Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Бабко Владислава Сергійовича*
(ПІБ)

академічної групи *121-20ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: Розробка мобільного застосунку трансформації даних агротехнічного моніторингу

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Лактіонов І.С.*			
розділів:				
спеціальний	*проф. Лактіонов І.С.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*ст. викладач Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-20ск-1* (група) *Бабко Владислава Сергійовича* (прізвище та ініціали)

тема кваліфікаційної роботи Розробка мобільного застосунку трансформації даних агротехнічного моніторингу

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *Проф. Лактіонов І.С.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Бабко В.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 76 с., 19 рис., 1 таблиця, 3 дод., 16 джерел.

Об'єкт розробки: мобільний застосунок для агротехнічного моніторингу, що орієнтований на відображення зібраних даних

Мета кваліфікаційної роботи полягає у підвищенні процесів збору, обробки та візуалізації даних агротехнічного моніторингу за рахунок розробки відповідного застосунку. Об'єктом дослідження є розробка програмного продукту, який відповідає галузі спеціальності «Інженерія програмного забезпечення» та враховує узагальнену тематику кваліфікаційних робіт.

У вступному розділі проводиться аналіз та огляд сучасного стану проблеми агротехнічного моніторингу, конкретизується мета кваліфікаційної роботи та вказується зв'язок зі спеціальністю "Інженерія програмного забезпечення". Обґрунтовується актуальність теми та визначаються основні завдання роботи.

У першому розділі проводиться детальний аналіз предметної галузі агротехнічного моніторингу. Визначається актуальність завдання розробки мобільного застосунку для збору, обробки та візуалізації даних. Формулюється постановка завдання, в якій визначаються вимоги до програмної реалізації, необхідні технології та програмні засоби.

У другому розділі аналізуються наявні рішення та обрано платформу для розробки мобільного застосунку. Виконується проектування та розробка застосунку, описується його робота, алгоритм та структура функціонування. Також розглядаються виклик та завантаження застосунку, визначаються вхідні та вихідні дані, охарактеризуються параметри технічних засобів, необхідних для роботи застосунку.

В економічному розділі проводиться оцінка трудомісткості розробки мобільного застосунку та підрахунок вартості робіт. Розраховується час, необхідний для створення застосунку.

Практичне значення роботи полягає в розробці та налаштуванні мобільного застосунку для збору, обробки та візуалізації даних агротехнічного моніторингу. Цей застосунок підвищить зручність у роботі сільськогосподарським підприємствам та допоможе в прийнятті обґрунтованих рішень для підвищення продуктивності та ефективності сільськогосподарської діяльності.

Ключові слова: МОБІЛЬНИЙ ЗАСТОСУНОК, АГРОТЕХНІЧНИЙ МОНІТОРИНГ, ДАНІ, ЗБІР ДАНИХ, ОБРОБКА ДАНИХ, ВІЗУАЛІЗАЦІЯ ДАНИХ, СІЛЬСЬКЕ ГОСПОДАРСТВО, C#, .NET, XAMARIN.

ABSTRACT

Explanatory note: 76 p., 19 fig., 1 table, 3 app., 16 sources.

Object of development: mobile application for agrotechnical monitoring focused on displaying collected data

The purpose of the qualification work is to improve the processes of collection, processing and visualization of agrotechnical monitoring data through the development of an appropriate application. The object of the research is the development of a software product that corresponds to the field of specialty "Software Engineering" and takes into account the general subject of qualification works.

The introductory section analyzes and reviews the current state of the problem of agrotechnical monitoring, specifies the purpose of the qualification work and indicates the connection with the field of study «Software Engineering». The relevance of the topic is substantiated and the main tasks of the work are defined.

The first section provides a detailed analysis of the subject area of agrotechnical monitoring. The relevance of the task of developing a mobile application for data collection, processing and visualization is determined. The task statement is formulated, which defines the requirements for software implementation, necessary technologies and software tools.

The second section analyzes existing solutions and selects a platform for developing a mobile application. The design and development of the application is carried out, its operation, algorithm and structure are described. It also describes the application call and download, defines the input and output data, and characterizes the parameters of the technical means required for the application operation.

The economic section assesses the labor intensity of mobile application development and calculates the cost of work. The time required to create the application is calculated.

The practical significance of the work is to develop and configure a mobile application for collecting, processing and visualizing agrotechnical monitoring data. It will provide convenience in the work of agricultural enterprises and help in making informed decisions to increase the productivity and efficiency of agricultural activities.

Keywords: MOBILE APPLICATION, AGROTECHNICAL MONITORING, DATA, DATA COLLECTION, DATA PROCESSING, DATA VISUALIZATION, AGRICULTURE, C#, .NET, XAMARIN.

ЗМІСТ

ВСТУП

Завдання даної кваліфікаційної роботи та об'єкт її діяльності безпосередньо пов'язані зі спеціальністю «Інженерія програмного забезпечення» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню та компетенціям, якими повинні володіти бакалаври спеціальності 121 «Інженерія програмного забезпечення».

Об'єкт дослідження даної роботи - розробка мобільного застосунку, який буде забезпечувати збір, обробку та візуалізацію даних агротехнічного моніторингу. Розробка такого застосунку вимагає володіння компетенціями з програмування, архітектури програмного забезпечення, розробки інтерфейсів та аналізу даних. Дана робота передбачає застосування знань і вмінь, отриманих під час навчання в рамках програми бакалаврської підготовки з комп'ютерних наук.

Застосунок буде розроблений з використанням фреймворку Xamarin.Forms та мови програмування C#. Він буде спрямований на платформу Android і надасть користувачам зручний та ефективний інструмент для збору, обробки та візуалізації даних агротехнічного моніторингу.

Основною метою розробки застосунку є створення безпечного та ефективного засобу для збору даних агротехнічного моніторингу, їх обробки та візуалізації. Застосунок надасть сільськогосподарським підприємствам зручність у роботі та допоможе у прийнятті обґрунтованих рішень для підвищення продуктивності та ефективності сільськогосподарської діяльності.

Результатом виконання даної кваліфікаційної роботи буде функціональний мобільний застосунок, який забезпечить безпечний та ефективний спосіб збору, обробки та візуалізації даних агротехнічного моніторингу. Застосунок буде містити необхідні функціональні можливості для зручного взаємодії з користувачем та аналізу отриманих даних.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Агротехнічний моніторинг є важливим етапом в сільському господарстві, який забезпечує збір та аналіз даних для покращення вирощування рослин і підвищення ефективності сільськогосподарських процесів. Цей розділ пропонує загальний огляд предметної галузі агротехнічного моніторингу.

Агротехнічний моніторинг передбачає збір і обробку даних про різні аспекти сільськогосподарських умов, таких як ґрунтові параметри, кліматичні умови, вологість, рослини та інші фактори. Ці дані дозволяють аграрним підприємствам та фермерам отримувати об'єктивну інформацію для прийняття рішень щодо поліпшення агротехніки, внесення агрохімікатів, поливу, розміщення рослин тощо.

У сучасній агротехнології використовуються різноманітні джерела даних, такі як сенсорні пристрої на полях, метеорологічні станції, супутникові знімки та бази даних. Ці джерела забезпечують збір актуальної інформації про різні аспекти сільськогосподарської діяльності.

У контексті даної роботи мобільний застосунок слугує засобом візуалізації і представлення отриманих даних. Він надає можливість користувачам переглядати статистику, графіки та інші візуальні дані, що відображають результати агротехнічного моніторингу. Мобільний застосунок дозволяє зручний доступ до цих даних, незалежно від місця та часу.

Основна функція мобільного застосунку полягає в тому, щоб користувачі могли швидко та зручно отримати доступ до актуальної статистики, що стосується агротехнічного моніторингу. Він може відображати інформацію про вологість ґрунту, температуру, вміст поживних речовин, фізичний стан рослин і інші показники. Крім того, мобільний застосунок може надати користувачам можливість порівнювати дані з різних часових періодів, використовувати

фільтри для вибору конкретних параметрів та здійснювати аналіз даних на основі графіків та діаграм.

Проте, варто зауважити, що мобільний застосунок сам по собі не займається збором або обробкою даних. Він покликаний виконувати роль інтерфейсу, який забезпечує зручний доступ до статистики, що була зібрана та оброблена за допомогою інших технологій та інструментів. Таким чином, мобільний застосунок трансформує інформацію з різних джерел у зручну форму для користувачів, що дозволяє їм зробити обґрунтовані рішення на основі надійних даних.

В рамках аналізу відомих розробок, були вивчені різноманітні мобільні застосунки, пов'язані з агротехнічним моніторингом та обробкою даних.

Розглянемо рішення MiniTab (див. рис. 1.1), що нам пропонує виробник Minitab [1]. У програмах із цього пакета можна візуалізувати, аналізувати, порівнювати дані для реалізації бізнес-завдань. приваблює простотою у використанні та точністю виконуваних операцій.

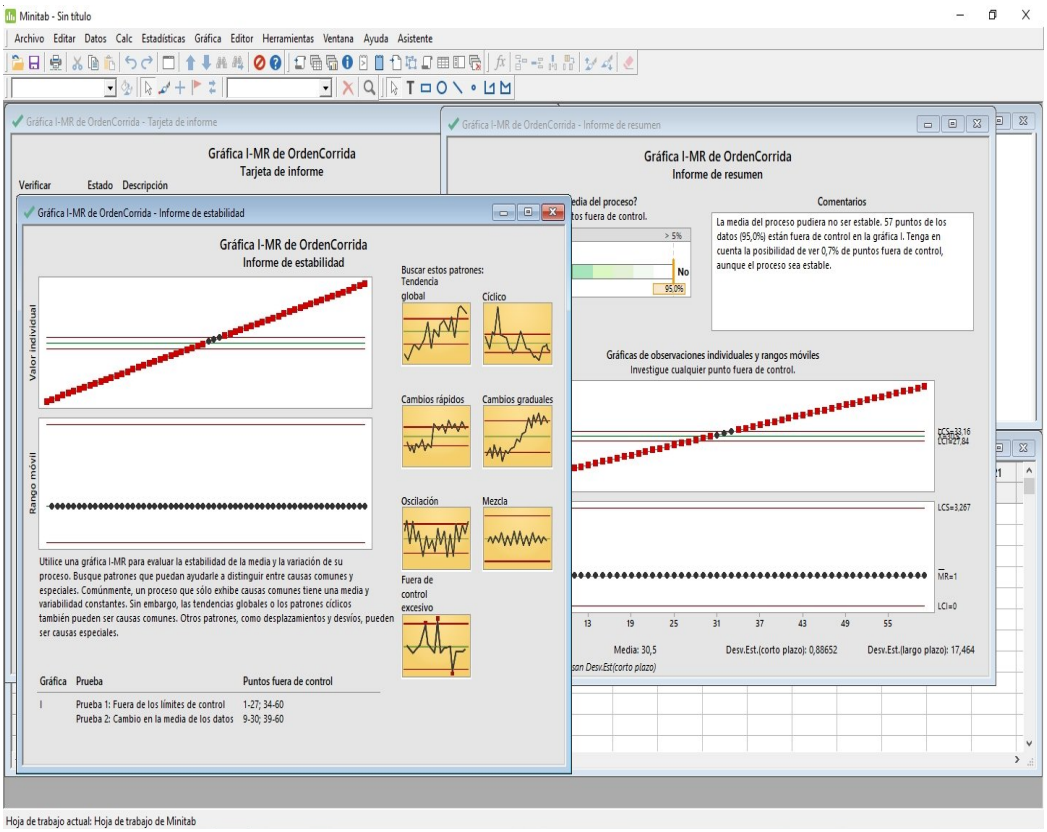

Рис. 1.1. Зовнішній вигляд програми MiniTab

Наступна програма COMSOL Multiphysics (див. рис. 1.2) від виробника COMSOL [2] Який є програмним середовищем, яке дозволяє проводити аналіз фізичних процесів, керувати моделями, застосунками. Працюючи у програмі, потрібно пройти всі етапи від створення геометричних моделей, присвоєння властивостей матеріалам до візуального відображення заключного проекту моделювання.

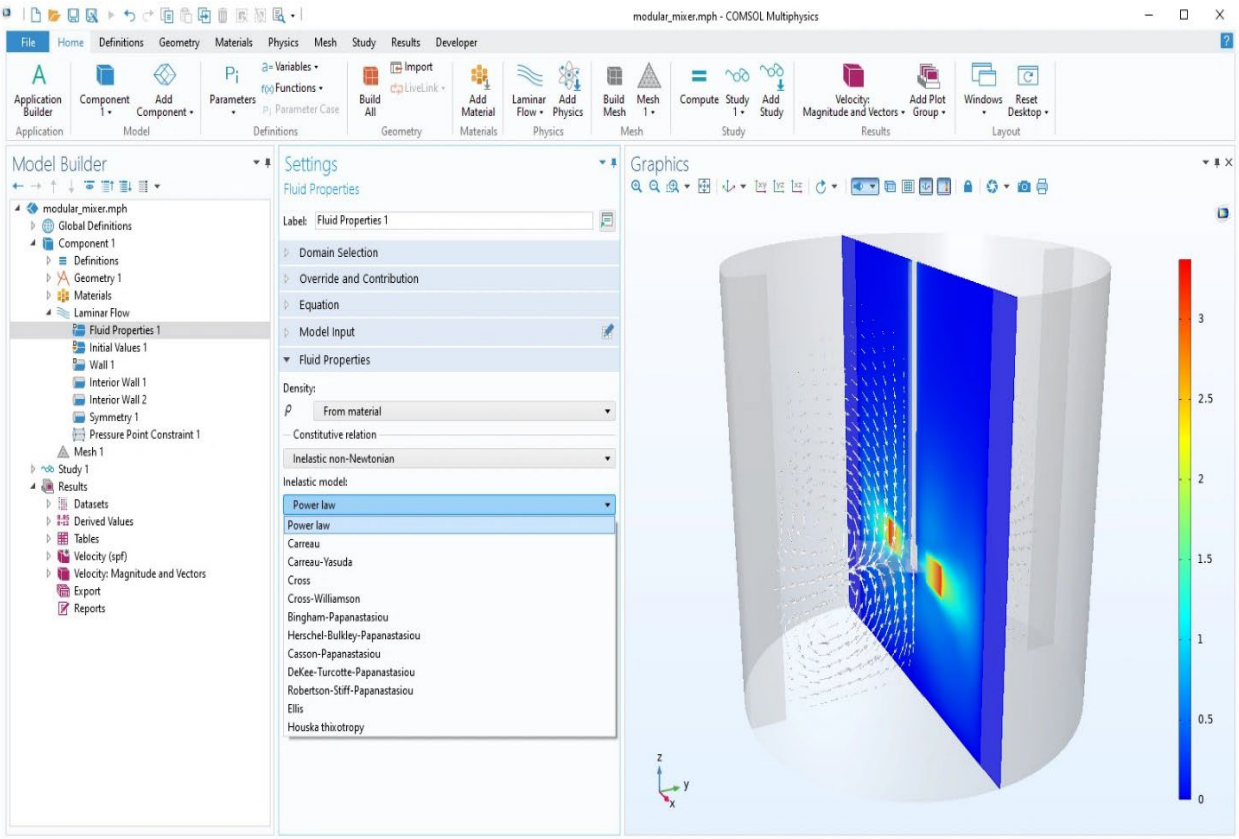

Рис. 1.2. Зовнішній вигляд програми COMSOL Multiphysics

Програма SAS (див. рис. 1.3) від виробника SAS Institute, Inc [3] інтерактивне та командне програмне середовище, що утворене з модулів для аналізу даних, статистики та написання звітів. Також забезпечує підключення до баз даних ORACLE та INGRES, забезпечує аналіз часових рядів та прогнозування, відтворює кольорові графіки, забезпечує матричне програмування та розвинену статистику.

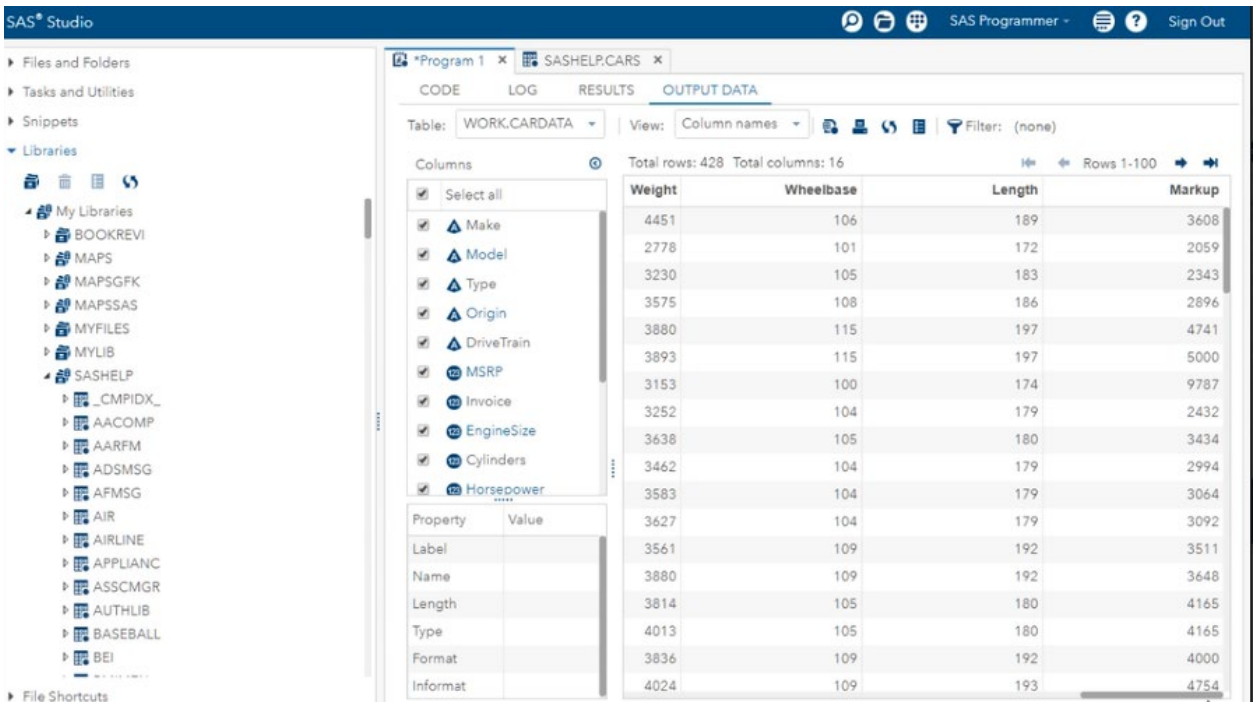

Рис. 1.3. Зовнішній вигляд програми SAS

Програма MS Excel (див. рис. 1.4) від виробника Microsoft [4] це популярний програмний продукт, що входить до складу пакету Microsoft Office. Він використовується для створення, редагування та аналізу електронних таблиць. Excel надає користувачам широкі можливості для організації та обробки даних, що дозволяє ефективно виконувати різноманітні завдання у багатьох сферах діяльності.

Основні особливості Microsoft Excel:

— Електронні таблиці: Excel надає можливість створювати структуровані електронні таблиці, які складаються з рядків і стовпців. Користувач може вводити дані в окремі комірки, форматувати їх, виконувати розрахунки та здійснювати аналіз даних.

— Функції та формули: Excel пропонує велику кількість вбудованих функцій і формул, які дозволяють виконувати різноманітні обчислення та маніпуляції з даними. Користувач може використовувати ці функції для вирішення математичних, статистичних, фінансових та інших завдань.

— Графіки та діаграми: Excel дозволяє візуалізувати дані за допомогою різних типів графіків та діаграм. Користувач може побудувати стовпчикові,

колонкові, кругові графіки та багато інших, що допомагає зрозуміти структуру даних та знаходити закономірності.

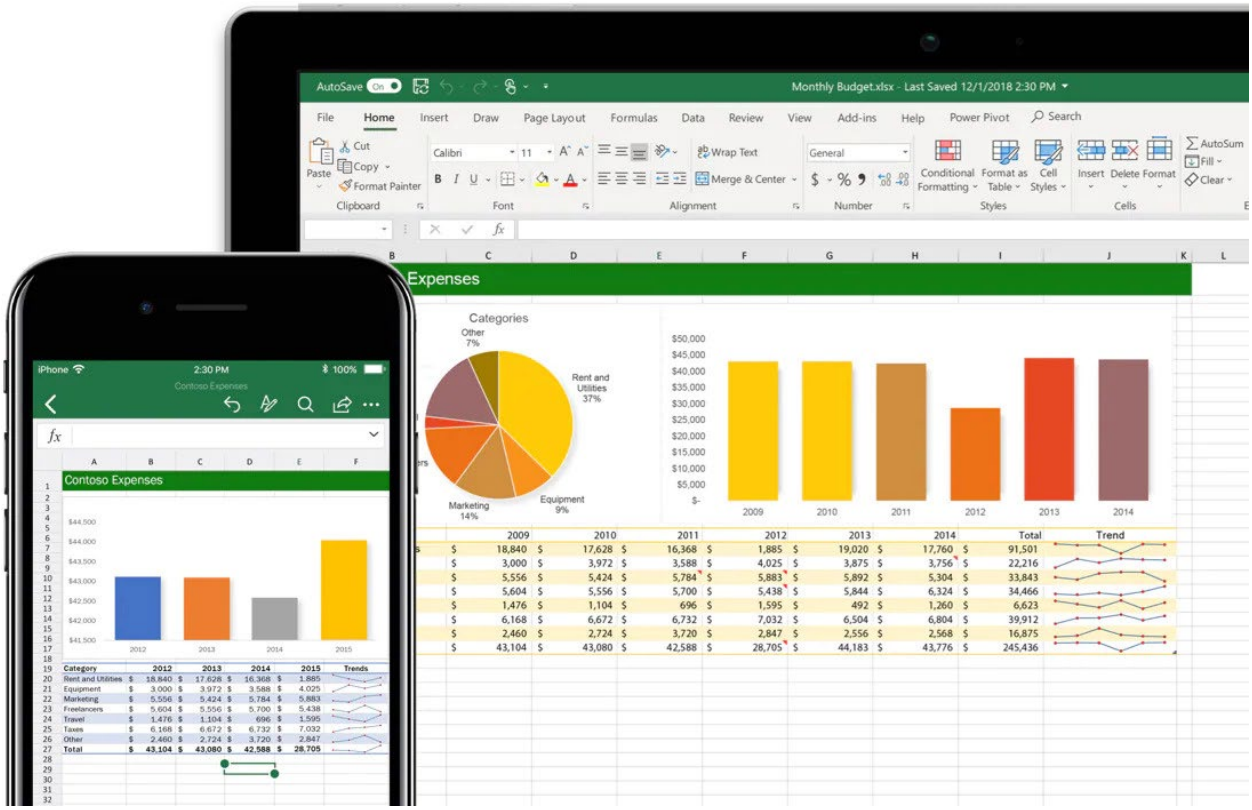

Рис. 1.4. Зовнішній вигляд програми Microsoft Excel

Для систематизації та порівняння отриманих даних, наведемо таблицю, яка відображає недоліки і переваги кожного аналогу.

Таблиця 1.

Переваги та недоліки наявних програмних застосунків

Назва застосунку	**Недоліки**	**Переваги**
MiniTab	Складний та застарілий інтерфейс	завантаження даних із інших програм
	Велика вартість для повної версії	Є пробна версія
	Відсутність підтримки мобільних платформ	Інтеграція зі сторонніми сервісами та пристроями

Продовження табл. 1

COMSOL Multiphysics	Складний інтерфейс	Можливість побудови 3д графіків
	Висока вартість та відсутність пробної версії	Програма має фізичні інтерфейси у готовому вигляді, вони дозволять створювати фізичні явища та матеріали
	Відсутність підтримки мобільних платформ	Велика функціональність
SAS	Відсутність підтримки мобільних платформ	Велика функціональність та гнучкість при роботі з даними.
	Складний інтерфейс та велика кількість команд, що можуть затримувати процес навчання та використання.	Складний інтерфейс та велика кількість команд, що можуть затримувати процес навчання та використання.
	Відсутність підтримки мобільних платформ	Складний інтерфейс та велика кількість команд, що можуть затримувати процес навчання та використання.
Microsoft Excel	Великі обсяги даних можуть сповільнювати роботу програми	Широкі можливості для організації та обробки даних
	Складні формули та функції можуть бути складні для розуміння та використання.	Складні формули та функції можуть бути складні для розуміння та використання.
	Обмежені можливості візуалізації даних та інтерактивного аналізу.	Присутність мобільного застосунка

Основна мета мобільного застосунку - забезпечити користувачам доступ до зібраних даних про стан ґрунту, рослин та агрокліматичних умов у зручному та зрозумілому форматі. Він дозволяє користувачам переглядати статистику, графіки, картографічні дані та інші візуальні представлення зібраних даних.

Мобільний застосунок може включати такі функції, як відображення поточних значень показників агротехнічного моніторингу, порівняння даних з попередніми періодами, а також інтерактивні функції, які дозволяють користувачам взаємодіяти з даними та виконувати аналіз.

При проектуванні мобільного застосунку для трансформації даних агротехнічного моніторингу важливо враховувати потреби та вимоги користувачів. Наприклад, можуть бути встановлені вимоги щодо зручного та інтуїтивно зрозумілого інтерфейсу, підтримки різних мов, можливості експорту даних або синхронізації з іншими пристроями та системами.

1.2 Призначення розробки та галузь застосування

Мобільний застосунок для трансформації даних агротехнічного моніторингу має на меті надати користувачам зручний та ефективний доступ до результатів збору та аналізу даних, пов'язаних з сільськогосподарською діяльністю. Основною метою розробки є покращення процесу прийняття рішень щодо агротехніки, ресурсного управління та оптимізації сільськогосподарських процесів.

Галузь застосування мобільного застосунку трансформації даних агротехнічного моніторингу охоплює різні аспекти сільськогосподарської діяльності, зокрема:

— Фермерські господарства та сільськогосподарські підприємства: Мобільний застосунок надає можливість фермерам та агрономам отримувати швидкий доступ до даних, пов'язаних з ґрунтовими параметрами, кліматичними умовами, ростом рослин та іншими факторами, що впливають на вирощування сільськогосподарських культур. Це допомагає забезпечити належний контроль і управління агротехнологічними процесами, а також вчасно реагувати на зміни в сільськогосподарському середовищі.

— Дослідницькі установи та університети: Мобільний застосунок може бути використаний у наукових дослідженнях та викладанні для збору, аналізу та візуалізації даних агротехнічного моніторингу. Він дозволяє вченим, студентам та дослідникам ефективно аналізувати зібрані дані, відображати їх у зрозумілій формі та проводити дослідження, спрямовані на вдосконалення агротехнологій та розвиток сільського господарства.

— Консультанти з сільського господарства: Мобільний застосунок може бути використаний консультантами, які надають підтримку сільськогосподарським підприємствам та фермерським господарствам. Він дозволяє консультантам швидко отримувати та аналізувати дані, необхідні для надання рекомендацій з питань агротехніки, внесення добрив, орошення та інших аспектів сільськогосподарського виробництва.

— Агротехнічні компанії: Мобільний застосунок може бути використаний компаніями, що надають агротехнологічні послуги та продукти. Він дозволяє забезпечити клієнтам зручний доступ до статистики, графіків та аналітичних даних, пов'язаних з використанням їхніх продуктів та послуг.

Таким чином, мобільний застосунок для трансформації даних агротехнічного моніторингу має широку галузь застосування, включаючи фермерські господарства, дослідницькі установи, університети, консультантів з сільського господарства та агротехнічні компанії. Він надає зручний доступ до важливих даних та допомагає зробити обґрунтовані рішення в галузі сільськогосподарського виробництва.

1.3. Підстава для розробки

В кінці навчання, студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою.

Підставою для розробки кваліфікаційної роботи на тему «Розробка мобільного застосунку трансформації даних агротехнічного моніторингу» є наказ по Національному технічному університету «Дніпровська політехніка» 350-с від 16.05.2023.

1.4. Постановка завдання

Основною метою розробки мобільного застосунку для трансформації даних агротехнічного моніторингу є створення зручного та ефективного

інструменту для візуалізації та доступу до даних, зібраних під час моніторингу сільськогосподарських процесів. З урахуванням цієї мети, постановка завдання перед розробкою мобільного застосунку включає наступні аспекти:

Визначення функціональних характеристик: Необхідно визначити основні функції та можливості мобільного застосунку. Це може включати відображення поточних та історичних даних про агротехнічний моніторинг, створення графіків, діаграм, порівняння даних та інтерактивний аналіз даних тощо.

Встановлення вимог до інтерфейсу: Необхідно визначити вимоги до зручного та інтуїтивно зрозумілого інтерфейсу мобільного застосунку. Це включає визначення елементів управління, організацію інформаційного простору, стиль, дизайн тощо.

Розробка системи збору та обробки даних: Необхідно розробити механізми збору, збереження та обробки даних, що стосуються агротехнічного моніторингу. Це включає визначення формату даних, розробку механізмів синхронізації та оновлення даних, забезпечення безпеки та конфіденціальності даних, а також розробку алгоритмів обробки та аналізу даних для отримання корисних висновків та інформації.

Забезпечення інформаційної безпеки: Важливим аспектом розробки мобільного застосунку є забезпечення безпеки обробки та передачі даних. Необхідно встановити заходи для захисту конфіденційності, цілісності та доступності даних, а також використовувати шифрування та інші техніки безпеки.

Тестування та валідація: Після розробки мобільного застосунку необхідно провести тестування та валідацію його функціональності та продуктивності. Це допоможе переконатися в правильності реалізації функцій, відсутності помилок та забезпечить високу якість застосунку.

Розгортання та підтримка: Після успішного тестування необхідно розгорнути мобільний застосунок на платформах, що підтримуються цільовою аудиторією, і забезпечити підтримку та оновлення застосунку з часом.

Отже, постановка завдання для розробки мобільного застосунку трансформації даних агротехнічного моніторингу включає визначення функціональних характеристик, вимог до інтерфейсу, розробку системи збору та обробки даних, забезпечення інформаційної безпеки, тестування та валідацію, розгортання та підтримку застосунку.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для мобільного застосунку, розробленого для платформи Android[5], встановлюються наступні вимоги до функціональних характеристик:

— Відображення статистики: Застосунок повинен забезпечувати зручне відображення статистичних даних, зібраних під час агротехнічного моніторингу. Це може включати графіки, діаграми, таблиці та інші візуалізації, що допомагають користувачам аналізувати та сприймати дані.

— Налаштування: Застосунок повинен надавати можливість користувачам налаштовувати параметри відображення даних. Наприклад, користувачі можуть змінювати діапазон часу, обирати показники для відображення, налаштовувати вигляд графіків та інші параметри.

— Збереження даних: Застосунок повинен надавати можливість зберігати відображені дані для подальшого використання. Користувачі мають мати можливість зберігати дані у внутрішню пам'ять пристрою або на зовнішнє сховище.

— Зручний і привабливий інтерфейс користувача: Застосунок повинен мати інтуїтивно зрозумілий і привабливий інтерфейс користувача, що забезпечує зручну навігацію та зрозуміле сприйняття відображених даних. Елементи керування повинні бути чіткими та легкими у використанні, а дизайн та кольорова схема повинні бути привабливими для користувача.

— Підтримка версій Android: Застосунок повинен бути сумісним з платформою Android версії 10.0 і вище, щоб забезпечити його оптимальну роботу на цих версіях операційної системи.

1.5.2. Вимоги до інформаційної безпеки

Для забезпечення інформаційної безпеки мобільного застосунку трансформації даних агротехнічного моніторингу, наступні вимоги повинні бути враховані:

— Захист даних: Застосунок повинен забезпечувати надійний захист конфіденційності, цілісності та доступності збережених даних. Це означає, що дані повинні бути зашифровані в пам'яті пристрою, а також під час передачі через мережу. Важливо забезпечити захист від несанкціонованого доступу до даних з боку сторонніх осіб.

— Безпека мережі: Застосунок повинен використовувати безпечні мережеві протоколи та методи зв'язку для забезпечення безпеки передачі даних між застосунком та віддаленим сервером. Це може включати використання HTTPS-протоколу[6] для захищеної передачі даних через Інтернет.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для мобільного застосунку трансформації даних агротехнічного моніторингу встановлюються наступні вимоги до складу та параметрів технічних засобів:

— Операційна система: Застосунок має підтримувати операційну систему Android версії 10.0 і вище. Він повинен бути розроблений з урахуванням можливостей та обмежень цієї версії операційної системи.

— Сервер: Для підтримки функціональності та обробки даних, необхідно мати сервер з підтримкою .NET 7 [7], що включає ASP.NET [8] та інші

необхідні компоненти. Сервер повинен мати 1 ГБ вільної оперативної пам'яті для забезпечення ефективної роботи застосунку.

Ці вимоги до сервера необхідні для забезпечення стабільної та продуктивної роботи мобільного застосунку, забезпечення швидкого доступу до даних та відповідної обробки запитів.

1.5.4. Вимоги до інформаційної та програмної сумісності

Вимоги до інформаційної та програмної сумісності:

— Інформаційна сумісність: Застосунок повинен підтримувати різні формати даних, такі як CSV, JSON, для взаємодії з іншими системами або пристроями.

— Версійна сумісність: Застосунок має бути сумісним з Android версії 10.0 і вище.

— Сумісність зі сторонніми сервісами: Застосунок повинен взаємодіяти зі сторонніми сервісами, такими як бази даних, веб-служби або хмарні сервіси.

— Сумісність з мобільними пристроями: Застосунок має працювати на різних моделях та виробниках мобільних пристроїв з операційною системою Android.

— Сумісність з .NET 7: Серверна частина застосунку вимагає підтримки платформи .NET 7 з необхідними ресурсами.

РОЗДІЛ 2.
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

Функціональне призначення розроблюваного мобільного застосунку полягає у забезпеченні зручного та ефективного збору, обробки та візуалізації даних агротехнічного моніторингу для сільськогосподарських підприємств. Застосунок буде виконувати наступні функції:

— Збір даних: Застосунок дозволить користувачам збирати різноманітні дані агротехнічного моніторингу, такі як кліматичні умови, рівень ґрунту, вміст рослинних поживних речовин тощо. Користувачі матимуть можливість вводити дані вручну або використовувати підключені датчики для автоматичного збору даних.

— Обробка даних: Застосунок забезпечить функціонал для обробки зібраних даних. Це включатиме перевірку та очищення даних, агрегацію та статистичний аналіз, що дозволить користувачам отримувати цінну інформацію про агротехнічний стан їхніх польових культур.

— Візуалізація даних: Розроблений застосунок надасть користувачам можливість візуального представлення зібраних та оброблених даних у зручній формі. Графіки, діаграми та інші візуальні елементи допоможуть користувачам зрозуміти та аналізувати дані ефективніше.

— Управління проектами: Застосунок також надасть можливість користувачам організовувати та управляти агротехнічними проектами. Це включатиме планування завдань, встановлення термінів, відстеження прогресу та спільну роботу над проектами з іншими учасниками.

— Звіти та експорт даних: Користувачі зможуть генерувати звіти на основі зібраних та оброблених даних. Застосунок також надасть можливість експорту даних в різних форматах, таких як CSV або PDF, для подальшого використання або обміну інформацією з іншими системами.

Функціональне призначення розроблюваного мобільного застосунку забезпечить зручний та ефективний процес збору, обробки та візуалізації даних агротехнічного моніторингу для покращення рішень у галузі аграрної діяльності (див. рис 2.1). Далі у розробці будуть розглянуті детальніше аспекти проектування та реалізації даних функцій.

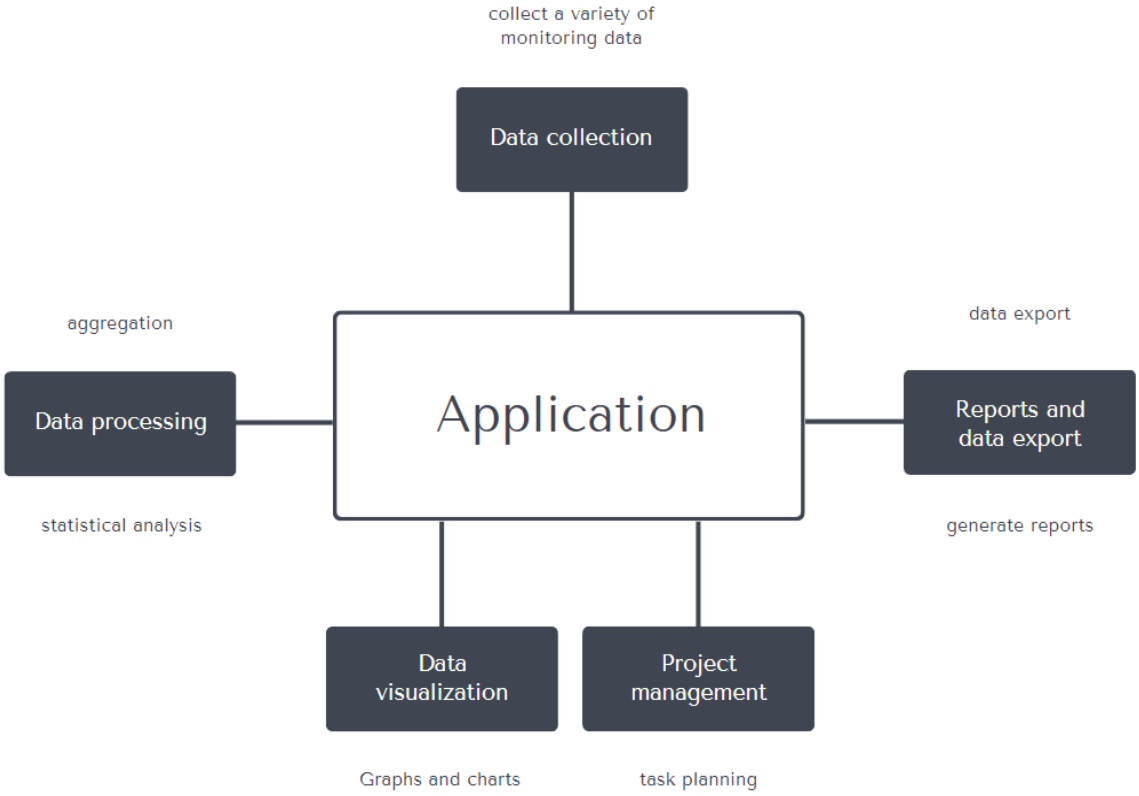

Рис. 2.1. Призначення програми

2.2 Опис застосованих математичних методів

Під час проектування та розробки даної програми використовувалися тільки арифметичні дії, такі як, додавання, віднімання, множення, ділення. Також були використані бібліотеки з алгоритмами, та готовими методами.

2.3 Опис використаної архітектури та шаблонів проектування

1. Під час розробки кваліфікаційної роботи була використана певна архітектура та шаблони проектування, що забезпечують ефективність, модульність та розширюваність системи. Нижче наведено опис використаної архітектури та деяких шаблонів проектування:

2. Model-View-Controller, MVC: Цей шаблон проектування використовується для розділення логіки програми на три основні компоненти: модель, вигляд та контролер. Модель відповідає за обробку даних та бізнес-логіку, вигляд відображає інтерфейс користувача, а контролер обробляє взаємодію між моделлю та виглядом. Це дозволяє досягти розділення відповідальностей та полегшити розробку та тестування.

3. Об'єктно-орієнтована архітектура: Використання об'єктно-орієнтованого підходу дозволяє розбити систему на набір класів, які взаємодіють один з одним через методи та властивості. Це сприяє повторному використанню коду, забезпечує зрозумілість та підтримку масштабованості.

4. Шаблон MVVM (Model-View-ViewModel)[9]: Цей шаблон проектування використовується для розробки інтерфейсу користувача в мобільних застосунках. Він розділяє логіку відображення даних (вигляд) від логіки бізнес-логіки (модель). Він також включає окремий компонент, відомий як ViewModel, який служить посередником між моделлю та виглядом, забезпечуючи зв'язок та взаємодію між ними.

5. Репозиторій (Repository) - це шаблон проєктування, який слугує прошарком між бізнес-логікою застосунку та доступом до даних. Він надає єдиний інтерфейс для роботи з даними та абстрагує бізнес-логіку від конкретної реалізації доступу до даних. Основна мета репозиторію полягає у спрощенні роботи з даними та забезпеченні їх централізованого управління.

6. Контейнер залежностей (Dependency Injection Container): Використання контейнера залежностей дозволяє керувати залежностями між компонентами системи. Це полегшує інтеграцію різних модулів, розширення та тестування. Контейнер залежностей дозволяє автоматично встановлювати залежності між класами та управляти їх життєвим циклом (рис. 2.2)[10].

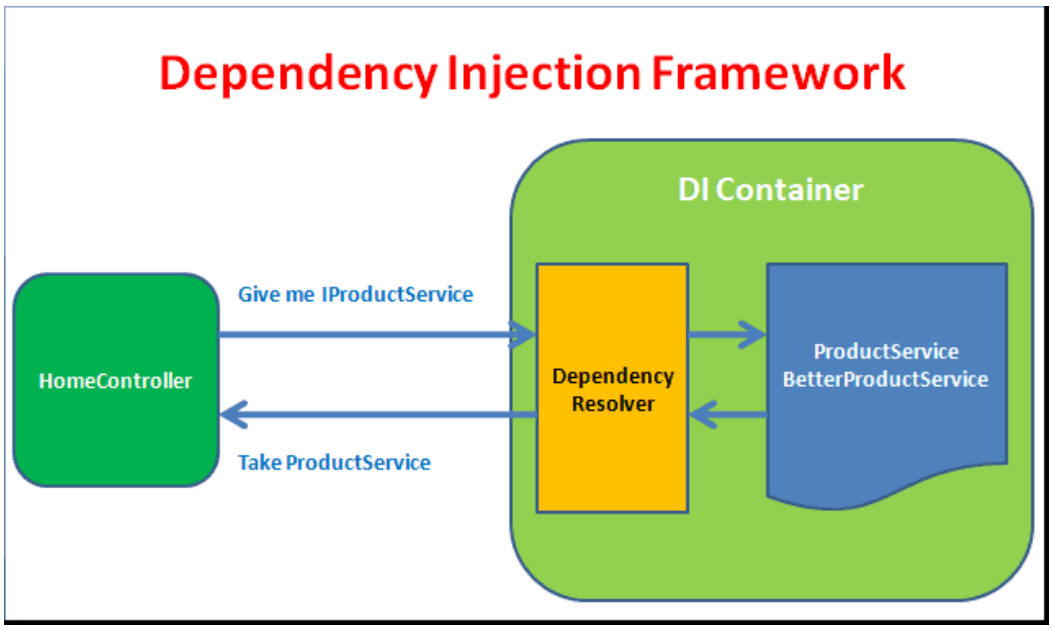

Рис. 2.2. Принцип роботи контейнера залежностей

2.4 Опис використаних технологій та мов програмування

Під час розробки програмного продукту були використані різноманітні технології та мови програмування, що забезпечують ефективну реалізацію функціональності та забезпечують сумісність з обраною архітектурою. Нижче наведено опис деяких з використаних технологій та мов програмування:

1. Xamarin.Forms[11]: Це фреймворк для розробки мобільних застосунків, який дозволяє створювати спільну кодову базу для різних платформ, включаючи Android та iOS. Використання Xamarin.Forms дозволяє ефективно розробляти інтерфейс користувача та логіку застосунка, забезпечуючи швидкий розвиток та підтримку мобільних платформ.

2. C#: Це мова програмування, яка використовується для розробки програмного продукту. C# є мовою з типізацією, орієнтованою на об'єкти та має багатий набір функціональності. Вона забезпечує потужність та гнучкість для розробки різних компонентів програмного продукту.

3. Realm [12]: Для зберігання та управління даними програмного продукту використовується база даних Realm. Він є швидким, мобільним та

легким рішенням для зберігання даних, що забезпечує простоту використання та зручну синхронізацію даних між різними пристроями.

4. Prism: є посібником, розробленим для того, щоб допомогти проєктувати та створювати насичені, гнучкі та легко підтримувані застосунки. Використовуючи патерни проектування, що втілюють важливі принципи архітектурного дизайну, такі як поділ відповідальності та слабка зв'язаність, допомагає проектувати та писати застосунки зі слабко пов'язаними компонентами, що можуть незалежно розвиватися та потім об'єднуватися в одне ціле з мінімальними зусиллями (див. рис. 2.3) [13].

5. JSON: Цей формат обміну даними використовується для структурування та передачі даних між різними компонентами програмного продукту. JSON є простим у використанні та розумінні форматом, який підтримується багатьма мовами програмування.

6. Git[14]: Це розподілена система керування версіями, яка використовується для зберігання та відстежування змін у програмному коді. Вона забезпечує ефективне співробітництво розробників та контроль над версіями програмного продукту.

7. Visual Studio: Це інтегроване середовище розробки (IDE), яке надає зручний інтерфейс для розробки та налагодження програмного продукту. Воно підтримує різні мови програмування, включаючи C#, та надає широкі можливості для розробки мобільних застосунків.

8. ASP.NET Core: Це відкрита, крос-платформена фреймворк для розробки веб-застосунків та веб-служб. ASP.NET Core надає потужні інструменти та функціональність для створення швидких та масштабованих веб-застосунків. Він підтримує використання C# або інших мов програмування для створення серверної логіки, а також надає можливості для роботи з HTTP-запитами, маршрутизації, автентифікації та іншими важливими аспектами веб-розробки.

Детальна інформація про використані технології та мови програмування може бути знайдена в документації, офіційних веб-сайтах та джерелах зі

спеціалізованою літературою, пов'язаною з кожною конкретною технологією та мовою програмування.

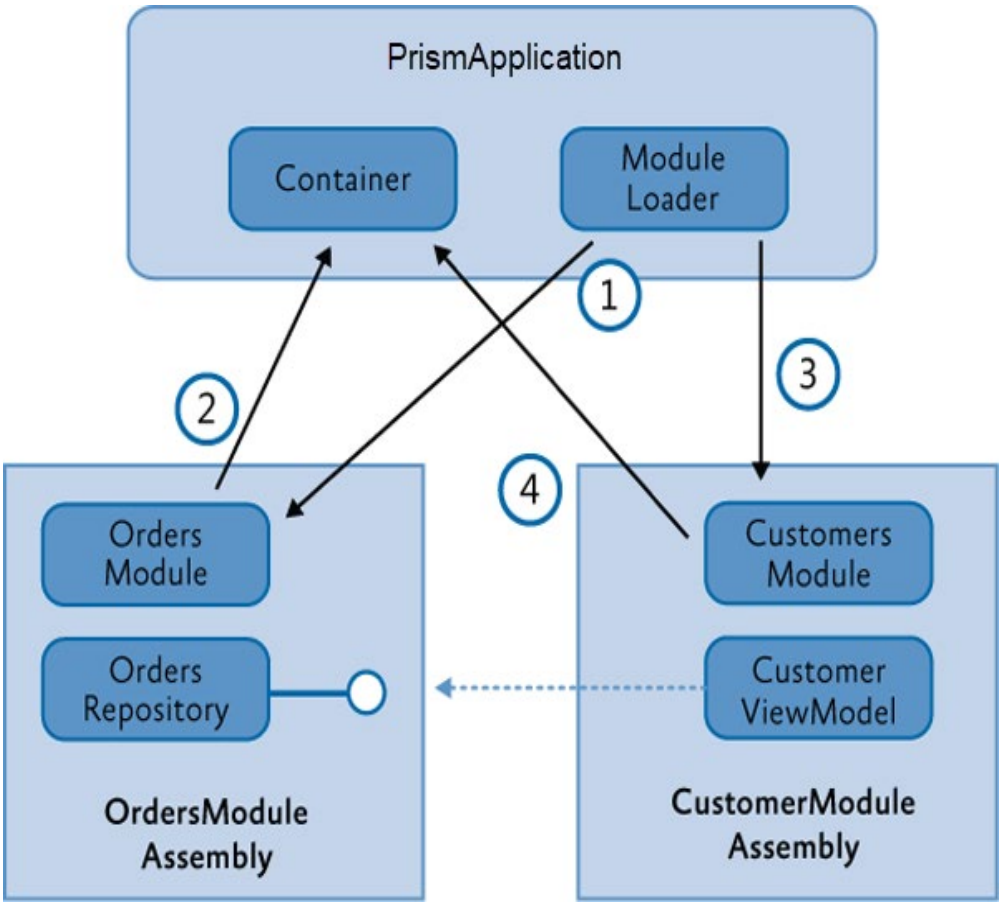

Рис. 2.3. Схема роботи prism застосунка

2.5 Опис структури програми та алгоритмів її функціонування

Структура програми: Програма для розробки мобільного застосунку трансформації даних агротехнічного моніторингу має наступну структуру (див. рис. 2.4):

1. Головний модуль: Цей модуль є точкою входу в програму. Він містить основні функції для ініціалізації та запуску застосунку.

2. Модуль користувацького інтерфейсу: Цей модуль відповідає за взаємодію з користувачем. Він містить різні екрани, форми та елементи керування, що дозволяють користувачеві взаємодіяти з програмою. Для

розробки користувацького інтерфейсу використовуються технології Xamarin Forms та XAML.

3. Модуль обробки даних: Цей модуль відповідає за обробку, збереження та доступ до даних. Використовується база даних для збереження агротехнічних даних, а також використовуються відповідні алгоритми для обробки та трансформації даних.

4. Модуль бізнес-логіки: Цей модуль містить основну логіку програми. Він обробляє дані, виконує розрахунки та виконує потрібні дії відповідно до функціональних вимог програми. Використовуються відповідні алгоритми та алгоритмічні структури даних для забезпечення ефективності та правильності роботи програми.

5. Модуль зв'язку з сервером: Цей модуль відповідає за комунікацію програми з сервером. Він включає в себе реалізацію протоколів зв'язку, обмін даними та інші важливі компоненти, що дозволяють програмі взаємодіяти з серверними ресурсами.

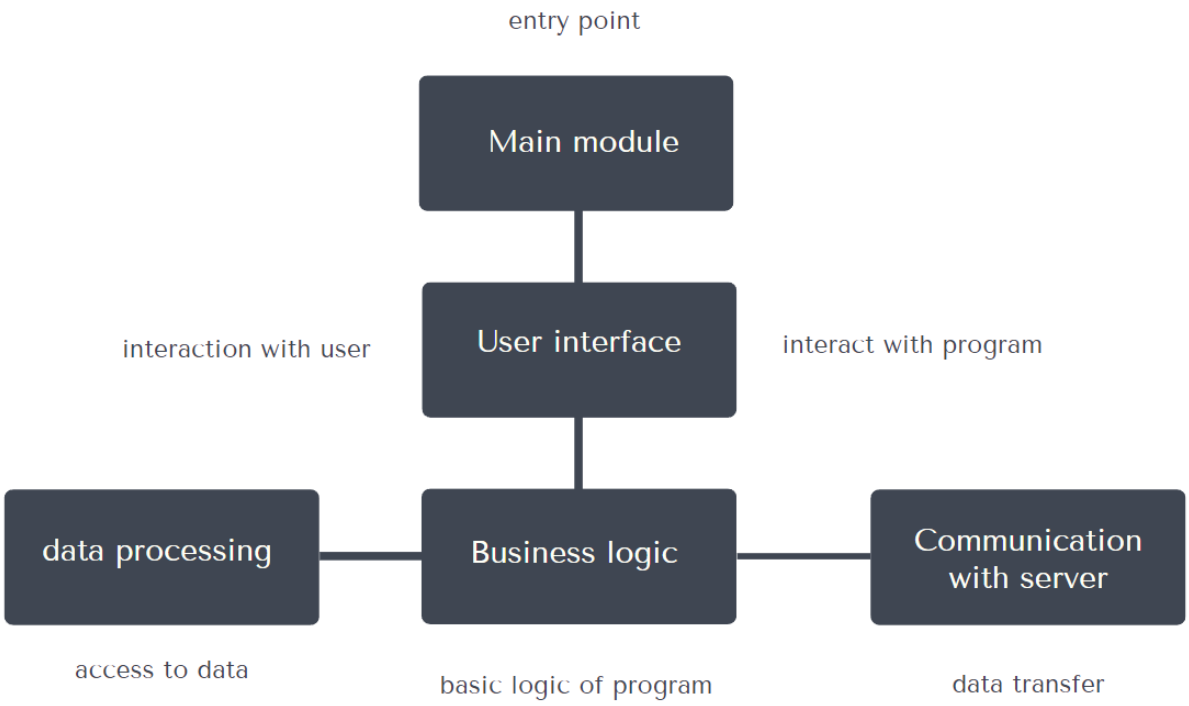

Рис. 2.4. Схема структури застосунку

Алгоритми функціонування: Для забезпечення коректності та ефективності роботи програмного продукту використовуються наступні алгоритми:

1. Алгоритми обробки даних: Використовуються алгоритми для обробки та трансформації агротехнічних даних. Ці алгоритми дозволяють проводити розрахунки, аналізувати дані та генерувати необхідну інформацію для користувача.

2. Алгоритми взаємодії з базою даних: Для збереження та доступу до даних використовується база даних. Відповідні алгоритми дозволяють здійснювати операції додавання, видалення, зміни та отримання даних з бази даних.

3. Алгоритми навігації та керування: Для забезпечення зручної навігації по різних екранах та формах програми використовуються відповідні алгоритми навігації та керування. Вони дозволяють користувачеві зручно переходити між різними частинами програми та взаємодіяти з ними.

4. Алгоритми синхронізації даних: У випадку, коли програма взаємодіє з сервером, використовуються алгоритми синхронізації даних. Вони дозволяють обмінюватись даними між програмою та сервером, забезпечуючи актуальність та цілісність інформації.

Ці алгоритми та структура програми були ретельно розроблені та протестовані, щоб забезпечити правильність, ефективність та стабільність функціонування програмного продукту.

2.5.1 Back-end технології

Для реалізації серверної частини програмного продукту використовуються наступні back-end технології:

— ASP.NET Core: Це відкрита платформа розробки, яка надає засоби для створення високопродуктивних та масштабованих веб-застосунків. ASP.NET

Core забезпечує швидкодію та безпеку програмного продукту, а також дозволяє розробникам ефективно працювати зі змінними потребами користувачів.

— C#: Це об'єктно-орієнтована мова програмування, яка використовується для розробки back-end логіки програмного продукту. C# є потужною та ефективною мовою, що дозволяє розробникам впроваджувати різноманітні функції та алгоритми для оптимальної роботи програми.

— REST API (див. рис. 2.5) [15]: Для забезпечення комунікації між мобільним застосунком та сервером використовується REST API. Цей підхід дозволяє створювати легкі та масштабовані API, які можуть передавати та отримувати дані у стандартному форматі, такому як JSON.

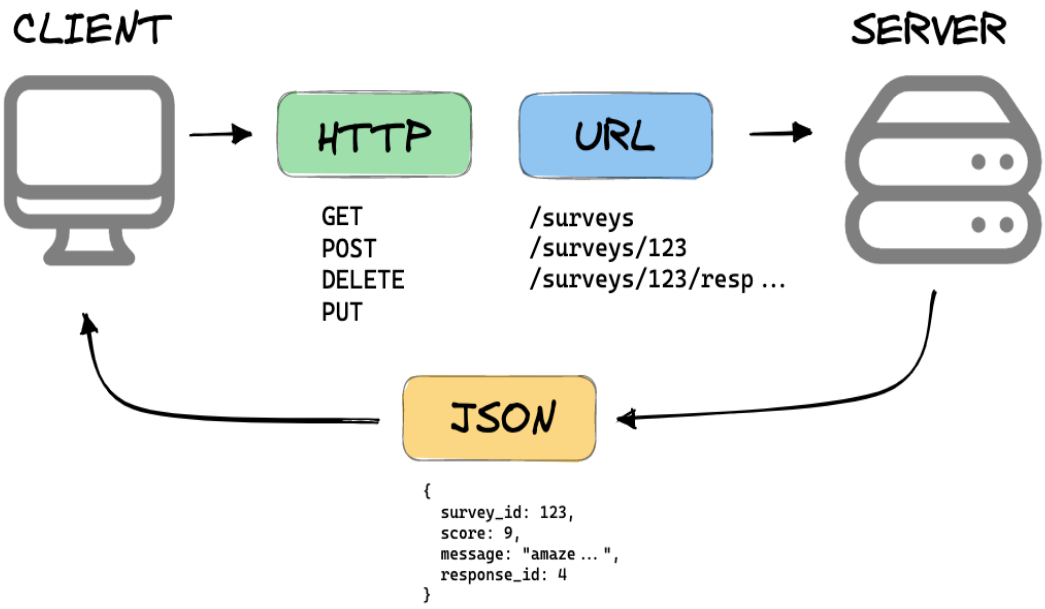

Рис. 2.5. Схема rest api

2.5.2 Front-end технології

Для розробки клієнтської частини програмного продукту використовуються наступні front-end технології:

— Xamarin.Forms: Це фреймворк, що дозволяє створювати мобільні застосунки для різних платформ, таких як Android та Windows, використовуючи

спільний код. Xamarin.Forms забезпечує можливість розробки інтерфейсу користувача в уніфікованому стилі та зручну навігацію між різними екранами.

— C#: Для програмування логіки клієнтської частини використовується мова програмування C#. Це дозволяє розробникам використовувати потужні можливості мови C# для створення функціональності, взаємодії з сервером та обробки даних на мобільному пристрої.

— XAML: Цей мовний стандарт використовується для опису інтерфейсу користувача в Xamarin.Forms. XAML дозволяє розробникам зручно створювати розмітку елементів у декларативному стилі, що спрощує розробку та налаштування вигляду застосунку (див. рис. 2.6) [16].

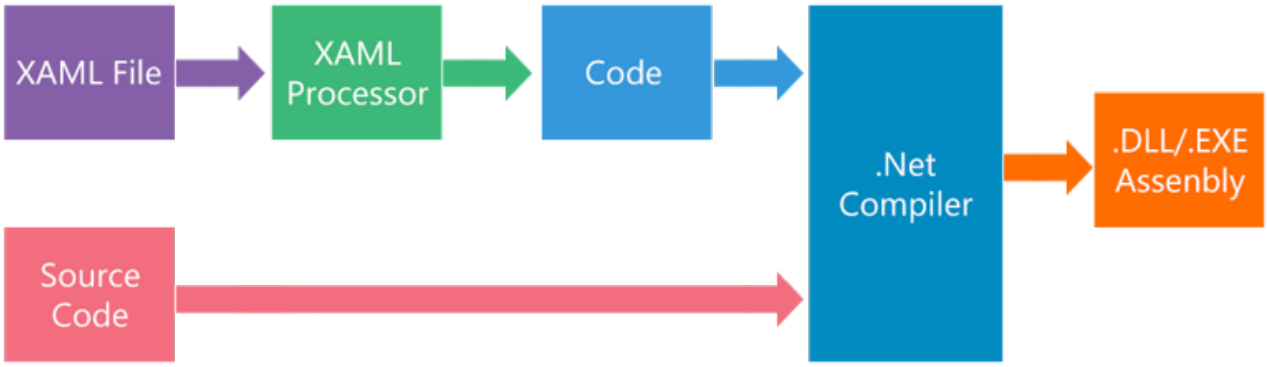

Рис. 2.6. Схема роботи XAML

— MVVM (Model-View-ViewModel): Цей шаблон проектування використовується для розділення логіки програми від її представлення. Використання MVVM дозволяє забезпечити модульність, розширюваність та тестованість клієнтської частини програми (див. рис. 2.7) [17].

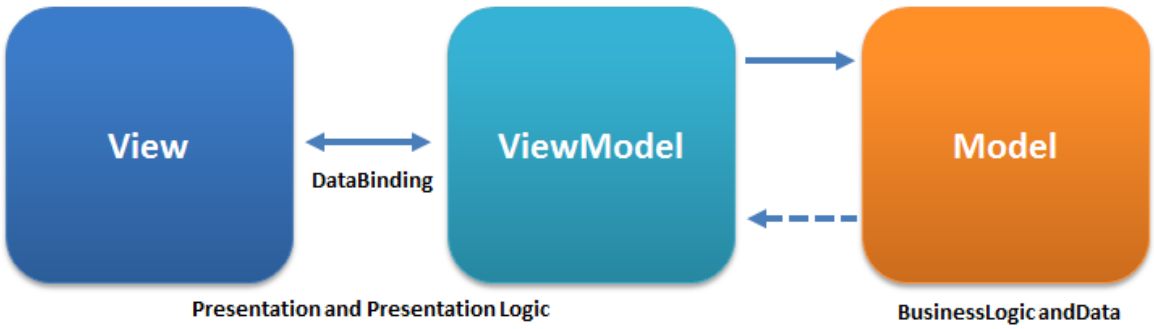

Рис. 2.7. Схема MVVM

2.6 Обґрунтування та організація вхідних та вихідних даних програми

Вхідні дані є ключовою складовою програми трансформації даних агротехнічного моніторингу. Вони надаються користувачами та іншими джерелами і використовуються для створення візуалізації даних та інших функцій програми. Організація та обробка цих вхідних даних відіграють важливу роль у досягненні мети програмного продукту.

— Конфігураційний файл, що підключається при запуску програми (*.json);

— Файл з локальними даними на сервері (*.CSV);

Вихідними даними є повідомлення, що відображаються у вікнах, повідомлення про результат спрацювання команди, а також лог роботи програми, що пишеться в консоль. Деякі вихідні дані можна зберегти у файли відповідних типів., а саме:

— Лог роботи програми (*.log);

— Графічні візуалізації: Результати візуалізації даних у вигляді діаграм, графіків або інших графічних елементів. Це допомагає користувачам легше сприймати та розуміти великі обсяги даних.

Організація вхідних та вихідних даних вимагає уваги до структури даних, їхнього формату та способів взаємодії з користувачем та іншими системами. Це забезпечує ефективну роботу програмного продукту і забезпечує задоволення потреб користувачів у розумінні та аналізі агротехнічних даних.

2.7 Опис роботи програмного продукту

2.7.1 Використані технічні засоби

Для роботи нашого програмного продукту, що розробляється для мобільних пристроїв на платформі, потрібні такі технічні засоби:

— Мобільний пристрій: Для використання програмного продукту необхідний мобільний пристрій, який підтримує операційну систему Android 10.0 версії і більше. Програма буде запускатися і працювати на цих мобільних платформах.

— Сервер: Для розробки програмного продукту необхідний достатніми ресурсами.

Ці технічні засоби дозволяють використовувати програмний продукт на мобільних пристроях з операційною системою Android.

2.7.2 Використані програмні засоби

Під час розроблення мобільного застосунку були використані такі програмні застосунки:

1. Visual Studio: Інтегроване середовище розробки (IDE), розроблене компанією Microsoft. Visual Studio надає розширені можливості для розробки програмного забезпечення, включаючи створення веб-застосунків, написання коду, налагодження та тестування.
2. Git: Розподілена система керування версіями, що використовується для контролю версій програмного коду. Git дозволяє ефективно працювати в команді, відстежувати зміни в коді та здійснювати злиття різних версій.
3. Postman: Інструмент для тестування та документування API. Postman дозволяє надсилати HTTP-запити до веб-сервера та перевіряти відповіді, що дозволяє впевнитись в правильному функціонуванні системи.

Ці програмні засоби були обрані з метою забезпечення зручного та ефективного процесу розробки. Вони надають потужні інструменти для створення, тестування, налагодження та розгортання системи.

2.7.3 Виклик та завантаження програми

Серверний застосунок можна або запустити на локальній машині, або запустити в хмарі Microsoft Azure [18] або Google Cloud, запуск в Microsoft Azure можна зробити засобами Visual Studio [19]. Для цього потрібно перейти у вкладку Publish та вибрати «Publish your application to the Microsoft Coud» і вибрати потрібні ресурси де розгортати застосунок (див. рис. 2.8). Якщо все добре, то побачимо повідомлення про успішне розгортання програми (див. рис. 2.9).

Після успішного встановлення BackEnd, потрібно завантажити apk файл на свій мобільний пристрій

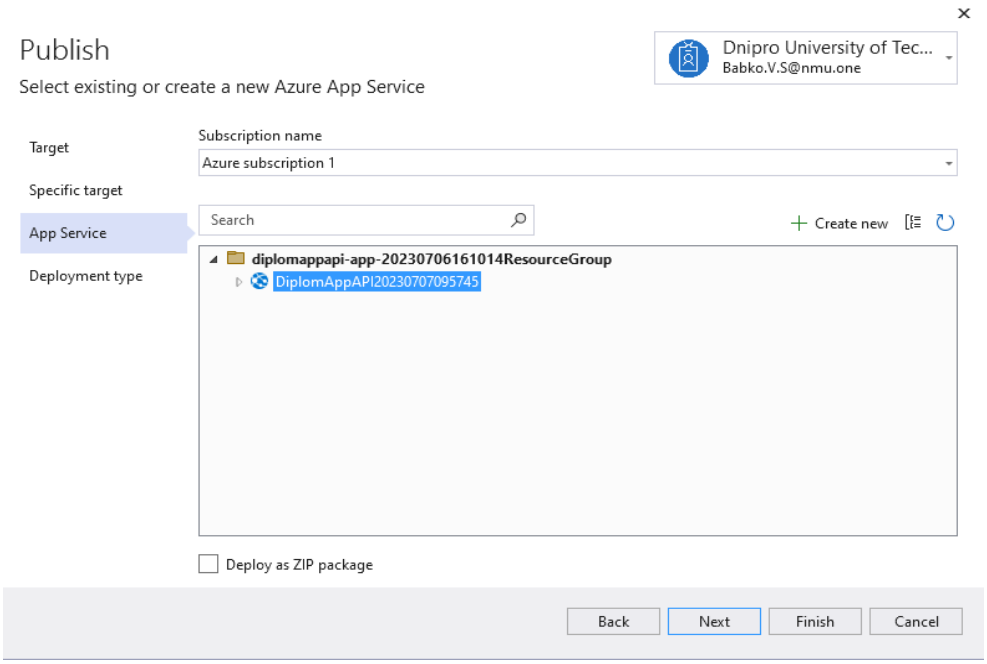

Рис. 2.8. Розгортання застосунку в Microsoft Azure

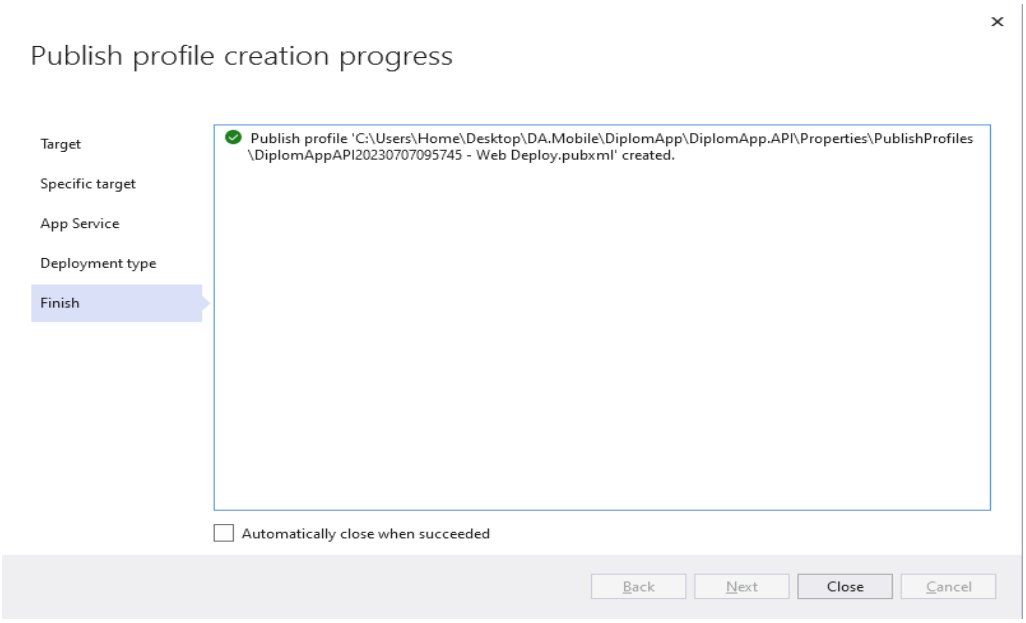

Рис. 2.9. Повідомлення про успішне розгортання застосунку

Після треба перейти з головного меню на сторінку Settings і завантажити нові дані для застосунка (див. рис. 2.10)

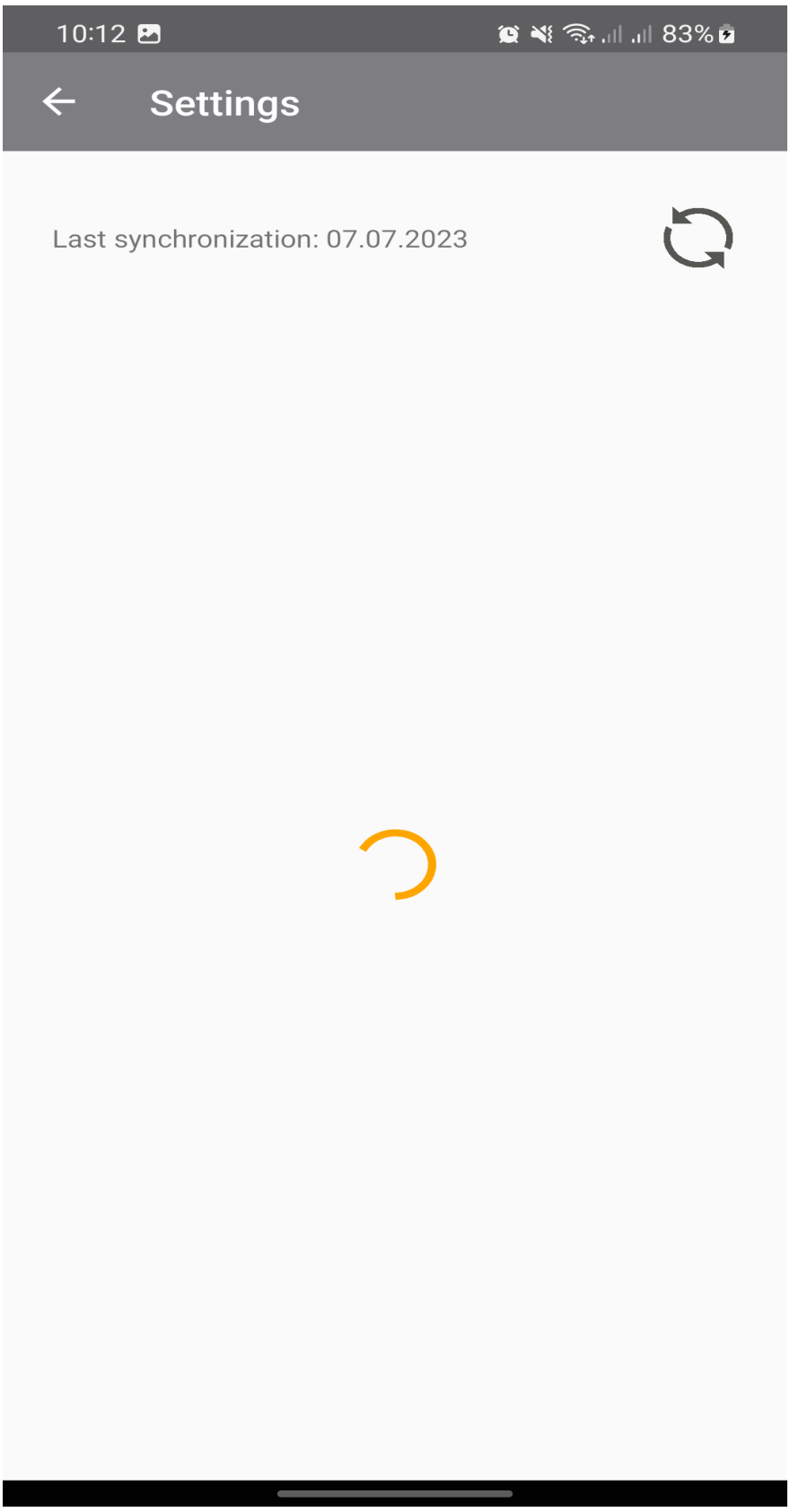

Рис. 2.10. Синхронізація даних для мобільного застосунка

2.7.4 Опис інтерфейсу користув

Після проведення синхронізації на головній сторінці (див. рис. 2.11), буде можливість вибрати метрику, за якою можна подивитися статистику

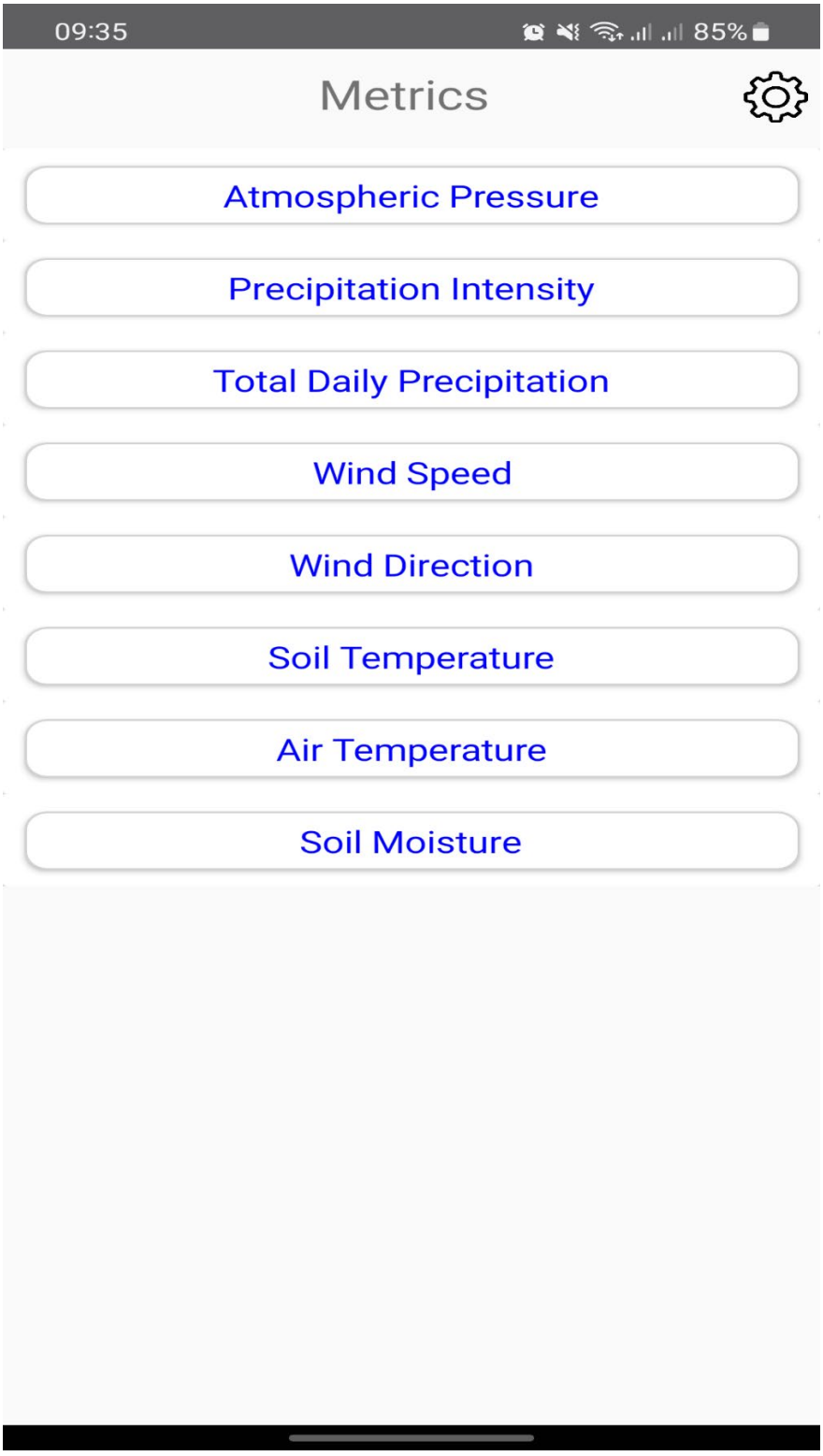

Рис. 2.11. Головне меню мобільного застосунка

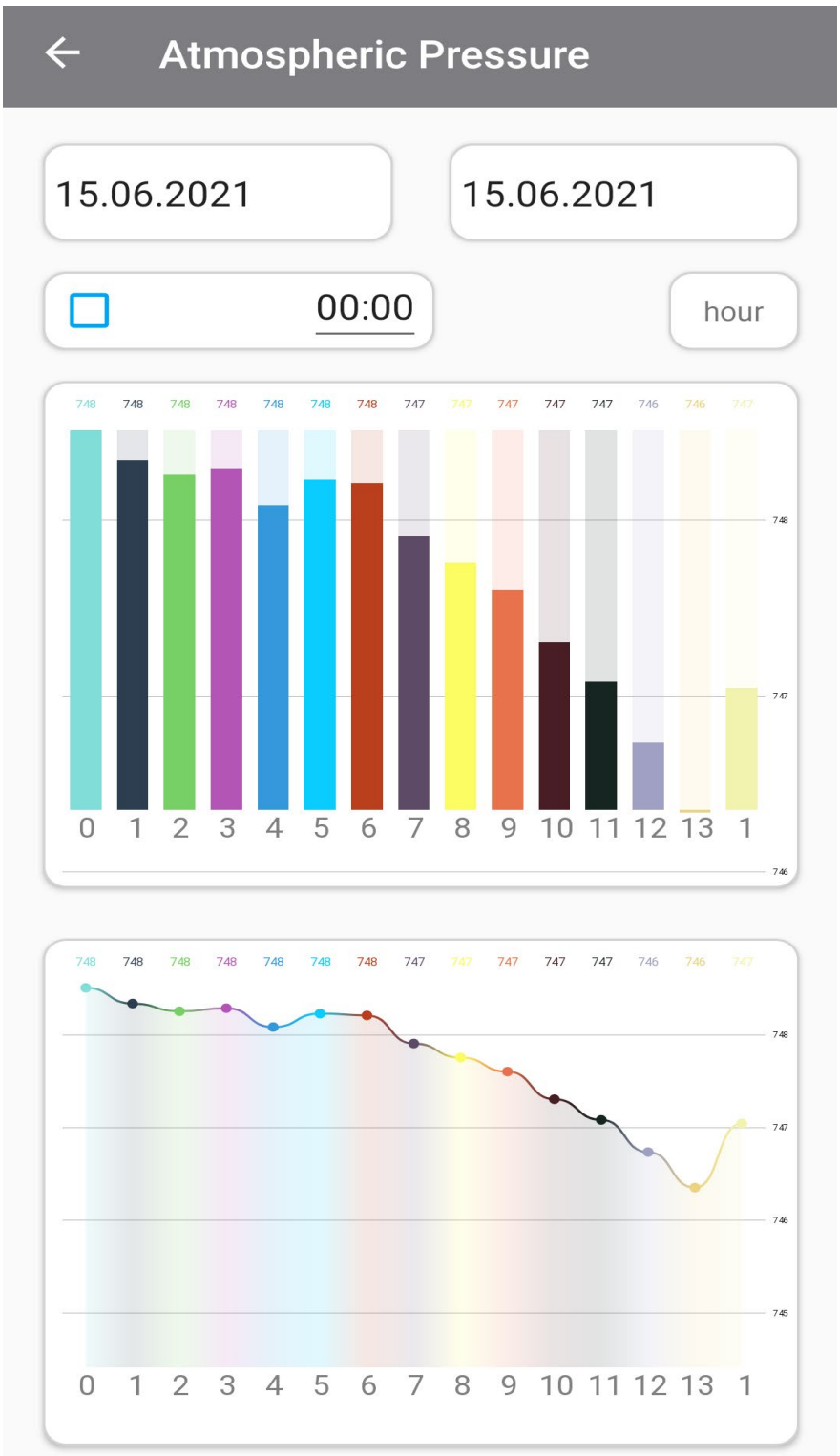

Рис. 2.12. Сторінка відображення статистики

Сторінка статистики (див. рис. 2.12) складається з частин:

1. Заголовка сторінки, за яким можна зрозуміти, які дані використовуються в статистиці.

2. Вибір відрізка, за яким буде відображатися статистика: хвилини\години\дні\тижні\місяці.

3. Діаграми в яких показані зміни в динаміці за певний відрізок.

4. Числові значення даних, які використовуються в діаграмах (див. рис. 2.13).

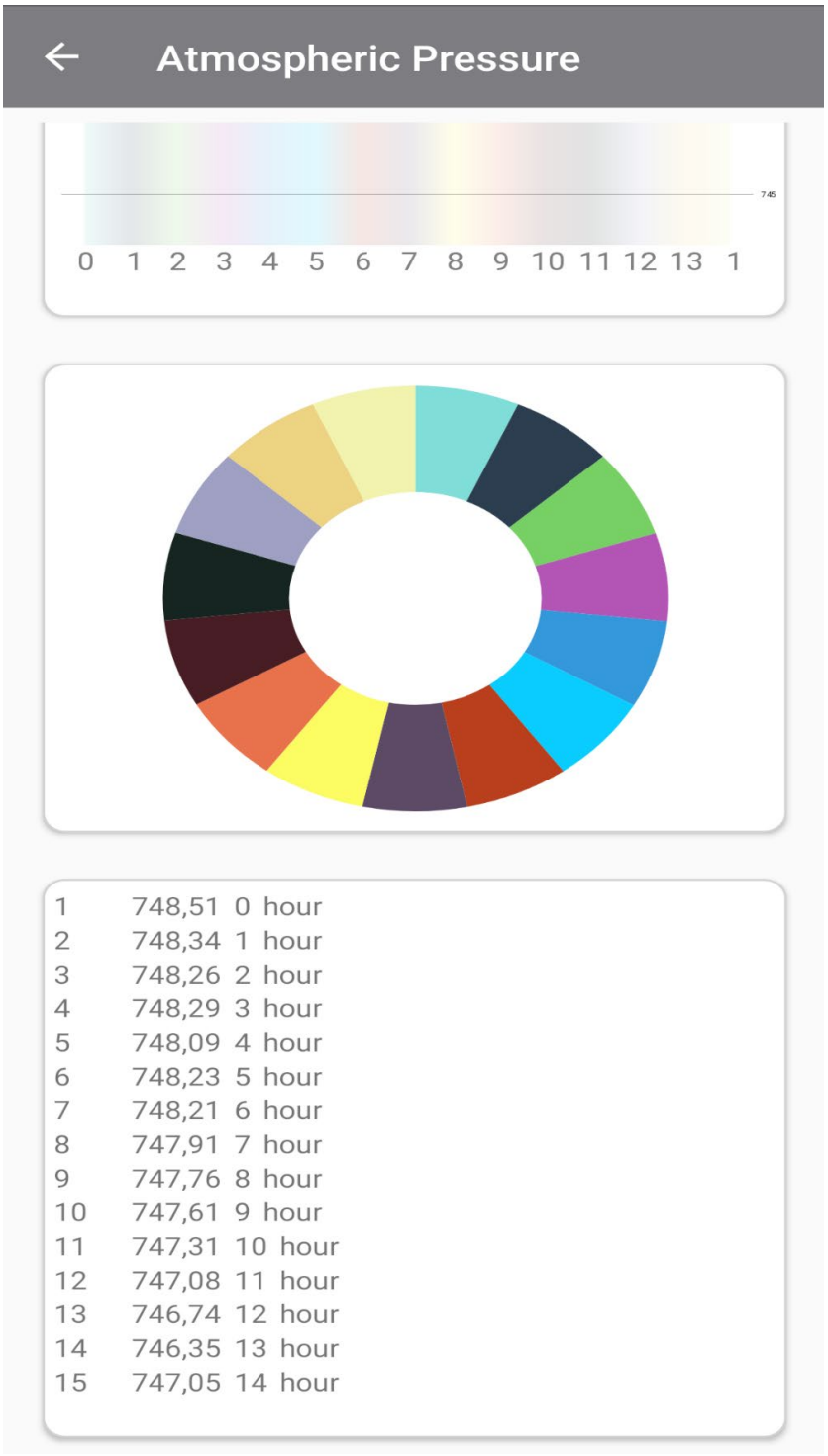

Рис. 2.13. Сторінка відображення статистики

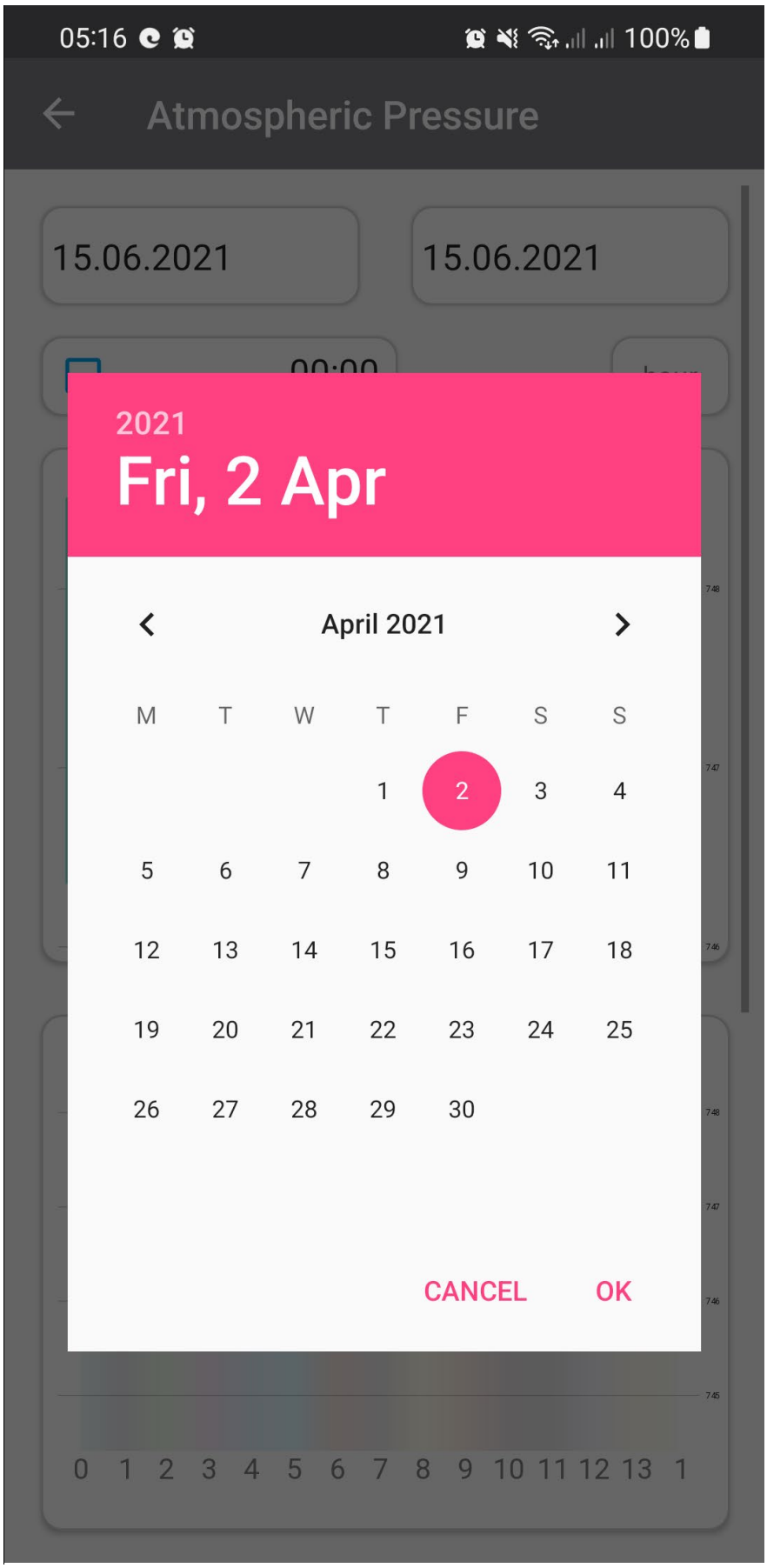

Рис. 2.14. Діалогове вікно вибору діапазону

Під час відображення статистики застосунок враховує кількість завантажених даних в обраному інтервалі (див. рис 2.14), що більший інтервал, то загальнішу статистику показують, а що менший, то конкретнішу (див. рис. 2.15).

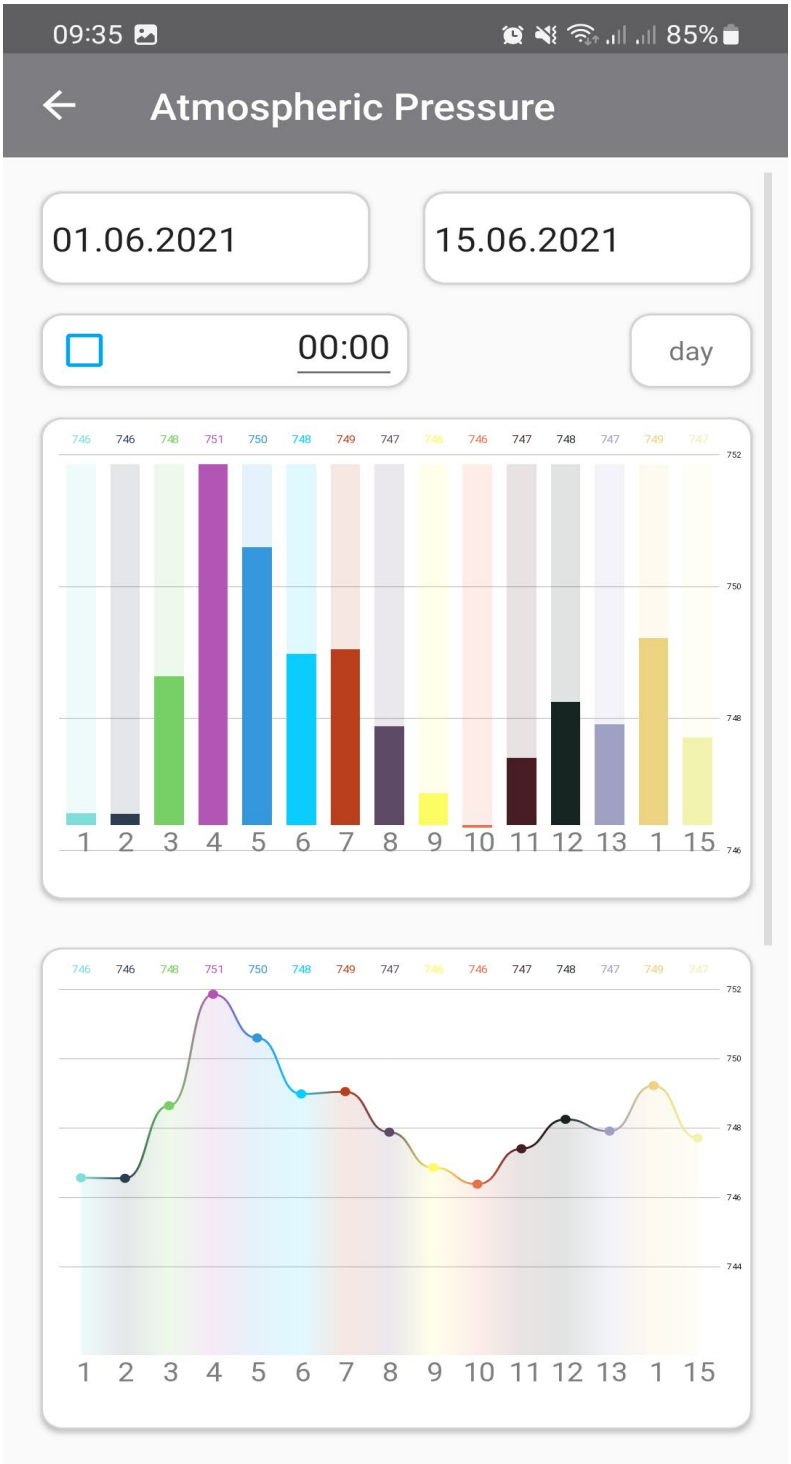

Рис. 2.15. Статистика за днями

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 300;
2. коефіцієнт складності програми – 1,5
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата програміста – 525 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,4;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,5;
7. вартість машино-години ЕОМ – 3.8 грн/год.

Орієнтовна ціна ЕОМ на Microsoft Azure коштує 73$ на місяць. Це буде приблизно (за курсом 1$ = 37грн) 2701 грн на місяць або 3.8 грн за год

За даними Djinni, в середньому, Xamarin Developer отримує зарплату близько 2500 доларів на місяць, що еквівалентно приблизно 92 500 гривень. Враховуючи 8-годинний робочий день та приблизно 22 робочий день на місяць, ставка для таких розробників становить приблизно 525 гривень.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_{\text{и}} + t_{\text{а}} + t_{\text{п}} + t_{\text{отл}} + t_{\text{д}}, \text{ людино-годин} \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 25);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$- праці на налагодження програми на ЕОМ;

$t_д$ - витрати витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p) \tag{3.2}$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 300 \cdot 1{,}5 \cdot (1 + 0{,}2) = 540$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього описузадачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даноїспеціальності;

Будемо вважати збільшення витрат праці внаслідок недостатнього опису завдання не більше 40% (*B* = 1,4). З урахуванням коефіцієнта кваліфікації *k* = 1,5, отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{540 \cdot 1{,}4}{85 \cdot 1{,}5} = 5{,}92 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot k}, \quad (3.4)$$

$$t_a = \frac{540}{25 \cdot 1{,}5} = 14{,}4 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_{\text{п}} = \frac{Q}{(20...25) \cdot k}, \quad (3.5)$$

$$t_{\text{п}} = \frac{540}{25} = 21{,}6 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

− за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5) \cdot k}, \quad (3.6)$$

$$\mathrm{t}_{\text{отл}} = \frac{540}{4 \cdot 1{,}5} = 90 \text{ людино-годин.}$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1{,}5 \cdot t_{\text{отл}} \text{ , людино-годин} \qquad (3.7)$$

$$t^{k}_{отл} = 1{,}5 \cdot 90 = 135 \text{ людино-годин.}$$

– за умови комплексного налагодження завдання:

Витрати праці на підготовку документації:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}} \text{ , людино-годин} \qquad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{\text{др}} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин,} \qquad (3.9)$$

$$\mathrm{t}_{\text{др}} = \frac{540}{15 \cdot 1{,}5} = 24 \text{ людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др} \text{ , людино-годин.} \qquad (3.10)$$

$$\mathrm{t}_{\partial \mathrm{o}} = 0{,}75 \cdot 24 = 18 \text{ людино-годин.}$$

$$t_{\partial} = 24 + 18 = 42 \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$t = 25 + 5{,}92 + 14{,}4 + 21{,}6 + 90 + 42 = 198{,}92 \text{ людино-годин.}$$

У результаті ми розрахували, що в загальній складності необхідно близько 198,92 людино-годин для розробки даного програмного забезпечення.

3.2 Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн} \qquad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн} \qquad (3.12)$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 525 грн / год, отримуємо:

$$З_{ЗП} = 198{,}92 \cdot 525 = 104\,433 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч}\text{, грн,} \qquad (3.13)$$

де: $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (1,18 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 90 \cdot 3.8 = 342 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 104\,433 + 342 = 104\,775 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}\text{, міс.} \qquad (3.14)$$

де: B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p= 180 годин).

Очікуваний період створення ПЗ:

$$T = 198{,}92 / 1 \cdot 180 \approx 1{,}1 \text{ міс.}$$

Висновки. Час розробки даного програмного забезпечення складає 198,92 людино-годин. Таким чином, очікувана тривалість розробки складе 1,1 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 180 годин), а витрати на створення програмного забезпечення складатимуть 104 775 грн.

ВИСНОВКИ

У даній кваліфікаційній роботі був розроблений мобільний застосунок для збору, обробки та візуалізації даних агротехнічного моніторингу в рамках спеціальності "Інженерія програмного забезпечення". Результатом роботи є функціональний застосунок, який надає зручність у роботі та сприяє прийняттю обґрунтованих рішень для підвищення продуктивності та ефективності сільськогосподарської діяльності.

Основна мета розробки застосунку полягала в створенні безпечного та ефективного інструменту для збору, обробки та візуалізації даних агротехнічного моніторингу. Застосування розробленого застосунку дозволяє сільськогосподарським підприємствам отримати зручний інструмент для збору та аналізу даних, що сприяє прийняттю обґрунтованих рішень для підвищення продуктивності та ефективності їх діяльності.

Застосунок був розроблений з використанням фреймворку Xamarin.Forms та мови програмування C#. Він спрямований на платформу Android, що робить його доступним для використання на різних мобільних пристроях.

Аналіз результатів роботи дозволяє зробити висновок про доцільність розробки мобільного застосунку для забезпечення ефективного збору, обробки та візуалізації даних агротехнічного моніторингу. Розроблений застосунок має ряд переваг, таких як зручний інтерфейс користувача, безпека та підтримка платформи Android, що робить його цінним інструментом для підвищення продуктивності та ефективності сільськогосподарської діяльності.

У результаті виконання даної кваліфікаційної роботи був розроблений функціональний мобільний застосунок, який забезпечує безпечний та ефективний спосіб збору, обробки та візуалізації даних агротехнічного моніторингу. Застосунок має можливості для зручної взаємодії з користувачем та аналізу отриманих даних. Використання розробленого застосунку спрощує процес збору та аналізу даних, що дозволяє сільськогосподарським

підприємствам приймати обґрунтовані рішення для підвищення продуктивності та ефективності їх діяльності.

У цілому, розробка мобільного застосунку для агротехнічного моніторингу є актуальною та корисною. Результати роботи можуть бути використані в сільськогосподарських підприємствах для покращення управління та прийняття обґрунтованих рішень, спрямованих на підвищення продуктивності та ефективності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Data Analysis, Statistical & Process Improvement Tools | Minitab. URL: https://www.minitab.com/.
2. COMSOL Multiphysics. Вікіпедія. URL: https://uk.wikipedia.org/wiki/COMSOL_Multiphysics.
3. SAS: Analytics, Artificial Intelligence and Data Management. URL: https://www.sas.com/.
4. Microsoft Excel. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft_Excel.
5. Android – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Android.
6. HTTPS – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/HTTPS
7. Troelsen A., Japikse P. Pro C# 7: With .NET and .NET Core. Apress, 2017. 1372 с
8. Marcano A. Programming Asp. net Core 5 Mvc and Web Api: Examples in C#. Independently Published, 2021.
9. Ortinau D., Snider E. Mastering Xamarin. Forms: App Architecture Techniques for Building Multi-Platform, Native Mobile Apps with Xamarin. Forms 4, 3rd Edition. Packt Publishing, Limited, 2019. 200 с.
10. Dependency Injection in ASP.NET Core. TekTutorialsHub. URL: https://www.tektutorialshub.com/asp-net-core/asp-net-core-dependency-injection/.
11. Versluis G., Thewissen S. Xamarin.Forms Solutions. Berkeley, CA : Apress, 2019. URL: https://doi.org/10.1007/978-1-4842-4134-9.
12. GitHub - realm/realm-dotnet: Realm is a mobile database: a replacement for SQLite & ORMs. GitHub. URL: https://github.com/realm/realm-dotnet.
13. Modular Application Development Using Prism Library | Prism. Prism Library. URL: https://prismlibrary.com/docs/modules.html.

14. Git – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Git.

15. REST APIs Explained. Howie Mann - Startup Strategy. URL: https://mannhowie.com/rest-api.

16. Chauhan S. XAML for Xamarin.Forms. Intensive Project Based Training | DotNetTricks. URL: https://www.dotnettricks.com/learn/xamarin/xaml.

17. Model-View-ViewModel. Википедия – свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Model-View-ViewModel.

18. Microsoft Azure. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft_Azure.

19. Publish an Azure cloud service - Visual Studio (Windows). Microsoft Learn. URL: https://learn.microsoft.com/en-us/visualstudio/azure/vs-azure-tools-publish-azure-application-wizard?view=vs-2022.

Додаток А

ЛІСТИНГ ПРОГРАМИ

Збірка DiplomApp.API

Файл Program.cs

```
using DiplomApp.API.Services;

var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.MapGet("/api", () => new MeteorologicalDataService().GetAllData());

app.Run();
```

Файл MeteorologicalDataService.cs

```
using DiplomApp.API.Models;
using System.Globalization;

namespace DiplomApp.API.Services;

public class MeteorologicalDataService
{
    public List<MeteorologicalData> GetAllData()
    {
        CultureInfo ci = (CultureInfo)CultureInfo.CurrentCulture.Clone();
        ci.NumberFormat.CurrencyDecimalSeparator = ".";

        var weatherDataList = new List<MeteorologicalData>();
        using (var reader = new StreamReader("Resources/feeds.csv"))
        {
            reader.ReadLine(); // skip header line
            string line;
            while ((line = reader.ReadLine()) != null)
            {
                var values = line.Split(';');
                var weatherData = new MeteorologicalData
                {
                    CreatedAt = DateTime.Parse(values[0]),
                    EntryId = int.Parse(values[1]),
                    AtmosphericPressure = double.Parse(values[2], NumberStyles.Any,
ci),
                    PrecipitationIntensity = double.Parse(values[3], NumberStyles.Any,
ci),
                    TotalDailyPrecipitation = double.Parse(values[4], NumberStyles.Any,
ci),
                    WindSpeed = double.Parse(values[5], NumberStyles.Any, ci),
                    WindDirection = double.Parse(values[6], NumberStyles.Any, ci),
                    SoilTemperature = double.Parse(values[7], NumberStyles.Any, ci),
                    AirTemperature = double.Parse(values[8], NumberStyles.Any, ci),
                    SoilMoisture = double.Parse(values[9], NumberStyles.Any, ci)
                };
                weatherDataList.Add(weatherData);
            }
        }
        return weatherDataList.ToList();
    }
}
```

Збірка DiplomApp
Файл App.xaml
```
<?xml version="1.0" encoding="utf-8" ?>
<prism:PrismApplication
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:prism="clr-namespace:Prism.Unity;assembly=Prism.Unity.Forms"
    x:Class="DiplomApp.App">
</prism:PrismApplication>
```

Файл App.xaml.cs
```
using DiplomApp.Services;
using DiplomApp.ViewModels;
using DiplomApp.Views;
using Prism;
using Prism.Ioc;
using Xamarin.Essentials.Implementation;
using Xamarin.Essentials.Interfaces;
using Xamarin.Forms;

namespace DiplomApp
{
    public partial class App
    {
        public App(IPlatformInitializer initializer)
            : base(initializer)
        {
        }

        protected override async void OnInitialized()
        {
            InitializeComponent();

            await NavigationService.NavigateAsync("NavigationPage/MainPage");
        }

        protected override void RegisterTypes(IContainerRegistry containerRegistry)
        {
            containerRegistry.RegisterSingleton<IAppInfo, AppInfoImplementation>();

            containerRegistry.RegisterForNavigation<NavigationPage>();
            containerRegistry.RegisterForNavigation<MainPage, MainPageViewModel>();
            containerRegistry.RegisterForNavigation<StatisticsDetailsPage,
StatisticsDetailsPageViewModel>();
            containerRegistry.RegisterForNavigation<SettingsPage,
SettingsPageViewModel>();

            containerRegistry.RegisterSingleton<IMeteorologicalService,
MeteorilogicalService>();
            containerRegistry.RegisterSingleton<IRealmService, RealmService>();
        }
    }
}
```

Файл MainPage.xaml
```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    x:Class="DiplomApp.Views.MainPage"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
```

```
    Title="{Binding Title}">

    <StackLayout Spacing="10">
        <Grid BackgroundColor="Transparent" ColumnDefinitions="*, 40">
            <Label
                Margin="40,10,0,10"
                FontSize="24"
                HorizontalOptions="Center"
                Text="Metrics" />
            <ImageButton
                Grid.Column="1"
                Margin="4,8,6,0"
                BackgroundColor="Transparent"
                Command="{Binding NavigateToSettingsCommand}"
                Source="Settings" />
        </Grid>

        <CollectionView ItemsSource="{Binding StatisticParameters}">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Frame Padding="10">
                        <Frame
                            Padding="5"
                            BackgroundColor="White"
                            BorderColor="LightGray"
                            CornerRadius="10">
                            <Label
                                FontSize="18"
                                HorizontalOptions="Center"
                                Text="{Binding Name}"
                                TextColor="Blue" />
                            <Frame.GestureRecognizers>
                                <TapGestureRecognizer CommandParameter="{Binding Name}"
Tapped="TapGestureRecognizer_Tapped" />
                            </Frame.GestureRecognizers>
                        </Frame>
                    </Frame>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </StackLayout>
</ContentPage>
```

Файл MainPage.xaml.cs

```
using DiplomApp.ViewModels;
using Xamarin.Forms;

namespace DiplomApp.Views
{
    public partial class MainPage
    {
        public MainPage()
        {
            NavigationPage.SetHasNavigationBar(this, false);

            InitializeComponent();
        }

        private void TapGestureRecognizer_Tapped(object sender, System.EventArgs e)
        {
```

```
            if ((BindingContext is MainPageViewModel context) && (e is TappedEventArgs
args))
            {
                context.NavigateToDataPageCommand.Execute(args.Parameter);
            }
        }
    }
}
```

Файл SettingsPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        Title="{Binding Title}"
    Padding="8"
    x:Class="DiplomApp.Views.SettingsPage">
    <ContentPage.Content>
        <Grid
            Margin="16"
            ColumnDefinitions="250, *, 40"
            RowDefinitions="50, *">
            <Label Text="{Binding LastDataString}" VerticalTextAlignment="Center" />
            <ImageButton
                Grid.Column="2"
                Source="refresh"
                BackgroundColor="Transparent"
                Command="{Binding LoadDataCommand}" />

            <ActivityIndicator
                Color="Orange"
                Grid.Row="1"
                IsRunning="{Binding IsRunning}"
                Grid.ColumnSpan="3"
                VerticalOptions="CenterAndExpand"
                HorizontalOptions="Center" />
        </Grid>
    </ContentPage.Content>
</ContentPage>
```

Файл SettingsPage.xaml.cs

```
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace DiplomApp.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SettingsPage : ContentPage
    {
        public SettingsPage()
        {
            InitializeComponent();
        }
    }
}
```

Файл StatisticsDetailsPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    x:Class="DiplomApp.Views.StatisticsDetailsPage"
    Title="{Binding Title}"
    Padding="8"
```

```
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:custom="clr-namespace:DiplomApp.Custom"
    xmlns:microcharts="clr-namespace:Microcharts.Forms;assembly=Microcharts.Forms"
    xmlns:mvvm="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"
    xmlns:local="clr-namespace:DiplomApp.Converters"
    mvvm:ViewModelLocator.AutowireViewModel="True">
    <ContentPage.Resources>
        <ResourceDictionary>
            <local:StatisticsDetailsModelsToStringConverter x:Key="StatisticsConverter"
/>
            <local:CollectionViewHeightConverter x:Key="collectionViewHeightConverter"
/>
            <local:DoubleToStringConverter x:Key="doubleToStringConverter" />
        </ResourceDictionary>
    </ContentPage.Resources>
    <ScrollView>

        <StackLayout>
            <Grid RowDefinitions="70">
                <Frame
                    Margin="10"
                    Padding="5"
                    BackgroundColor="White"
                    BorderColor="LightGray"
                    CornerRadius="10">
                    <custom:BorderlessDatePicker
                        MinimumDate="{Binding StartBorderDate.DateTime}"
                        MaximumDate="{Binding EndBorderDate.DateTime}"
                        Date="{Binding StartDate}" />
                </Frame>

                <Frame
                    Grid.Column="1"
                    Margin="10"
                    Padding="5"
                    BackgroundColor="White"
                    BorderColor="LightGray"
                    CornerRadius="10">

                    <custom:BorderlessDatePicker
                        MinimumDate="{Binding StartDate}"
                        MaximumDate="{Binding EndBorderDate.DateTime}"
                        Date="{Binding EndDate}" />
                </Frame>

            </Grid>

            <Grid
                Margin="0"
                Padding="0"
                RowDefinitions="40, *"
                ColumnDefinitions="200, *, 80">

                <Frame
                    Margin="10,0,10,0"
                    Padding="5,0,5,0"
                    BackgroundColor="White"
                    BorderColor="LightGray"
                    CornerRadius="10">

                    <Grid>
```

```
                    <CheckBox IsChecked="{Binding IsTimeEnabled}" />

                    <TimePicker
                        Format="HH:mm"
                        HorizontalOptions="End"
                        Time="{Binding Time}"
                        Grid.Column="1" />
                </Grid>
            </Frame>


            <Frame
                Grid.Column="2"
                Margin="10,0,10,0"
                Padding="5"
                BackgroundColor="White"
                BorderColor="LightGray"
                CornerRadius="10">

                <Label
                    Text="{Binding Data, Converter={StaticResource
StatisticsConverter}}"
                    HorizontalOptions="CenterAndExpand"
                    VerticalOptions="CenterAndExpand" />

            </Frame>

            <Frame
                Margin="10"
                Padding="5"
                BackgroundColor="White"
                BorderColor="LightGray"
                Grid.ColumnSpan="3"
                Grid.Row="1"
                CornerRadius="10">
                <microcharts:ChartView
                    Grid.Row="1"
                    Grid.ColumnSpan="2"
                    BackgroundColor="White"
                    Chart="{Binding BarChart}"
                    EnableTouchEvents="True"
                    HeightRequest="250" />
            </Frame>
        </Grid>

        <Grid Margin="0" Padding="0">


            <Frame
                Margin="10"
                Padding="5"
                BackgroundColor="White"
                BorderColor="LightGray"
                CornerRadius="10">
                <microcharts:ChartView
                    BackgroundColor="White"
                    Chart="{Binding LineChart}"
                    HeightRequest="250" />
            </Frame>
        </Grid>

        <Grid Margin="0" Padding="0">
```

```
                <Frame
                    Margin="10"
                    Padding="5"
                    BackgroundColor="White"
                    BorderColor="LightGray"
                    CornerRadius="10">
                    <microcharts:ChartView
                        BackgroundColor="White"
                        Chart="{Binding DonutChart}"
                        HeightRequest="250" />
                </Frame>
            </Grid>

            <Frame
                Margin="10"
                Padding="5"
                BackgroundColor="White"
                BorderColor="LightGray"
                Grid.ColumnSpan="3"
                Grid.Row="1"
                CornerRadius="10">
                <CollectionView ItemsSource="{Binding Data}" HeightRequest="{Binding
Data, Converter={StaticResource collectionViewHeightConverter}}">
                    <CollectionView.ItemTemplate>
                        <DataTemplate>
                            <Grid ColumnDefinitions="30, auto, auto, auto">
                                <Label Text="{Binding Index}" />
                                <Label Text="{Binding Value, Converter={StaticResource
doubleToStringConverter}}" Grid.Column="1" />
                                <Label Text="{Binding DateString}" Grid.Column="2" />
                                <Label Text="{Binding CompressionDetails}"
Grid.Column="3" />
                            </Grid>
                        </DataTemplate>
                    </CollectionView.ItemTemplate>
                </CollectionView>
            </Frame>
        </StackLayout>
    </ScrollView>
</ContentPage>
```

Файл StatisticsDetailsPage.xaml.cs

```
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace DiplomApp.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class StatisticsDetailsPage : ContentPage
    {
        public StatisticsDetailsPage()
        {
            InitializeComponent();
        }
    }
}
```

Файл ViewModelBase.cs

```
using Prism.Mvvm;
using Prism.Navigation;
```

```
namespace DiplomApp.ViewModels
{
    public class ViewModelBase : BindableBase, IInitialize, INavigationAware,
IDestructible
    {
        protected INavigationService NavigationService { get; private set; }

        private string _title;
        public string Title
        {
            get { return _title; }
            set { SetProperty(ref _title, value); }
        }

        public ViewModelBase(INavigationService navigationService)
        {
            NavigationService = navigationService;
        }

        public virtual void Initialize(INavigationParameters parameters)
        {
        }

        public virtual void OnNavigatedFrom(INavigationParameters parameters)
        {
        }

        public virtual void OnNavigatedTo(INavigationParameters parameters)
        {
        }

        public virtual void Destroy()
        {
        }
    }
}
```

Файл MainPageViewModel.cs

```
using DiplomApp.Constants;
using DiplomApp.Model;
using DiplomApp.Views;
using Prism.Navigation;
using System.Collections.Generic;
using System.Windows.Input;
using Xamarin.Forms;

namespace DiplomApp.ViewModels
{
    public class MainPageViewModel : ViewModelBase
    {
        private ICommand _navigateToDataPageCommand;
        public ICommand NavigateToDataPageCommand => _navigateToDataPageCommand ??
(_navigateToDataPageCommand = new Command<string>(OnNavigateToDataPage));

        private ICommand _navigateToSettingsCommand;
        public ICommand NavigateToSettingsCommand => _navigateToSettingsCommand ??
(_navigateToSettingsCommand = new Command(OnNavigateToSettingsPage));
```

```
        public List<SectionsModel> StatisticParameters { get; set; } = new
List<SectionsModel>()
        {
            new SectionsModel() { Name = "Atmospheric Pressure",  },
            new SectionsModel() { Name = "Precipitation Intensity" },
            new SectionsModel() { Name = "Total Daily Precipitation" },
            new SectionsModel() { Name = "Wind Speed" },
            new SectionsModel() { Name = "Wind Direction" },
            new SectionsModel() { Name = "Soil Temperature" },
            new SectionsModel() { Name = "Air Temperature" },
            new SectionsModel() { Name = "Soil Moisture" },
        };

        public MainPageViewModel(INavigationService navigationService)
            : base(navigationService)
        {
            Title = "Main Page";
        }

        private void OnNavigateToDataPage(string path)
        {
            var parameters = new Prism.Navigation.NavigationParameters();
            parameters.Add(NavigationConstants.StatisticParameter, path);

            NavigationService.NavigateAsync(nameof(StatisticsDetailsPage), parameters);
        }

        private void OnNavigateToSettingsPage()
        {
            NavigationService.NavigateAsync(nameof(SettingsPage));
        }
    }
}
```

Файл SettingsPageViewModel.cs

```
using DiplomApp.API.Models;
using DiplomApp.Extensions;
using DiplomApp.Model;
using DiplomApp.Services;
using Prism.Navigation;
using Realms;
using System;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Input;
using Xamarin.Forms;

namespace DiplomApp.ViewModels
{
    public class SettingsPageViewModel : ViewModelBase
    {
        private const string Last_Synchronization = "Last synchronization: ";
        private readonly IMeteorologicalService _meteorologicalService;
        private readonly IRealmService _realmService;

        public SettingsPageViewModel(INavigationService navigationService,
IMeteorologicalService meteorologicalService, IRealmService realmService) :
base(navigationService)
        {
            _meteorologicalService = meteorologicalService;
            _realmService = realmService;
```

```
            Title = "Settings";
        }

        private string _lastDataString;
        public string LastDataString
        {
            get => _lastDataString;
            set => SetProperty(ref _lastDataString, value);
        }

        private bool _isRunning;
        public bool IsRunning
        {
            get => _isRunning;
            set => SetProperty(ref _isRunning, value);
        }

        private ICommand _loadDataCommand;
        public ICommand LoadDataCommand => _loadDataCommand ?? (_loadDataCommand = new
Command(OnLoadDataAsync));

        public override void Initialize(INavigationParameters parameters)
        {
            try
            {
                Realm _realm = _realmService.GetInstance();

                var lastData = _realm.All<LoadedDataModel>().ToList();

                if (lastData != null)
                {
                    LastDataString = Last_Synchronization +
lastData.Last().Date.DateTime.ToShortDateString();
                }
            }
            catch (Exception ex)
            {
            }

            base.Initialize(parameters);
        }

        private void OnLoadDataAsync()
        {
            Task.Run(async () =>
            {
                Device.BeginInvokeOnMainThread(() =>
                {
                    IsRunning = true;
                });

                Realm _realm = _realmService.GetInstance();

                var newData = new LoadedDataModel()
                {
                    Date = DateTime.Now,
                };

                var last = _realm.All<LoadedDataModel>().LastOrDefault();
                newData.Id = (last?.Id ?? 0) + 1;
```

```
                var meteorological = await _meteorologicalService.GetAll();

                if (meteorological.Success)
                {
                    try
                    {
                        _realm.Write(() =>
_realm.RemoveAll<MeteorologicalDataRealmDTO>());
                    }
                    catch (Exception)
                    {
                    }

                    _realm.Write(() => _realm.Add(meteorological.Data.ToRealmDTO()));
                }

                _realm.Write(() => _realm.Add<LoadedDataModel>(newData));

                LastDataString = Last_Synchronization +
newData.Date.DateTime.ToShortDateString();

                IsRunning = false;
            });
        }
    }
}
```

Файл StatisticsDetailsPageViewModel.cs

```
using DiplomApp.API.Models;
using DiplomApp.Constants;
using DiplomApp.Extensions;
using DiplomApp.Model;
using DiplomApp.Services;
using Microcharts;
using Prism.Navigation;
using Realms;
using SkiaSharp;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Globalization;
using System.Linq;
using System.Xml.Schema;

namespace DiplomApp.ViewModels
{
    public class StatisticsDetailsPageViewModel : ViewModelBase
    {
        private const int _maximalDataCount = 15;
        private readonly Realm _realm;

        private readonly string[] _colors =
         {
            "#80ded9",
            "#2c3e50",
            "#77d065",
            "#b455b6",
            "#3498db",
            "#0acdff",
            "#ba3f1d",
            "#5d4a66",
```

```
            "#fcfc62",
            "#e9724c",
            "#481d24",
            "#162521",
            "#9fa0c3",
            "#edd382",
            "#f2f3ae",
            "#7cb518",
        };

        public StatisticsDetailsPageViewModel(INavigationService navigationService,
IRealmService realmService) : base(navigationService)
        {
            _realm = realmService.GetInstance();
        }

        private DateTimeOffset _startBorderDate;
        public DateTimeOffset StartBorderDate
        {
            get => _startBorderDate;
            set => SetProperty(ref _startBorderDate, value);
        }
        private DateTimeOffset _endBorderDate;
        public DateTimeOffset EndBorderDate
        {
            get => _endBorderDate;
            set => SetProperty(ref _endBorderDate, value);
        }

        private DateTime _startDate;
        public DateTime StartDate
        {
            get => _startDate;
            set => SetProperty(ref _startDate, value);
        }
        private DateTime _endDate;
        public DateTime EndDate
        {
            get => _endDate;
            set => SetProperty(ref _endDate, value);
        }

        private TimeSpan _time;
        public TimeSpan Time
        {
            get => _time;
            set => SetProperty(ref _time, value);
        }

        private bool _isTimeEnabled;
        public bool IsTimeEnabled
        {
            get => _isTimeEnabled;
            set => SetProperty(ref _isTimeEnabled, value);
        }

        private List<StatisticsDetailsModel> _data;

        public List<StatisticsDetailsModel> Data
        {
            get => _data;
            set => SetProperty(ref _data, value);
```

```
        }

        private BarChart _barChart;
        public BarChart BarChart
        {
            get => _barChart;
            set => SetProperty(ref _barChart, value);
        }

        private LineChart _lineChart;
        public LineChart LineChart
        {
            get => _lineChart;
            set => SetProperty(ref _lineChart, value);
        }
        private DonutChart _donutChart;
        public DonutChart DonutChart
        {
            get => _donutChart;
            set => SetProperty(ref _donutChart, value);
        }

        public override void Initialize(INavigationParameters parameters)
        {
            if (parameters.TryGetValue(NavigationConstants.StatisticParameter, out
string name))
            {
                Title = name;
            }

            try
            {
                if (!string.IsNullOrWhiteSpace(Title))
                {
                    EndBorderDate =
_realm.All<MeteorologicalDataRealmDTO>().Last().CreatedAt;
                    StartBorderDate =
_realm.All<MeteorologicalDataRealmDTO>().First().CreatedAt;

                    EndDate =
_realm.All<MeteorologicalDataRealmDTO>().Last().CreatedAt.DateTime;
                    StartDate =
_realm.All<MeteorologicalDataRealmDTO>().Last().CreatedAt.DateTime;
                }
            }
            catch (Exception ex)
            {
            }

            base.Initialize(parameters);
        }

        protected override void OnPropertyChanged(PropertyChangedEventArgs args)
        {
            try
            {
                if ((args.PropertyName == nameof(StartDate) || args.PropertyName ==
nameof(EndDate) || args.PropertyName == nameof(IsTimeEnabled) || (IsTimeEnabled &&
(args.PropertyName == nameof(Time)))) && EndDate != default && StartDate != default)
                {
                    var hour = Time.Hours;
```

```
                    Data = _realm.All<MeteorologicalDataRealmDTO>().ToList().Where(x =>
x.CreatedAt.DateTime.Date >= StartDate.Date && EndDate.Date >=
x.CreatedAt.DateTime.Date)
                        .Where(x => IsTimeEnabled ? x.CreatedAt.Hour == hour :
true).Select(x => new StatisticsDetailsModel()
                        {
                            Date = x.CreatedAt.DateTime,
                            Value = Title == "Atmospheric Pressure" ?
x.AtmosphericPressure :
                                Title == "Precipitation Intensity" ?
x.PrecipitationIntensity :
                                Title == "Total Daily Precipitation" ?
x.TotalDailyPrecipitation :
                                Title == "Wind Speed" ? x.WindSpeed :
                                Title == "Wind Direction" ? x.WindDirection :
                                Title == "Soil Temperature" ? x.SoilTemperature :
                                Title == "Air Temperature" ? x.AirTemperature :
                                Title == "Soil Moisture" ? x.SoilMoisture : 0,
                        }).ToList();

                    DataCompression();

                    if (!Data.Any())
                    {
                        return;
                    }

                    var charts = Data.Select(x => new ChartEntry((float)x.Value)
                    {
                        Color = SKColor.Parse(_colors[Data.IndexOf(x)]),
                        Label = x.DateString = LabelNameByDataDetails(x),
                        ValueLabel = x.Value.ToString("N2"),
                        ValueLabelColor = SKColor.Parse(_colors[Data.IndexOf(x)]),
                    }).ToList();

                    var min = (float)Data.Select(x => x.Value).Min();
                    var max = (float)Data.Select(x => x.Value).Max();

                    var dif = max - min;

                    BarChart = new BarChart
                    {
                        ShowYAxisLines = true,
                        ShowYAxisText = true,
                        YAxisPosition = Position.Right,
                        LabelTextSize = 40,
                        ValueLabelOrientation = Orientation.Horizontal,
                        LabelOrientation = Orientation.Horizontal,
                        MinValue = dif != 0 ? min + dif * 0.9f : 0,
                        MaxValue = max,
                        Entries = charts,
                    };

                    LineChart = new LineChart
                    {
                        ShowYAxisLines = true,
                        ShowYAxisText = true,
                        YAxisPosition = Position.Right,
                        LabelTextSize = 40,
                        ValueLabelOrientation = Orientation.Horizontal,
                        LabelOrientation = Orientation.Horizontal,
                        MinValue = dif != 0 ? min - dif * 0.9f : 0,
```

```
                        MaxValue = max,
                        Entries = charts
                    };

                    DonutChart = new DonutChart
                    {
                        LabelTextSize = 40,
                        MinValue = dif != 0 ? min + dif * 0.9f : 0,
                        MaxValue = max,
                        Entries = charts,
                        GraphPosition = GraphPosition.AutoFill,
                        LabelMode = Data.Count() <= 8 ? LabelMode.RightOnly :
LabelMode.None,
                    };
                }
            }
            catch (Exception ex)
            {
                var charts = new List<ChartEntry>()
                {
                    new ChartEntry(0),
                };

                BarChart = new BarChart() { Entries = charts };
                LineChart = new LineChart() { Entries = charts };
                DonutChart = new DonutChart() { Entries = charts };
            }

            base.OnPropertyChanged(args);
        }

        private void DataCompression()
        {
            int counter;

            if (Data.Count <= _maximalDataCount)
            {
                counter = 1;

                Data = Data.GroupBy(x => new { x.Date.Year, x.Date.Month, x.Date.Day,
x.Date.Hour, x.Date.Minute }).Select(x =>
                {
                    x.First().Value = x.Select(v => v.Value).Average();
                    x.First().CompressionDetails = "minute";
                    x.First().Index = counter;
                    counter++;

                    return x.First();
                }).ToList();
            }

            if (Data.Count > _maximalDataCount)
            {
                counter = 1;

                Data = Data.GroupBy(x => new { x.Date.Year, x.Date.Month, x.Date.Day,
x.Date.Hour }).Select(x =>
                {
                    x.First().Value = x.Select(v => v.Value).Average();
                    x.First().CompressionDetails = "hour";
                    x.First().Index = counter;
                    counter++;
```

```
                    return x.First();
                }).ToList();
            }

            if (Data.Count > _maximalDataCount)
            {
                counter = 1;

                if (Data.Select(x => x.Date.Day).All(x => x == Data.First().Date.Day))
                {
                    return;
                }

                Data = Data.GroupBy(x => new { x.Date.Year, x.Date.Month, x.Date.Day
}).Select(x =>
                {
                    x.First().Value = x.Select(v => v.Value).Average();
                    x.First().CompressionDetails = "day";
                    x.First().Index = counter;
                    counter++;

                    return x.First();
                }).ToList();
            }

            if (Data.Count > _maximalDataCount)
            {
                counter = 1;

                DateTimeFormatInfo dfi = DateTimeFormatInfo.CurrentInfo;
                Calendar cal = dfi.Calendar;

                Data = Data.GroupBy(x =>
                {
                    var week = cal.GetWeekOfYear(x.Date, dfi.CalendarWeekRule,
DayOfWeek.Monday);
                    return new { x.Date.Year, week };
                }).Select(x =>
                {
                    x.First().Value = x.Select(v => v.Value).Average();
                    x.First().CompressionDetails = "week";
                    x.First().Index = counter;
                    counter++;

                    return x.First();
                }).ToList();
            }

            if (Data.Count > _maximalDataCount)
            {
                counter = 1;

                Data = Data.GroupBy(x => new { x.Date.Year, x.Date.Date.Month
}).Select(x =>
                {
                    x.First().Value = x.Select(v => v.Value).Average();
                    x.First().CompressionDetails = "month";
                    x.First().Index = counter;
                    counter++;

                    return x.First();
```

```
                }).ToList();
            }
        }

        private string LabelNameByDataDetails(StatisticsDetailsModel x)
        {
            if (x.CompressionDetails == "minute")
            {
                return $"{x.Date.Minute}";
            }
            if (x.CompressionDetails == "hour")
            {
                return $"{x.Date.Hour}";
            }
            if (x.CompressionDetails == "day")
            {
                return $"{x.Date.Day}";
            }

            if (x.CompressionDetails == "week")
            {
                DateTimeFormatInfo dfi = DateTimeFormatInfo.CurrentInfo;
                Calendar cal = dfi.Calendar;

                var week = cal.GetWeekOfYear(x.Date, dfi.CalendarWeekRule,
DayOfWeek.Monday);

                return $"{week}";
            }

            if (x.CompressionDetails == "month")
            {
                return $"{x.Date.Month}";
            }

            return $"{x.Date.Hour}";
        }
    }
}
```

Файл IMeteorologicalService.cs

```
using DimplomApp.Common.Models;
using DiplomApp.API.Models;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace DiplomApp.Services
{
    public interface IMeteorologicalService
    {
        Task<ServiceResponse<IEnumerable<MeteorologicalData>>> GetAll();
    }
}
```

Файл IRealmService.cs

```
using Realms;

namespace DiplomApp.Services
{
    public interface IRealmService
    {
```

```
        Realm GetInstance();
    }
}
```

Файл MeteorilogicalService.cs

```
using DimplomApp.Common.Models;
using DiplomApp.API.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Threading.Tasks;

namespace DiplomApp.Services
{
    public class MeteorilogicalService : IMeteorologicalService
    {
        public async Task<ServiceResponse<IEnumerable<MeteorologicalData>>> GetAll()
        {
            ServiceResponse<IEnumerable<MeteorologicalData>> result = new
ServiceResponse<IEnumerable<MeteorologicalData>>();

            try
            {
                var httpClient = new HttpClient();
                var response = await
httpClient.GetAsync("https://diplomappapi20230707110817.azurewebsites.net/api");
                var json = await response.Content.ReadAsStringAsync();
                result.Data =
JsonConvert.DeserializeObject<List<MeteorologicalData>>(json);
            }
            catch (Exception ex)
            {
                result.ErrorMessage = ex.Message;
                result.Success = false;
            }

            return result;
        }
    }
}
```

Файл RealmService.cs

```
using Realms;

namespace DiplomApp.Services
{
    public class RealmService : IRealmService
    {
        public Realm GetInstance()
        {
            var config = RealmConfiguration.DefaultConfiguration;

            config.SchemaVersion = 2;

            return Realm.GetInstance(config);
        }
    }
}
```

Файл LoadedDataModel.cs

```
using Realms;
using System;

namespace DiplomApp.Model
{
    public class LoadedDataModel : RealmObject
    {
        [PrimaryKey]
        public int Id { get; set; }
        public DateTimeOffset Date { get; set; }
    }
}
```

Файл MeteorologicalDataRealmDTO.cs

```
using Realms;
using System;

namespace DiplomApp.Model
{
    public class MeteorologicalDataRealmDTO : RealmObject
    {
        public DateTimeOffset CreatedAt { get; set; }
        [PrimaryKey]
        public int EntryId { get; set; }
        public double AtmosphericPressure { get; set; }
        public double PrecipitationIntensity { get; set; }
        public double TotalDailyPrecipitation { get; set; }
        public double WindSpeed { get; set; }
        public double WindDirection { get; set; }
        public double SoilTemperature { get; set; }
        public double AirTemperature { get; set; }
        public double SoilMoisture { get; set; }
    }
}
```

Файл SectionsModel.cs

```
namespace DiplomApp.Model
{
    public class SectionsModel
    {
        public string Name { get; set; }
        public string Property { get; set; }
        public string Image { get; set; }
    }
}
```

Файл StatisticsDetailsModel.cs

```
using System;

namespace DiplomApp.Model
{
    public class StatisticsDetailsModel
    {
        public DateTime Date { get; set; }
        public double Value { get; set; }
        public string CompressionDetails { get; set; }
        public int Index { get; set; }
        public string DateString { get; set; }
    }
}
```

Файл DTOExtensions.cs

```
using DiplomApp.API.Models;
using DiplomApp.Model;
using System.Collections.Generic;
using System.Linq;

namespace DiplomApp.Extensions
{
    public static class DTOExtensions
    {
        public static IEnumerable<MeteorologicalDataRealmDTO> ToRealmDTO(this
IEnumerable<MeteorologicalData> data)
        {
            if (data == null)
            {
                return null;
            }

            return data.Select(x => new MeteorologicalDataRealmDTO()
            {
                EntryId = x.EntryId,
                CreatedAt = x.CreatedAt,
                WindSpeed = x.WindSpeed,
                SoilMoisture = x.SoilMoisture,
                WindDirection = x.WindDirection,
                AirTemperature = x.AirTemperature,
                SoilTemperature = x.SoilTemperature,
                AtmosphericPressure = x.AtmosphericPressure,
                PrecipitationIntensity = x.PrecipitationIntensity,
                TotalDailyPrecipitation = x.TotalDailyPrecipitation,
            });
        }

        public static IEnumerable<MeteorologicalData> ToModel(this
IEnumerable<MeteorologicalDataRealmDTO> data)
        {
            if (data == null)
            {
                return null;
            }

            return data.Select(x => new MeteorologicalData()
            {
                EntryId = x.EntryId,
                CreatedAt = x.CreatedAt.DateTime,
                WindSpeed = x.WindSpeed,
                SoilMoisture = x.SoilMoisture,
                WindDirection = x.WindDirection,
                AirTemperature = x.AirTemperature,
                SoilTemperature = x.SoilTemperature,
                AtmosphericPressure = x.AtmosphericPressure,
                PrecipitationIntensity = x.PrecipitationIntensity,
                TotalDailyPrecipitation = x.TotalDailyPrecipitation,
            });
        }
    }
}
```

Файл BorderlessDatePicker.cs

```
using Xamarin.Forms;

namespace DiplomApp.Custom
{
    public class BorderlessDatePicker : DatePicker
    {
    }
}
```

Файл CollectionViewHeightConverter.cs

```
using DiplomApp.Model;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using Xamarin.Forms;

namespace DiplomApp.Converters
{
    internal class CollectionViewHeightConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            if (value is IEnumerable<StatisticsDetailsModel> result)
            {
                return result.Count() * 20;
            }

            return 0;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            return null;
        }
    }
}
```

Файл DoubleToStringConverter.cs

```
using System;
using System.Globalization;
using Xamarin.Forms;

namespace DiplomApp.Converters
{
    internal class DoubleToStringConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            if (value is double result)
            {
                return result.ToString("N2");
            }

            return value;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
```

```
        {
            return null;
        }
    }
}
```

Файл StatisticsDetailsModelsToStringConverter.cs

```
using DiplomApp.Model;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using Xamarin.Forms;

namespace DiplomApp.Converters
{
    internal class StatisticsDetailsModelsToStringConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            if (value is IEnumerable<StatisticsDetailsModel> result)
            {
                return result.First().CompressionDetails;
            }

            return "";
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            return "";
        }
    }
}
```

Файл NavigationConstants.cs

```
namespace DiplomApp.Constants
{
    public class NavigationConstants
    {
        public const string StatisticParameter = nameof(StatisticParameter);
    }
}
```

Збірка DiplomApp.Android

Файл MainApplication.cs

```
using System;
using Android.App;
using Android.Runtime;

namespace DiplomApp.Droid
{
    [Application(Theme = "@style/MainTheme")]
    public class MainApplication : Application
    {
        public MainApplication(IntPtr javaReference, JniHandleOwnership transfer)
            : base(javaReference, transfer)
```

```
        {
        }

        public override void OnCreate()
        {
            base.OnCreate();
            Xamarin.Essentials.Platform.Init(this);
        }
    }
}
```

Файл MainActivity.cs

```
using Android.App;
using Android.Content.PM;
using Android.OS;
using Prism;
using Prism.Ioc;

namespace DiplomApp.Droid
{
    [Activity(Theme = "@style/MainTheme",
              ConfigurationChanges = ConfigChanges.ScreenSize |
ConfigChanges.Orientation | ConfigChanges.UiMode | ConfigChanges.ScreenLayout |
ConfigChanges.SmallestScreenSize)]
    public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

            global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
            LoadApplication(new App(new AndroidInitializer()));
        }

        public override void OnRequestPermissionsResult(int requestCode, string[]
permissions, Android.Content.PM.Permission[] grantResults)
        {
            Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode,
permissions, grantResults);

            base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }

    public class AndroidInitializer : IPlatformInitializer
    {
        public void RegisterTypes(IContainerRegistry containerRegistry)
        {
        }
    }
}
```

Файл SplashActivity.cs

```
using Android.App;
using Android.Content;
using AndroidX.AppCompat.App;

namespace DiplomApp.Droid
{
```

```
    [Activity(Theme = "@style/MainTheme.Splash",
              MainLauncher = true,
              NoHistory = true)]
    public class SplashActivity : AppCompatActivity
    {
        protected override void OnResume()
        {
            base.OnResume();
            StartActivity(new Intent(Application.Context, typeof(MainActivity)));
        }
    }
}
```

Файл BorderlessDatePickerRenderer.cs

```
using Android.Graphics.Drawables;
using DiplomApp.Custom;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;

[assembly: ExportRenderer(typeof(BorderlessDatePicker),
typeof(BorderlessDatePickerRenderer))]

public class BorderlessDatePickerRenderer : DatePickerRenderer
{
    public static void Init()
    { }

    protected override void OnElementChanged(ElementChangedEventArgs<DatePicker> e)
    {
        base.OnElementChanged(e);
        if (e.OldElement == null)
        {
            Control.Background = null;

            var layoutParams = new MarginLayoutParams(Control.LayoutParameters);
            layoutParams.SetMargins(0, 0, 0, 0);
            LayoutParameters = layoutParams;
            GradientDrawable gd = new GradientDrawable();
            gd.SetStroke(0, Android.Graphics.Color.LightGray);
            Control.SetBackgroundDrawable(gd);
            Control.LayoutParameters = layoutParams;
            Control.SetPadding(0, 0, 0, 0);
            SetPadding(0, 0, 0, 0);
        }
    }
}
```

Додаток Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Кваліфікаційна_робота_Бабко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна_робота_Бабко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Program.rar	Архів. Містить коди програми і скомпільований застосунок.
Презентація	
Презентація_Бабко.pptx	Презентація кваліфікаційної роботи.