МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«МОБІЛЬНИЙ ДОДАТОК ПРОГНОЗУ ПОГОДИ НА ОСНОВІ ДАНИХ GFS»**

Виконав: студент 2 курсу, групи 8.1210

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

Т. С. Степура
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Горбенко В. І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент зав.кафедри фундаментальної та прикладної математики, доцент, д.т.н. Гребенюк С. М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

(підпис)

« 09 » 06 2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Степурі Тимуру Сергійовичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Мобільний додаток прогнозу погоди на основі даних GFS

керівник роботи (проекту) Горбенко Віталій Іванович, к.ф.-м.н., доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 09 » червня 2021 року № 851-с

2. Строк подання студентом роботи 24.11.2021

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області.
2. Проектування мобільного додатку.
3. Розробка мобільного додатку.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Горбенко В. І., доцент, к.ф.-м.н.	23.04.2021	03.05.2021
2	Горбенко В. І., доцент, к.ф.-м.н.	03.05.2021	05.09.2021
3	Горбенко В. І., доцент, к.ф.-м.н.	05.09.2021	12.10.2021
Додатки	Горбенко В. І., доцент, к.ф.-м.н.	12.10.2021	19.10.2021

7. Дата видачі завдання 23.04.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	22.04.2021	
2.	Збір вихідних даних.	01.05.2021	
3.	Обробка методичних та теоретичних джерел.	18.05.2021	
4.	Розробка першого і другого розділу.	29.07.2021	
5.	Розробка третього розділу.	06.09.2021	
6.	Оформлення та нормоконтроль кваліфікаційної роботи.	23.11.2021	
7.	Захист кваліфікаційної роботи.	17.12.2021	

Студент _____ Т. С. Степура
(підпис) (ініціали та прізвище)

Керівник роботи _____ В. І. Горбенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ С. П. Швидка
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Мобільний додаток прогнозу погоди на основі даних GFS»: 63 с., 25 рис., 1 табл., 10 джерел, 3 додатки.

ПРОГНОЗУВАННЯ ПОГОДИ, МОБІЛЬНИЙ ДОДАТОК, ДАНІ GFS, ДОДАТОК, МЕТЕОМОДЕЛІ, ГЛОБАЛЬНА СИСТЕМА ПРОГНОЗУВАННЯ, МЕТАДАНІ, СУПЕРКОМП'ЮТЕРИ, SWOT-АНАЛІЗ, ОПЕРАЦІЙНА СИСТЕМА, ANDROID, JAVA, СМАРТФОН, РЕЗУЛЬТАТИ, ЕФЕКТИВНІСТЬ, API, ІНТЕРФЕЙС.

Об’єкт дослідження – мобільний додаток прогнозу погоди на основі даних глобальної моделі прогнозування GFS.

Мета роботи: на основі аналізу досліджень вчених проблеми створення та використання глобальних комп'ютерних моделей прогнозу погоди створити мобільний додаток на основі даних моделі GFS.

Метод дослідження – аналітичний, порівняльний, дослідницький.

У кваліфікаційній роботі розглядаються проблеми дослідження прогнозу погоди із застосуванням суперкомп'ютерів, які використовують глобальну числову модель прогнозування погоди, дані якої згодом використовуються на регіональному рівні для будь-якої точки планети.

Розроблено програму для операційної системи Android. Використання цього мобільного додатка може бути широко використане в різних сферах життя людини, оскільки портативність та швидкодія є важливими характеристиками застосування мобільних пристроїв.

SUMMARY

Master’s qualifying paper "Mobile weather forecast application based on GFS data": 63 pages, 25 figures, 1 tables, 10 references, 3 supplements.

WEATHER FORECASTING, MOBILE APPLICATION, GFS DATA, APPLICATION, METEOROLOGICAL MODELS, GLOBAL FORECASTING SYSTEM, METADATA, SUPERCOMPUTERS, SWOT ANALYSIS, OPERATING SYSTEM, ANDROID, JAVA, SMARTPHONE, RESULTS, EFFICIENCY, API, INTERFACE.

The object of the study is mobile weather forecast application based on data from the global GFS forecasting model.

The aim of the study is creation of a mobile application based on GFS model data based on the analysis of scientific research problems of creating and using global computer models of weather forecasting.

The methods of research are analytical, comparative, research.

The qualification work examines the problems of weather forecast research using supercomputers that use a global numerical model of weather forecasting, the data of which are then used at the regional level for any part of the world.

Developed an application for the Android operating system. The use of this mobile application can be widely used in various spheres of human life, as portability and speed are important characteristics of the use of mobile devices.

ЗМІСТ

ВСТУП

Прогнозування погоди – це застосування науки і технологій для прогнозування стану атмосфери у певному місці та у певний час. Люди намагалися передбачити погоду неформально упродовж тисячоліть, а формально – з 19 століття. Прогнози погоди складаються шляхом збору кількісних даних про поточний стан атмосфери в даному місці та використання метеорології для прогнозування змін атмосфери.

Прогнозування погоди тепер ґрунтується на комп'ютерних моделях, які враховують багато атмосферних факторів. Прогноз погоди – науково обґрунтоване припущення про майбутній стан погоди у певному пункті чи регіоні на певний період. Складається (розробляється) метеорологічними службами на основі методів метеорології.

Прогнози діляться за завчасністю періоду, на який дається прогноз (прогноз поточної погоди, короткостроковий, довгостроковий і т.д.), а також за типами в залежності від цілей, для яких вони розроблені (загального користування, авіаційні, морські та річкові, сільськогосподарські та інші), за масштабом (приватні, місцеві, регіональні, галузеві, країнові, глобальні) та за відповідальністю (особисті, на рівні підприємства або організації, на рівні державних органів).

У сучасному суспільстві майже всі сфери життя тісно пов'язані з комп'ютерними технологіями. Не становить виняток і застосування різноманітних мобільних додатків. Велику популярність набули мобільні додатки, пов'язані з прогнозом погоди, оскільки різні сфери життя людини диктують необхідність використання спеціальної техніки та пристроїв, що дозволяють бути в епіцентрі подій.

Мета дипломної роботи – створення мобільного додатку прогнозу погоди на основі даних GFS.

Перший розділ присвячений обґрунтуванню та аналізу теми та підбору завдання для її реалізації.

Другий розділ присвячений підбору технологій, що застосовуються для реалізації задачі та перерахування функціональних вимог додатка.

Третій розділ присвячений опису додатка, що розробляється, та дослідженню інтерполяції погодних даних та SWOT-аналізу мобільного додатка.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Метеомоделі прогнозу погоди

Всі прогнози погоди складаються на основі одних і тих самих даних: це дані з метеостанцій, розкиданих по всьому світу, дані з метеозондів, які збирають інформацію про погоду на висоті, та знімки з супутників. Таким чином, метеорологи отримують величезну кількість цифр, з яких і складається прогноз. Метеодані поміщають у спеціальні програми, які називаються метеомоделями. Щоб мати можливість прогнозувати погоду, потрібні різні чисельні моделі. Ці числові моделі набувають значення атмосферних змінних, і за допомогою складних рівнянь стан цих змінних може бути відомий у найближчому майбутньому. Чим досконаліша модель і чим потужніший комп'ютер, на якому вона розраховується, тим більшу кількість метеоданих він здатний обробити і тим точніше буде прогноз.

На цей час існує кілька великих моделей, з яких багато мобільних програм беруть інформацію: це GFS (Global Forecast System), WRF (Weather Research and Forecasting), NAM (North American Mesoscale), ECMWF (European Centre for Medium-Range Weather Forecast), NMM (Nonhydrostatic Mesoscale Model) ті інші. Ці професійні моделі, які обробляються дуже продуктивними комп'ютерами, створені великою кількістю професіоналів у держаних інтересах, оскільки метеорології приділяють велику увагу. Для прогнозування погоди використовуються одні з найпотужніших комп'ютерів у світі. Коштують вони дорого, а їхня продуктивність більше не збільшується колишніми темпами: кремнієві мікросхеми майже нікуди вдосконалювати. Немає підстав не довіряти цим системам, але й на 100% точного прогнозу вони не дадуть, особливо на тривалий період. Є ще суттєвий мінус – у цих моделях не завжди враховуються локальні особливості рельєфу.

1.2 Глобальна система прогнозування погоди (GFS)

Найбільш популярна чисельна модель прогнозу погоди – GFS, що містить глобальну комп'ютерну модель та варіаційний аналіз, який виконує Національна служба погоди США (NWS). Глобальна система прогнозів (GFS) – це модель прогнозу погоди національних центрів екологічного прогнозування (NCEP), яка генерує дані для десятків змінних атмосфери та ґрунту, включаючи температуру, вітер, опади, вологість ґрунту та концентрацію атмосферного озону. Система поєднує чотири окремі моделі (атмосферу, модель океану, модель землі ґрунту та морський лід), які працюють разом, щоб точно відображати погодні умови. Додаток NCEI Data Access пропонує широкий вибір варіантів завантаження та піднабору для зростаючого збору екологічних даних. Хоча поточні пропозиції обмежуються в основному інформацією про погоду та клімат, додаток має незалежну від даних інфраструктуру, призначену для підтримки широкого спектру форматів спостережень з різних наукових дисциплін.

Незважаючи на свій (як для оперативної моделі) поважний вік, GFS була і залишається однією з двох найкращих у світі глобальних моделей прогнозу погоди (інша європейська ECMWF Integrated Forecast System). Global Forecast System (GFS) – глобальна чисельна модель прогнозу погоди, створена та експлуатована Національною адміністрацією з дослідження океанів та атмосфери США (NOAA). Перша версія GFS з'явилася в 1988 році в Національному центрі передбачення стану навколишнього середовища США (NCEP) як частина Глобальної системи асиміляції даних (GDAS) та прогнозу для покращення глобального прогнозу та прогнозу синоптичних систем довгохвильового характеру. Модель постійно розвивається, збільшується кількість прогнозованих метеорологічних величин, підвищується якість самих прогнозів, удосконалюються її експлуатаційні характеристики.

В даний час у Сполучених Штатах Америки експлуатується ціла низка версій цієї моделі, що відрізняються роздільною здатністю та фізичними схемами. GFS є єдиною у світі глобальною прогностичною моделлю, більшість вихідних даних яких у вільному доступі в інтернеті (це регламентовано законодавством США) і є основою для діяльності численних недержавних компаній, що займаються прогнозами погоди. Модель виконується чотири рази на день і робить прогнози терміном до 16 днів уперед, але з зниженою просторовою роздільною здатністю через 10 днів. Рівень прогнозування зазвичай зменшується з часом (як і у разі будь-якої чисельної моделі прогнозування погоди), і для більш довгострокових прогнозів лише більш великі шкали зберігають значну точність. Є однією з переважаючих моделей середнього діапазону синоптичного масштабу загального використання.

- GFS модель оновлюється чотири рази на день
(00:00, 06:00, 12:00 та 18:00 UTC) і видає
прогнози до 16 діб (на 384 години).
- Модель виконується у двох частинах:
  - перша частина має більш високу роздільну здатність та виходить до 192 годин (8 днів);
  - друга частина проходить від 192 до 384 годин (16 днів).

Як і в більшості робіт уряду США, дані GFS доступні безкоштовно у відкритому доступі відповідно до положень законодавства США. Через це модель є основою для прогнозів численних приватних, комерційних та закордонних погодних компаній.

Добре відомо, що ці прогнози не є повністю надійними, оскільки динаміку атмосфери можна легко змінити. Характеристики атмосфери та переважна погода залежать від значення багатьох змінних одночасно. На більшість із цих змінних безпосередньо впливає кількість сонячної радіації, що впливає на нашу планету. Відповідно до кількості сонячної радіації та

інші змінні змінюються, починаючи з температури та вітрового режиму. Припустимо, що прогнози моделі GFS не забезпечують нам високої надійності через 7 днів. Можна навіть сказати, що за 3-4 дні це вже не зовсім точно. Більшість національних метеорологічних інститутів та агенцій відмовляються від більшості результатів цієї моделі, особливо тих, які перевищують 10 днів по тому. Український центр прогнозів використовує дані погоди за 5-6 днів.

GFS покриває всю земну кулю з базовою горизонтальною роздільною здатністю 18 миль (28 кілометрів) між точками сітки, що використовується оперативними синоптиками, які передбачають погоду на строк до 16 днів у майбутньому. Горизонтальна роздільна здатність падає до 44 миль (70 кілометрів) між точками сітки для прогнозів від одного до двох тижнів.

Національні центри прогнозування довкілля (NCEP) розробляють початкові умови теплого старту системи глобального прогнозування NOAA (GFS) для виконання оперативних детермінованих середньострокових чисельних прогнозів погоди. GFS побудована з використанням динамічного ядра GFDL з кубічною сферою кінцевого обсягу (FV3) та системи засвоєння даних статистичної інтерполяції за точками сітки (GSI).

Поточна діюча GFS працює на 64 рівнях по вертикалі, що тягнуться від поверхні до верхніх шарів стратосфери, і на шести осередках кубічної сфери з роздільною здатністю C768 або 13 км по горизонталі. Нова версія GFS зі 127 рівнями, що тягнуться до мезопаузи, була реалізована для роботи 3 лютого 2021 року. Ці початкові умови стають доступними чотири рази на день для виконання прогнозів у циклах 00Z, 06Z, 12Z та 18Z відповідно. Для кожного циклу набір даних містить перше припущення про стан атмосфери, знайдене в каталозі ./gdas.yyyymmdd/hh-6/RESTART, яке є 6-годинним прогнозом GDAS з останнього циклу, і збільшення атмосферного аналізу та аналізу поверхні для поточного циклу, знайдені у каталозі ./gfs.yyyymmdd/hh, що виробляються системами асиміляції даних.

Екологічні дані NCEI охоплюють широкий спектр наукових дисциплін, методів архівування, угод про імена, форматів файлів та стратегій управління. NCEI розробляє програмне забезпечення, API-інтерфейси, методи візуалізації та інші послуги для покращення доступу до даних, їх виявлення та взаємодії [1, 3].

1.3 Робота з метаданими на NOAA OneStop

OneStop – це система зберігання, управління та виявлення метаданих з відкритим вихідним кодом, створена дослідниками CIRES з гранту Національних центрів екологічної інформації NOAA. Оскільки обсяг даних збільшується із року у рік, відданість OneStop принципам хмарного дизайну знімає проблеми з масштабованістю та доступом.

Проект складається із чотирьох «компонентів» високого рівня:

- Потік подій / база даних для отримання, зберігання та забезпечення гнучкого наступного використання метаданих;
- Багатофункціональний RESTful Search API, створений на основі метаданих необробленого потоку подій;
- Ультрасучасний інтерфейс користувача на основі браузера (UI), який має спеціальні можливості, вбудовані в кожну функцію;
- Інтерфейс командного рядка (CLI), що забезпечує більш ефективне виконання скриптів та алгоритмів під час використання інформації, отриманої з Search API.

NOAA OneStop можна використовувати для пошуку будь-яких даних NOAA із записом метаданих. Це сумісний інструмент, призначений для вивчення даних з різних наукових дисциплін, форматів, періодів часу та місць, це засіб класифікації, впорядкування та характеристики даних або вмісту.

Метадані надають описову, контекстну інформацію про набори даних та продукти, щоб полегшити їх виявлення, використання та розуміння. Записи метаданих – це живі документи, які необхідно як мінімум щорічно переглядати для внесення змін до даних і змісту метаданих або для включення інформації, яка робить запис більш корисним. Geoportal виконує пошук в індексованих полях метаданих для всіх заархівованих наборів даних, поєднуючи різні веб-служби виявлення та метадані в структуру, орієнтовану на користувача. Планування розробки метаданих на початкових етапах планування проекту може заощадити час, зусилля та гроші у довгостроковій перспективі. Встановлюється формат або протокол для документування даних, а потім слід дотримуватися його.

Документується все під час розробки даних: набагато простіше отримати цю інформацію у режимі реального часу, ніж заднім числом задокументувати. Метадані часто являють собою компіляцію інформації з багатьох джерел. Також необхідно документувати, систематизувати та відслідковувати джерела даних протягом усього проекту, ретельно дотримуватися стандарту при написанні метаданих та вивчати його перед тим, як почати писати, щоб визначити, як і де документувати конкретну інформацію [4].

1.4 Порівняння європейської та американської моделей GFS

Європейська модель GFS має безліч переваг перед своїм найкращим суперником, створеним урядом Сполучених Штатів Америки. Обидві моделі мають дуже хороші характеристики і досить добре пророкують погоду. Корпорація, що працює незалежно один від одного, ще не провела об'єктивних тестів, щоб визначити, яка з двох моделей найкраще здатна передбачати погоду.

Незважаючи на те, що жодна з двох моделей не є переможницею над іншою, більшість фахівців у цій галузі обирають європейську модель. Одна з головних відмінностей цієї моделі від американської – це технологія. Вона має складніші та дорогі комп'ютерні системи, які дозволяють їм працювати більш ефективно. За допомогою цієї складнішої технології досягаються більш точні, скориговані і більш роздільно здатні атмосферні проекції.

Більшість експертів кажуть, що європейська модель GFS набагато краща, ніж американська, з погляду моделювання даних. Аргумент, який вони використовують, полягає в тому, що він значно повніший і надає більший обсяг перевіреної інформації. Щоб дати приклад, щоб можна було бачити різницю між європейською та американською моделями, європейська модель здатна проводити 50 симуляцій атмосфери за цикл прогнозу, тоді як у Північній Америці можна запускати лише 20 моделювань за раз [2].

1.4.1 Особливості моделі GFS, її переваги та недоліки

GFS – спочатку створена з урахуванням квазіоптичної моделі поширення потоків (не враховуючи інтерференцій), не враховує рельєфу суші, наявність невеликих островів, обриси берегової лінії материків і великих островів. При цьому дає досить точний прогноз середньої погоди для квадрата 0,5х0,5 (а тепер уже у багатьох випадках і 0,25х0,25) для відкритих океанів, але зовсім не придатна для місць поблизу суші, а також закритих водойм, таких як , наприклад, Середземне море, і вже точно дає цілком випадковий результат для річок та озер, закритих бухт тощо. В даний час модель вдосконалена і її основною перевагою є регулярний (кілька разів на день) розрахунок погоди для всієї планети, який проводиться незалежно у кількох гідрометцентрах у різних країнах.

Основні переваги прогнозу погоди, розрахованого на основі моделі GFS:

- регулярний розрахунок прогнозу погоди для поверхні всієї планети у кількох гідрометцентрах у різних країнах;

- доступність прогнозів;
- прийнятно-точне передбачення погоди для відкритих океанів.

Основні недоліки прогнозу погоди, розрахованого на основі моделі GFS:

- середня погода для квадрата 0,25x0,25 (15nm x 15nm), а в багатьох місцях і 0,5x0,5 (30nm x 30nm);
- непридатність прогнозу для ділянок, що межують із сушею та на суші.

Прогноз погоди, виконаний на основі моделі GFS доступний у 3 форматах:

- Meteocharts;
- GRIB;
- Табличний.

Водночас у кожному гідрометцентрі світу (а в деяких країнах їх по кілька) розроблено метеорологічні моделі, оптимізовані для конкретного регіону.

Якщо порівняти прогноз, що видається гідрометцентром однієї країни для іншої країни, наприклад, прогноз гідрометцентру України по Мадриду з прогнозом по Мадриду, що видається іспанською La Agencia Estatal de Meteorología. Саме прогноз погоди, розрахований за оптимізованою моделлю, передається по радіо, телебаченню, VHF, Navtex. Відповідно, для кожного регіону нас насамперед цікавить офіційний прогноз погоди, розрахований за найбільш оптимізованою для цієї зони моделі. Такі прогнози доступні за VHF та Navtex і, з деякими застереженнями, на сайті weather.gmdss.org. Офіційні прогнози даються як середня погода для регіонів («морських квадратів») та у форматі shipping forecast.

Разом з тим, ціла низка гідрометцентрів по всьому світу, окрім видачі прогнозу на підставі моделей GFS, офіційного shipping forecast, видають додаткові прогнози у різних форматах для різних ділянок та ступенів

деталізації території. У багатьох випадках такі прогнози виявляються найчастіше точнішими для заданих ділянок, хоча можуть пропустити глобальні погодні ефекти.

На підставі вищесказаного очевидно, що алгоритм роботи з джерелами прогнозу погоди наступний:

- Оцінити прогноз погоди по моделі GFS (по Meteochart'ам або GRIB). Виділити основні погодні ефекти (депресії, урагани, фронти) та оцінити основні погодні ефекти на їх основі (посилення вітру, ймовірність поривів, звів тощо);
- Ознайомитись з офіційним прогнозом погоди у форматі shipping forecast;
- Ознайомитись із додатковими, місцевими джерелами прогнозу погоди.

GFS надає глобальні прогнози, повторний аналіз клімату та спеціальні набори даних, розроблені для задоволення різноманітних вимог користувачів. Вони доступні через Інтернет, двоточкове розповсюдження, сервери даних та широкомовне розсилання.

Операційні прогнози GFS націлені на те, щоб показати, як погода, швидше за все, розвиватиметься. Для цього Центр здійснює ансамбль прогнозів. Окремо вони є повними описами еволюції погоди. Разом вони вказують на можливість ряду майбутніх погодних сценаріїв. Наша інтегрована система прогнозування (IFS) забезпечує прогнози та відповідну перевірку з різною роздільною здатністю та для різних часових діапазонів. Перевірка забезпечує важливий зворотний зв'язок про якість системи прогнозування.

GFS знаходиться на передньому краї досліджень у галузі чисельного прогнозування погоди. Тут використовуються наукові досягнення у таких галузях, як асиміляція даних, моделювання земних систем, передбачуваність та повторний аналіз з метою покращення прогнозів. GFS працює спільно з

вченими по всьому світу над усіма аспектами систем прогнозування, а також над багатьма галузями, які підтримують створення прогнозів [1].

1.4.2 Використання суперкомп'ютерів у моделі GFS

Щодня в GFS потрібно фільтрувати сотні мільйонів спостережень та комбінувати їх із недавніми короткостроковими прогнозами. Цей процес відомий як асиміляція даних і він забезпечує вхідні дані для складної числової моделі системи Землі, яка імітує погоду по всьому світу. Чисельна модель заснована на фізичних законах, які керують змінами в атмосфері, суші та океанах. Щоб зробити прогноз, вирішуються математичні рівняння, що описують ці закони, з використанням аналізованого стану земної системи як відправну точку. Цей тип процедури прогнозування відомий як чисельне прогнозування погоди (ЧПП). На виконання цих операцій необхідні комп'ютери особливого класу – суперкомп'ютери.

Суперкомп'ютери складаються з багатьох тисяч процесорів, які можуть ефективно обмінюватися інформацією та працювати як одна потужна машина. Тільки суперкомп'ютер може обробляти безліч математичних обчислень і обробки даних, які необхідні для сучасного прогнозування погоди.

Обчислювальні потужності GFS також зберігають та розповсюджують прогнози серед держав-членів та Співпрацюючих держав, а також низки інших користувачів – і все це у суворих часових рамках. Час на суперкомп'ютері ретельно контролюється.

Половина доступного часу приділяється на щоденне оперативне прогнозування. Основний операційний прогноз виконується двічі на день (полудні та опівночі). Є також ще два проміжні прогони моделі, які в основному використовуються для забезпечення граничних умов для локальних моделей, керованих нашими державами-членами та державами, що співпрацюють. Держави-члени також виділяють ресурси на

суперкомп'ютер для таких дій, як виконання деяких власних операційних прогнозів або для роботи з конкретними проектами.

Інша половина доступного часу на суперкомп'ютері відводиться на дослідження GFS, які є життєво важливими для постійного поліпшення прогнозів. Він також включає в себе тестування наступного оновлення системи прогнозування. Він запускається паралельно з операційною системою протягом тривалого періоду тестування перед впровадженням в експлуатацію.

Розвиток суперкомп'ютерів допомагає надавати точніші прогнози погоди. Поліпшення представлення процесів земної системи в моделях, виконання ансамблевих прогнозів з вищою роздільною здатністю та покращення засвоєння даних спостережень за погодою – все це може покращити навички прогнозування. Однак такі розробки вимагають все більшої обчислювальної потужності, тому суперкомп'ютер GFS оновлюється в середньому кожні чотири або п'ять років. Збільшення потужності комп'ютерів при одночасному підвищенні енергоефективності та збереженні доступності – все це запорука майбутнього. Суперкомп'ютер GFS складається з двох ідентичних систем Cray XC, які забезпечують два автономні кластери процесорів та сховища.

З'являється нове покоління обчислювальних систем із ексафлопними можливостями (1018 або мільярд мільярдів обчислень за секунду), які обіцяють підвищену енергоефективність. Однак важлива не тільки чиста обчислювальна потужність, необхідні радикальні зміни, щоб оптимізувати систему прогнозування, щоб використати весь потенціал ексафлопних машин.

Основна програма масштабованості GFS вирішує цю проблему за допомогою розробників метеорологічних моделей, фахівців з інформатики та постачальників обладнання з усього світу у рамках спільного підходу до розробки програмного та апаратного забезпечення. Ключові завдання включають: адаптацію комп'ютерних кодів для ефективної роботи з низкою

нових комп'ютерних архітектур, покращення обробки даних та використання машинного навчання.

Робота на стику між обчислювальною наукою та прогнозуванням погоди дозволить GFS і надалі розширювати межі навичок прогнозування погоди [5].

1.5 Актуальність теми, постановка задачі

Проаналізувавши дослідження вчених з проблеми створення та використання глобальних комп'ютерних моделей прогнозу погоди, можна помітити, що в умовах інформаційних технологій, що стрімко розвиваються, – створення мобільних додатків, які враховуватимуть регіональні особливості місцевості, використовуючи популярні глобальні моделі – дуже актуально. На сьогоднішній день є необхідність введення в життя людини все більш удосконалених та ефективних портативних пристроїв з адаптованими на них наочними мобільними додатками прогнозу погоди. У дипломному проекті треба:

- на основі аналізу досягнень вчених, а також вже застосовуваних моделей та пристроїв, запропоновати варіант мобільного додатка прогнозу погоди на основі даних глобальної моделі GFS;
- підібрати технології та перераховані функціональні вимоги до додатку;
- описати додаток, що розробляється, обговорені отримані результати, в тому числі і ефективність роботи додатку при різних варіантах вихідних даних;
- провести дослідження інтерполяції проміжних значень погодних даних та провести SWOT-аналіз.

Цей додаток має бути дійсно – мобільним та ефективним. У рамках дипломного проекту мобільний додаток буде створено на смартфон, використовуючи операційну систему Android.

2 ТЕХНОЛОГІЧНА ЧАСТИНА

2.1 Середовище програмування Android Studio

Android Studio – це інтегроване середовище розробки (IDE) для роботи з платформою Android, анонсована 16 травня 2013 року на конференції Google I / O. Є офіційним середовищем розробки додатків для платформи Android.

Особливості:

- Розширений редактор макетів: WYSIWYG, здатність працювати з UI компонентами за допомогою Drag-and-Drop, функція попереднього перегляду макета на декількох конфігураціях екрану;
- Збірка додатків, заснована на Gradle;
- Різні види збірок і генерація кількох .apk файлів;
- Рефакторинг коду;
- Статичний аналізатор коду (Lint), що дозволяє знаходити проблеми продуктивності, несумісності версій і інше;
- Вбудований ProGuard і утиліта для підписки додатків;
- Шаблони основних макетів і компонентів Android;
- Підтримка розробки додатків для Android Wear і Android TV;
- Вбудована підтримка Google Cloud Platform, яка включає в себе інтеграцію з сервісами Google Cloud Messaging і App Engine.

Для розробки додатків використовується високорівневий прикладний інтерфейс програмування Java для Android, за допомогою якого можна створювати додатки для кінцевих користувачів Android.

Комплект для розробки програмного забезпечення (SDK) для Android поставляється з Android Studio, плагіном, який називається набором інструментальних засобів для розробки на Android (ADT). Цей інструмент

розробки IDE (IDE) для створення, налагодження та тестування додатків на Java. Android SDK може використовуватися без ADT; замість інструментів, можна використовувати інструменти командного рядка. Емулятор підтримує використання обох підходів, і з його допомогою можна запустити, виправити і перевірити додатки. 90% розробки додатків може бути завершена, навіть без використання реального пристрою. Повністю функціональний емулятор для Android відтворює найбільш вивчені характеристики пристрою. Серед тих функцій, які не можуть бути імітовані в емуляторі, є USB-підключення, робота камери і відео, імітація роботи навушників, батареї і технології Bluetooth.

Android Emulator базується на технології з відкритим вихідним кодом "імітація" процесора під назвою QEMU, який був розроблений Беллар. Та ж сама технологія може емулювати одну операційну систему на 16 інших, незалежно від того, яка використовується процесором. QEMU забезпечує емуляцію рівня процесора.

2.2 Онлайн сервіс OpenWeatherMap

OpenWeatherMap – онлайн сервіс, який надає платний (є функціонально обмежена безкоштовна версія) API для доступу до даних про поточну погоду, прогнози та історичні дані. Як джерело даних використовуються офіційні метеорологічні служби, дані з метеостанцій аеропортів та дані з приватних метеостанцій.

Особливості:

- OpenWeatherMap використовує API, щоб надати дані поточної погоди, прогнозу та карт з погодними явищами, такими як хмари, вітер, тиск та опади. Усі погодні дані можна отримати у форматах JSON, XML або HTML;

- Поточні погодні дані можуть бути знайдені по місту або з географічних координат. Дані оновлюються кожні 10 хвилин;
- OpenWeatherMap надає такі види прогнозних даних:
  - Хвилинний прогноз на 1 годину;
  - Часовий прогноз на 4 дні;
  - Денний прогноз на 16 днів;
  - Кліматичний прогноз на 30 днів.
- Система геокодування OpenWeatherMap дозволяє знайти міста за назвою, країною, поштовим індексом або географічними координатами. Пошук можливий за частиною імені міста;
- OpenWeatherMap надає безліч карт погоди, включаючи карти опадів, хмарності, атмосферного тиску, температури, вітру та багато інших. Картки можуть бути підключені до мобільних додатків та веб-сайтів. Погодні картки можуть бути підключені як шари для багатьох постачальників карт, у тому числі статичних тайлів, WMS, OpenLayers, Leaflet, карт Google. Сервіс використовує карти OpenStreetMap для побудови погодних карт;
- Всі дані OpenWeatherMap поширюються ліцензією Creative Common license CC-BY-SA 2.0 [6].

2.3 GFS

Глобальна система прогнозів (GFS) — це модель прогнозу погоди Національних центрів екологічного прогнозування (NCEP), яка генерує дані для десятків атмосферних і грунтових змінних, включаючи температуру, вітер, опади, вологість ґрунту та концентрацію озону в атмосфері. Система поєднує чотири окремі моделі (атмосфера, модель океану, модель землі/ґрунту та морський лід), які працюють разом, щоб точно відображати погодні умови.

Особливості:

- Модель GFS має різну роздільну здатність 27 км. Роздільна здатність — це відстань між двома точками сітки моделі погоди. Більша роздільна здатність розміром від 50 до 10 км зазвичай розгортається на відносно рівнинній місцевості, тоді як гірські хребти вимагають, щоб вузли були набагато ближче один до одного, зазвичай 5, 2 або 1 км;
- Глибина прогнозу моделі погоди GFS27 становить 10 днів. Глибина прогнозу погоди – це кількість годин або днів, на які складається прогноз. Як правило, чим нижче глибина, тим точніше прогноз;
- Крок прогнозу GFS становить 1 годину. Крок прогнозу — це те, як довго ви можете бачити прогноз у додатку або на веб-сайті. Існує два основних типи кроків: 1 год і 3 год;
- Частота оновлення GFS27 становить 4 рази на день. Частота оновлення прогнозу – це регулярний інтервал часу, після якого від суперкомп’ютерів надходять нові прогнозні дані [3].

2.4 API

API (Application Programming Interface) являє собою сукупність різних інструментів, функцій, реалізованих у вигляді інтерфейсу для створення нових додатків, завдяки якому одна програма взаємодіятиме з іншою.

Основне завдання API – надання можливості програмістам суттєво полегшити завдання розробки різних додатків з допомогою використання вже готового коду.

Особливості:

- API спрощує та прискорює створення нових продуктів. Розробникам не доводиться щоразу створювати нові компоненти. Є можливість

взяти API готового компонента і впровадити його у своє програмне забезпечення;

- Програмний інтерфейс підвищує безпеку розробки. За допомогою нього можна винести ряд функцій в окремий додаток, унеможлививши їх некоректне використання;
- API спрощує налаштування зв'язків між різними сервісами та програмами. Інтерфейс нівелює необхідність у тісному співробітництві творців різних додатків. Розробники можуть впроваджувати підтримку сторонніх сервісів, не контактуючи їх творцями;
- Наявність готових інтерфейсів дозволяє заощадити фінанси, із якими часто пов'язане створення нових програмних рішень [7, 8].

2.5 Бібліотека SQLite

SQLite – це вбудована бібліотека, яка реалізує автономний, безсерверний, нульовий конфігурації, механізм транзакції СУБД SQL. Це база даних, яка налаштована на нуль, що означає, що її не потрібно налаштовувати у системі. SQLite безпосередньо звертається до своїх файлів зберігання.

Особливості:

- Транзакції є атомарними, послідовними, ізольованими та довготривалими навіть після системних збоїв і збоїв живлення;
- Нульова конфігурація – не потрібно налаштування або адміністрування;
- Повнофункціональна реалізація SQL з розширеними можливостями, такими як часткові індекси, індекси виразів, JSON, загальні табличні вирази та віконні функції;

- Повна база даних зберігається в одному кросплатформному дисковому файлі. Чудово підходить для використання як формат файлу програми;
- Підтримує терабайтні бази даних і гігабайтні рядки та великі об’єкти;
- Невелике значення коду: менше 600 КБ повністю налаштовано або набагато менше без додаткових функцій;
- Простий, легкий у використанні API;
- Швидко: у деяких випадках SQLite швидше, ніж пряме введення-виведення файлової системи;
- Написано в ANSI-C. Прив’язки TCL включені. Прив’язки для десятків інших мов доступні окремо;
- Добре прокоментований вихідний код із 100% покриттям тестуванням гілок;
- Доступний у вигляді одного файлу вихідного коду ANSI-C, який легко компілювати і, отже, легко додати до більшого проекту;
- Автономність: немає зовнішніх залежностей;
- Кросплатформенність: Android, *BSD, iOS, Linux, Mac, Solaris, VxWorks та Windows (Win32, WinCE, WinRT) підтримуються з коробки. Легко переносити на інші системи;
- Джерела знаходяться у відкритому доступі;
- Постачається з автономним клієнтом інтерфейсу командного рядка (CLI), який можна використовувати для адміністрування баз даних SQLite [9, 10].

2.6 Функціональні вимоги додатку

У даній дипломній роботі буде розроблений додаток під назвою “WeatherAPP”, який буде спрямовано на можливість отримання даних про погоду, а також їх збереження до бази даних для подальшого перегляду.

Воно повинно відповідати таким вимогам:

- додаток повинен надавати можливість введення країни або міста, а також коду країни в поля введення;
- додаток повинен надавати інформацію про поточну погоду;
- додаток повинен надавати інформацію про прогноз погоди на 5 днів;
- додаток повинен мати таймер за допомогою якого буде відбуватися автоматичний запит на отримання інформації про поточну погоду через певну кількість часу;
- додаток повинен мати кнопку зупинки таймера;
- додаток повинен мати архів, в якому будуть зберігатися дані про поточну погоду, які будуть отримані після автоматичного запиту за таймром у форматі: місто або країна (код країни) - дата і час - температура, вологість, швидкість вітру, хмарність, тиск. Для кожного типу погодних даних повинен бути свій розділ.

Повну структуру вимог можна побачити на рисунку 2.1.

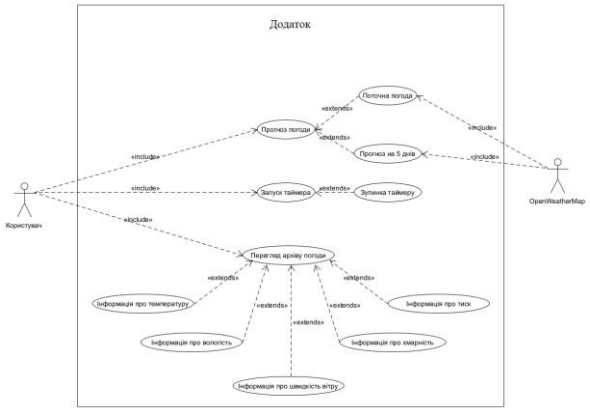


Рисунок 2.1 – Діаграма варіантів використання додатку

3 РЕАЛІЗАЦІЯ ПРОЕКТУ

У цьому розділі буде реалізований додаток під назвою “WeatherAPP”, який буде спрямовано на можливість отримання даних про погоду, а також їх збереження до бази даних для подальшого перегляду.

3.1 Загальна архітектура додатку

Додаток складається з 2 шарів та 5 класів.

Серед шарів, є такі як:

- activity_main.xml – шар головної сторінки додатку;
- activity_archive_menu.xml – шар архіву погодних даних.

Серед класів:

- MainActivity.java – клас пов'язаний із шаром activity_main.xml, в якому реалізовано запит на отримання даних про поточну погоду та прогноз на 5 днів; кнопка старту таймера, який зациклює запит на прогноз про поточну погоду та кнопка зупинки таймера; кнопка архіву погоди, що відкриває шар activity_archive_menu.xml; також у ньому реалізовано команду, яка збирає отримані дані з прогнозу поточної погоди, що здійснюється за допомогою таймера та відправляє їх до класу MyDbManager.java, в якому ці дані зберігаються до бази даних;
- ArchiveMenuActivity.java – клас пов'язаний із шаром activity_archive_menu.xml, в якому реалізовані кнопки, що виводять збережені дані з бази даних на екран;
- MyConstans.java – клас, у якому реалізовано структуру бази даних;

- MyDbManager.java – допоміжний клас, в якому реалізовано збереження даних до бази даних, які були відправлені з класу MainActivity.java. Також у ньому реалізовано зчитування даних із бази даних, які після натискання однієї з кнопок у класі ArchiveMenuActivity.java виводяться на екран шару activity_archive_menu.xml;
- MyDbHelper.java – клас, у якому реалізовано створення бази даних та таблиці в ній, а також метод, який очищатиме базу даних після зміни версії самої бази даних.

Структуру класів та взаємодію між компонентами можна побачити на рисунку 3.1 – 3.2.

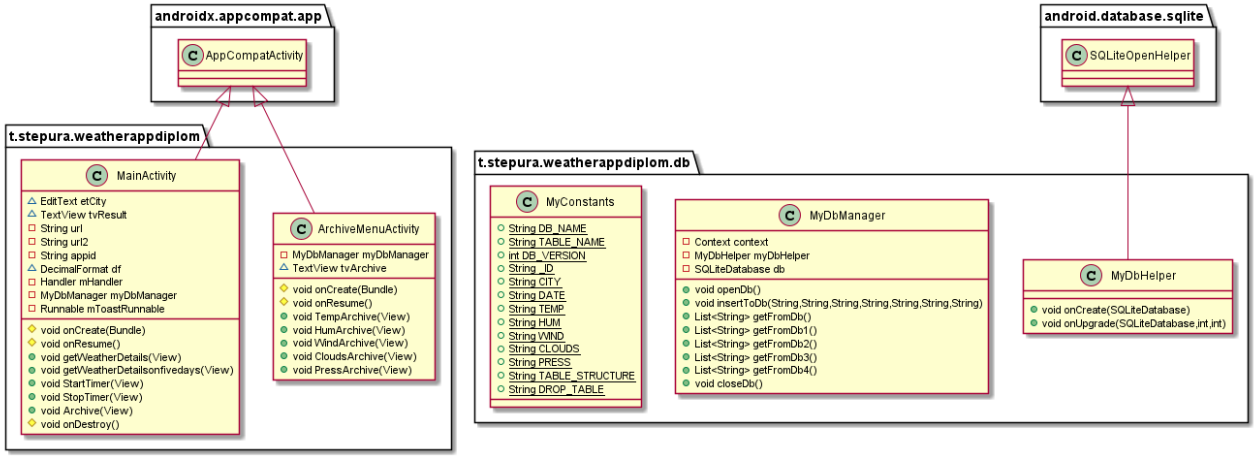


Рисунок 3.1 – Діаграма класів

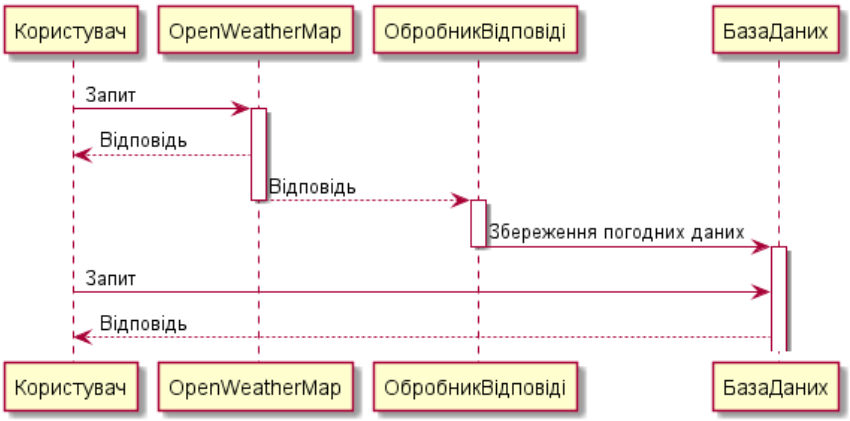


Рисунок 3.2 – Діаграма послідовності

3.2 Графічний інтерфейс користувача

Екран головного меню являє собою фон з полем введення назви міста чи країни та полем введення коду країни, з кнопками виведення поточної погоди, прогнозу на 5 днів, запуску та зупинки таймера, архіву збережених погодних даних та екраном виведення інформації (див. рис. 3.3). Інтерфейс цього меню реалізований в файлі activity_main.xml.

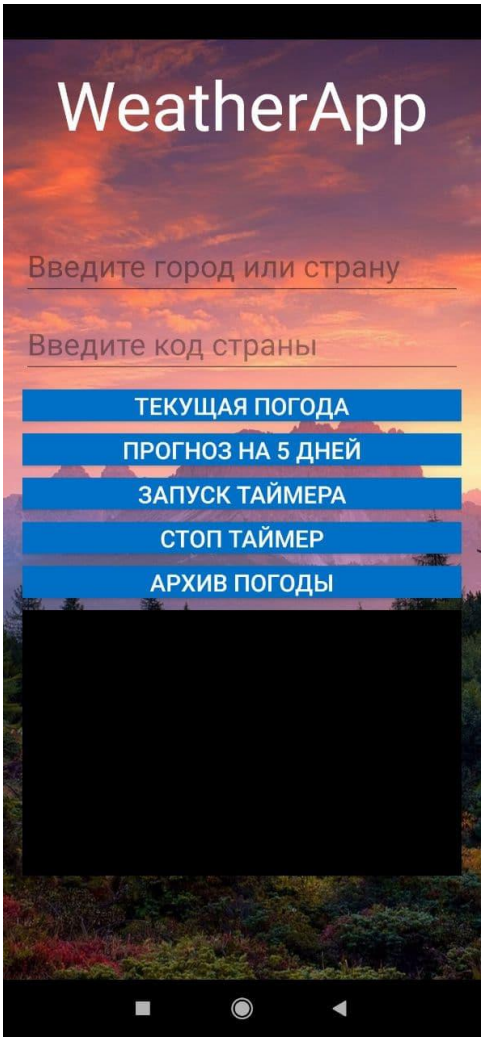


Рисунок 3.3 – Екран головного меню

Після введення назви міста (Наприклад: “Запоріжжя”) та натискання на кнопку “Поточна погода”, на екрані виведення інформації з'являються дані про погоду на даний момент (див. рис. 3.4).

Рисунок 3.4 – Виведення даних поточної погоди на екран

Якщо при введеному місті в полі введення міста натиснути на кнопку "Прогноз на 5 днів", на екрані виведення інформації з'являються дані про погоду на 5 днів з інтервалом 3 години (див. рис. 3.5).

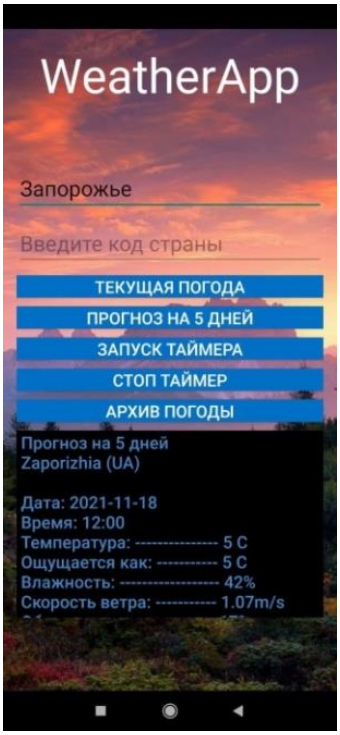


Рисунок 3.5 – Виведення даних прогнозу погоди на 5 днів на екран

Якщо погодних даних багато, то їх можна скролити (див. рис. 3.6).

Рисунок 3.6 – Скролінг даних прогнозу погоди на 5 днів

Якщо ввести в поле введення міста назву міста (Наприклад: “Александрія”), яке існує у двох країнах і не ввести код країни та натиснути на кнопку виведення погодних даних (Наприклад: “Поточна погода”), тоді будуть виведені погодні дані міста однієї з країн. У данному випадку це Єгипет (див. рис. 3.7).

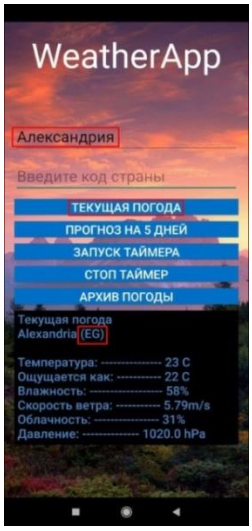


Рисунок 3.7 – Приклад прогнозу погоди по назві міста без її коду

Якщо при введеній назві міста ввести код країни (Наприклад: “US” – це код США) у поле введення коду країни і натиснути на кнопку виведення погодних даних (Наприклад: “Поточна погода”), тоді виведуться погодні дані конкретної країни (див. рис. 3.8).

Рисунок 3.8 – Приклад прогнозу погоди по назві міста з її кодом

Якщо натиснути на кнопку "Запуск таймера", тоді буде відбуватися автоматичний запит на сайт OpenWeatherMap на отримання даних про поточну погоду через кожні 30 секунд, після чого дані зберігаються в базі даних. Кнопка “Стоп таймер” зупиняє таймер.

Якщо натиснути кнопку "Архів погоди", то відкриється меню яке являє собою фон з екраном виведення інформації та кнопками виведення збережених погодних даних температури, вологості, швидкості вітру,

хмарності, тиску. Інтерфейс цього меню реалізований в файлі activity_archive_menu.xml (див. рис. 3.9).

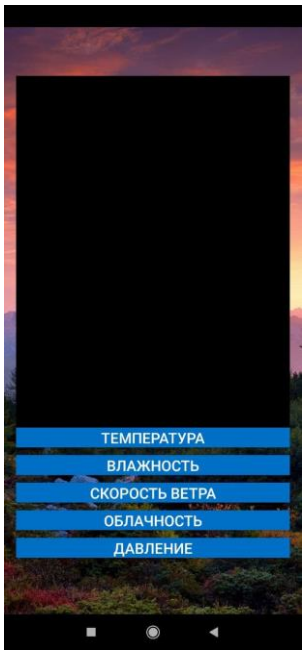


Рисунок 3.9 – Екран меню архіву

Якщо натиснути на кожну з кнопок, буде виведено збережені погодні дані автоматичного запиту таймера у форматі “Назва міста-[дд/мм/рр]-значення” (див. рис. 3.10 – 3.14).

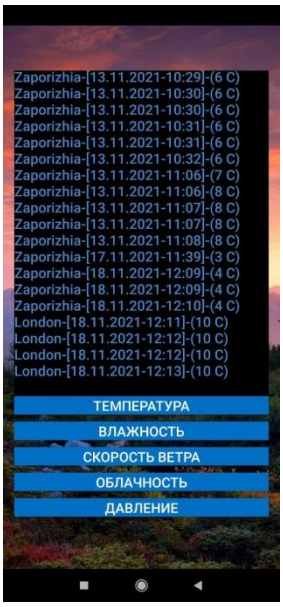


Рисунок 3.10 – Температура

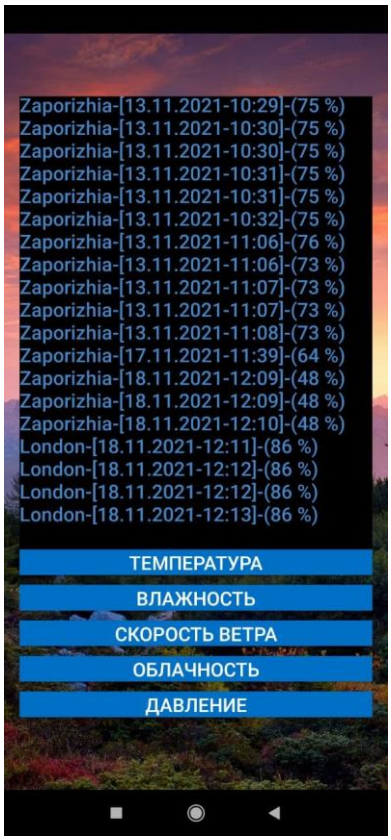


Рисунок 3.11 – Вологість

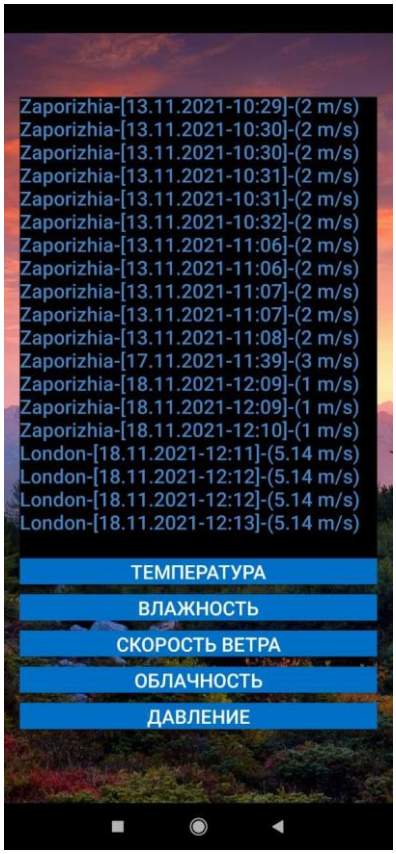


Рисунок 3.12 – Швидкість вітру

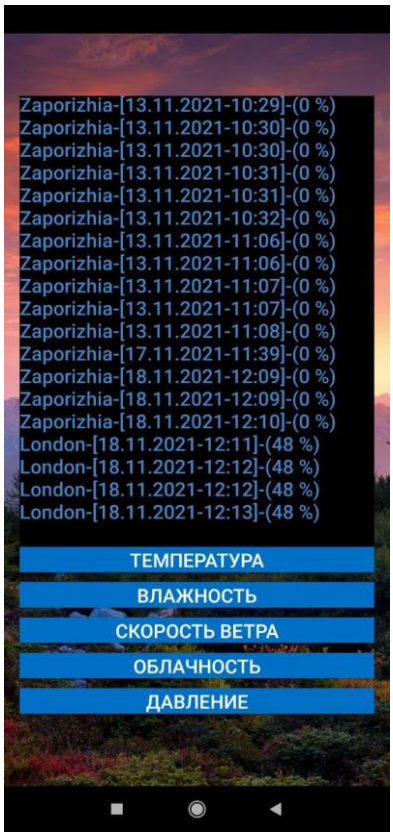


Рисунок 3.13 – Хмарність

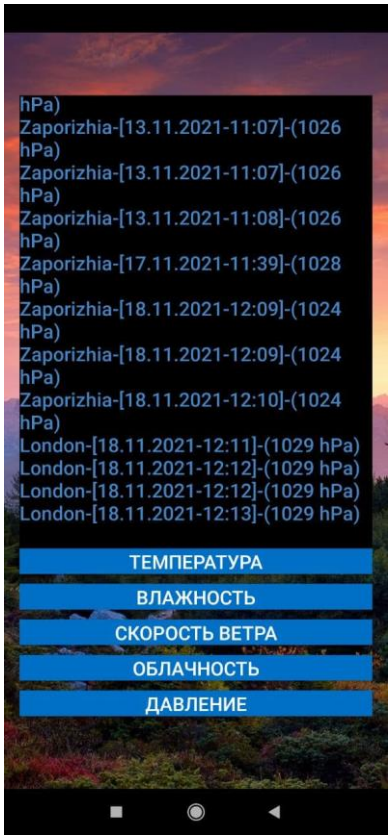


Рисунок 3.14 – Тиск

3.3 Робота додатку

При запуску додатку починає роботу метод onCreate() класу MainActivity.java. У ньому ініціалізуються такі елементи:

– інтерфейс головної сторінки програми

```
setContentView(R.layout.activity_main);
```

– поле введення міста чи країни

```
etCity = findViewById(R.id.etCity);
```

– поле введення коду країни

```
etCountry = findViewById(R.id.etCountry);
```

– екран виведення інформації

```
tvResult = findViewById(R.id.tvResult);
```

– допоміжний клас для взаємодії з базою даних

```
myDbManager = new MyDbManager(this);
```

При методі onResume() додаток взаємодіє з користувачем. У цьому методі реалізовано відкриття бази даних після запуску Activity:

```
myDbManager.openDb();
```

Далі йде метод getWeatherDetails (це кнопка) який визначає поточну погоду. У ньому реалізовано:

– оголошення змінної для утримання повної URL-адреси

```
String tempUrl = "";
```

– оголошення змінних для взяття введеного тексту з полів введення міста чи країни та введення коду країни

```
String city = etCity.getText().toString().trim();
String country = etCountry.getText().toString().trim();
```

– умова при якій, якщо поле введення міста чи країни порожнє, виводиться повідомлення про порожнє поле

```
if(city.equals("")){
   tvResult.setText("Поле не может быть пустым!");
```

– умова при якій, якщо поле введення коду країни не порожнє, тоді в URL-адресу входить і код країни. А якщо поле порожнє, тоді в URL-адресі використовується лише назва міста чи країни.

```
}else{
   if(!country.equals("")){
      tempUrl = url + "?q=" + city + "," + country + "&appid=" + appid;
   }else{
      tempUrl = url + "?q=" + city + "&appid=" + appid;}
```

– Запит із вже сформованою URL-адресою на сайт OpenWeatherMap

```
StringRequest stringRequest = new StringRequest(Request.Method.POST, tempUrl, new
Response.Listener<String>()
```

– оголошення змінної для збереження даних у ній, одержаних після запиту на сайт OpenWeatherMap

```
String output = "";
```

– отримання та збереження погодних даних у змінні

```
JSONObject jsonResponse = new JSONObject(response);
JSONArray jsonArray = jsonResponse.getJSONArray("weather");
JSONObject jsonObjectWeather = jsonArray.getJSONObject(0);
JSONObject jsonObjectMain = jsonResponse.getJSONObject("main");
double temp = jsonObjectMain.getDouble("temp") - 273.15;
double feelsLike = jsonObjectMain.getDouble("feels_like") - 273.15;
float pressure = jsonObjectMain.getInt("pressure");
int humidity = jsonObjectMain.getInt("humidity");
JSONObject jsonObjectWind = jsonResponse.getJSONObject("wind");
String wind = jsonObjectWind.getString("speed");
JSONObject jsonObjectClouds = jsonResponse.getJSONObject("clouds");
String clouds = jsonObjectClouds.getString("all");
JSONObject jsonObjectSys = jsonResponse.getJSONObject("sys");
String countryName = jsonObjectSys.getString("country");
String cityName = jsonResponse.getString("name");
```

– визначення кольору тексту на екрані виведення інформації

```
tvResult.setTextColor(Color.rgb(68, 134, 199));
```

– збереження погодних даних у заданому форматі у змінну output та подальшим виведенням цих даних на екран виводу інформації

```
output += " Текущая погода"
        + "\n " + cityName + " (" + countryName + ")"
        + "\n"
        + "\n Температура: " + "---------- " + df.format(temp) + " C"
        + "\n Ощущается как: " + "---------- "  + df.format(feelsLike) + " C"
        + "\n Влажность: " + "---------- " + humidity + "%"
        + "\n Скорость ветра: " + "---------- " + wind + "m/s"
        + "\n Облачность: " + "---------- " + clouds + "%"
        + "\n Давление: " + "---------- " + pressure + " hPa";
tvResult.setText(output);
```

– поява помилки у вигляді спливаючого повідомлення

```
}, new Response.ErrorListener(){


    @Override
    public void onErrorResponse(VolleyError error) {
        Toast.makeText(getApplicationContext(), error.toString().trim(),
Toast.LENGTH_SHORT).show();
    }
});
```

– створення екземпляра черги запитів

```
RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
```

– додавання строкового запиту до черги запитів

```
requestQueue.add(stringRequest);
```

Далі йде метод getWeatherDetailsonfivedays (це кнопка) який визначає прогноз погоди на 5 днів. Він схожий на метод getWeatherDetails, за винятком URL-адреси, що відрізняється. Також він відрізняється великим обсягом даних, оскільки розрахунок ведеться на 5 днів з інтервалом о 3 годині.

Далі йдуть методи:

– StartTimer() – кнопка яка запускає таймер

– StopTimer() – кнопка яка зупиняє таймер

– Archive() – кнопка яка відкриває сторінку з архівом погодних даних

Далі йдуть змінні:

– для таймера

```
private Handler mHandler = new Handler();
```

– объекта класса MyDbManager

```
private MyDbManager myDbManager;
```

Метод, який запускається після натискання кнопки StartTimer і зупиняється після натискання кнопки StopTimer

```
private Runnable mToastRunnable = new Runnable() {
   @Override
   public void run() {
```

Зчитування поточної дати та часу з пристрою

```
Date currentDate = new Date();
DateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy", Locale.getDefault());
//Текущая дата
String dateText = dateFormat.format(currentDate);
```

```
DateFormat timeFormat = new SimpleDateFormat("HH:mm", Locale.getDefault()); //Текущее время

String timeText = timeFormat.format(currentDate);
```

Далі йде команда, яка збирає дані отримані після запиту та відправляє їх методу insertToDb(), який збереже ці дані до бази даних, який знаходиться в класі MyDbManager

```
myDbManager.insertToDb(cityName24, date, tempstr, humiditystr, wind24, clouds24, pressurestr);
```

Далі йде таймер із затримкою за часом на 30 секунд

```
mHandler.postDelayed(this, 30000);
```

Далі йде метод, який закриває базу даних

```
protected void onDestroy() {
   super.onDestroy();
   myDbManager.closeDb();  //Закрытие базы данных
}
```

Клас MyConstans.java містить структуру бази даних. Також там оголошено змінну необхідну для скидання бази даних

```
public class MyConstants {
   public static final String DB_NAME = "my_db.db";
   public static final String TABLE_NAME = "my_table";
   public static final int DB_VERSION = 4;
   public static final String _ID = "_id";
   public static final String CITY = "city";
   public static final String DATE = "date";
```

```
    public static final String TEMP = "temp";

    public static final String HUM = "hum";

    public static final String WIND = "wind";

    public static final String CLOUDS = "clouds";

    public static final String PRESS = "press";

    public static final String TABLE_STRUCTURE = "CREATE TABLE IF NOT EXISTS " +

            TABLE_NAME + " (" + CITY + " TEXT," + _ID + " INTEGER PRIMARY KEY," +
DATE
+ " TEXT," + TEMP + " INTEGER," + HUM + " INTEGER," +
WIND + " INTEGER," + CLOUDS + " INTEGER," + PRESS + " INTEGER)";


    public static final String DROP_TABLE = "DROP TABLE IF EXISTS " + TABLE_NAME;

}
```

Клас MyDbHelper,java створює базу даних, таблицю в ній, а також очищає базу даних після зміни версії бази

```
        public class MyDbHelper extends SQLiteOpenHelper {


            public MyDbHelper(@Nullable Context context) {
                super(context, MyConstants.DB_NAME, null, MyConstants.DB_VERSION);
            }


            @Override
            public void onCreate(SQLiteDatabase db) {
                db.execSQL(MyConstants.TABLE_STRUCTURE);
            }


            @Override
```

```
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL(MyConstants.DROP_TABLE);
        onCreate(db);
    }
}
```

Метод openDb() класу MyDbManager.java відкриває базу даних

```
public void openDb(){
    db = myDbHelper.getWritableDatabase();
}
```

Метод insertToDb() класу MyDbManager.java приймає надіслані погодні дані з класу MainActivity.java та зберігає їх у базі даних

```
public void insertToDb(String cityName24, String date, String tempstr, String
humiditystr, String wind24, String clouds24, String pressurestr){
    ContentValues cv = new ContentValues();

    cv.put(MyConstants.CITY, cityName24);
    cv.put(MyConstants.DATE, date);
    cv.put(MyConstants.TEMP, tempstr);
    cv.put(MyConstants.HUM, humiditystr);
    cv.put(MyConstants.WIND, wind24);
    cv.put(MyConstants.CLOUDS, clouds24);
    cv.put(MyConstants.PRESS, pressurestr);
    db.insert(MyConstants.TABLE_NAME, null, cv);
}
```

Метод getFromDb() класу MyDbManager.java зчитує дані з бази даних, для подальшого виведення їх на екран виведення інформації

```
public List<String> getFromDb(){
    List<String> tempList = new ArrayList<>();
    Cursor cursor = db.query(MyConstants.TABLE_NAME, null, null, null, null, null,
null);

    while (cursor.moveToNext()){
```

```
            String cityy = cursor.getString(cursor.getColumnIndex(MyConstants.CITY));
            String datee = cursor.getString(cursor.getColumnIndex(MyConstants.DATE));
            String tempp = cursor.getString(cursor.getColumnIndex(MyConstants.TEMP));

            String ex = cityy + "-[" + datee + "]-" + "(" + tempp + " C" + ")";

            tempList.add(ex);

        }

        cursor.close();
        return tempList;
    }
```

Метод closeDb() класу MyDbManager закриває базу даних, для того, щоб не навантажувати систему

```
public void closeDb(){
    myDbHelper.close();
}
```

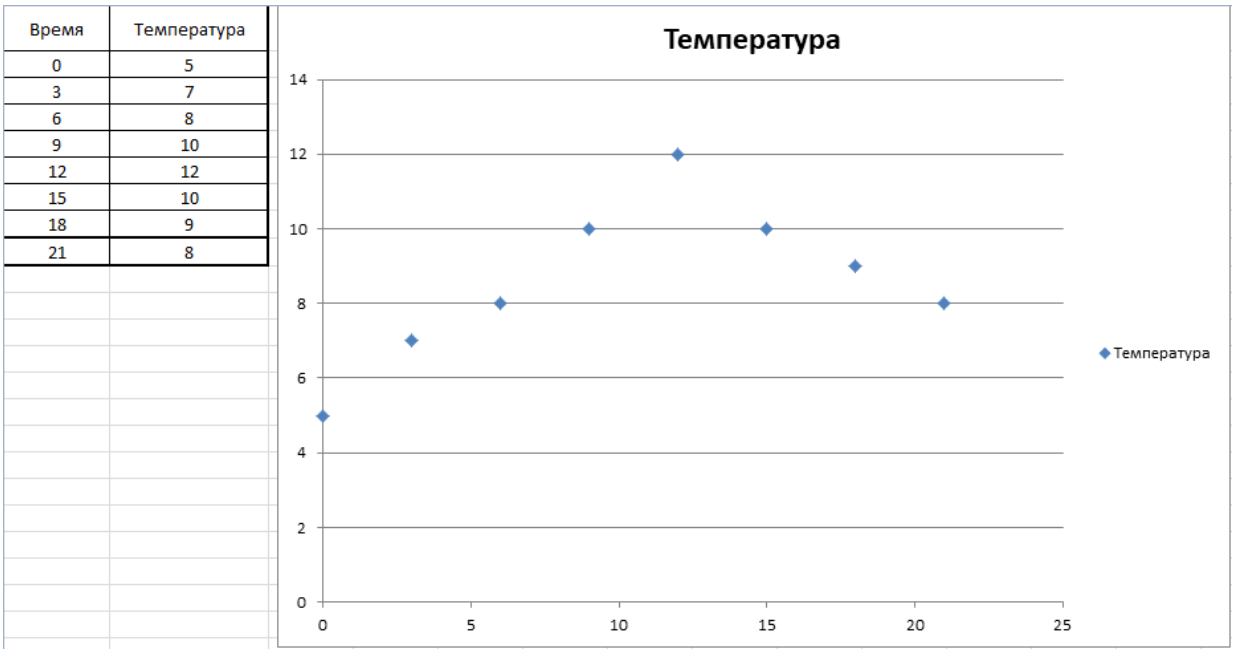
У класі ArchiveMenuActivity.java реалізовані кнопки, які виводять погодні дані на екран.

3.4 Дослідницька частина

3.4.1 Інтерполяція погодних даних

Для обчислення інтерполяції погодних даних як приклад візьмемо дані прогнозу на 5 днів (температура, вологість, швидкість вітру, хмарність, тиск) за 27 листопада 2021 року у діапазоні від 00:00 до 21:00 з інтервалом у 3 години.

Спочатку візьмемо дані температури (див. рис. 3.15).



Время	Температура
0	5
3	7
6	8
9	10
12	12
15	10
18	9
21	8

Рисунок 3.15 – Дані граничних значень часу та температури

Наприклад візьмемо такі проміжні значення часу:

$$x = [1.5, 4.5, 7.5, 10.5, 13.5, 16.5, 19.5].$$

Необхідно знайти y – це невідомі проміжні значення (в даному прикладі температури).

Обчислення відбуватимуться за формулою лінійної інтерполяції:

$$y = y_1 + ((x-x_1)/(x_2-x_1) * (y_2-y_1)),$$

де x_1 и x_2 – граничні значення часу, а y_1 и y_2 – граничні значення температури.

Після вирішення рівняння знаходимо проміжні значення температури:

$$y = [6, 7.5, 9, 11, 11, 9.5, 8.5].$$

Формується відповідна таблиця, в якій сірим кольором виділено проміжні значення часу та знайденої температури (див. рис. 3.16).

Время	Температура
00:00	5° C
01:30	6° C
03:00	7° C
04:30	7,5° C
06:00	8° C
07:30	9° C
09:00	10° C
10:30	11° C
12:00	12° C
13:30	11° C
15:00	10° C
16:30	9,5° C
18:00	9° C
19:30	8,5° C
21:00	8° C

Рисунок 3.16 – Граничні та проміжні значення часу та температури

Обчислимо інтерполяцію для інших погодних даних.

Обчислимо дані вологості (див. рис. 3.17 – 3.18).

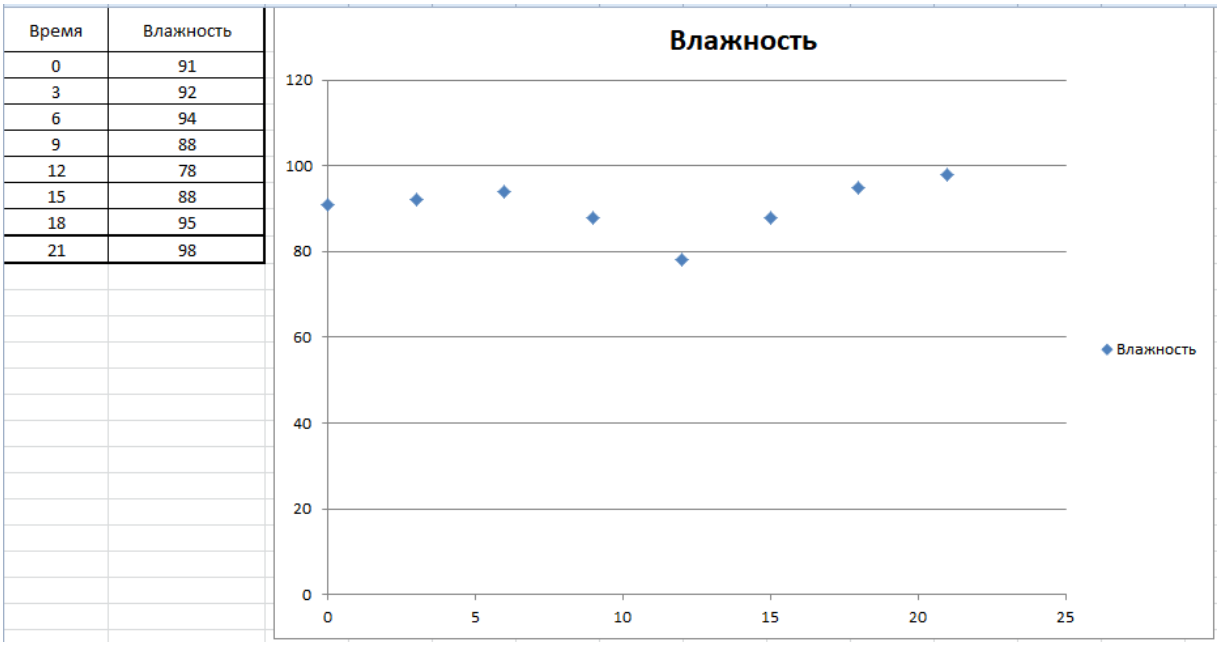


Рисунок 3.17 – Дані граничних значень часу та вологості

Проміжні дані часу

x = [1.5, 4.5, 7.5, 10.5, 13.5, 16.5, 19.5].

Проміжні дані вологості

y = [91.5, 93, 91, 83, 83, 91.5, 96.5].

Время	Влажность
00:00	91%
01:30	91,50%
03:00	92%
04:30	93%
06:00	94%
07:30	91%
09:00	88%
10:30	83%
12:00	78%
13:30	83%
15:00	88%
16:30	91,50%
18:00	95%
19:30	96,50%
21:00	98%

Рисунок 3.18 – Граничні та проміжні значення часу та вологості

Обчислимо дані швидкості вітру (див. рис. 3.19 – 3.20).

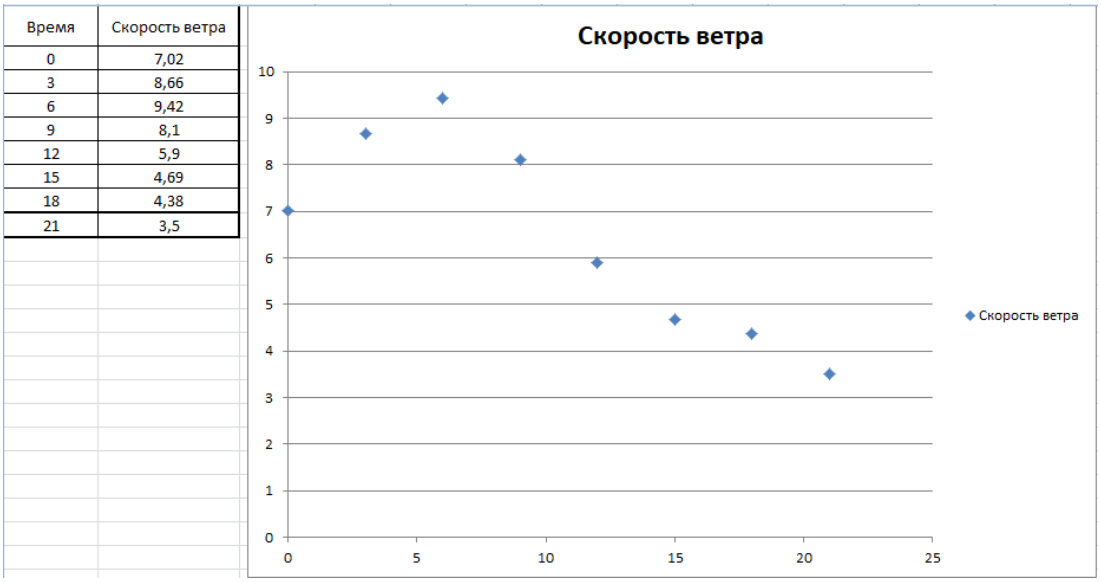


Рисунок 3.19 – Дані граничних значень часу та швидкості вітру

Проміжні дані часу

x = [1.5, 4.5, 7.5, 10.5, 13.5, 16.5, 19.5].

Проміжні дані швидкості вітру

y = [7.84, 9.04, 8.76, 7, 5.29, 4.53, 3.94].

Время	Скорость ветра
00:00	7,02m/s
01:30	7,84m/s
03:00	8,66m/s
04:30	9,04m/s
06:00	9,42m/s
07:30	8,76m/s
09:00	8,1m/s
10:30	7m/s
12:00	5,9m/s
13:30	5,29m/s
15:00	4,69m/s
16:30	4,53m/s
18:00	4,38m/s
19:30	3,94m/s
21:00	3,5m/s

Рисунок 3.20 – Граничні та проміжні значення часу та швидкості вітру

Обчислимо дані хмарності (див. рис. 3.21 – 3.22).

Время	Облачность
0	67
3	100
6	100
9	98
12	93
15	100
18	88
21	95

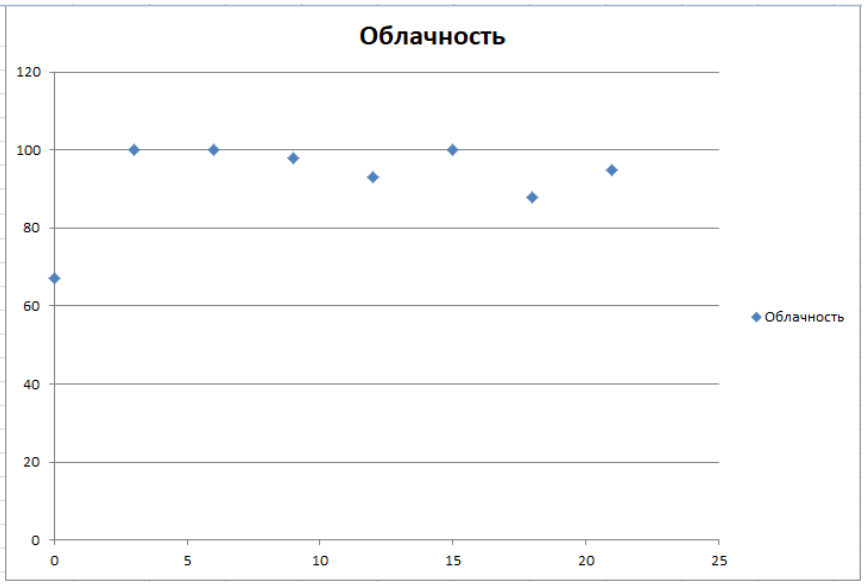


Рисунок 3.21 – Дані граничних значень часу та хмарності

Проміжні дані часу

x = [1.5, 4.5, 7.5, 10.5, 13.5, 16.5, 19.5].

Проміжні дані хмарності

y = [83.5, 100, 99, 95.5, 96.5, 94, 91.5].

Время	Облачность
00:00	67%
01:30	83,50%
03:00	100%
04:30	100%
06:00	100%
07:30	99%
09:00	98%
10:30	95,50%
12:00	93%
13:30	96,50%
15:00	100%
16:30	94%
18:00	88%
19:30	91,50%
21:00	95%

Рисунок 3.22 – Граничні та проміжні значення часу та хмарності

Обчислимо дані тиску (див. рис. 3.23 – 3.24).

Время	Давление
0	1010
3	1007
6	1006
9	1005
12	1005
15	1005
18	1006
21	1006

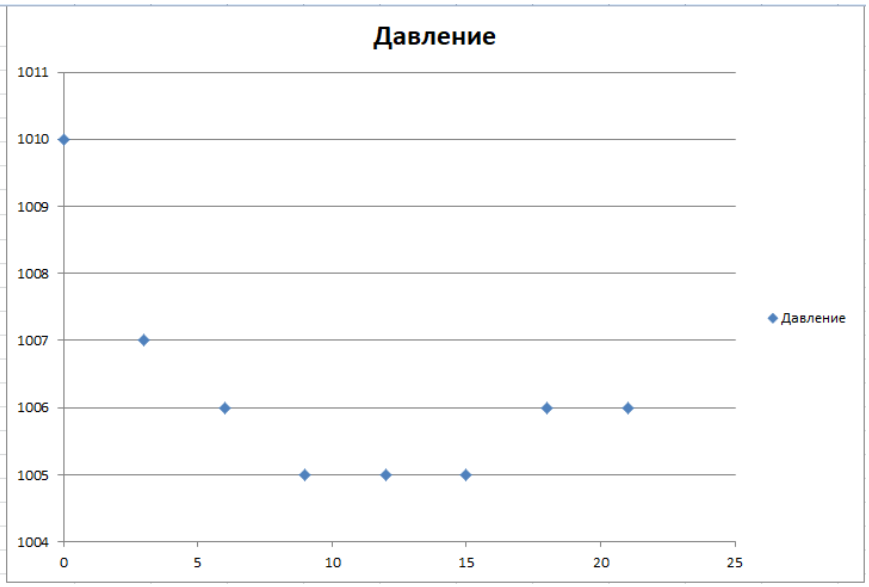


Рисунок 3.23 – Дані граничних значень часу та тиску

Проміжні дані часу

x = [1.5, 4.5, 7.5, 10.5, 13.5, 16.5, 19.5].

Проміжні дані тиску

y = [1008.5, 1006.5, 1005.5, 1005.0, 1005.0, 1005.5, 1006.0].

Время	Давление
00:00	1010 hPa
01:30	1008,5 hPa
03:00	1007 hPa
04:30	1006,5 hPa
06:00	1006 hPa
07:30	1005,5 hPa
09:00	1005 hPa
10:30	1005 hPa
12:00	1005 hPa
13:30	1005 hPa
15:00	1005 hPa
16:30	1005,5 hPa
18:00	1006 hPa
19:30	1006 hPa
21:00	1006 hPa

Рисунок 3.24 – Граничні та проміжні значення часу та тиску

Під час обчислення інтерполяції було отримано проміжні значення погодних даних, які доповнюють та уточнюють прогноз погоди.

3.4.2 SWOT-аналіз

SWOT-аналіз є одним із найпоширеніших методів стратегічного планування, при якому визначаються сильні, слабкі сторони, можливості та загрози проекту (див. табл. 3.1).

Завдання SWOT-аналізу – дати структурований опис ситуації, щодо якої потрібно ухвалити будь-яке рішення.

Висновки, зроблені на його основі, мають описовий характер, без рекомендацій та розміщення пріоритетів.

Таблиця 3.1 – SWOT-аналіз мобільного додатку

Сильні сторони	**Слабкі сторони**
• Простота у використанні • Чуйність • Швидка робота	• Не реалізовано можливість зміни значення таймера • Не реалізовано можливість очищення бази даних
Можливості	**Загрози**
• Збереження погодних даних кожні 30 секунд • Перегляд збережених погодних даних • Прогноз на поточний час та на 5 днів	• Прихід іншого розробника з подібним покращеним додатком • Додаток не має реклами, що може погано позначиться на фінансових успіхах

В результаті SWOT-аналізу було розглянуто сильні та слабкі сторони розробки мобільного додатку для прогнозу погоди. Особливістю цього додатку є можливість циклічного збереження погодних даних через певний час. Слабкі сторони по можливості необхідно мінімізувати, базуючись насамперед на сильних сторонах та потенційних можливостях розробки.

ВИСНОВКИ

Кваліфікаційна робота присвячена проектуванню та розробці Android-додатку. Використання інформаційних технологій широко використовується у народно-господарській діяльності людини. Це дозволило уможливити застосування мобільних пристроїв для щоденного та повсюдного володіння інформацією у питаннях прогнозу погоди.

У кваліфікаційній роботі створено мобільний додаток, який буде встановлений на смартфон із використанням операційної системи Android.

Метою дипломної роботи було створення мобільного додатку прогнозу погоди на основі даних GFS.

Структурно робота складається з трьох розділів.

Перший розділ був присвячений обгрунтуванню та аналізу теми та підбору завдання для її реалізації.

У другому розділі був виконаний підбір технологій, спрямованих на реалізацію завдання і перерахування функціональних вимог додатку.

У третій частині було виконано опис розробленого додатка. Також було зроблено дослідження інтерполяції погодних даних та SWOT-аналіз мобільного додатка.

Мобільний додаток прогнозу погоди на основі даних GFS – це ефективний інструмент у руках людини, у розробці якого приділено багато уваги не тільки технічній стороні, але й зручності у використанні.

ПЕРЕЛІК ПОСИЛАНЬ

1. Модель GFS. URL : https://www.meteorologiaenred.com/ru/modelo-gfs.html (дата звернення: 30.07.2021).

2. Який прогноз погоди обрати та чому. URL : https://forcesail.ru/meteorology/#.Yaz6QCpByUn (дата звернення: 02.08.2021).

3. Global Forecast System. URL : https://www.ncei.noaa.gov/products/weather-climate-models/global-forecast (дата звернення: 03.08.2021).

4. Metadata. URL : https://www.ncei.noaa.gov/resources/metadata (дата звернення: 05.08.2021).

5. Fact sheet: Supercomputing at ECMWF. URL : https://www.ecmwf.int/en/about/media-centre/focus/2021/fact-sheet-supercomputing-ecmwf (дата звернення: 05.08.2021).

6. OpenWeatherMap. URL : https://ru.wikipedia.org/wiki/OpenWeatherMap (дата звернення: 08.08.2021).

7. Що таке API. URL : https://wiki.merionet.ru/servernye-resheniya/24/chto-takoe-api-prostaya-statya-dlya-vashej-babushki/ (дата звернення: 12.08.2021).

8. Що таке API. URL : https://timeweb.com/ru/community/articles/chto-takoe-api (дата звернення: 14.08.2021).

9. SQLite – Вступ. URL : https://unetway.com/tutorial/sqlite (дата звернення: 15.08.2021).

10. Features of SQLite. URL : https://www.sqlite.org/features.html (дата звернення: 20.08.2021).

ДОДАТОК А

Вміст файлу activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".MainActivity"
    android:background="@drawable/background"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:textColor="@android:color/white"
        android:textSize="40sp"
        android:text="WeatherApp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical">

        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/etCity"
```

```
    android:layout_marginBottom="10dp"
    android:ems="10"
    android:hint="Введите город или страну"
    android:inputType="textPersonName" />
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/etCountry"
    android:layout_marginBottom="10dp"
    android:ems="10"
    android:hint="Введите код страны"
    android:inputType="textPersonName" />

<Button
    android:id="@+id/btnGet"
    android:layout_width="match_parent"
    android:layout_height="26dp"
    android:layout_marginBottom="10dp"
    android:background="#0070c7"
    android:onClick="getWeatherDetails"
    android:text="Текущая погода"
    android:textColor="@android:color/white"
    app:backgroundTint="@null" />

<Button
    android:layout_width="match_parent"
    android:layout_height="26dp"
    android:id="@+id/btnGetfivedays"
    android:layout_marginBottom="10dp"
    android:background="#0070c7"
    android:textColor="@android:color/white"
    android:onClick="getWeatherDetailsonfivedays"
    android:text="Прогноз на 5 дней"
    app:backgroundTint="@null" />
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="26dp"
    android:id="@+id/btnStartTimer"
    android:layout_marginBottom="10dp"
    android:background="#0070c7"
    android:textColor="@android:color/white"
    android:onClick="StartTimer"
    android:text="ЗАПУСК ТАЙМЕРА"
    app:backgroundTint="@null" />

<Button
    android:layout_width="match_parent"
    android:layout_height="26dp"
    android:id="@+id/btnStopTimer"
    android:layout_marginBottom="10dp"
    android:background="#0070c7"
    android:textColor="@android:color/white"
    android:onClick="StopTimer"
    android:text="СТОП ТАЙМЕР"
    app:backgroundTint="@null" />

<Button
    android:layout_width="match_parent"
    android:layout_height="26dp"
    android:id="@+id/btnArchive"
    android:layout_marginBottom="10dp"
    android:background="#0070c7"
    android:textColor="@android:color/white"
    android:onClick="Archive"
    android:text="АРХИВ ПОГОДЫ"
    app:backgroundTint="@null" />


<ScrollView
```

```
        android:layout_width="match_parent"
        android:layout_height="219dp"
        android:background="#000000">

        <TextView
            android:id="@+id/tvResult"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:fontFamily="sans-serif-medium"
            android:shadowColor="@color/text_shadow"
            android:shadowDx="5"
            android:shadowDy="5"
            android:shadowRadius="2"
            android:textSize="20dp" />


    </ScrollView>


</LinearLayout>


</LinearLayout>
```

ДОДАТОК Б

Вміст файлу activity_archive_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".ArchiveMenuActivity"
    android:background="@drawable/background"
    android:orientation="vertical"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical">

        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="464dp"
            android:background="#000000">

            <TextView
                android:id="@+id/tvArchive"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:fontFamily="sans-serif-medium"
                android:shadowColor="@color/text_shadow"
                android:shadowDx="5"
```

```
            android:shadowDy="5"
            android:shadowRadius="2"
            android:textSize="20dp" />


    </ScrollView>

<Button
    android:id="@+id/btnTEMP"
    android:layout_width="match_parent"
    android:layout_height="26dp"
    android:layout_marginBottom="10dp"
    android:background="#0070c7"
    android:onClick="TempArchive"
    android:text="ТЕМПЕРАТУРА"
    android:textColor="@android:color/white"
    app:backgroundTint="@null" />

<Button
    android:id="@+id/btnHUMIDITY"
    android:layout_width="match_parent"
    android:layout_height="26dp"
    android:layout_marginBottom="10dp"
    android:background="#0070c7"
    android:onClick="HumArchive"
    android:text="ВЛАЖНОСТЬ"
    android:textColor="@android:color/white"
    app:backgroundTint="@null" />

<Button
    android:id="@+id/btnWIND"
    android:layout_width="match_parent"
    android:layout_height="26dp"
    android:layout_marginBottom="10dp"
    android:background="#0070c7"
```

```
        android:onClick="WindArchive"
        android:text="СКОРОСТЬ ВЕТРА"
        android:textColor="@android:color/white"
        app:backgroundTint="@null" />

    <Button
        android:id="@+id/btnCLOUDS"
        android:layout_width="match_parent"
        android:layout_height="26dp"
        android:layout_marginBottom="10dp"
        android:background="#0070c7"
        android:onClick="CloudsArchive"
        android:text="ОБЛАЧНОСТЬ"
        android:textColor="@android:color/white"
        app:backgroundTint="@null" />

    <Button
        android:id="@+id/btnPRESSURE"
        android:layout_width="match_parent"
        android:layout_height="26dp"
        android:layout_marginBottom="10dp"
        android:background="#0070c7"
        android:onClick="PressArchive"
        android:text="ДАВЛЕНИЕ"
        android:textColor="@android:color/white"
        app:backgroundTint="@null" />



    </LinearLayout>

</LinearLayout>
```

ДОДАТОК В

Посилання на Git

https://github.com/Glaynder/Diplom