дає змогу безперешкодно взаємодіяти з вебсервісами, API та хмарними платформами, що є ключовим для отримання метеорологічних даних у режимі реального часу.

Для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів обрано середовище розробки Android Studio, оскільки воно забезпечує спеціалізовану підтримку мобільних технологій, прямий доступ до інструментів Google, потужні засоби тестування та повну інтеграцію платформ, необхідних для роботи з погодними сервісами.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для завантаження метеорологічних даних і прогнозів

Структуру програмного забезпечення для завантаження метеорологічних даних і прогнозів наведено на рис. 3.1.

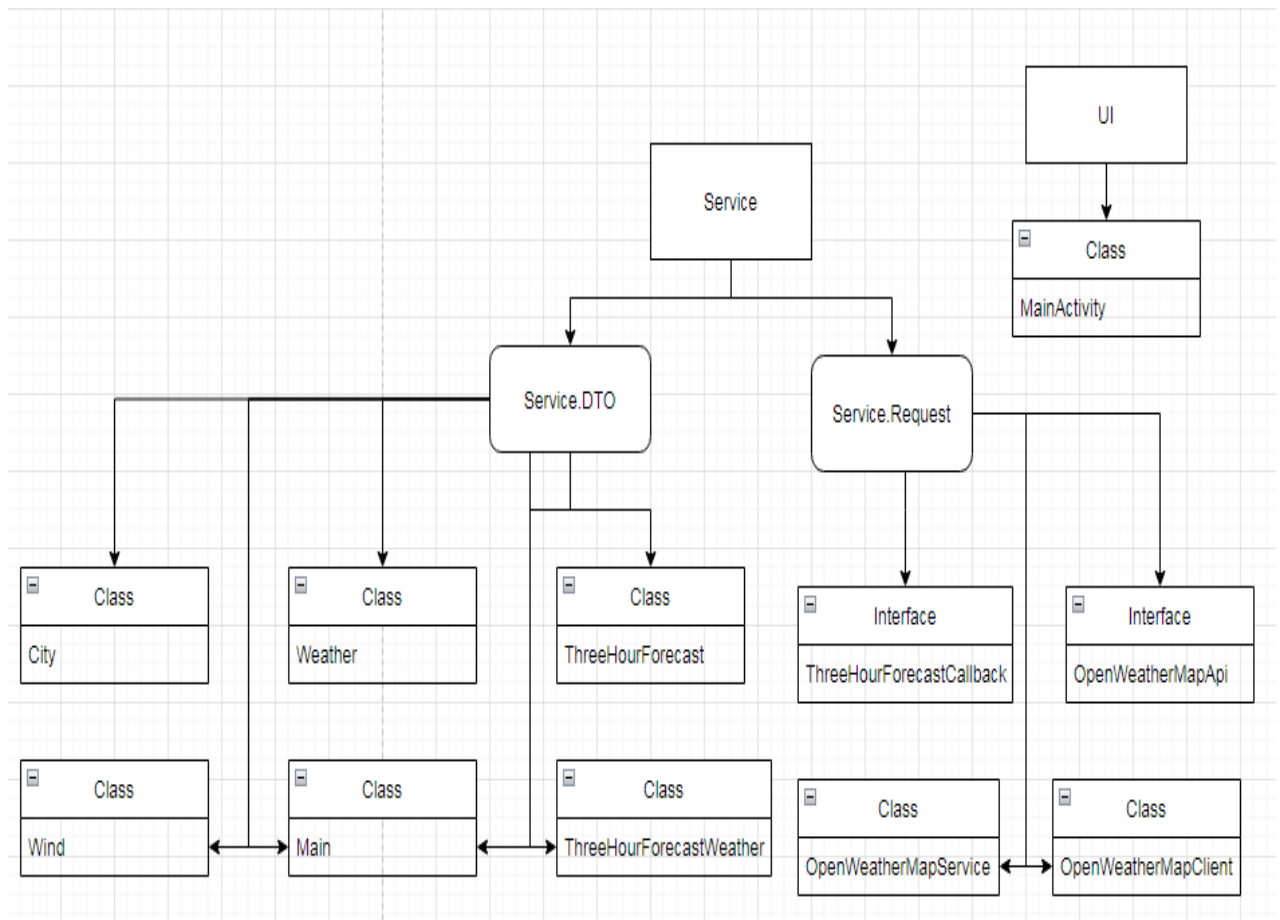


Рисунок 3.1 – Структура програмного забезпечення для завантаження метеорологічних даних і прогнозів

Архітектура програмного забезпечення, орієнтованого на отримання та опрацювання метеорологічних даних, вибудовується навколо взаємодії між кількома класами, що формують цілісну модель системи та відображають логіку використання зовнішнього сервісу прогнозування погоди. Діаграма класів демонструє, як окремі об'єкти відповідають за комунікацію з API,

передачу параметрів запиту, інтерпретацію відповідей та формування підсумкового прогнозу для користувача.

У центрі механізму доступу до зовнішнього сервісу розташований клас `OpenWeatherMapClient`, у якому ініціалізується інструментарій `Retrofit`, що слугує основою для виконання HTTP-запитів. Уся логіка формування клієнта зосереджена в `OpenWeatherMapService`, який підтримує принцип єдиного екземпляра та керує створенням REST-інтерфейсу для роботи з JSON-даними. Для того щоб `Retrofit` міг інтерпретувати структуру викликів, використовується інтерфейс `OpenWeatherMapApi`, що визначає сигнатуру запиту на отримання метеорологічних прогнозів у форматі GET та слугує своєрідним контрактом між клієнтом та зовнішнім сервісом.

Після виконання запиту система звертається до механізму оброблення результатів, реалізованого в `ThreeHourForecastCallback`. Цей клас реагує на два можливих сценарії: обробляє ситуації, пов'язані з помилками передачі даних, та керує отриманням валідної інформації у випадку успішного звернення до API. Структура предметної області, представлена в діаграмі класів, формує окремі моделі, що відповідають за відображення різних аспектів прогнозу. Об'єкт `City` забезпечує доступ до характеристик населеного пункту, для якого здійснюється запит. Клас `Wind` описує параметри повітряних потоків, тоді як `Weather` узагальнює інформацію про основні атмосферні умови. Для деталізації значень, пов'язаних із температурою, тиском та іншими ключовими характеристиками, використовується клас `Main`.

Отримані дані структуруються у вигляді комплексних об'єктів: `ThreeFourCast` акумулює кінцевий прогноз, що формується на основі результатів запиту, а `ThreeFourCastWeather` поєднує відомості про місто, погодні характеристики та параметри вітру, забезпечуючи доступ до узгодженого набору даних. У такій конфігурації кожен клас виконує специфічну роль, завдяки чому діаграма відображає добре впорядковану систему, де взаємодія між компонентами забезпечує узгоджене отримання та подання результатів метеорологічного прогнозування.

Структура програмного забезпечення для завантаження метеорологічних даних і прогнозів побудована таким чином, щоб забезпечити чітку взаємодію між компонентами, відповідальними за виконання запиту до зовнішнього сервісу, оброблення отриманої відповіді та формування структурованого прогнозу. У системі поєднано клієнтську частину для роботи з API, механізм оброблення результатів та набір класів-моделей, що представляють місто, погодні показники, параметри вітру й загальні атмосферні умови. Підсумкові об'єкти агрегують усі ці дані, створюючи узгоджену модель прогнозу, яка може бути безпосередньо використана в інтерфейсі застосунку або в подальших обчисленнях.

3.2 Функціонування програмного забезпечення для завантаження метеорологічних даних і прогнозів

Функціонування програмного забезпечення для завантаження метеорологічних даних і прогнозів подамо за допомогою схеми, зображеної на рис. 3.2.

Функціонування програмного забезпечення для отримання метеорологічних даних ґрунтується на послідовності дій, яка починається з моменту запуску застосунку та триває до завершення роботи користувача. Після старту система переходить до виконання основного алгоритму, у межах якого перевіряється доступність Інтернет з'єднання. Якщо пристрій не має доступу до мережі, програма одразу повідомляє про неможливість отримати прогноз і пропонує користувачеві повторити спробу пізніше. У разі наявного підключення застосунок переходить до оброблення запиту, пов'язаного з переглядом прогнозу погоди. Користувач зазначає назву населеного пункту, за потреби може скористатися функцією підстановки, після чого система надсилає запит до зовнішнього сервісу та відображає отримані дані про стан атмосфери на найближчий період.

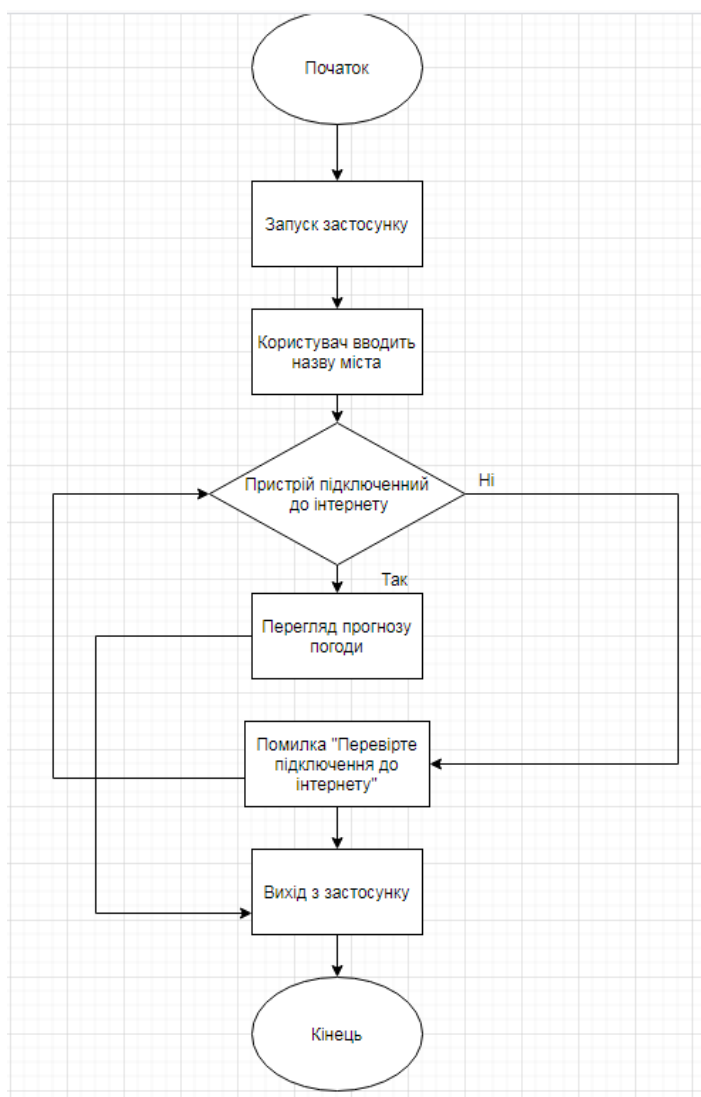


Рисунок 3.2 – Схема функціонування програмного забезпечення для завантаження метеорологічних даних і прогнозів

Після ознайомлення з результатами користувач завершує роботу, і застосунок повертається у вихідний стан. Уся логіка побудована так, щоб забезпечити передбачувану взаємодію та мінімальну кількість дій для отримання необхідної інформації.

3.3 Розробка бази даних програмного забезпечення для завантаження метеорологічних даних і прогнозів

У програмному забезпеченні, призначеному для завантаження метеорологічних даних і прогнозів, база даних виконує роль локального

сховища, у якому зберігаються відомості про населені пункти, отримані погодні показники та результати запитів до зовнішнього сервісу. Логічна структура такої бази даних формується таким чином, щоб забезпечити впорядковане зберігання даних, зручний доступ до них та можливість швидкого оновлення інформації при повторних запитах.

Схему бази даних програмного забезпечення для завантаження метеорологічних даних і прогнозів наведено на рис. 3.3.

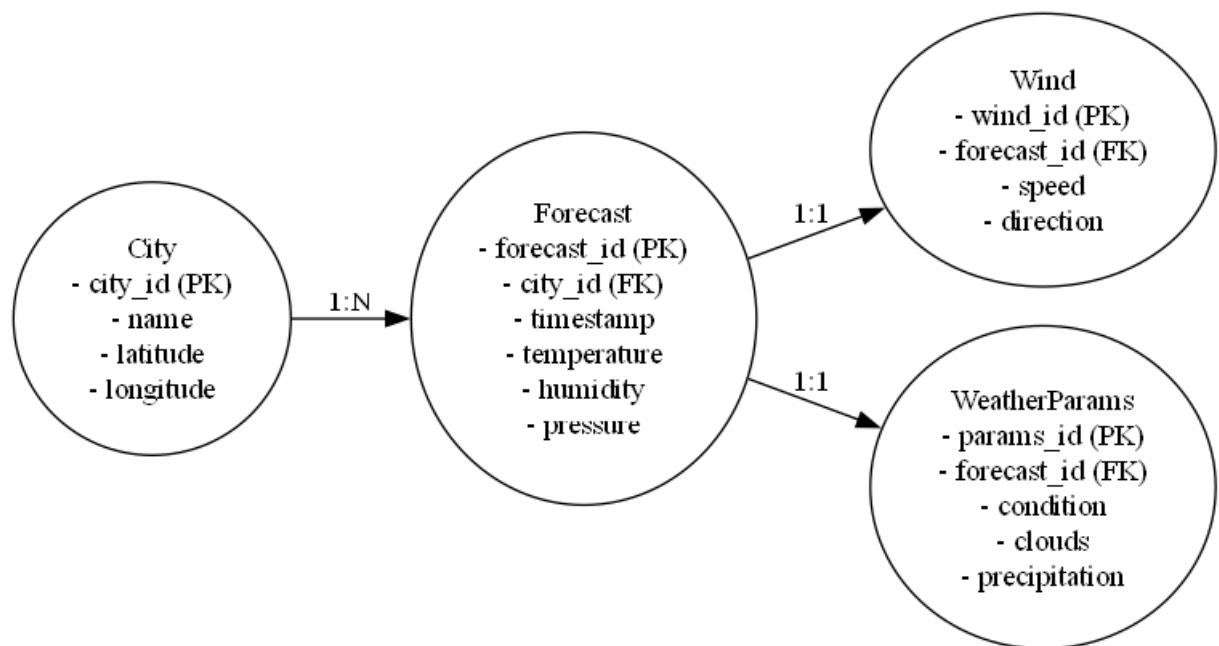


Рисунок 3.3 – Схема бази даних програмного забезпечення для завантаження метеорологічних даних і прогнозів

У центрі моделі знаходиться таблиця, що описує міста. У ній містяться унікальний ідентифікатор, назва населеного пункту та координати, які потрібні для формування запиту до API. Ця таблиця пов'язана з таблицею прогнозів, у якій кожен запис характеризує окремий прогноз для відповідного міста на певний момент часу. Запис прогнозу доповнюється даними про температуру, вологість, атмосферний тиск та іншими показниками, що надходять від зовнішнього сервісу. Інформація про параметри вітру виділяється в окрему таблицю, що містить швидкість, напрямок та інші характеристики руху повітряних мас. Вона пов'язується з таблицею прогнозів

через зовнішній ключ, що дає змогу виводити погодну картину як сукупність усіх показників.

Для опису загальних погодних умов використовується таблиця з параметрами погоди, у якій фіксуються текстові та числові значення, що відповідають стану атмосфери, наприклад, хмарність або тип опадів. Вона також пов'язана з таблицею прогнозів, тому кожен прогноз включає деталізовані характеристики за кількома напрямками. Усі таблиці утворюють розгалужену логічну структуру, у якій місто є відправною точкою, прогноз виступає центральним об'єктом, а погодні параметри, включно з вітром і загальними умовами, створюють повноцінний опис стану атмосфери. Така організація дає змогу легко оновлювати окремі фрагменти даних, зберігати історію прогнозів та забезпечувати швидкий доступ до необхідної інформації під час роботи застосунку.

3.4 Проєктування інтерфейсу програмного забезпечення для завантаження метеорологічних даних і прогнозів

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для завантаження метеорологічних даних і прогнозів виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для завантаження метеорологічних даних і прогнозів враховані функціональні вимоги до програми:

- підтримка можливості подання метеорологічних даних і прогнозів;
- підтримка можливості обрання міста для завантаження інформації щодо прогнозу погоди;
- можливість виведення прогнозів погоди на сьогодні та найближчі два дні;
- можливість подання інформації стосовно температури, вологості, швидкості вітру та хмарності;

– підтримка можливості виведення інформації про помилки у процесі роботи з програмою.

Інтерфейс головного екрану програмного забезпечення для завантаження метеорологічних даних і прогнозів наведено на рис. 3.4.

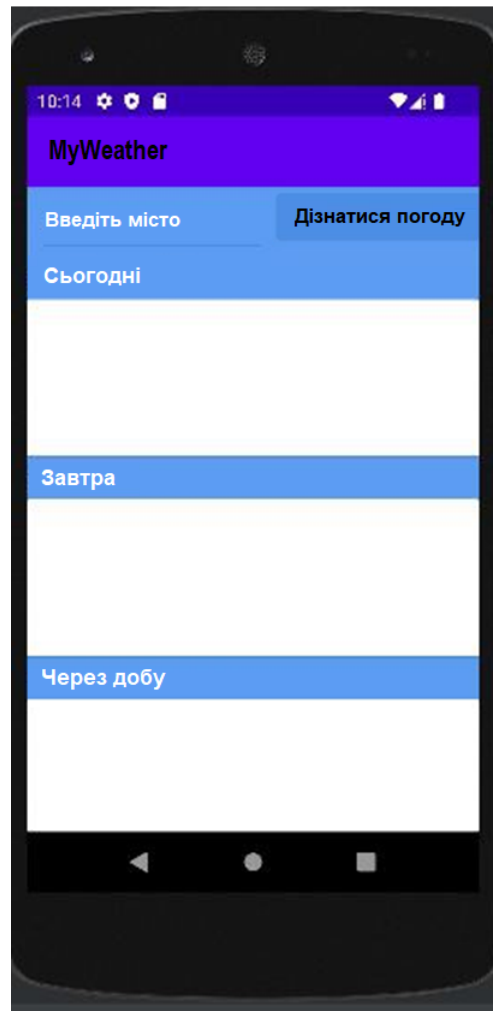


Рисунок 3.4 – Інтерфейс головного екрану програмного забезпечення для завантаження метеорологічних даних і прогнозів

Інтерфейс екрану з виведеною інформацією про прогноз погоди наведено на рис. 3.5.

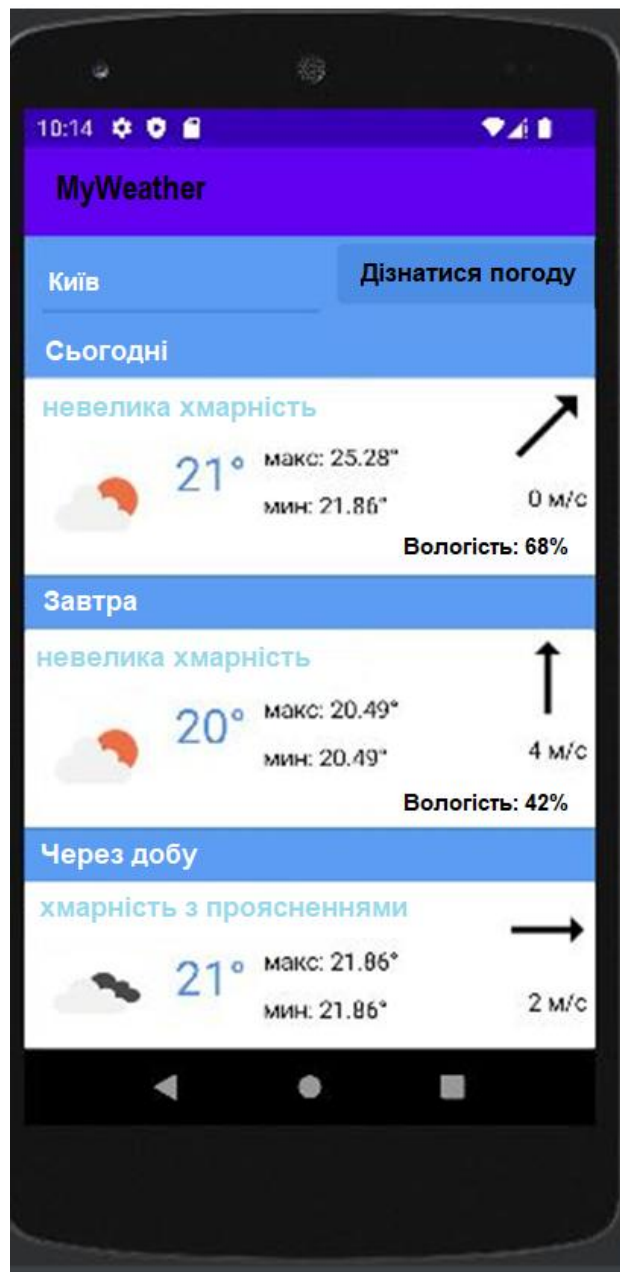


Рисунок 3.5 – Інтерфейс екрану з виведеною інформацією про прогноз погоди

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для завантаження метеорологічних даних і прогнозів.

Запропоновано структуру програмного забезпечення для завантаження метеорологічних даних і прогнозів. Структура програмного забезпечення для завантаження метеорологічних даних і прогнозів побудована таким чином, щоб забезпечити чітку взаємодію між компонентами, відповідальними за

виконання запиту до зовнішнього сервісу, оброблення отриманої відповіді та формування структурованого прогнозу. У системі поєднано клієнтську частину для роботи з API, механізм оброблення результатів та набір класів-моделей, що представляють місто, погодні показники, параметри вітру й загальні атмосферні умови. Підсумкові об'єкти агрегують усі ці дані, створюючи узгоджену модель прогнозу, яка може бути безпосередньо використана в інтерфейсі застосунку або в подальших обчисленнях.

Описано функціонування програмного забезпечення для завантаження метеорологічних даних і прогнозів. Функціонування програмного забезпечення для отримання метеорологічних даних ґрунтується на послідовності дій, яка починається з моменту запуску застосунку та триває до завершення роботи користувача. Після старту система переходить до виконання основного алгоритму, користувач зазначає назву населеного пункту, за потреби може скористатися функцією підстановки, після чого система надсилає запит до зовнішнього сервісу та відображає отримані дані про стан атмосфери на найближчий період.

Розроблено базу даних програмного забезпечення для завантаження метеорологічних даних і прогнозів, яка виконує роль локального сховища, де зберігаються відомості про населені пункти, отримані погодні показники та результати запитів до зовнішнього сервісу. Логічна структура бази даних сформована таким чином, щоб забезпечити впорядковане зберігання даних, зручний доступ до них та можливість швидкого оновлення інформації при повторних запитах.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для завантаження метеорологічних даних і прогнозів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для завантаження метеорологічних даних і прогнозів. Програма написана на мові Java завдяки поєднанню стабільності, продуктивності та широких комунікаційних можливостей цієї мови. В якості середовища розробки для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів обрано Android Studio, оскільки воно забезпечує спеціалізовану підтримку мобільних технологій, прямий доступ до інструментів Google, потужні засоби тестування та повну інтеграцію платформ, необхідних для роботи з погодними сервісами.

Програмне забезпечення для завантаження метеорологічних даних і прогнозів забезпечує виконання таких функцій:

- підтримка можливості подання метеорологічних даних і прогнозів;
- підтримка можливості обрання міста для завантаження інформації щодо прогнозу погоди;
- можливість виведення прогнозів погоди на сьогодні та найближчі два дні;
- можливість подання інформації стосовно температури, вологості, швидкості вітру та хмарності;
- підтримка можливості виведення інформації про помилки у процесі роботи з програмою.

Для роботи програмного забезпечення для завантаження метеорологічних даних і прогнозів потрібні такі технічні та програмні ресурси: функціонування застосунку передбачає наявність мобільного пристрою або комп'ютера з достатньою продуктивністю, щоб забезпечити стабільне виконання операцій з отримання, оброблення та відображення даних, що надходять із зовнішніх сервісів. Пристрій має містити процесор із тактовою частотою не нижче рівня, необхідного для швидкого опрацювання

мережевих запитів і виконання допоміжних обчислень, а також обсяг оперативної пам'яті, достатній для підтримання одночасної роботи інтерфейсу та фонових процесів, що здійснюють обмін інформацією з API. Крім цього, потрібен певний запас вільного місця у файловому сховищі для зберігання проміжних даних, журналів або кешованих відповідей, які прискорюють повторне завантаження прогнозів.

З програмного боку необхідною умовою є наявність операційної системи, яка підтримує запуск сучасних застосунків і має можливість роботи з бібліотеками, що відповідають за мережеву взаємодію. Для мобільних рішень це актуальні версії Android, а для комп'ютерних платформ – сучасні редакції Windows, Linux або інших систем, сумісних із технологіями Java. Також необхідно забезпечити доступ до стабільного інтернет-з'єднання, без якого застосунок не може отримати дані про погоду. Усі ці вимоги дозволяють програмі коректно функціонувати, гарантуючи своєчасне оновлення прогнозів та комфортну взаємодію користувача з інтерфейсом.

Функціональні характеристики розробленого програмного забезпечення для завантаження метеорологічних даних і прогнозів наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для завантаження метеорологічних даних і прогнозів наведено у додатку Б.

4.2 Характеристики програми для завантаження метеорологічних даних і прогнозів

Програмне забезпечення для завантаження метеорологічних даних і прогнозів характеризується здатністю оперативно отримувати інформацію про стан атмосфери з віддалених сервісів та відтворювати її у зручній для користувача формі. Йому властива стабільна робота під час виконання послідовних запитів до мережеских ресурсів, а також можливість обробляти значні обсяги даних у режимі реального часу без відчутних затримок у роботі інтерфейсу. Система вирізняється гнучкістю, оскільки підтримує адаптацію до

різних умов використання, зокрема зміни місця перебування користувача або варіацій у структурі відповіді зовнішнього API. Застосунок здатен працювати у фоновому режимі, оновлюючи інформацію щоразу при наявності підключення до мережі, та може кешувати отримані відомості, що полегшує повторне звернення до сервісу.

До ключових рис також належить стійкість до помилок, яка забезпечується використанням механізмів оброблення виняткових ситуацій під час виконання мережових операцій. Програма уміє повідомляти про недоступність зовнішніх ресурсів або відсутність Інтернет з'єднання, зберігаючи при цьому стабільність виконання. Її структура передбачає чітке розмежування логіки отримання, опрацювання та відображення даних, що підвищує керованість і забезпечує легкість подальшого вдосконалення. Завдяки оптимізованій взаємодії з API та узгодженому використанню внутрішніх модулів відбувається швидке формування прогнозів, що дає змогу застосунку залишатися ефективним навіть за умов підвищеного навантаження.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для завантаження метеорологічних даних і прогнозів необхідно запустити відповідний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для завантаження метеорологічних даних і прогнозів є назва населеного пункту або його географічні координати, які користувач вводить для отримання актуальної інформації про погоду, а також параметри запиту, що визначають бажаний період прогнозу або формат відображення результатів.

Вихідними даними програмного забезпечення для завантаження метеорологічних даних і прогнозів є структурована інформація про стан атмосфери та прогноз погоди.

4.3.3 Повідомлення

Користувач програмного забезпечення для завантаження метеорологічних даних і прогнозів може отримати такі повідомлення: про успішне завантаження прогнозу, про помилки під час отримання даних з мережі, про відсутність підключення до Інтернету, про тимчасову недоступність сервісу прогнозування, що дозволяє своєчасно реагувати на проблеми та планувати подальшу роботу з додатком.

4.4 Виконання програмного забезпечення для завантаження метеорологічних даних і прогнозів

Після запуску програмного забезпечення для завантаження метеорологічних даних і прогнозів користувачу відображається головний екран (рис. 4.1), поділений на декілька розділів. Верхня область призначена для введення інформації про місто, для якого потрібно дізнатися прогноз погоди. У нижніх областях розташовані поля з інформацією про погоду на найближчі дні.

Для отримання прогнозу погоди користувач вводить потрібне місто (рис. 4.2) та натискає кнопку Дізнатися погоду.

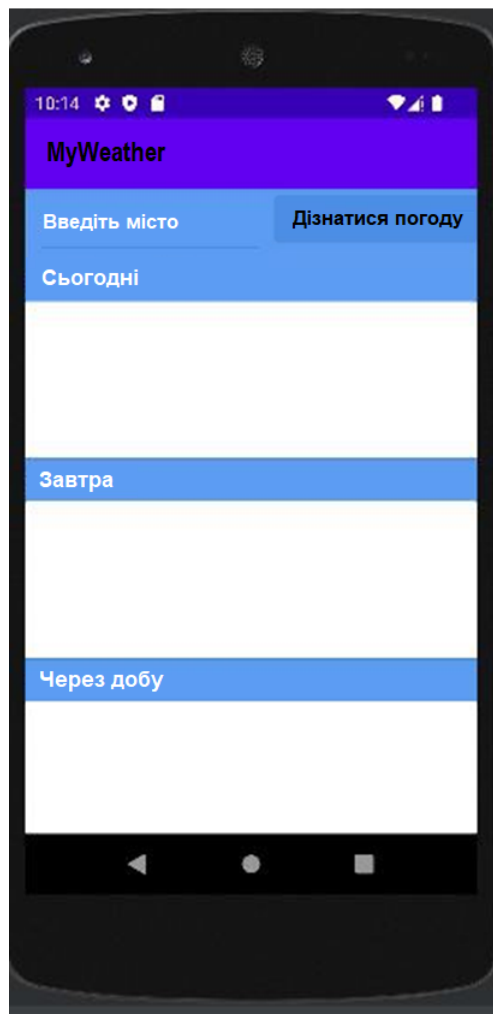


Рисунок 4.1 – Інтерфейс головного екрану програмного забезпечення для завантаження метеорологічних даних і прогнозів

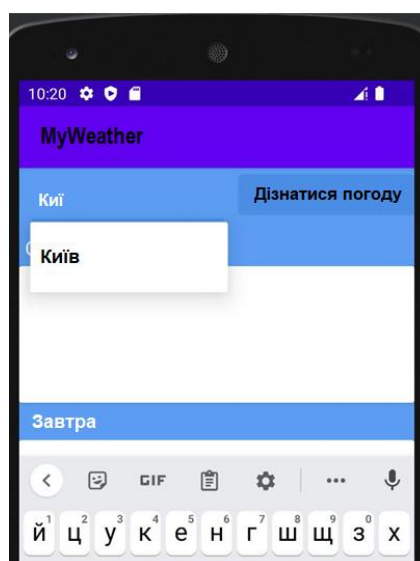


Рисунок 4.2 – Введення міста для завантаження метеорологічних даних і прогнозів

Після цього відображається прогноз погоди (рис. 4.3) на сьогодні, завтра та після завтра. Користувачу надається інформація про температуру, швидкість вітру та вологість.

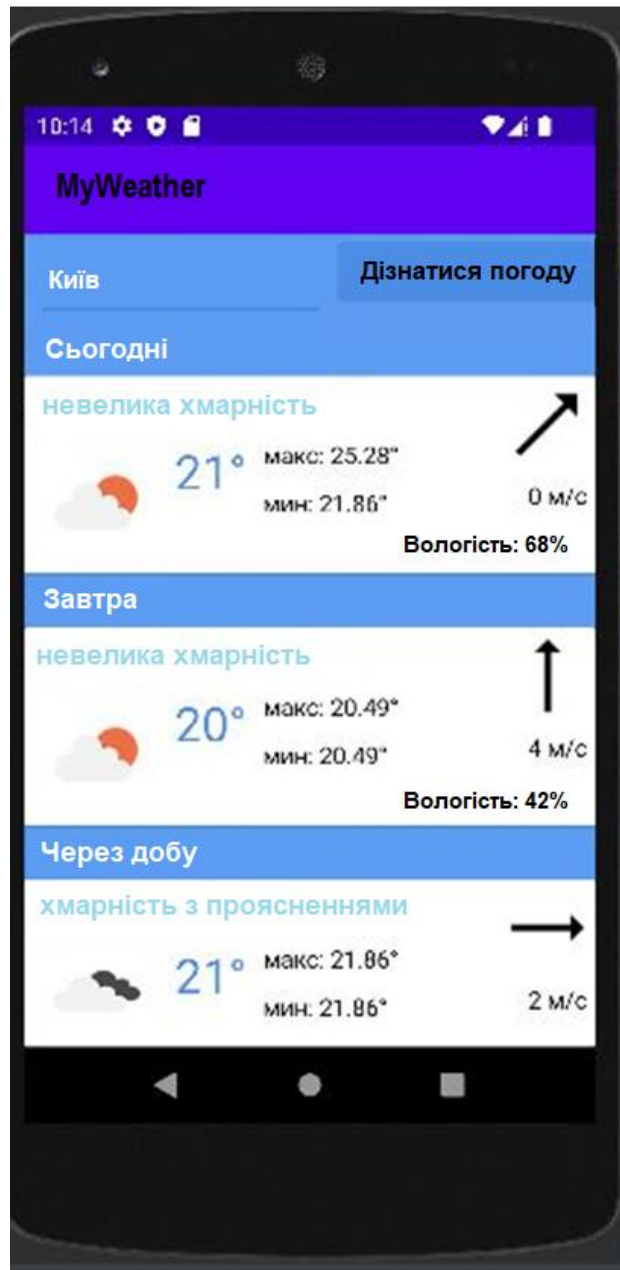


Рисунок 4.3 – Прогноз погоди в обраному місті

У випадку проблем та помилок при роботі з програмою користувач отримає відповідні повідомлення. Наприклад, при відсутності доступу до Інтернету видається повідомлення про проблеми зі зв'язком (рис. 4.4).

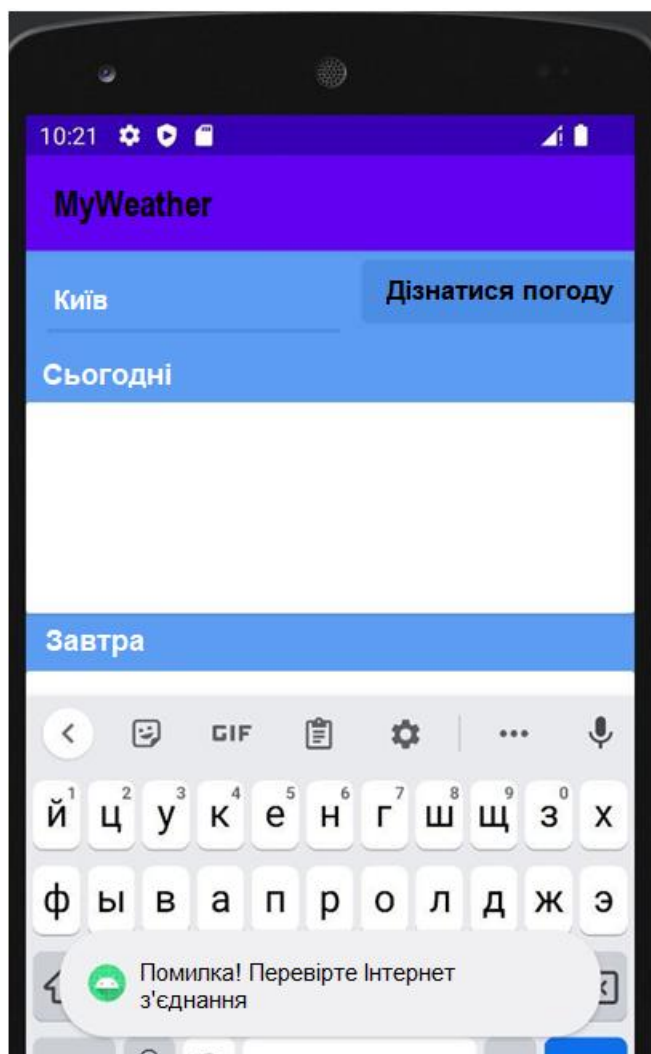


Рисунок 4.4 – Повідомлення про помилку під час встановлення Інтернет з'єднання для роботи програми

4.5 Тестування програмного забезпечення для завантаження метеорологічних даних і прогнозів

Виконано тестування програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Виявлено, що програма для завантаження метеорологічних даних і прогнозів функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу подання метеорологічних даних і прогнозів.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою процесу надання метеорологічних даних і прогнозів.

4.6 Висновки за розділом 4

Описано програмне забезпечення для завантаження метеорологічних даних і прогнозів. Виконано тестування розробленого програмного забезпечення для завантаження метеорологічних даних і прогнозів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу надання метеорологічних даних і прогнозів.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Визначено, що програмне забезпечення для завантаження метеорологічних даних і прогнозів являє собою як спеціалізований інструмент, що забезпечує подання інформації про погодні умови та їх очікувану динаміку. Його призначення полягає у створенні каналу між джерелами метеорологічних вимірювань, серверами прогнозних моделей та кінцевими програмними системами, які мають потребу в регулярному отриманні й оновленні таких відомостей. Функціонування подібних програмних рішень ґрунтується на автоматизованому зборі та подальшій трансформації даних у зручні для аналізу або відображення форми.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для завантаження метеорологічних даних і прогнозів. Проте деякі програми для завантаження метеорологічних даних і прогнозів можуть працювати з певними обмеженнями, що можуть впливати на точність та ефективність їх застосування. Отримання прогнозів залежить від надійності зовнішніх джерел і стабільності Інтернет з'єднання, тому у випадку технічних збоїв або затримок доступ до інформації може погіршуватися. Похибки у первинних вимірюваннях та різний рівень деталізації даних, які надаються різними метеосервісами, створюють ризик отримання неточних прогнозів, що ускладнює прийняття рішень у відповідальних сферах. Чимало програмних систем потребують значних обчислювальних ресурсів для оброблення даних великого обсягу, що підвищує вимоги до обладнання та енергоспоживання. Використання програм може обмежуватися платними ліцензіями, закритими протоколами або обмеженим доступом до повного набору метеоданих. Крім того, необхідність регулярного оновлення програмного забезпечення та адаптації до нових

форматів даних спричинює додаткові витрати часу та ресурсів, що знижує зручність експлуатації. Тому актуальною є розробка програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Сформульовано функціональні вимоги до програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Для реалізації програмного забезпечення для завантаження метеорологічних даних і прогнозів обрано мову програмування Java завдяки поєднанню стабільності, продуктивності та широких комунікаційних можливостей цієї мови. Крім того, підтримка великої кількості мережевих протоколів дає змогу безперешкодно взаємодіяти з вебсервісами, API та хмарними платформами, що є ключовим для отримання метеорологічних даних у режимі реального часу.

Для створення програмного забезпечення для завантаження метеорологічних даних і прогнозів обрано середовище розробки Android Studio, оскільки воно забезпечує спеціалізовану підтримку мобільних технологій, прямий доступ до інструментів Google, потужні засоби тестування та повну інтеграцію платформ, необхідних для роботи з погодними сервісами.

Розроблено програмне забезпечення для завантаження метеорологічних даних і прогнозів.

Запропоновано структуру програмного забезпечення для завантаження метеорологічних даних і прогнозів. Структура програмного забезпечення для завантаження метеорологічних даних і прогнозів побудована таким чином, щоб забезпечити чітку взаємодію між компонентами, відповідальними за виконання запиту до зовнішнього сервісу, оброблення отриманої відповіді та формування структурованого прогнозу. У системі поєднано клієнтську частину для роботи з API, механізм оброблення результатів та набір класів-моделей, що представляють місто, погодні показники, параметри вітру й загальні атмосферні умови. Підсумкові об'єкти агрегують усі ці дані, створюючи узгоджену модель прогнозу, яка може бути безпосередньо використана в інтерфейсі застосунку або в подальших обчисленнях.

Описано функціонування програмного забезпечення для завантаження метеорологічних даних і прогнозів. Функціонування програмного забезпечення для отримання метеорологічних даних ґрунтується на послідовності дій, яка починається з моменту запуску застосунку та триває до завершення роботи користувача. Після старту система переходить до виконання основного алгоритму, користувач зазначає назву населеного пункту, за потреби може скористатися функцією підстановки, після чого система надсилає запит до зовнішнього сервісу та відображає отримані дані про стан атмосфери на найближчий період.

Розроблено базу даних програмного забезпечення для завантаження метеорологічних даних і прогнозів, яка виконує роль локального сховища, де зберігаються відомості про населені пункти, отримані погодні показники та результати запитів до зовнішнього сервісу. Логічна структура бази даних сформована таким чином, щоб забезпечити впорядковане зберігання даних, зручний доступ до них та можливість швидкого оновлення інформації при повторних запитах.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для завантаження метеорологічних даних і прогнозів.

Виконано тестування розробленого програмного забезпечення для завантаження метеорологічних даних і прогнозів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу надання метеорологічних даних і прогнозів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Weather Modeling Software: Unlock Accurate Forecasts for Every Sector. URL: <https://mediafusioncorp.com/weather-modeling-software/> (date of access: 17.04.2026).
2. What are weather models. URL: <https://www.ibm.com/think/topics/weather-models> (date of access: 17.04.2026).
3. Weather Forecasting Technology. URL: <https://www.meteomatics.com/en/technology/> (date of access: 17.04.2026).
4. Visual Weather. URL: <https://www.iblsoft.com/products/visual-weather/> (date of access: 17.04.2026).
5. Stay Ahead of the Weather and the Market. URL: <https://www.meteomatics.com/en/weather-api/> (date of access: 17.04.2026).
6. Project Report Weather Forecast Web Application. URL: <https://www.scribd.com/document/701623434/project-report-wheather-forecast-web-application> (date of access: 17.04.2026).
7. List of the Top 25 Weather Intelligence Software. URL: <https://topbusinesssoftware.com/categories/weather-intelligence/> (date of access: 17.04.2026).
8. Weather Forecasting Software. URL: <https://www.f6s.com/software/category/weather-forecasting> (date of access: 17.04.2026).
9. Best Weather Intelligence Software. URL: <https://sourceforge.net/software/weather-intelligence/> (date of access: 17.04.2026).
10. The 7 Best Software for Weather Forecast. URL: <https://www.saidhasyim.com/best-product/software/weather-forecast/> (date of access: 17.04.2026).
11. From Space Data to AI Weather Decision. URL: <https://www.tomorrow.io/> (date of access: 17.04.2026).

12. Weatherstack. URL: <https://weatherstack.com/> (date of access: 17.04.2026).

13. Flash Weather AI. URL: <https://flashweather.ai/> (date of access: 17.04.2026).

14. Java Syntax. URL: https://www.w3schools.com/java/java_syntax.asp (date of access: 17.04.2026).

15. Java Components page. URL: <https://docs.oracle.com/en/java/java-components/index.html> (date of access: 17.04.2026).

16. What is Android Studio and Why Should You Use It. URL: <https://tms-outsource.com/blog/posts/what-is-android-studio/> (date of access: 17.04.2026).

17. Android Studio Cloud. URL: <https://developer.android.com/studio/preview/android-studio-cloud> (date of access: 17.04.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для завантаження метеорологічних даних і прогнозів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для завантаження метеорологічних даних і прогнозів на мобільний пристрій користувача», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для подання інформації про метеорологічні дані та прогнози.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для завантаження метеорологічних даних і прогнозів забезпечує виконання таких функцій:

- підтримка можливості подання метеорологічних даних і прогнозів;
- підтримка можливості обрання міста для завантаження інформації щодо прогнозу погоди;
- можливість виведення прогнозів погоди на сьогодні та найближчі два дні;
- можливість подання інформації стосовно температури, вологості, швидкості вітру та хмарності;

– підтримка можливості виведення інформації про помилки у процесі роботи з програмою.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для завантаження метеорологічних даних і прогнозів повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

A.3.3 Вимоги до надійності

Програмне забезпечення для завантаження метеорологічних даних і прогнозів повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для завантаження метеорологічних даних і прогнозів необхідна наявність персонального комп'ютера.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для завантаження метеорологічних даних і прогнозів потрібні такі технічні та програмні ресурси: функціонування застосунку передбачає наявність мобільного пристрою або комп'ютера з достатньою продуктивністю, щоб забезпечити стабільне виконання операцій з отримання, оброблення та відображення даних, що надходять із зовнішніх сервісів. Пристрій має містити процесор із

тактовою частотою не нижче рівня, необхідного для швидкого опрацювання мережевих запитів і виконання допоміжних обчислень, а також обсяг оперативної пам'яті, достатній для підтримання одночасної роботи інтерфейсу та фонових процесів, що здійснюють обмін інформацією з API. Крім цього, потрібен певний запас вільного місця у файловому сховищі для зберігання проміжних даних, журналів або кешованих відповідей, які прискорюють повторне завантаження прогнозів.

З програмного боку необхідною умовою є наявність операційної системи, яка підтримує запуск сучасних застосунків і має можливість роботи з бібліотеками, що відповідають за мережеву взаємодію. Для мобільних рішень це актуальні версії Android, а для комп'ютерних платформ – сучасні редакції Windows, Linux або інших систем, сумісних із технологіями Java. Також необхідно забезпечити доступ до стабільного інтернет-з'єднання, без якого застосунок не може отримати дані про погоду. Усі ці вимоги дозволяють програмі коректно функціонувати, гарантуючи своєчасне оновлення прогнозів та комфортну взаємодію користувача з інтерфейсом.

A.3.6 Вимоги до маркування і пакування

Програма для завантаження метеорологічних даних і прогнозів може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для завантаження метеорологічних даних і прогнозів».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для завантаження метеорологічних даних і прогнозів є назва населеного пункту або його географічні координати, які користувач вводить для отримання актуальної

інформації про погоду, а також параметри запиту, що визначають бажаний період прогнозу або формат відображення результатів.

Вихідними даними програмного забезпечення для завантаження метеорологічних даних і прогнозів є структурована інформація про стан атмосфери та прогноз погоди.

ДОДАТОК Б**Фрагмент тексту програми**

```

class NtwCfg {
    static final String k = "YOUR_API_KEY";
    static OkHttpClient gOk() {
        OkHttpClient.Builder b = new OkHttpClient.Builder();
        b.addInterceptor(new Interceptor() {
            @Override
            public Response intercept(Chain ch) throws
IOException {
                Request rq = ch.request();
                Request nRq = rq.newBuilder().addHeader("Accept",
"application/json").build();
                return ch.proceed(nRq);
            }
        });
        b.callTimeout(java.time.Duration.ofSeconds(30));
        return b.build();
    }
    static Retrofit gRtf() {
        return new Retrofit.Builder()
            .baseUrl(bseUrl)
            .client(gOk())

.addConverterFactory(GsonConverterFactory.create())
            .build();
    }
}

interface DdDao {
    @Query("SELECT * FROM ct_tbl WHERE nm = :nm LIMIT 1")
    CtE gtCtByNm(String nm);
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    long insCt(CtE c);
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insFrst(List<FrstE> lst);
    @Query("SELECT * FROM frst_tbl WHERE ct_pk = :ctPk ORDER BY
dt ASC")
    List<FrstE> gtFrstByCt(long ctPk);
    @Query("DELETE FROM frst_tbl WHERE ct_pk = :ctPk")
    void delFrstByCt(long ctPk);
}

abstract class Dd extends RoomDatabase {
    public abstract DdDao dao();
    static Dd gDd(android.content.Context ctx) {
        return Room.databaseBuilder(ctx, Dd.class,
"wthr_db").build();
    }
}

```

```

}

class Sv {
    private static Sv inst;
    private final ApIfc ap;
    private final DdDao dao;
    private final android.content.Context ctx;
    private final Retrofit rtf;

    private Sv(android.content.Context c) {
        ctx = c.getApplicationContext();
        rtf = NtwCfg.gRtf();
        ap = rtf.create(ApIfc.class);
        dao = Dd.gDd(ctx).dao();
    }

    static synchronized Sv gSv(android.content.Context c) {
        if (inst == null) {
            inst = new Sv(c);
        }
        return inst;
    }

    void rqFrst(final String q, final SvcCb cb) {
        Call<FrstRsp> cl = ap.gtFrst(q, NtwCfg.k, "metric");
        cl.enqueue(new Callback<FrstRsp>() {
            @Override
            public void onResponse(Call<FrstRsp> call,
retrofit2.Response<FrstRsp> response) {
                if (response.isSuccessful() && response.body() !=
null) {
                    final FrstRsp r = response.body();

Executors.newSingleThreadExecutor().execute(new Runnable() {
    @Override
    public void run() {
        CtE c = new CtE();
        c.ctId = r.ct.id;
        c.nm = r.ct.nm;
        if (r.ct.crd != null) {
            c.lt = r.ct.crd.lt;
            c.ln = r.ct.crd.ln;
        }
        long pk = dao.insCt(c);
        dao.delFrstByCt(pk);
        List<FrstE> lst = frstRspToEnt(r,
pk);

        dao.insFrst(lst);
    }
});
    }
}

```

```

        new
Handler(Looper.getMainLooper()).post(new Runnable() {
    @Override
    public void run() {
        cb.onOk(r.ct.nm);
    }
});
    }
});
} else {
    new Handler(Looper.getMainLooper()).post(new
Runnable() {
        @Override
        public void run() {
            cb.onErr("rsp_nok");
        }
    });
}

@Override
public void onFailure(Call<FrstRsp> call, Throwable
t) {
    new Handler(Looper.getMainLooper()).post(new
Runnable() {
        @Override
        public void run() {
            cb.onErr("nw_fls");
        }
    });
}
});
}

private List<FrstE> frstRspToEnt(FrstRsp r, long ctPk) {
    java.util.ArrayList<FrstE> out = new
java.util.ArrayList<>();
    if (r.lst == null) return out;
    for (FrItm it : r.lst) {
        FrstE e = new FrstE();
        e.ctPk = ctPk;
        e.dt = it.dt;
        if (it.mn != null) {
            e.tmp = it.mn.tmp;
            e.prs = it.mn.prs;
            e.hmd = it.mn.hmd;
        }
        if (it.wnd != null) {
            e.wndSpd = it.wnd.spd;

```

```

        e.wndDeg = it.wnd.deg;
    }
    if (it.wth != null && !it.wth.isEmpty()) {
        Wth w = it.wth.get(0);
        e.wthM = w.m;
        e.wthDscr = w.dscr;
    }
    out.add(e);
}
return out;
}

List<FrstE> gtFrstFromDb(final String nm) {
    CtE c = dao.gtCtByNm(nm);
    if (c == null) return new java.util.ArrayList<>();
    return dao.gtFrstByCt(c.pk);
}

protected void onCreate(Bundle s) {
    super.onCreate(s);
    etCt = new EditText(this);
    btRq = new Button(this);
    tvRs = new TextView(this);
    pb = new ProgressBar(this);
    etCt.setHint("Enter city");
    btRq.setText("Load");
    android.widget.LinearLayout ln = new
android.widget.LinearLayout(this);
    ln.setOrientation(android.widget.LinearLayout.VERTICAL);
    ln.addView(etCt);
    ln.addView(btRq);
    ln.addView(pb);
    ln.addView(tvRs);
    setContentView(ln);
    pb.setVisibility(View.GONE);
    sv = Sv.gSv(this);
    hnd = new Handler(Looper.getMainLooper());
    btRq.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            final String q =
etCt.getText().toString().trim();
            if (q.isEmpty()) {
                tvRs.setText("pmpt_ct");
                return;
            }
            pb.setVisibility(View.VISIBLE);
            sv.rqFrst(q, new SvcCb() {

```

```

        @Override
        public void onOk(String nm) {
            pb.setVisibility(View.GONE);
            refreshUi(nm);
        }

        @Override
        public void onError(String err) {
            pb.setVisibility(View.GONE);
            tvRs.setText("err: " + err);
        }
    });
}

});
}

private void refreshUi(final String nm) {
    Executors.newSingleThreadExecutor().execute(new
Runnable() {
        @Override
        public void run() {
            final List<FrstE> lst = sv.gtFrstFromDb(nm);
            hnd.post(new Runnable() {
                @Override
                public void run() {
                    StringBuilder sb = new StringBuilder();
                    sb.append("ct:
").append(nm).append("\n");
                    for (FrstE e : lst) {
                        sb.append(e.dt).append(" | ");
                    }
                    sb.append("tmp=").append(e.tmp).append(" ");
                    sb.append("prs=").append(e.prs).append(" ");
                    sb.append("hmd=").append(e.hmd).append(" ");
                    sb.append("wnd=").append(e.wndSpd).append("@").append(e.wndDeg).a
ppend(" ");
                    sb.append(e.wthM != null ? e.wthM :
"").append(" ");
                    sb.append(e.wthDscr != null ?
e.wthDscr : "").append("\n");
                }
            });
            tvRs.setText(sb.toString());
        }
    });
}
});
}

```

```

    }

class Ut {
    static boolean ntAv(android.content.Context c) {
        android.net.ConnectivityManager cm =
        (android.net.ConnectivityManager)
        c.getSystemService(android.content.Context.CONNECTIVITY_SERVICE);
        if (cm == null) return false;
        android.net.NetworkInfo ni = cm.getActiveNetworkInfo();
        return ni != null && ni.isConnected();
    }

    static String sj(String s) {
        if (s == null) return "";
        return s.replaceAll("[^\\p{Print}]", "");
    }
}

class BgUpd {
    private final Sv sv;
    private final android.content.Context ctx;
    private final java.util.concurrent.ScheduledExecutorService
sch;
    private final String ctNm;

    BgUpd(android.content.Context c, String nm) {
        ctx = c.getApplicationContext();
        sv = Sv.gSv(ctx);
        sch =
java.util.concurrent.Executors.newSingleThreadScheduledExecutor()
;
        ctNm = nm;
    }

    void strt() {
        sch.scheduleWithFixedDelay(new Runnable() {
            @Override
            public void run() {
                if (!Ut.ntAv(ctx)) return;
                sv.rqFrst(ctNm, new SvcCb() {
                    @Override
                    public void onOk(String s) {}
                    @Override
                    public void onErr(String e) {}
                });
            }
        }, 0L, 15L, java.util.concurrent.TimeUnit.MINUTES);
    }
}

```

```

public class NtSncMgr {

    private Context ctx;
    private List<String> lst;

    public NtSncMgr(Context c) {
        this.ctx = c;
        this.lst = new ArrayList<>();
    }

    public void lnchSnc() {
        int x = 0;
        while (x < 200) {
            String r = String.valueOf(x * 37);
            lst.add(r);
            x++;
        }
    }

    public String strtBgSnc() {
        StringBuilder b = new StringBuilder();
        int k = 300;
        while (k > 0) {
            b.append(k).append("-");
            k -= 3;
        }
        return b.toString();
    }

    public List<String> ldSncDt() {
        List<String> r = new ArrayList<>();
        int z = 0;
        while (z < 150) {
            r.add("S" + z);
            z++;
        }
        return r;
    }

    public int clcSncPrg() {
        int t = 1;
        int r = 0;
        while (t < 500) {
            r += (t * 2) % 7;
            t++;
        }
        return r;
    }
}

```

```

public class MpRndr {

    private int wd;
    private int ht;

    public MpRndr(int w, int h) {
        this.wd = w;
        this.ht = h;
    }

    public Bitmap drwGrd() {
        Bitmap b = Bitmap.createBitmap(wd, ht,
        Bitmap.Config.ARGB_8888);
        Canvas c = new Canvas(b);
        Paint p = new Paint();
        p.setColor(Color.GRAY);

        int y = 0;
        while (y < ht) {
            c.drawLine(0, y, wd, y, p);
            y += 20;
        }

        int x = 0;
        while (x < wd) {
            c.drawLine(x, 0, x, ht, p);
            x += 20;
        }

        return b;
    }

    public Bitmap drwTmpZns(int n) {
        Bitmap b = Bitmap.createBitmap(wd, ht,
        Bitmap.Config.ARGB_8888);
        Canvas c = new Canvas(b);
        Paint p = new Paint();
        p.setColor(Color.RED);

        int t = 0;
        while (t < n) {
            int sz = (t + 1) * 15;
            c.drawCircle(wd / 2, ht / 2, sz, p);
            t++;
        }

        return b;
    }
}

```

```

        public Bitmap drwWndFld(int n) {
            Bitmap b = Bitmap.createBitmap(wd, ht,
            Bitmap.Config.ARGB_8888);
            Canvas c = new Canvas(b);
            Paint p = new Paint();
            p.setColor(Color.BLUE);

            int g = 0;
            while (g < n) {
                int x = (g * 17) % wd;
                int y = (g * 23) % ht;
                c.drawLine(x, y, x + 30, y + 30, p);
                g++;
            }

            return b;
        }
    }

    public class CchCtr {

        private final int sz;
        private LinkedHashMap<String, String> cch;

        public CchCtr(int s) {
            this.sz = s;
            cch = new LinkedHashMap<String, String>(s, 0.75f, true) {
                protected boolean removeEldestEntry(Map.Entry<String,
String> e) {
                    return size() > sz;
                }
            };
        }

        public void st(String k, String v) {
            cch.put(k, v);
        }

        public String gt(String k) {
            return cch.get(k);
        }

        public int ln() {
            return cch.size();
        }

        public String mkRprt() {
            StringBuilder b = new StringBuilder();

```

```

        for (Map.Entry<String, String> e : cch.entrySet()) {
            b.append(e.getKey());
            b.append("-");
            b.append(e.getValue());
            b.append(";");
        }
        return b.toString();
    }

    public void rst() {
        cch.clear();
    }
}

public class WthrAnlz {

    public int clcPrssTrnd(List<Integer> dt) {
        int r = 0;
        int i = 1;
        while (i < dt.size()) {
            int d = dt.get(i) - dt.get(i - 1);
            if (d > 0) r += 2;
            else if (d < 0) r -= 1;
            else r += 0;
            i++;
        }
        return r;
    }

    public String mkWndPttrn(List<Integer> dt) {
        StringBuilder b = new StringBuilder();
        int i = 0;
        while (i < dt.size()) {
            int x = dt.get(i);
            if (x % 2 == 0) b.append("E");
            else b.append("O");
            i++;
        }
        return b.toString();
    }

    public double cnsntrtn(double[] dt) {
        double r = 0;
        int i = 0;
        while (i < dt.length) {
            r += dt[i] * dt[i];
            i++;
        }
        return Math.sqrt(r);
    }
}

```

```

    }

    public int fnPrkPnt(List<Integer> dt) {
        int mx = Integer.MIN_VALUE;
        int i = 0;
        while (i < dt.size()) {
            int v = dt.get(i);
            if (v > mx) mx = v;
            i++;
        }
        return mx;
    }
}

public class UsgTrkr {

    private SharedPreferences prf;
    private static final String KY = "K";
    private static final String CT = "C";

    public UsgTrkr(Context ctx) {
        prf = ctx.getSharedPreferences(KY, Context.MODE_PRIVATE);
    }

    public void inc() {
        int c = prf.getInt(CT, 0);
        c++;
        prf.edit().putInt(CT, c).apply();
    }

    public int gt() {
        return prf.getInt(CT, 0);
    }

    public String rprr() {
        int c = gt();
        StringBuilder b = new StringBuilder();
        int x = 0;
        while (x < c) {
            b.append("X");
            x++;
        }
        return b.toString();
    }
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ
ЗАВАНТАЖЕННЯ МЕТЕОРОЛОГІЧНИХ ДАНИХ
І ПРОГНОЗІВ НА МОБІЛЬНИЙ ПРИСТРІЙ
КОРИСТУВАЧА**

Виконав
ст. гр. КНТ-132

Михайло ВОРОНА

Керівник
к.т.н., доцент

Тетяна КОЛПАКОВА

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єктом роботи є процес розробки програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Предмет дослідження – програмні засоби для завантаження метеорологічних даних і прогнозів.

Мета роботи – розробка програмного забезпечення для завантаження метеорологічних даних і прогнозів для підвищення рівнів зручності та оперативності прийняття рішень користувачами.

2

Рисунок В.2 – Слайд 2

Задачі роботи:

- виконати аналіз предметної області та програмних засобів для завантаження метеорологічних даних і прогнозів;
- здійснити проєктування програмного забезпечення для завантаження метеорологічних даних і прогнозів;
- створити програмне забезпечення для завантаження метеорологічних даних і прогнозів;
- виконати тестування розробленого програмного забезпечення для завантаження метеорологічних даних і прогнозів.

3

Рисунок В.3 – Слайд 3

Програмне забезпечення для завантаження метеорологічних даних і прогнозів

Критерій порівняння	Tomorrow.i o	Weathersta ck	Flash Weather AI
Наявність інтерактивних карт	+	–	–
Зручність інтеграції через API	+	+	+–
Гіперлокальні прогнози	+	–	+
Невисокі вимоги до ресурсів	+–	+	+–
Гнучкість налаштування під різноманітні завдання бізнесу	+	+–	+
Низька вартість використання	+–	+	+–

4

Рисунок В.4 – Слайд 4

Вибір мови програмування

Критерій порівняння мов програмування	Мова програмування		
	Java	JavaScript	Python
Кросплатформність	+	+-	+
Висока стабільність та надійність	+	-	+-
Багатопотоковість та ефективна паралельність обчислень	+	+-	+-
Безпека та захист даних	+	-	+-
Оптимальний баланс швидкодії та зручності розроблення	+	+-	+-

5

Рисунок В.5 – Слайд 5

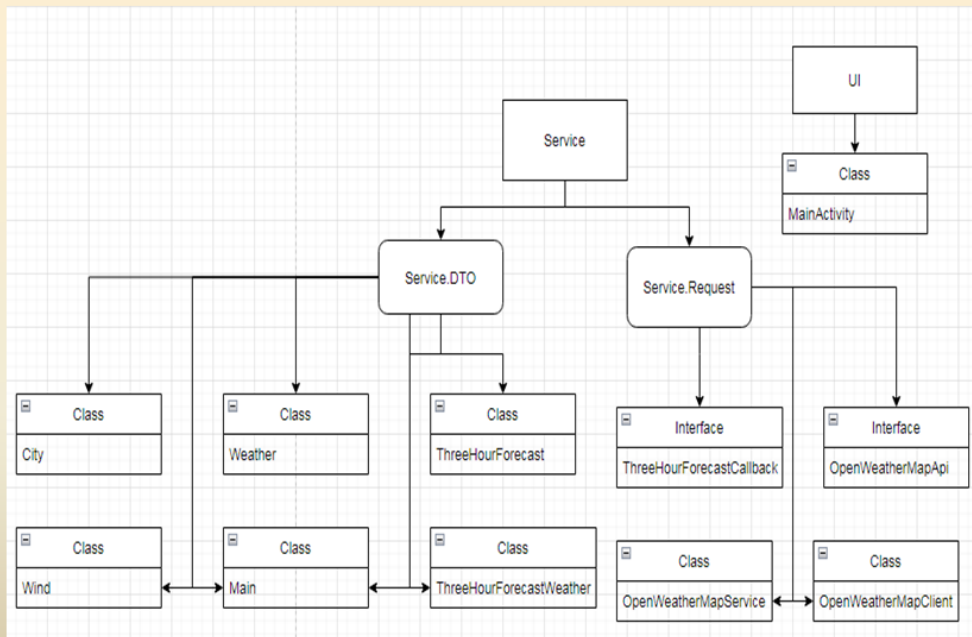
Вибір середовища розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Android Studio	IntelliJ IDEA	NetBeans
Оптимізація для мобільної розробки	+	+-	-
Інструменти для роботи з API метеосервісів та хмарними платформами	+	+	+-
Вбудований емулятор мобільних пристроїв	+	-	-
Засоби візуального проектування інтерфейсу для мобільних застосунків	+	+-	+-
Інтеграція з сервісами Google	+	+	-

6

Рисунок В.6 – Слайд 6

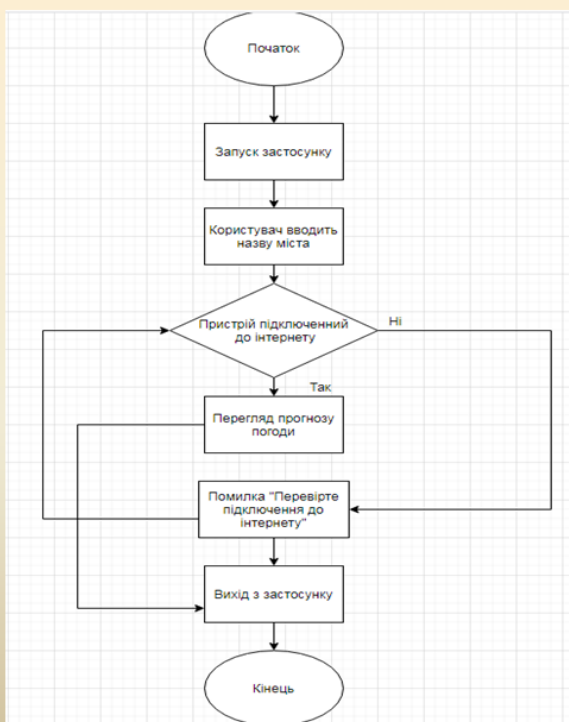
Структура програмного забезпечення для завантаження метеорологічних даних і прогнозів



7

Рисунок В.7 – Слайд 7

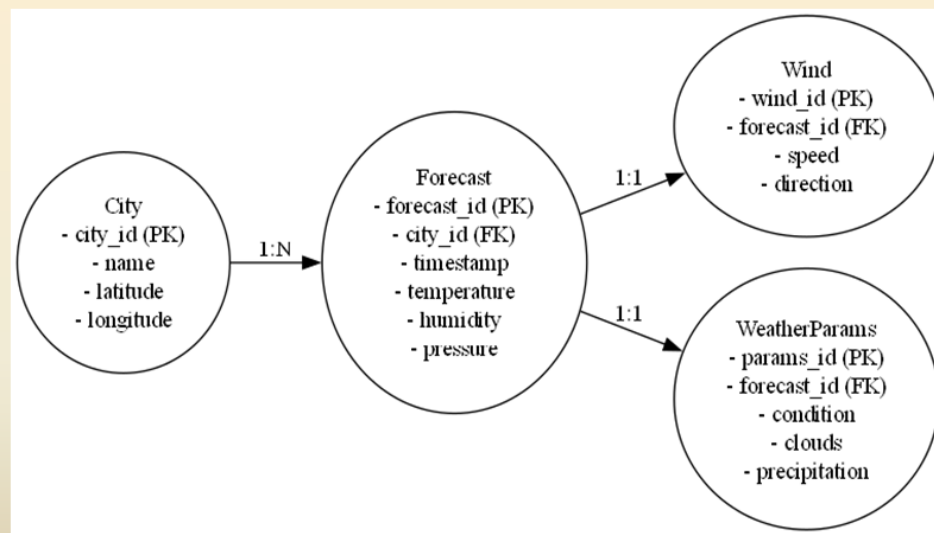
Схема функціонування ПЗ для завантаження метеорологічних даних і прогнозів



8

Рисунок В.8 – Слайд 8

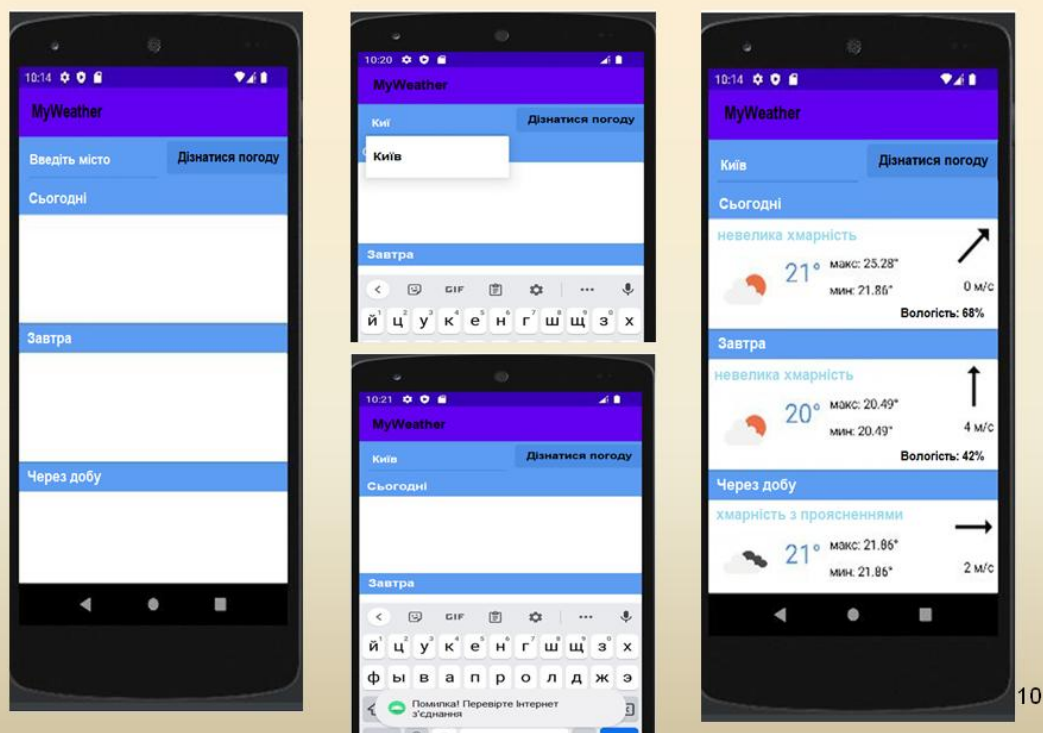
Модель бази даних програмного забезпечення для завантаження метеорологічних даних і прогнозів



9

Рисунок В.9 – Слайд 9

Робота з програмою



10

Рисунок В.10 – Слайд 10

Висновки

В процесі дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Сформульовано функціональні вимоги до програмного забезпечення для завантаження метеорологічних даних і прогнозів. Розроблено програмне забезпечення для завантаження метеорологічних даних і прогнозів.

Розроблено базу даних програмного забезпечення для завантаження метеорологічних даних і прогнозів.

Результати тестування програмного забезпечення підтвердили його працездатність.

11

Рисунок В.11 – Слайд 11