МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____Скарга-Бандурова І.С.

«____»_____ 20__ р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Клієнтський додаток для мобільної платформи Android

Освітньо-кваліфікаційний рівень "бакалавр"
Спеціальність 6.050102 – "Комп'ютерна інженерія"

Керівник проекту:	_____ (підпис)	Щербакова М.Є. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я.О. (ініціали, прізвище)
Здобувач вищої освіти:	_____ (підпис)	Стріщенко Т.В. (ініціали, прізвище)
Група:		КІ-14з

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки _____
(шифр і назва)
Спеціальність 6.050102 – "Комп'ютерна інженерія"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Стріщенко Тетяни Вікторівни
(прізвище, ім'я, по батькові)

1. Тема роботи Клієнтський додаток для мобільної платформи Android

керівник проекту (роботи) Щербакова М.Є., к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від _____ _____
2. Термін подання студентом роботи 11.06.2018
3. Вихідні дані до роботи Матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області та постановка задачі; вибір засобів для розробки; розробка клієнтського додатку; охорона праці та безпека в надзвичайних ситуаціях.

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	ст. викл. Критська Я.О.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Ознайомлення з предметною галуззю	30.04.18 – 05.05.18	
2	Аналіз існуючих аналогів	06.05.18 – 10.05.18	
3	Вибір засобів для розробки	11.05.18 – 19.05.18	
4	Розробка клієнтського додатку	20.05.18 – 05.06.18	
5	Розробка розділу «Охорона праці та	06.06.18 – 07.06.18	
	безпека в надзвичайних ситуаціях»		
6	Оформлення пояснювальної записки	08.06.18 – 11.06.18	

Студент _____ Стріщенко Т.В.
(підпис) (прізвище та ініціали)

Керівник _____ Щербакова М.Є.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту (роботи) бакалавра складається з чотирьох розділів, містить 48 сторінок, 15 рисунків, 3 таблиці, 12 джерел.

Об’єкт розробки: клієнтський додаток для відображення погоди.

Мета роботи: створення мобільного додатку для ОС Android, що отримує дані від онлайн-сервера погоди, і дозволяє користувачам переглядати прогноз погоди зі зручним інтерфейсом на своєму смартфоні, уникаючи реклами.

В дипломному проекті:

– проаналізовані переваги і недоліки існуючих програм відображення погоди;

– розглянуті засоби розробки клієнтських програм для мобільних платформ, а також можливості використання погодних онлайн-серверів;

– розроблений мобільний клієнтський додаток для відображення погодних умов.

Отримані наступні результати: мобільний додаток з інформацією про погоду у визначеному населеному пункті.

Практичне значення, галузь застосування роботи: додаток може бути використаний для отримання даних про погоду на пристроях з ОС Android.

ANDROID, ANDROID STUDIO, MYSQL, ОПЕРАЦІЙНА СИСТЕМА, ПОГОДА, ПРОГНОЗ ПОГОДИ.

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А, м. Сєвєродонецьк, 93400.

ЗМІСТ

ВСТУП

Мобільні пристрої - невід'ємна частина життя сучасної людини. Це комунікації між людьми, прослуховування улюбленої музики, перегляд аудіовізуальної інформації, блокнот, і ще безліч функцій. Мобільний смартфон - це копія комп'ютера, яку завжди можна мати при собі. В наш час, найпоширенішою є операційна система (ОС) Android.

Android підтримує велику кількість пристроїв від різних виробників. я

Чимало мобільних додатків використовують архітектуру клієнт-сервер. Сервер представляє собою програму, що працює на віддаленому комп'ютері, і реалізує функціонал «спілкування» з додатками-клієнтами (слухає запити, розпізнає передані параметри і значення, коректно відповідає на них). Клієнт - програма на мобільному пристрої, яка вміє формувати зрозумілий серверу запит і читати отриману відповідь. Інтерфейс взаємодії - це формат і спосіб передачі та отримання запитів і відповідей обома сторонами. Взаємодія клієнта і сервера може відбуватися шляхом HTTP-запитів і JSON-відповідей. JSON (JavaScript Object Notation) - текстовий формат обміну даними, заснований на JavaScript. JSON-текст представляє собою або набір пар ключ: значення, або впорядкований набір значень. Клієнт-серверні мобільні додатки - це, наприклад, інтернет-месенджери, додатки для туристичних компаній, додатки для електронної комерції.

Основним завданням роботи є створення клієнтського додатка на базі ОС Android для відображення даних про погоду у вказаному населеному пункті.

1 МОБІЛЬНІ ДОДАТКИ ДЛЯ ОС ANDROID

1.1 Основні поняття

Android - операційна система для смартфонів, інтернет - планшетів, електронних книг, цифрових програвачів, наручних годинників, ігрових нетбуків, смартбуків, окулярів Google, телевізорів та інших пристроїв.

Android SDK - середовище розробки додатків для однойменної операційної системи.

Hosting - послуга з надання ресурсів для розміщення інформації на сервері, що постійно перебуває в мережі (звичайний Інтернет).

Серіалізация (в програмуванні) - процес перекладу будь - якої структури даних в послідовність бітів.

JSON (JavaScriptObjectNotation) - простий формат обміну даними, зручний для читання і написання, як людиною, так і комп'ютером.

База даних - представлена в об'єктивній формі сукупність самостійних матеріалів (статей, розрахунків, нормативних актів, судових рішень та інших подібних матеріалів), систематизованих таким чином, щоб ці матеріали могли бути знайдені і оброблені за допомогою електронної обчислювальної машини (ЕОМ).

Карти Google (GoogleMaps) - набір додатків, побудованих на основі безкоштовного картографічного сервісу і технології, що надаються компанією Google створених в 2005 році.

Геокодування - процес призначення географічних ідентифікаторів (таких як географічні координати, виражені у вигляді широти і довготи) об'єктів карти і записів даних.

Геолокація - визначення географічного розташування інтернет-користува Java-являє собою мову програмування і платформу обчислень, яка була вперше випущена SunMicrosystems в 1995 р.

JavaDevelopmentKit (скорочено JDK) - безкоштовно поширюваний компанією OracleCorporation (раніше SunMicrosystems) комплект розробника додатків на мові Java, що включає в себе компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різні утиліти і виконавчу систему Java (JRE) .

Eclipse- вільне інтегроване середовище розробки модульних россплатформенних додатків. Розвивається і підтримується Eclipse

Foundation.EmbarcaderoJBuilder - пропріетарне середовище розробки на мові JavaкомпаніїEmbarcadero, входить до складу EmbarcaderoRADStudio. середовище розроблено відповідно до концепції візуального програмування.

JDeveloper – безкоштовне інтегроване середовище розробки програмного забезпечення, розроблена корпорацією Oracle. Надає можливість для розробки на мовах програмування Java, JavaScript, BPEL, PHP, SQL, PL / SQL і на мовах розмітки HTML, XML. JDeveloper покриває весь життєвий цикл розробки програмного забезпечення від проектування, кодування, налагодження, оптимізації та профілювання до його разгортування.

API (інтерфейс програмування додатків, інтерфейс прикладного програмування) (англ.Application programming interface) —набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) або операційною системою для використання у зовнішніх програмних продуктах. Використовується програмістами при написанні всіляких додатків.

Система управління базами даних (СКБД) – сукупність програмних і лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням і використанням баз даних.

1.2 Операційна система Android

Android - операційна система для смартфонів, планшетних комп'ютерів, електронних книг, цифрових програвачів, "розумних" наручних годинників, ігрових приставок, нетбуків, смартбуків, окулярів Google, телевізорів, ситем автоматичного керування автомобілем та інших пристроїв. ОС заснована на ядрі Linux і власної реалізації віртуальної машини Java від Google. Спочатку розроблялася компанією AndroidInc., Яку в 2005 році купила Google.

Згодом Google ініціювала створення альянсу OpenHandsetAlliance (OHA), який зараз займається підтримкою і подальшим розвитком платформи. Android дозволяє створювати Java - додатки, що керують пристроєм через розроблені Google бібліотеки. AndroidNativeDevelopmentKit дозволяє перенести (але не налагоджувати) бібліотеки і компоненти додатків, написані на Сі та інших мовах. ОС Androidвстановлена на 86% смартфонів (2014 року).

1.3 Ключові особливості Android

Android є найпоширенішою операційною системою (ОС) для мобільних пристроїв - телефонів і планшетів. Дана система має безліч характерних рис, які роблять її популярною і привабливою для великої кількості користувачів по всьому світу.

Операційна система Android є невимогливою і здатна працювати на різнихконфігураціях. Саме тому більшість світових виробників оснащують свої пристрої даною ОС, оскільки інші програмні продукти призначені для окремих апаратів, що відповідають певній специфікації.

Така гнучкість Android пов'язана з тим, що система побудована на ядрі Linux, що має відкритий програмний код, що дає необмежені можливості розробникам.

Android може бути запущений на пристроях, що мають об'єм оперативної пам'яті менше 256 Мб. Найбільш нові версії системи вимагають 512 Мб оперативної пам'яті, що також є невеликим значенням для сучасних апаратів.

Система не вимагає наявності високопродуктивного процесора і може працювати на пристроях, оснащених ядром з частотою 600 МГц.

Операційна система дає можливість установки додатків з офіційного репозиторію Google, який надає найбільшу в світі базу програм. Це пов'язано з тим, що кожен розробник може самостійно написати будь-яку програму для апарату і розмістити її в магазині.

Можливість також реалізована завдяки відкритості операційної системи. Варто відзначити, що додатки на пристрої під управлінням Android можуть бути встановлені як безпосередньо з телефону або планшета, так і через комп'ютер шляхом завантаження файлу .apk і його подальшої установки на апараті.

Відмінною особливістю Android є його інтегрованість з сервісами Google -Gmail, Hangouts, VoiceSearch і т.п. На Android офіційно реалізована підтримка Chrome, що дозволяє синхронізувати відкриваються в браузері вкладки на смартфоні з комп'ютерним браузером.

Наприклад, ви можете почати перегляд сторінок з вашого телефону і при бажанні продовжити вивчати інформацію, відкрившицю ж вкладку на комп'ютері, не вдаючись до допомоги повторного пошуку

«Андроїд» має досить простий і інтуїтивно зрозумілий інтерфейс. Всі потрібні програми розміщуються одночасно на головному екрані і в меню апарату, яке викликається натисканням на центральну сенсорну клавішу або відповідну кнопку на екрані. Всі настройки розташовуються в секції «Налаштування», а кожна дія користувача пояснюється коментарями і підказками при першому запуску апарату.

Операційна система швидко реагує на натискання користувача і проводить інсталяцію і завантаження потрібних програм і файлів зі швидкістю, яка не програє іншим сучасним мобільним ОС.

1.4 Загальна схема роботи додатка Android

Додатки для Android в своїй роботі використовують вікна (аналогічноWindows), проте в даній системі вищевказані вікна носять іншу назву - Activity. Як і в Windows, кожне вікно має свій життєвий цикл і свої особливості. При створенні нового вікна викликається метод onCreate, при розробці даний метод переопределяется і в ньому відбувається ініціалізація програми та його компонентів. Далі викликаються методи onStart (і onResume. Обидва методи викликаються перед Відображенням вікна при його створенні, або відновленні (при перемиканні з іншої програми, при розгортанні згорнутого додатки і тп). При згортанні викликаються методи onPause і onStop.

При закритті програми і вікна викликається onDestory (), в данному методі можна зберегти призначені для користувача дані і параметри. Повний опис та послідовність виклику методів можна знайти на офіційному сайті.

Загальна схема життєвого циклу додатку для Android представлена на рис. 1.1.

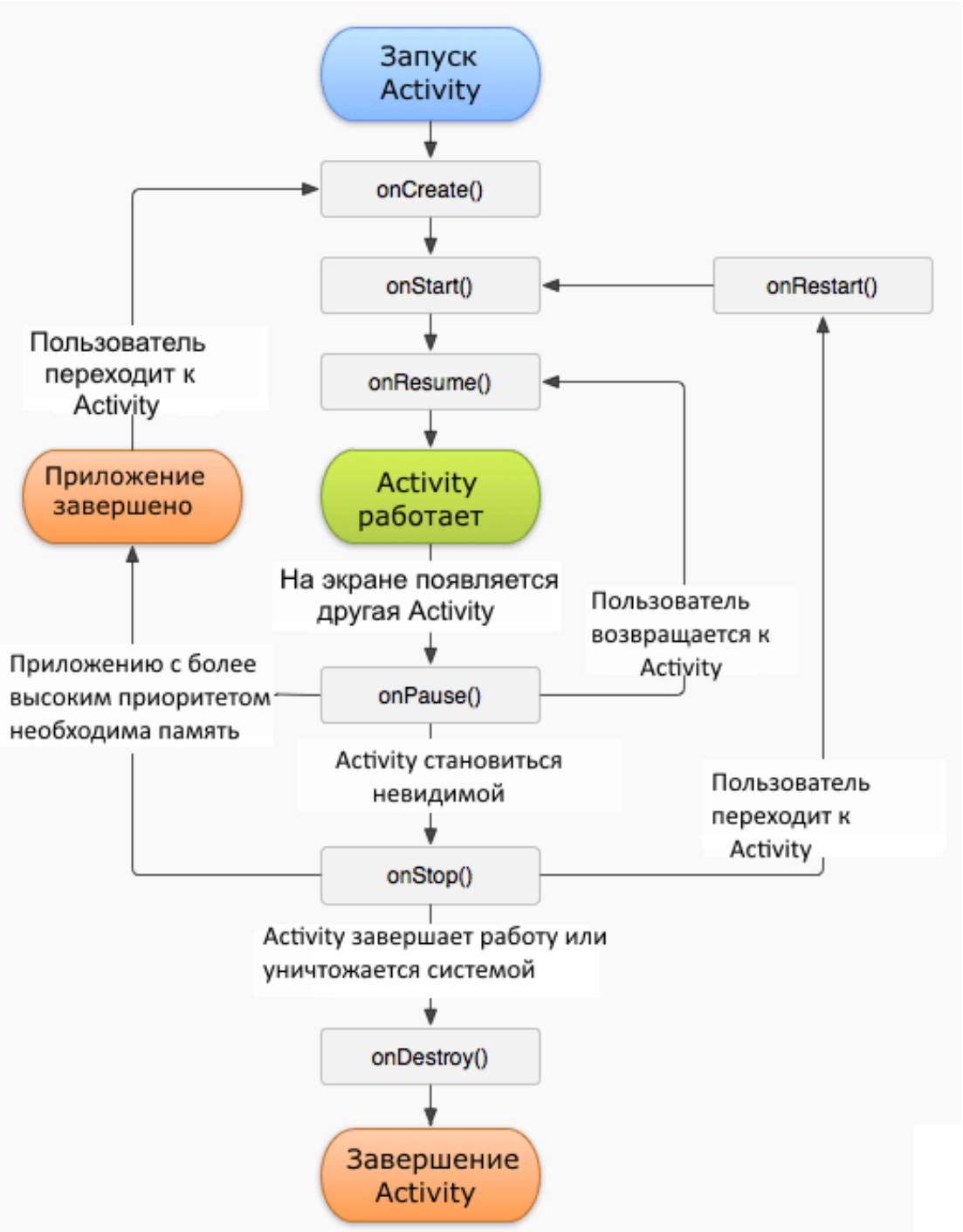


Рисунок 1.1 - Життєвий цикл додатку для системи під управлінням Android

1.5 Постановка завдання на розробку клієнтської програми для мобільних пристроїв

Потрібно розробити клієнт – серверний додаток на базі ОС Android для відображення поточних погодних умов у вказаному населеному пункті. Додаток має використовувати погодний сервер для отримання даних.

Основні функції додатка:

- відображення у зручному форматі температури, атмосферного тиску, опадів, тощо;
- можливість пошуку населеного пункту за його назвою
- зберігання інформації про останній населений пункт до наступного запуску програми
- отримання даних і відображення погодних умов на наступні декілька діб;
- відображення карт температури, швидкості вітру, опадів.

Висновки до першого розділу

Android є найпоширенішою операційною системою для мобільних платформ. Створення додатків під ОС Android значно простіше, ніж для Windows Phone або iOS. Існує багато програмних бібліотек компонетов, що покращують якість програм для ОС Android.

2 ЗАСОБИ РОЗРОБКИ МОБІЛЬНИХ КЛІЄНТСЬКИХ ДОДАТКІВ

2.1 Інструменти розробки програм для мобільних платформ

Існує кілька найпоширеніших операційних систем для смартфонів, такі як iOS, Android, WindowsPhone, BlackBerry, Simbian.

Для створення мого програми обраний Android, тому що:

– Android -операційна система з відкритим вихідним кодом.

– Поширеність ОС Android, вказана на рис. 2.1.

– Доступ до розробки будь-якому користувачеві.

– Абсолютно безкоштовна для розробки.

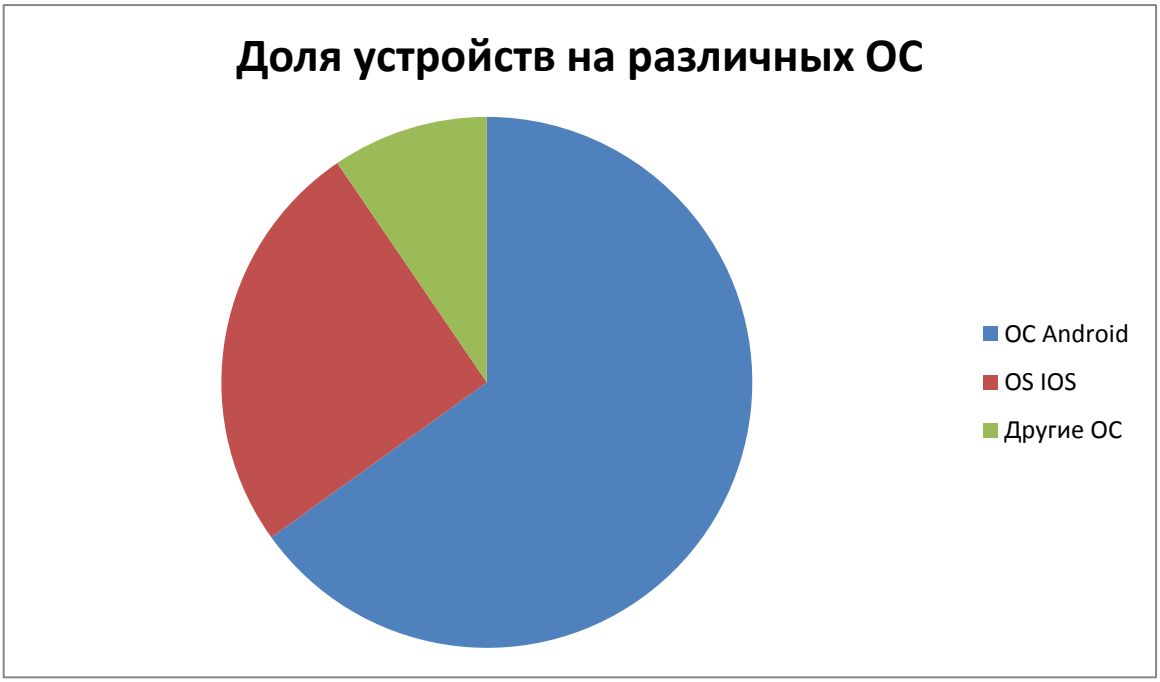


Рисунок 2.1 - Частка пристроїв на різних ОС

Для створення програми на ОС Android, можуть бути обрані різні середовища розробки, такі як, Eclipse, Embarcadero JBuilder, JDeveloper і т.д, але для роботи обрана AndroidStudio від компанії Google. Причиною вибору став

простий спосіб підключення Google-карт, а так само зручність інтерфейсу і швидкодії програми.

Android Studio - це інтегроване середовище розробки (IDE) для роботи з платформою Android, анонсована 16.05.2013 року на конференції Google I/O.

IDE перебувала у вільному доступі починаючи з версії 0.1, опублікованій в травні 2013, а потім перейшла в стадію бета - тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року, тоді ж припинилася підтримка плагіна Android Development Tools (ADT) для Eclipse.

AndroidStudio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, офіційне засіб розробки Android додатків. Дане середовище розробки доступна для Windows, OS X і Linux.

За рахунок своїй лаконічності в порівнянні з XML, формат JSON може бути більш підходящим для сериализации складних структур. Якщо говорити про веб - додатках, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP - інтерфейси).

Оскільки формат JSON є підмножиною синтаксису мови JavaScript, то він може бути швидко десеріалізован вбудованою функцією eval (). Крім того, можлива вставка цілком працездатних JavaScript - функцій. У мові PHP, починаючи з версії 5. 2.0, підтримка JSON включена в ядро у вигляді функцій json_decode () і json_encode (), які самі перетворять типи даних JSON в відповідні типи PHP і навпаки.

Як значення в JSON можуть бути використані:

Об'єкт - це неврегульована безліч пар ключ: значення, заключѐнное в фігурні дужки «{}». Ключ описується рядком, між ним і значенням стоїть символ «:». Пари ключ - значення відокремлюються один від одного комами.

Масив (одновимірний) - це впорядкована множина значень. Масив полягає в квадратні дужки «[]». Значення розділяються комами.

Літерали true, false і null.

Рядок - це впорядкована множина з нуля або більше символів юникода, укладену в подвійні лапки. Символи можуть бути вказані з використанням escape - послідовностей, що починаються з зворотної косої межі «\» (підтримуються варіанти \ ", \\, \ /, \ t, \ n, \ r, \ f і \ b), або записані шістнадцятковим кодом в кодуванні UTF - 8 у вигляді \ uFFFF.

Карти Google (Googlemaps) - набір додатків, побудованих на основі безкоштовного картографічного сервісу і технології, що надаються компанією Google. Створено в 2005 році. Сервіс являє собою карту та супутникові знімки планети Земля.

За допомогою Google Maps Android API можна додавати в свій додаток карти на основі даних Google Карт. Цей API - інтерфейс автоматично управляє доступом до серверів Google Карт, завантаженням даних, відображенням карт і реакцією на жести, виконувані на картах.

Крім того, ви можете використовувати виклики API, щоб додавати маркери, багатокутники і накладення до основної картки, а також змінювати спосіб відображення певної області на карті. Ці об'єкти надають додаткову інформацію про місця на карті і забезпечують можливості взаємодії користувачів з картою. API дозволяє додавати на карту такі графічні елементи:

– значки, пов'язані з певними місцями на карті (маркери);

– набори сегментів ліній (ламані лінії);

– замкнуті сегменти ліній (багатокутники);

– растрові графічні елементи, пов'язані з певними місцями на карті (наземні накладення);

– набори зображень, які відображаються поверх листів основної картки (мозаїчні накладення).

Технологія Java включає в себе мову програмування Java, засоби трансляції вихідного тексту програми вихідного коду в спеціальну форму, придатну для виконання комп'ютером, і кошти виконання Java-програм на різних платформах, тобто в різних операційних системах і на різному

апаратному забезпеченні. Основна особливість JAVA-технології в тому, що перетворена на етапі трансляції в спеціальний код Java-програма повністю "машінонезавісіма".

Якщо виконуваний код, отриманий з програм на інших распространѐнних мовами, зазвичай не придатний для виконання комп'ютером "іншої платформи", то до виконуваного коду Java таке обмеження не стосується. Правда, необхідно, щоб для "цільової платформи" була реалізація так званої Java-машини-середовища виконання JAVA-програм.

JAVA-технології, активно просуває компанією SUN, набули широкого поширення (далеко не тільки в web-рішення). А платформонезавісимость Java, що дозволила інтегрувати засоби виконання Java-програм в браузери, що працюють в самих різних операціоннихсістемах, визначила поширення Java в якості елемента Web-технологій.

JAVA використовується для створення складних інтерактивних елементів, пов'язаних з web-сайтом. Наприклад, на Java реалізуються складні інструменти для роботи з базами даних, размещѐннимі в Web. Або графічні інтерфейси, що вимагають виведення складних інтерактивних елементів. І, звичайно, багато іншого, від мережевих шахових програм до засобів редагування звукових файлів. Для зберігання даних використовують різні СУБД, такі як, SQLite, MySQL, Oracle, але для роботи обрана MySQL, так як:

– підтримання хостингом;

– гнучкість, яка забезпечується підтримкою великої кількості типів таблиць;

– продукт класу Open Source (відкриті вихідні тексти), який можна отримати безкоштовно.

Для відображення об'єктів на карті використовуються різні картографічні сервіси такі як, Яндекс карти, Googlemap, NASA World Wind, EarthDesk, але для роботи обраний Googlemap, тому що:

– прості в з'єднанні з AndroidStudio, так як Googlecоздатель обох продуктів.

– швидке знаходження адреси.

2.2 Огляд програмних продуктів для відображення погоди

На даний момент додатків з відображенням погодних умов дуже багато. Нижче наведені приклади деяких з них і описані їх переваги та недоліки.

2.2.1 Додаток Bright Weather

Базова версія програми є безкоштовною і включає рекламу. Версія Pro, без реклами, варто 3.5 USD. Заявлена підтримка Android 4.0. Споживання оперативної пам'яті - 165 МБ.

При першому включенні додаток пропонує вибрати відповідні одиниці виміру і додаткові функції: повідомлення, показ температури в рядку стану, активація екрану блокування. Пізніше зміни вносяться на прихованої панелі зліва, так як пункт налаштувань фактично відсутня.

Початковий екран містить інформацію про поточні погодні умови, включаючи погодинної і щоденний прогноз. Так само можна переглянути додаткові дані: вологість, опади, температура вітру, час сходу сонця і заходу, тиск і% хмарності. Крім того програма дозволяє подивитися розширений прогноз для кожного наступного дня.

На рис. 2.2 зображено роботу програми Bright Weather.

Рисунок 2.2 - Робота програми Bright Weather

Основні переваги:

– простота подачі інформації;

– наявність додаткових метіоусловій;

– погодинний та щоденний прогноз;

Основні недоліки:

– наявність реклами;

– слабка оптимізація і велике споживання ОЗУ;

– висока вартість за оновлення до Pro версії;

– некоректне переклад і локалізація додатки;

– не підтримується оновлення погоди закінченню певного часового періоду.

2.2.2 Додаток Daily & Hourly Weather Clock Widget

Погодний додаток завантажується безкоштовно, але містить агресивну рекламу. Споживання ОЗУ - 312 МБ. Підтримуються пристрої з Android 4.1 або вище. На рис. 2.3 зображено роботу програми Daily & Hourly Weather Clock Widget

Рисунок 2.3 - Робота програми Daily & Hourly Weather Clock Widget

На стартовій сторінці, вгорі, відображається поточна температура доданого міста. Нижче містяться додаткові дані метіоусловій: вітер, важливість, видимість, тиск, індекс ультрафіолетового випромінювання і час сходу / заходу сонця. Більш детальна інформація повідомить денний час і фазу місяця посредствам зображення.

Так само на головній сторінці міститься інформація мінімального / максимального значення температури, на сьогоднішній і завтрашній день.

Погодинний прогноз і на 14 днів вперед, з відповідним графіком зміни температури.

Налаштування дозволяють задати формат часу, одиниці виміру, інтервал поновлення погодних даних. Ще задати колір фону, який можна застосувати тільки до розділу налаштувань.

Основні переваги:

- погодинний прогноз і на 14 днів вперед;
- показ метіоусловій;
- можливість задати інтервал поновлення погоди;

Основні недоліки:

- наявність реклами;
- високе споживання оперативної пам'яті;
- мало корисних налаштувань;
- некоректна зміна графічного оформлення.

2.2.3 Погода, радар і віджет

Погодний додаток безкоштовне, але напхане рекламою, за відключення якої доведеться заплатити 3.86 USD. Перед встановленням зверніть увагу Android не нижче версії 4.1. Споживання оперативки - 169 МБ.

Крім співзвучної назви з першим представником даної збірки, так само схожий інтерфейс і функціонал. Назва розробників відрізняються, але не виключено, що створенням займається одна контора.

На рис. 2.4 зображено роботу програми «Погода, радар і віджет»

Рисунок 2.4 - Робота програми Погода, радар і віджет

Основні переваги:

– додаток не перевантажено;
– буде відображено додаткові метіоусловія;
– є погодинний і щоденний прогноз;

Основні недоліки:

– присутній реклама;
– велике споживання оперативної пам'яті;
– дорога версія без реклами;
– місцями відсутній переклад;
– не можна задати інтервал поновлення температури.

Висновки до другого розділу

Безумовно, розглянуті вище програми досить опрацьовані і професійні, але вони мають суттєві недоліки. Розробка власного додатку дасть можливість переглядати прогноз погоди без реклами, у стриманому зовнішньому вигляді.

3 РОЗРОБКА КЛІЄНТСЬКОГО ДОДАТКУ ВІДОБРАЖЕННЯ ПОГОДИ

Метою дипломного проекту є створення Android-додатку, що дозволяє ознайомити користувачів з прогнозом погоди. Інтерфейс програми зроблений дуже зручним і простим для користувачів (рис. 3.1).

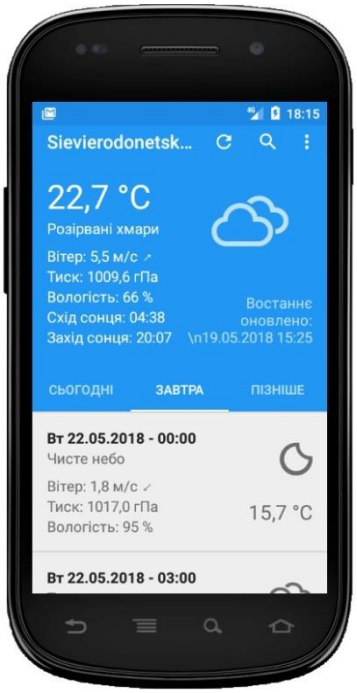


Рисунок 3.1 - Інтерфейс програми

З допомогою цієї програми користувачі можуть з легкістю ознайомитися з прогнозом погоди. Основні можливості додатку: візуалізація інформації про поточну погоду та погоду на наступні декілька днів; карти опадів, швидкості вітру, температури, а також графіки, що показують зміну атмосферного тиску, рівня опадів, температури (рис. 3.2).

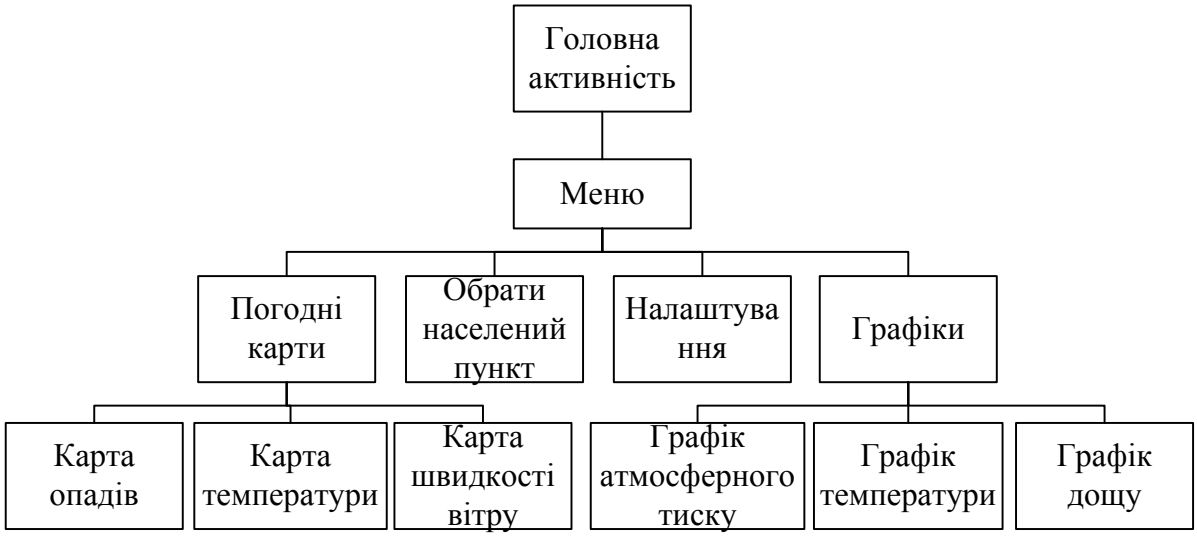


Рисунок 3.2 – Структурна схема додатку

Розроблений клієнтський додаток отримує дані від погодного сервісу за допомогою API-функцій, що надаються сервісом. Програмний інтерфейс програми API (application programming interface) представляє собою набір готових класів і функцій, що надаються серверним додатком для використання в зовнішніх (зокрема, клієнтських) програмних продуктах.

3.1 Отримання відомостей від сервісу погоди

Розроблений додаток отримує відомості про погоду від сервісу OpenWeatherMap (електронна адреса https://openweathermap.org). Це онлайн сервіс, який надає API для доступу до даних про поточну погоду та її прогнозів для web-сервісів і мобільних додатків. Архівні дані за минулі дні доступні тільки на комерційній основі. Як джерело даних використовуються офіційні метеорологічні служби, дані з метеостанцій аеропортів, і дані з приватних метеостанцій [1].

Інформація обробляється OpenWeatherMap (OWM), після чого на основі отриманих даних будується прогноз погоди і погодні карти, наприклад, карти

хмарності та опадів. Основною ідеєю сервісу OWM є використання приватних погодних станцій, які допомагають підвищити точність початкової погодної інформації і, як наслідок, точність прогнозів погоди.

На вході сервіс отримує дані від погодних станцій (рис. 3.3), а також прогнози метеорологічних служб і наукових лабораторій. Ці дані зберігаються в базі даних OWM, а після обробки за допомогою математичних алгоритмів вони перетворюються в інтерпольовані дані про поточну погоду в будь-якій точці світу, а також в карти з погодними явищами. До всіх даних про погоду надається API, включаючи карти з погодними явищами.

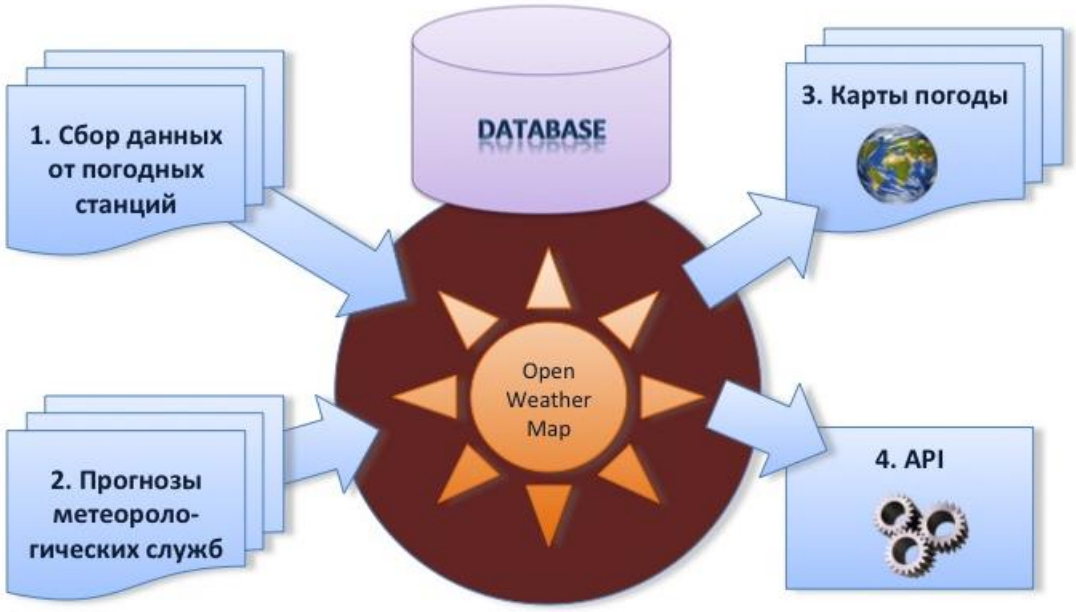


Рисунок 3.3 - Обмін даними в сервісі OpenWeatherMap

Для отримання доступу до сервісу погоди була пройдена процедура реєстрації на сайті OpenWeatherMap.org. Реєстрація потрібна для отримання рядка для ідентифікації користувача App Id, що складається з набору букв і цифр.

Щоб знайти населений пункт в базі даних, формується такий рядок запиту:

```
https://api.openweathermap.org/data/2.5/forecast?q=" +
URLEncoder.encode(sp.getString("city", Constants.DEFAULT_CITY),
"UTF-8") + "&lang="+ language +"&mode=json&appid=" + apiKey);
```

де вказується потрібне місто, мова та свій ідентифікатор користувача. У відповідь на сформований запит розроблений клієнтський додаток отримує пакет в форматі JSON. Необхідно розібрати пакет і отримати потрібні значення за назвами полів:

```
JSONObject main = reader.getJSONObject("main");
todayWeather.setTemperature(main.getString("temp"));
todayWeather.setDescription(reader.getJSONArray("weather").getJSON
Object(0)
.getString("description"));
```

3.2 Основні функції головної активності додатка

Функція onCreate() задає початкову установку параметрів при ініціалізації активності:

```
protected void onCreate(Bundle savedInstanceState) {
...
todayTemperature = (TextView)
findViewById(R.id.todayTemperature);
todayWind = (TextView) findViewById(R.id.todayWind);
todayPressure = (TextView) findViewById(R.id.todayPressure);
todayHumidity = (TextView) findViewById(R.id.todayHumidity);
}
```

Також в цій функції задаються налаштування програми в об'єкті класу android.content.SharedPreferences. Дані зберігаються у вигляді пар ключ/значення. Для будь-якого конкретного набору налаштувань Android зберігає тільки один об'єкт цього класу з тим, щоб зберегти цілісність даних. Загальні настройки зберігаються у внутрішній пам'яті та видаляються при

деінсталяції програми. У головній активності в настройках зберігається інформація про обране місто, про час останнього оновлення даних, про мову інтерфейсу програми.

3.3 Пошук відомостей про населений пункт

Якщо в меню обраний значок пошуку, то створюється і виводиться на екран діалогове вікно AlertDialog. Воно є розширенням класу Dialog і використовується, коли потрібен діалог з кнопками «Так» і «Ні» (рис. 3.4).

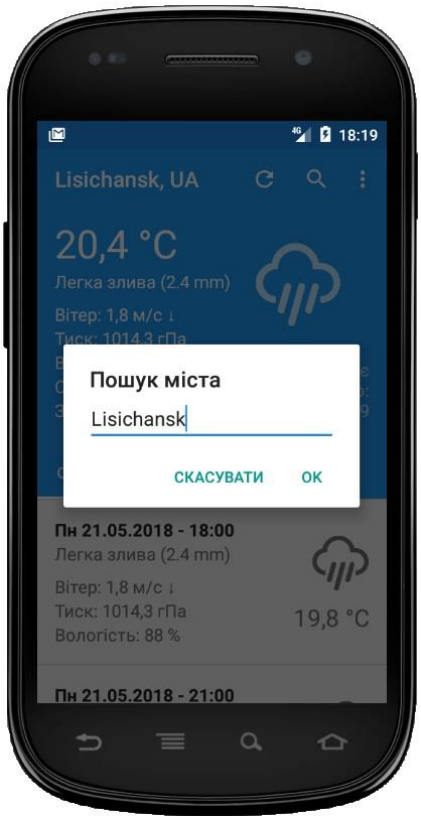


Рисунок 3.4 - Пошук міста

Спочатку був створений об'єкт класу AlertDialog.Builder, якому в якості параметра передане посилання на головну активність додатку, потім задається заголовок для створюваного діалогу (метод setTitle ()), і для відображення вікна викликається метод show():

```
AlertDialog.Builder alert = new AlertDialog.Builder(this);
alert.setTitle(this.getString(R.string.search_title));
alert.show();
```

Сама обробка натискання кнопок всередині діалогового вікна задається всередині методів setPositiveButton() і setNegativeButton(), які приймають в якості параметрів напис для кнопки і інтерфейс DialogInterface. OnClickListener, що визначає дію при натисканні:

```
     alert.setPositiveButton(R.string.dialog_ok, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
        String result = input.getText().toString();
        if (!result.isEmpty()) {
            saveLocation(result);
        } } });
alert.setNegativeButton(R.string.dialog_cancel, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
        // Cancelled
    } });
```

Якщо погодний сервіс знаходить вказаний населений пункт, інформація про погоду в головній активності буде замінена.

3.4 Локалізація ресурсів програми

Функція, яка зберігає поточне місцезнаходження, щоб відобразити відомості про погоду тією мовою, що встановлена в операційній системі в якості основної:

```
    private String localize(SharedPreferences sp, String
preferenceKey, String defaultValueKey) {
    return localize(sp, this, preferenceKey, defaultValueKey);
}
```

Поки що в додатку реалізована українська мова інтерфейсу. Щоб додати інші мови, можна в каталозі res/values/strings проекту додати файли strings.xml з назвами ресурсів на інших мовах. Наприклад, замість

```
<string name="wind">Вітер</string>
<string name="pressure">Тиск</string>
```

в англомовному файлі ресурсів можна написати:

```
<string name="wind">Wind</string>
<string name="pressure">Pressure</string>
```

Наступна функція виводить на екран поточну погоду в обраній користувачем країні і місті:

```
    private void updateTodayWeatherUI() {
        String city = todayWeather.getCity();
    String country = todayWeather.getCountry();

    SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(MainActivity.this);

    // Temperature
    float temperature =
UnitConvertor.convertTemperature(Float.parseFloat(todayWeather.get
Temperature()), sp);
    if (sp.getBoolean("temperatureInteger", false)) {
        temperature = Math.round(temperature);
    }
  }
```

3.5 Основне меню програми

На рис. 3.5 представлене основне меню програми. У нього входить погодна карта, графіки, пошук населеного пункту, налаштування і інформація про програму. Визначення елементів меню знаходиться в файлі main_menu.xml каталогу res/menu проекту. Коли меню відкривається вперше, Android викликає метод onCreateOptionsMenu(), передаючи в якості параметра об'єкт Menu. Далі метод getMenuInflater отримує об'єкт MenuInflater і викликає його метод inflate(). Цей метод в якості першого параметра приймає ресурс, який представляє декларативний опис меню в xml, і наповнює їм об'єкт menu, переданий в якості другого параметра:

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

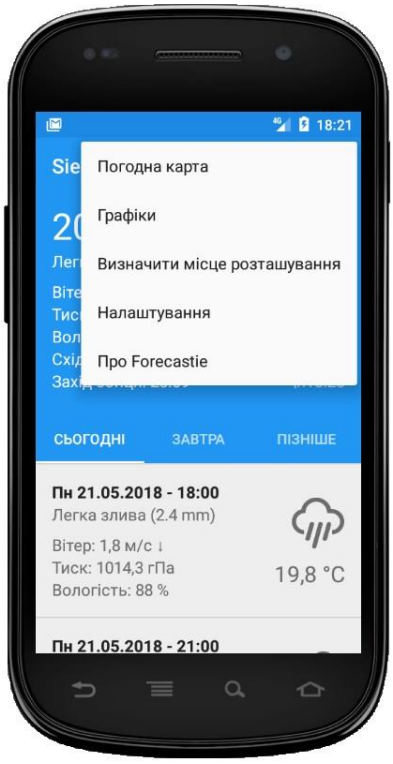


Рисунок 3.5 - Меню програми

3.6 Відображення карти погоди

У пункті меню «Погодна карта» можна обрати три види карт: дощ, вітер і температура. Якщо ми обираємо розділ «температура», можна побачити, де тепло, холодно або жарко завдяки кольорам. Теплу погоду відображає додаток жовтим кольором, жарку червоним і холодну білим. Так само, місця, в яких йде дощ, ми бачимо з затемненням (рис. 3.6).

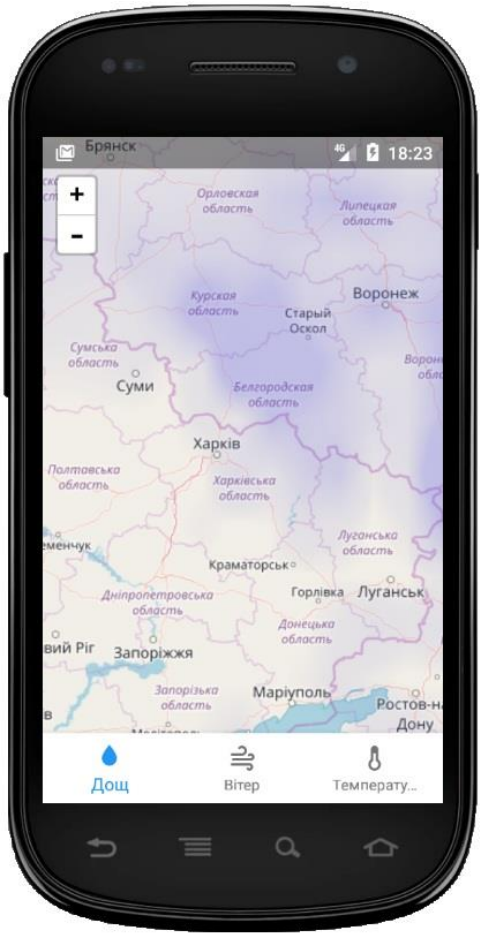


Рисунок 3.6 - Карта опадів

Сервіс OpenWeatherMap надає веб-карти з погодними умовами, для доступу до яких використовується плагін JQuery. Інтерактивність - це одна з особливостей JQuery, яка дозволяє користувачам взаємодіяти з інтерфейсом ресурсу, в нашому випадку з картами. В активності, що відображає погодні

карти, використовується компонент WebView, який дозволяє вбудовувати веб-сторінки в мобільні додатки:

```
WebView webView = (WebView) findViewById(R.id.webView);
webView.getSettings().setJavaScriptEnabled(true);
```

3.7 Відображення погоди за кілька днів

Також в програмі можливо переглянути погоду на найближчі дні на вкладках «Сьогодні», «Завтра» і «Пізніше». Ці можливості відображені на рис. 3.7.

Рисунок 3.7 - Процеси перегляду погоди на найближчі дні

В наведеному нижче коді зчитується інформація про погоду на даний день і на наступні дні. Також ця функція відображає в списку інформацію про погоду на вкладках «Сьогодні», «Завтра» і в наступні дні.

```
      private void updateLongTermWeatherUI() {
  ViewPagerAdapter viewPagerAdapter = new
ViewPagerAdapter(getSupportFragmentManager());

    Bundle bundleToday = new Bundle();
    bundleToday.putInt("day", 0);
    RecyclerViewFragment recyclerViewFragmentToday = new RecyclerViewFragment();
    recyclerViewFragmentToday.setArguments(bundleToday);
    viewPagerAdapter.addFragment(recyclerViewFragmentToday,
getString(R.string.today));
//…
    int currentPage = viewPager.getCurrentItem();

    viewPagerAdapter.notifyDataSetChanged();
    viewPager.setAdapter(viewPagerAdapter);
    tabLayout.setupWithViewPager(viewPager);

    if (currentPage == 0 && longTermTodayWeather.isEmpty()) {
        currentPage = 1;
    }
    viewPager.setCurrentItem(currentPage, false);
}
```

В залежності від погоди в цьому коді ми підставляємо потрібну нам картинку. Якщо дощ, то це дощова хмара, якщо погода не віщує жодних опадів, це сонце (рис. 3.8).

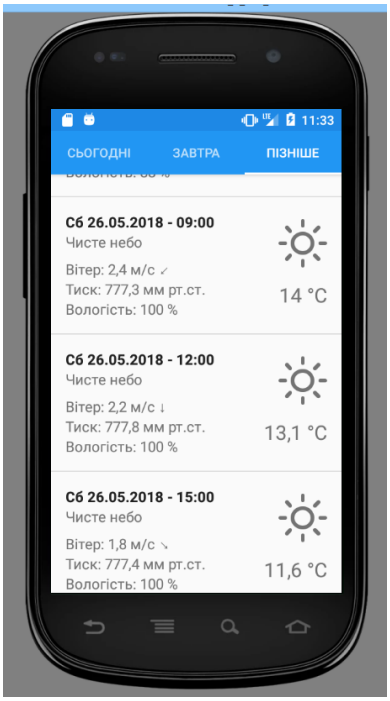


Рисунок 3.8 - Відображення малюнку, який відповідає за стан погоди

```
final String idString =
reader.getJSONArray("weather").getJSONObject(0).getString("id");
todayWeather.setId(idString);
todayWeather.setIcon(setWeatherIcon(Integer.parseInt(idString),
Calendar.getInstance().get(Calendar.HOUR_OF_DAY)));
```

Так само в налаштуваннях можна зайти в пункт меню «Графіки», де будуть відображатися три діаграми: температура, дощ і тиск. За шкалою можна визначити перепади одного з цих пунктів. Знизу у нас дні тижня, а зліва за шкалою одиниці виміру відповідно до обраного пункта (рис. 3.9).

Рисунок 3.9 - Графіки відображення тиску, дощу та температури

Даний пункт меню в коді програми реалізований наступним чином.

В Функції OnCreate() цієї активності викликаються функції відображення графіків температури, опадів і тиску за допомогою об'єкта WebView:

```
    if (parseLongTermJson(lastLongterm) == ParseResult.OK) {
   temperatureGraph();
   rainGraph();
   pressureGraph();
}
else {
   Snackbar.make(findViewById(android.R.id.content), R.string.msg_err_parsing_json,
Snackbar.LENGTH_LONG).show();
}
```

Відображення графіків виконується з використанням об'єкта Paint:

```
Paint paint = new Paint();
paint.setStyle(Paint.Style.STROKE);
paint.setAntiAlias(true);
paint.setColor(Color.parseColor("#333333"));
paint.setPathEffect(new DashPathEffect(new float[]{10, 10}, 0));
paint.setStrokeWidth(1);
```

3.8 Основні класи проекту

Клас Weather використовується для зберігання інформації про погоду за один день. У головній активності програми використовуються масиви об'єктів Weather з інформацією про погоду вранці, вдень, ввечері і вночі сьогодні, а також в наступні декілька днів:

```
private List<Weather> longTermWeather = new ArrayList<>();
```

Клас Weather отримує інформацію з сервісу OpenWeatherMap. Ця інформація згодом відображається активностями клієнтського додатку (рис. 3.10).

Рисунок 3.10 - Структура класу Weather

Поля класу призначені для зберігання усіх показників: температури, швидкості і напряму вітру, атмосферного тиску, вологості, наявності опадів, а також дати, часу заходу і світанку.

Клас WeatherViewHolder є похідним від класу RecyclerView.ViewHolder:

```
public class WeatherViewHolder extends RecyclerView.ViewHolder {…}
```

RecyclerView - це компонент інтерфейсу користувача, який дозволяє створювати прокручуваний список і використовується для відображення погоди в різні дні. Він підтримується, починаючи з версії Android Lollipop 5.0, і він подібний до компоненту ListView [5].

Він використаний для відображення погоди замість ListView, тому що використовує більш ефективний спосіб реалізації списку і розділяє зони відповідальності між класами.

Патерн ViewHolder запобігає непотрібним викликам функції findViewById(). Суть шаблону ViewHolder - це уникнути постійного сканування елементів в списку при його заповненні за допомогою методу findViewById(), який споживає багато системних ресурсів [6]. Для цього і був створений похідний від ViewHolder клас WeatherViewHolder, який постійно містить посилання на потрібні елементи. Замість того, щоб постійно викликати findViewById(), це робиться один раз і посилання зберігається у ViewHolder.

Висновки до третього розділу

Основні можливості розробленого додатку: відображення погодних умов на декілька днів у вказаному населеному пункті, побудова графіків зміни погодних показників, отримання від серверу і перегляд карт погоди.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Вступ

У даному розділі проводиться аналіз середовища, в якому проводилося дослідження програмного продукту на основі санітарних норм України. Проводилося дослідження роботи з розробки клієнтського додатку на ОС Android «Погодний інформер», напряму пов'язані з роботою на комп'ютері. Робота персонального комп’ютеру та тестувальних пристроїв (смартфон, планшет) впливають на фізичні і хімічні чинники середовища: електромагнітні випромінювання, статична електрика, температура і вологість повітря, вміст кисню і озону. Повітря забруднюється шкідливими хімічними речовинами антропогенного походження за рахунок деструкції полімерних матеріалів, що використовуються для обробки приміщень та обладнання. Організація робочого місця з порушенням норм призводить до загальної та локальної напруги м'язів шиї, тулуба, верхніх кінцівок, викривлення хребта і розвитку остеохондрозу, зміні внутрішнього тиску у оці та інших проблем з органами зору.

4.2 Аналіз умов праці в приміщенні

Дослідження в даній дипломній роботі проводилося в приміщенні.

Приміщення має одностороннє природне освітлення (вікно: висота = 1,5 м, ширина = 2.5 м) і загальне штучне освітлення. Стіни і стеля обклеєні світлими шпалерами, підлога вкрита світлим ламінатом. У приміщенні відсутні сильні вібрації та шкідливі речовини. Склад повітря в нормі.

Приміщення має довжину 3.5 м, ширину 2.5 м, висоту стелі 2,7 м. Кількість робочих місць - одне. Приміщення знаходиться на другому поверсі двохповерхової цегляної будівлі. Площа – 8.75 м 2 , об'єм – 23.625 м 3 . Виходячи з цього, отримаємо дані, наведені в таблиці 4.1.

Таблиця 4.1 – Фактичні та нормативні значення параметрів приміщення

Параметр	Норма	Реальні параметри
Площа, S	не менше 6 м ^{2}	8.75 м ^{2}
Об’єм , V	не менше 15 м ^{3}	23.625 м ^{3}

Можна зробити висновок, що отримані показники відповідають існуючим нормам та вимогам. Розглянемо тепер відповідність характеристик робочого місця нормативним. Для цього зведемо основні вимоги до організації робочого місця відповідні фактичні значення для робочого місця, за яким виконується робота, у таблиці 4.2:

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Значення	
	Фактичне	Нормативне
1	2	3
Висота робочої поверхні, мм	700	680 - 800
Висота простору для ніг, мм	700	не менше 600
Ширина простору для ніг, мм	700	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	500	400 - 500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	500	не менше 300
Ширина опорної поверхні спинки, мм	400	не менше 380

1	2	3
Радіус кривини спини в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	700	700 - 800

Крісло є підйомно-поворотним, має підлокітники і можливість регулювання за висотою і кутом нахилу спинки. Екран монітору знаходиться на відстані 0.7 м. Таким чином, за всіма параметрами робоче місце відповідає нормативним вимогам.

З обладнання в приміщенні знаходиться 1 комп’ютер та 2 монітори ACER VH27IPS 27". На все обладнання є паспорт та інструкція з експлуатації 75 українською мовою. згідно з супроводжувальній документації обладнання відповідає стандартам України і його можна використовувати без загрози здоров’ю та життю працюючого.

4.3 Мікрокліматичні умови

Цю роботу можна віднести до категорії легка 1а. Джерелами тепла в цьому приміщенні є люди, електроустаткування, освітлювальні прилади в темний час доби і система опалювання зимою. Оператором виділяється до 120 Ккал теплової енергії за годину. Оптимальні та фактичні значення параметрів мікроклімату приведені в таблиці 4.3.

Таблиця 4.3 – Значення мікроклімату

Період року	Параметр	Оптимальний	Фактичний
Теплий	Температура	23 – 25 °С	23 °С
	Вологість	40 – 60 %	40 %
	Швидкість повітря	≤ 0.1 м/с	
Холодний	Температура	22 – 24 °С	22 °С
	Вологість	40 – 60 %	50 %
	Швидкість повітря	≤ 0.1 м/с	

Всі показники задовольняють вимогам зазначеним для робіт категорії легка 1а і є задовільними для здоров'я людини.

4.4 Освітлення

Природне освітлення здійснюється за допомогою вікна, площа якого складає S' = 2,5*1,5 = 3.75м 2 та являється боковим освітленням.

Штучне освітлення здійснюється за допомогою світлодіодних ламп білого світіння.

Робота за дисплеєм ПЕОМ за розрядом зорових робіт відноситься до III розряду. При загальному висвітленні освітленість робочого місця повинна становити від 200 до 400 лк.

При штучному освітленні нормуються:

- Е (лк) - найменша припустима освітленість;
- М - показник дискомфорту;
- Кп (%) - коефіцієнт пульсації освітленості;

Перевіримо відповідність нормам фактичних параметрів штучного освітлення в приміщенні. Номінальний світловий потік лампи білого світіння ЛБ-40 Фл = 3120 лм.

У приміщенні застосовуються світильники, у яких встановлені 2 лампи.

Висоту підвісу світильника визначимо з формули:

$h = H - h^c - h_p - h^n$

H - висота приміщення, м; h_c - висота світильника, м; h_n - відстань від стелі до підвісу, м; h_p - висота робочої поверхні, м.

Для розглянутого приміщення :

H = 2,7 м; h_c = 0,2 м; h_n = 0,18 м; h_p = 0.7 м.

звідси :

h = 2,7 - 0,2 – 0,18 – 0,7 = 1,62 м.

Світильники розташовані в 1 ряд. Відстань між світильниками 2 метра, відстань від ряду до стіни 0,8 метра. Приміщення має наступні габарити:

довжина A = 3.5 метрів,

ширина B = 2.5 метрів.

Визначимо освітленість у робочій точці. Для розрахунку загальної рівномірної освітленості при горизонтальній робочій поверхні використаємо метод коефіцієнта використання світлового потоку.

Розрахункова формула для світлового потоку світильника має вигляд:

$$Ф_{л}=\frac{E*К_З*S*Z}{N*n} \quad (4.1)$$

N - число світильників у приміщенні, N = 2*2=4;

n - коефіцієнт використання світлового потоку;

$Ф_л$ - світловий потік ламп;

$К_з$ - коефіцієнт запасу, $К_з$ = 1.5;

Z - коефіцієнт нерівномірності;

S - площа приміщення;

E - освітленість, створювана всіма світильниками.

Звідси одержуємо формулу для розрахунку освітленості на робочому місці :

$$E = \frac{\Phi_л * N * n}{К_З * S * Z} \quad (4.2)$$

Коефіцієнт використання світлового потоку залежить від:

ККД, кривій розподілу сили світла світильника;

Коефіцієнта відбиття стелі Rc і стін Rc;

Висоти підвісу світильників hп;

4.5 Шум і вібрація

Джерелом шуму в приміщенні є комп'ютер. Кулери комп'ютера є сучасними і мають низький рівень шуму за рахунок нового під шипника зі самостабілізуючимся тиском рідини (SSO2). Згідно технічній документації шум обумовлений кулером в блоці живлення складає 7-12 дБ, кулером процесора – 7-12 дБ, загальний, - 20 дБ. Враховуючи незначний рівень шуму від персонального комп'ютера та незначний рівень фонового шуму від іншого устаткування - сумарний рівень шумового забруднення приміщення не перевищує максимально допустимий рівень коригованої звукової потужності і складає не більше 50 дБ.

При роботі з персональним комп'ютером в робочому приміщенні значення характеристик вібрації на робочих місцях не повинна перевищувати допустимих значень.

4.6 Випромінювання

У приміщенні відсутні інфрачервоні, ультрафіолетові та електромагнітні випромінювання, бо усі монітори ПК вироблені на основі OLED матриці, підсвітка якої здійснюється за рахунок органічних світлодіодів, що не має сильного електромагнітного випромінювання і сертифіковані в Україні.

4.7 Оцінка умов безпеки праці

4.7.1 Вимоги електробезпеки

Технічні заходи із запобігання електротравм від контакту з нормально струмовідними елементами електроустаткування

- електромережа в приміщенні розведена в спеціальних каналах стін і підлоги.

- величина напруги мережі 380х 220В (міжфазна лінійна і фазна);

- всі нормально струмовідні елементів (в першу чергу електричні дроти) вкриті ізоляційними матеріалами;

- в джерелі безперебійного живлення персонального комп’ютера використовується механічне захисне блокування, що забезпечує вимикання напруги при його відкриванні;

Можна зробити висновок, що дане приміщення задовольняє нормам електробезпеки для встановлення ПК.

4.7.2 Оцінка пожежної безпеки приміщення

У приміщенні, що розглядається, можуть горіти: вироби з дерева, пластмас, тканини і паперу, ламінат. Горючі рідини, пил та волокна у приміщенні не використовуються і не виділяються, тому воно відноситься, відповідно до нормативної документації, до зони П-IIа [10] і до категорії пожежної небезпеки В.

Ймовірними причинами виникнення пожежі можуть бути несправна робота електрообладнання (кабелів, розеток), короткі замикання внаслідок виходу з ладу чи експлуатації несправного електроустаткування (ПЕОМ, периферійних пристроїв), порушення правил протипожежної безпеки тощо.

Експлуатація ліній електромережі практично повністю унеможливлює виникнення електричного джерела загоряння в наслідок короткого замикання та перевантаження проводів, а автоматичні вимикачі миттєво відключать ділянку мережі у разі перевантаження чи замикання. Застосовуються дроти з важкогорючою і негорючою ізоляцією. Блок живлення комп'ютеру оснащений автоматичним вимикачем у разі перевантаження чи стрибка напруги у мережі.

Комплекс заходів для своєчасного попередження пожеж у приміщенні та оперативного реагування у разі їх виникнення:

- заборона використання відкритого вогню у приміщенні;
- автомати на кожній ланці мережі для запобігання перевантаження та замикання;
- обов'язковий інструктаж персоналу з пожежної безпеки;
- наявність системи автоматичної пожежної сигналізації з димовими пожежними оповіщувачами;
- ступінь вогнестійкості будівлі, у якій розташовано приміщення – II;
- наявність шляхів евакуації при виникненні пожежі;
- призначення особи, відповідаючої за пожежну безпеку;

- розміщення схеми евакуації людей при пожежі і ознайомлення з нею персоналу.

Приміщення має один вихід, оскільки в ньому працює менше 25 чоловік. Будинок має два виходи – головний і запасних. Коридор між приміщеннями має два виходи на різні сходи, одні з яких ведуть до головного виходу, а другі - до спеціального евакуаційного виходу

Для гасіння пожежі кожна кімната обладнана ручними вуглекислотними вогнегасниками ВВК-1,4 [11]. У загальному коридорі встановлені пінні вогнегасники ВВП. Призначена особа, що відповідає за дотримання персоналом вимог пожежної безпеки. Розроблено план евакуації персоналу і найбільш коштовного устаткування (майна). Співробітники ознайомлені з порядком і планом евакуації під розпис. Можна зробити висновок, що шляхи евакуації з приміщення повністю відповідають нормам.

Висновки до четвертого розділу

Аналіз умов праці в розглянутому робочому приміщенні показав, що умови праці з ПЕОМ відповідають вимогам , оскільки площа та об’єм не менше нормативних значень, рівні шуму, вібрації і загазованості не перевищують нормативних обмежень.

Для підтримання параметрів мікроклімату в приміщенні встановлено радіатор центральної водяної системи опалення, що складається з 7 секцій.

Ергономіка робочого місця і режим зорової роботи задовольняють вимогам і сприяють зниженню втоми.

ВИСНОВКИ

Тільки додатки можуть зробити будь-яку операційну систему придатною для роботи, розваги, виходу в Інтернет, перегляду веб-сторінок і багато чого іншого, що перетворює звичайний телефон в маленький кишеньковий комп'ютер з повним набором функціональних можливостей. В дипломному проекті був розроблений клієнтський додаток, що дозволяє користувачу переглядати на смартфоні з ОС Android дані про погоду в указаному населеному пункті. Основні можливості розробленої програми:

- пошук будь-якого населеного пункту;
- відображення показників погоди: температури, атмосферного тиску, опадів, швидкості вітру, хмарності;
- отримання даних і відображення погодних умов на тиждень;
- відображення карт температури, швидкості вітру, опадів.

Для розробки додатку була використана мова програмування Java і середовище програмування Android Studio 3.1. Зовнішній вигляд активностей (форм) додатку зберігається у файлах xml-формату.

Були розроблені заходи з охорони праці та безпеки в надзвичайних ситуаціях.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сервер OpenWeatherMap [Електронний ресурс]. - Режим доступу: https://ru.wikipedia.org/wiki/OpenWeatherMap

2. Steve Prettyman Learn PHP 7: Object-Oriental Modular Programming using HTML5, CSS, JavaScript, XML, JSON, and MySQL. – Stone Mountain, Georgia USA, 2016.

3. Дунаев В. В. Базы данных. Язык SQL для студента: 2-е изд. доп. и перераб. – СПб.: БХВ-Петербург, 2017. – 320 с.

4. ECMAScript® 2015 Language Specification [Електронний ресурс]. - Режим доступу: http://www.ecma-international.org/ecma-262/6.0/index.html#sec-ecmascript-function-objects.

5. Dave MacLean Pro Android 5 / Dave MacLean, Satya Komatineni, Grant Allen. - Springer Science+Business Media, New York, 2015. – 813 pp.

6. Flanagan D. JavaScript: The Definitive Guide, Sixth Edition / David Flanagan. - Sebastopol : O'Reilly Media, Inc., 2011. – 1098 p.

7. Revill L. jQuery 2.0 Development Cookbook / Leon Revill. - Birmingham : Packt Publishing Ltd, 2014. – 410 p.

8. Neuburg M. Programming iOS 7. Fourth Edition / Matt Neuburg . - Sebastopol : O'Reilly Media, Inc., 2014. – 929 p.

9. Gargenta M. Learning Android, Second Edition / Marko Gargenta, Masumi Nakamura. – Sebastopol : O'Reilly Media, Inc., 2014. – 288 p.

10. Программирование под Android. Уроки разработки, примеры кода [Електронний ресурс]. - Режим доступу: http://androiddocs.ru/pattern-viewholder-v-realizacii-spiska-listview/

11. Жидецький В.Ц., Джигирей В.С ., Мельников О.В. Основи охорони праці , 2000 – 350 с.

12. Закон України «Про пожежну безпеку»

Додаток А

Лістинг програми

Файл MainActivity.java

```
import android.Manifest;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.content.res.TypedArray;
import android.graphics.Color;
import android.graphics.Typeface;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.provider.Settings;
import android.support.design.widget.Snackbar;
import android.support.design.widget.TabLayout;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.InputType;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.webkit.WebView;
import android.widget.EditText;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.text.DateFormat;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import cz.martykan.forecastie.AlarmReceiver;
import cz.martykan.forecastie.Constants;
import cz.martykan.forecastie.R;
import cz.martykan.forecastie.adapters.ViewPagerAdapter;
import cz.martykan.forecastie.adapters.WeatherRecyclerAdapter;
import cz.martykan.forecastie.fragments.RecyclerViewFragment;
```

```
import cz.martykan.forecastie.models.Weather;
import cz.martykan.forecastie.tasks.GenericRequestTask;
import cz.martykan.forecastie.tasks.ParseResult;
import cz.martykan.forecastie.tasks.TaskOutput;
import cz.martykan.forecastie.utils.UnitConvertor;
import cz.martykan.forecastie.widgets.AbstractWidgetProvider;
import cz.martykan.forecastie.widgets.DashClockWeatherExtension;

public class MainActivity extends AppCompatActivity implements LocationListener
{
    protected static final int MY_PERMISSIONS_ACCESS_FINE_LOCATION = 1;

    // Time in milliseconds; only reload weather if last update is longer ago
than this value
    private static final int NO_UPDATE_REQUIRED_THRESHOLD = 300000;

    private static Map<String, Integer> speedUnits = new HashMap<>(3);
    private static Map<String, Integer> pressUnits = new HashMap<>(3);
    private static boolean mappingsInitialised = false;

    Typeface weatherFont;
    Weather todayWeather = new Weather();

    TextView todayTemperature;
    TextView todayDescription;
    TextView todayWind;
    TextView todayPressure;
    TextView todayHumidity;
    TextView todaySunrise;
    TextView todaySunset;
    TextView lastUpdate;
    TextView todayIcon;
    ViewPager viewPager;
    TabLayout tabLayout;

    View appView;

    LocationManager locationManager;
    ProgressDialog progressDialog;

    int theme;
    boolean destroyed = false;

    private List<Weather> longTermWeather = new ArrayList<>();
    private List<Weather> longTermTodayWeather = new ArrayList<>();
    private List<Weather> longTermTomorrowWeather = new ArrayList<>();

    public String recentCity = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // Initialize the associated SharedPreferences file with default values
        PreferenceManager.setDefaultValues(this, R.xml.prefs, false);

        SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(this);
        setTheme(theme = getTheme(prefs.getString("theme", "fresh")));
        boolean darkTheme = theme == R.style.AppTheme_NoActionBar_Dark ||
                theme == R.style.AppTheme_NoActionBar_Classic_Dark;
        boolean blackTheme = theme == R.style.AppTheme_NoActionBar_Black ||
                theme == R.style.AppTheme_NoActionBar_Classic_Black;

        // Initiate activity
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scrolling);
        appView = findViewById(R.id.viewApp);

        progressDialog = new ProgressDialog(MainActivity.this);

        // Load toolbar
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        if (darkTheme) {
            toolbar.setPopupTheme(R.style.AppTheme_PopupOverlay_Dark);
        } else if (blackTheme) {
            toolbar.setPopupTheme(R.style.AppTheme_PopupOverlay_Black);
        }

        // Initialize textboxes
        todayTemperature = (TextView) findViewById(R.id.todayTemperature);
        todayDescription = (TextView) findViewById(R.id.todayDescription);
        todayWind = (TextView) findViewById(R.id.todayWind);
        todayPressure = (TextView) findViewById(R.id.todayPressure);
        todayHumidity = (TextView) findViewById(R.id.todayHumidity);
        todaySunrise = (TextView) findViewById(R.id.todaySunrise);
        todaySunset = (TextView) findViewById(R.id.todaySunset);
        lastUpdate = (TextView) findViewById(R.id.lastUpdate);
        todayIcon = (TextView) findViewById(R.id.todayIcon);
        weatherFont = Typeface.createFromAsset(this.getAssets(),
"fonts/weather.ttf");
        todayIcon.setTypeface(weatherFont);

        // Initialize viewPager
        viewPager = (ViewPager) findViewById(R.id.viewPager);
        tabLayout = (TabLayout) findViewById(R.id.tabs);

        destroyed = false;

        initMappings();

        // Preload data from cache
        preloadWeather();
        updateLastUpdateTime();

        // Set autoupdater
        AlarmReceiver.setRecurringAlarm(this);
    }

    public WeatherRecyclerAdapter getAdapter(int id) {
        WeatherRecyclerAdapter weatherRecyclerAdapter;
        if (id == 0) {
            weatherRecyclerAdapter = new WeatherRecyclerAdapter(this,
longTermTodayWeather);
        } else if (id == 1) {
            weatherRecyclerAdapter = new WeatherRecyclerAdapter(this,
longTermTomorrowWeather);
        } else {
            weatherRecyclerAdapter = new WeatherRecyclerAdapter(this,
longTermWeather);
        }
        return weatherRecyclerAdapter;
    }

    @Override
    public void onStart() {
        super.onStart();
```

```
        updateTodayWeatherUI();
        updateLongTermWeatherUI();
    }

    @Override
    public void onResume() {
        super.onResume();
        if
(getTheme(PreferenceManager.getDefaultSharedPreferences(this).getString("theme",
"fresh")) != theme) {
            // Restart activity to apply theme
            overridePendingTransition(0, 0);
            finish();
            overridePendingTransition(0, 0);
            startActivity(getIntent());
        } else if (shouldUpdate() && isNetworkAvailable()) {
            getTodayWeather();
            getLongTermWeather();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        destroyed = true;

        if (locationManager != null) {
            try {
                locationManager.removeUpdates(MainActivity.this);
            } catch (SecurityException e) {
                e.printStackTrace();
            }
        }
    }

    private void preloadWeather() {
        SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(MainActivity.this);

        String lastToday = sp.getString("lastToday", "");
        if (!lastToday.isEmpty()) {
            new TodayWeatherTask(this, this,
progressDialog).executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR,
"cachedResponse", lastToday);
        }
        String lastLongterm = sp.getString("lastLongterm", "");
        if (!lastLongterm.isEmpty()) {
            new LongTermWeatherTask(this, this,
progressDialog).executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR,
"cachedResponse", lastLongterm);
        }
    }

    private void getTodayWeather() {
        new TodayWeatherTask(this, this, progressDialog).execute();
    }

    private void getLongTermWeather() {
        new LongTermWeatherTask(this, this, progressDialog).execute();
    }

    private void searchCities() {
        AlertDialog.Builder alert = new AlertDialog.Builder(this);
```

```
        alert.setTitle(this.getString(R.string.search_title));
        final EditText input = new EditText(this);
        input.setInputType(InputType.TYPE_CLASS_TEXT);
        input.setMaxLines(1);
        input.setSingleLine(true);
        alert.setView(input, 32, 0, 32, 0);
        alert.setPositiveButton(R.string.dialog_ok, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                String result = input.getText().toString();
                if (!result.isEmpty()) {
                    saveLocation(result);
                }
            }
        });
        alert.setNegativeButton(R.string.dialog_cancel, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                // Cancelled
            }
        });
        alert.show();
    }

    private void saveLocation(String result) {
        SharedPreferences preferences =
PreferenceManager.getDefaultSharedPreferences(MainActivity.this);
        recentCity = preferences.getString("city", Constants.DEFAULT_CITY);

        SharedPreferences.Editor editor = preferences.edit();
        editor.putString("city", result);
        editor.commit();

        if (!recentCity.equals(result)) {
            // New location, update weather
            getTodayWeather();
            getLongTermWeather();
        }
    }

    private void aboutDialog() {
        AlertDialog.Builder alert = new AlertDialog.Builder(this);
        alert.setTitle("Forecastie");
        final WebView webView = new WebView(this);
        String about = "<p>1.6.1</p>" +
                "<p>A lightweight, opensource weather app.</p>" +
                "<p>Developed by <a href='mailto:t.martykan@gmail.com'>Tomas
Martykan</a></p>" +
                "<p>Data provided by <a
href='https://openweathermap.org/'>OpenWeatherMap</a>, under the <a
href='http://creativecommons.org/licenses/by-sa/2.0/'>Creative Commons
license</a>" +
                "<p>Icons are <a href='https://erikflowers.github.io/weather-
icons/'>Weather Icons</a>, by <a href='http://www.twitter.com/artill'>Lukas
Bischoff</a> and <a href='http://www.twitter.com/Erik_UX'>Erik Flowers</a>,
under the <a href='http://scripts.sil.org/OFL'>SIL OFL 1.1</a> licence.";
        TypedArray ta = obtainStyledAttributes(new
int[]{android.R.attr.textColorPrimary, R.attr.colorAccent});
        String textColor = String.format("#%06X", (0xFFFFFF & ta.getColor(0,
Color.BLACK)));
        String accentColor = String.format("#%06X", (0xFFFFFF & ta.getColor(1,
Color.BLUE)));
        ta.recycle();
```

```
        about = "<style media=\"screen\" type=\"text/css\">" +
                "body {\n" +
                "    color:" + textColor + ";\n" +
                "}\n" +
                "a:link {color:" + accentColor + "}\n" +
                "</style>" +
                about;
        webView.setBackgroundColor(Color.TRANSPARENT);
        webView.loadData(about, "text/html", "UTF-8");
        alert.setView(webView, 32, 0, 32, 0);
        alert.setPositiveButton(R.string.dialog_ok, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {

            }
        });
        alert.show();
    }

    private String setWeatherIcon(int actualId, int hourOfDay) {
        int id = actualId / 100;
        String icon = "";
        if (actualId == 800) {
            if (hourOfDay >= 7 && hourOfDay < 20) {
                icon = this.getString(R.string.weather_sunny);
            } else {
                icon = this.getString(R.string.weather_clear_night);
            }
        } else {
            switch (id) {
                case 2:
                    icon = this.getString(R.string.weather_thunder);
                    break;
                case 3:
                    icon = this.getString(R.string.weather_drizzle);
                    break;
                case 7:
                    icon = this.getString(R.string.weather_foggy);
                    break;
                case 8:
                    icon = this.getString(R.string.weather_cloudy);
                    break;
                case 6:
                    icon = this.getString(R.string.weather_snowy);
                    break;
                case 5:
                    icon = this.getString(R.string.weather_rainy);
                    break;
            }
        }
        return icon;
    }

    public static String getRainString(JSONObject rainObj) {
        String rain = "0";
        if (rainObj != null) {
            rain = rainObj.optString("3h", "fail");
            if ("fail".equals(rain)) {
                rain = rainObj.optString("1h", "0");
            }
        }
        return rain;
    }
```

```
    private ParseResult parseTodayJson(String result) {
        try {
            JSONObject reader = new JSONObject(result);

            final String code = reader.optString("cod");
            if ("404".equals(code)) {
                return ParseResult.CITY_NOT_FOUND;
            }

            String city = reader.getString("name");
            String country = "";
            JSONObject countryObj = reader.optJSONObject("sys");
            if (countryObj != null) {
                country = countryObj.getString("country");
                todayWeather.setSunrise(countryObj.getString("sunrise"));
                todayWeather.setSunset(countryObj.getString("sunset"));
            }
            todayWeather.setCity(city);
            todayWeather.setCountry(country);

            JSONObject coordinates = reader.getJSONObject("coord");
            if (coordinates != null) {
                SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(this);
                sp.edit().putFloat("latitude", (float)
coordinates.getDouble("lon")).putFloat("longitude", (float)
coordinates.getDouble("lat")).commit();
            }

            JSONObject main = reader.getJSONObject("main");

            todayWeather.setTemperature(main.getString("temp"));

todayWeather.setDescription(reader.getJSONArray("weather").getJSONObject(0).getS
tring("description"));
            JSONObject windObj = reader.getJSONObject("wind");
            todayWeather.setWind(windObj.getString("speed"));
            if (windObj.has("deg")) {
                todayWeather.setWindDirectionDegree(windObj.getDouble("deg"));
            } else {
                Log.e("parseTodayJson", "No wind direction available");
                todayWeather.setWindDirectionDegree(null);
            }
            todayWeather.setPressure(main.getString("pressure"));
            todayWeather.setHumidity(main.getString("humidity"));

            JSONObject rainObj = reader.optJSONObject("rain");
            String rain;
            if (rainObj != null) {
                rain = getRainString(rainObj);
            } else {
                JSONObject snowObj = reader.optJSONObject("snow");
                if (snowObj != null) {
                    rain = getRainString(snowObj);
                } else {
                    rain = "0";
                }
            }
            todayWeather.setRain(rain);

            final String idString =
reader.getJSONArray("weather").getJSONObject(0).getString("id");
```

```
            todayWeather.setId(idString);
            todayWeather.setIcon(setWeatherIcon(Integer.parseInt(idString),
Calendar.getInstance().get(Calendar.HOUR_OF_DAY)));

            SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(MainActivity.this).edit();
            editor.putString("lastToday", result);
            editor.commit();

        } catch (JSONException e) {
            Log.e("JSONException Data", result);
            e.printStackTrace();
            return ParseResult.JSON_EXCEPTION;
        }

        return ParseResult.OK;
    }

    private void updateTodayWeatherUI() {
        try {
            if (todayWeather.getCountry().isEmpty()) {
                preloadWeather();
                return;
            }
        } catch (Exception e) {
            preloadWeather();
            return;
        }
        String city = todayWeather.getCity();
        String country = todayWeather.getCountry();
        DateFormat timeFormat =
android.text.format.DateFormat.getTimeFormat(getApplicationContext());
        getSupportActionBar().setTitle(city + (country.isEmpty() ? "" : ", " +
country));

        SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(MainActivity.this);

        // Temperature
        float temperature =
UnitConvertor.convertTemperature(Float.parseFloat(todayWeather.getTemperature())
, sp);
        if (sp.getBoolean("temperatureInteger", false)) {
            temperature = Math.round(temperature);
        }

        // Rain
        double rain = Double.parseDouble(todayWeather.getRain());
        String rainString = UnitConvertor.getRainString(rain, sp);

        // Wind
        double wind;
        try {
            wind = Double.parseDouble(todayWeather.getWind());
        } catch (Exception e) {
            e.printStackTrace();
            wind = 0;
        }
        wind = UnitConvertor.convertWind(wind, sp);

        // Pressure
        double pressure = UnitConvertor.convertPressure((float)
Double.parseDouble(todayWeather.getPressure()), sp);
```

```
        todayTemperature.setText(new DecimalFormat("0.#").format(temperature) +
" " + sp.getString("unit", "°C"));
        todayDescription.setText(todayWeather.getDescription().substring(0,
1).toUpperCase() +
                todayWeather.getDescription().substring(1) + rainString);
        if (sp.getString("speedUnit", "m/s").equals("bft")) {
            todayWind.setText(getString(R.string.wind) + ": " +
                    UnitConvertor.getBeaufortName((int) wind) +
                    (todayWeather.isWindDirectionAvailable() ? " " +
getWindDirectionString(sp, this, todayWeather) : ""));
        } else {
            todayWind.setText(getString(R.string.wind) + ": " + new
DecimalFormat("0.0").format(wind) + " " +
                    localize(sp, "speedUnit", "m/s") +
                    (todayWeather.isWindDirectionAvailable() ? " " +
getWindDirectionString(sp, this, todayWeather) : ""));
        }
        todayPressure.setText(getString(R.string.pressure) + ": " + new
DecimalFormat("0.0").format(pressure) + " " +
                localize(sp, "pressureUnit", "hPa"));
        todayHumidity.setText(getString(R.string.humidity) + ": " +
todayWeather.getHumidity() + " %");
        todaySunrise.setText(getString(R.string.sunrise) + ": " +
timeFormat.format(todayWeather.getSunrise()));
        todaySunset.setText(getString(R.string.sunset) + ": " +
timeFormat.format(todayWeather.getSunset()));
        todayIcon.setText(todayWeather.getIcon());
    }

    public ParseResult parseLongTermJson(String result) {
        int i;
        try {
            JSONObject reader = new JSONObject(result);

            final String code = reader.optString("cod");
            if ("404".equals(code)) {
                if (longTermWeather == null) {
                    longTermWeather = new ArrayList<>();
                    longTermTodayWeather = new ArrayList<>();
                    longTermTomorrowWeather = new ArrayList<>();
                }
                return ParseResult.CITY_NOT_FOUND;
            }

            longTermWeather = new ArrayList<>();
            longTermTodayWeather = new ArrayList<>();
            longTermTomorrowWeather = new ArrayList<>();

            JSONArray list = reader.getJSONArray("list");
            for (i = 0; i < list.length(); i++) {
                Weather weather = new Weather();

                JSONObject listItem = list.getJSONObject(i);
                JSONObject main = listItem.getJSONObject("main");

                weather.setDate(listItem.getString("dt"));
                weather.setTemperature(main.getString("temp"));

weather.setDescription(listItem.optJSONArray("weather").getJSONObject(0).getStri
ng("description"));
                JSONObject windObj = listItem.optJSONObject("wind");
                if (windObj != null) {
```

```
                    weather.setWind(windObj.getString("speed"));
                    weather.setWindDirectionDegree(windObj.getDouble("deg"));
                }
                weather.setPressure(main.getString("pressure"));
                weather.setHumidity(main.getString("humidity"));

                JSONObject rainObj = listItem.optJSONObject("rain");
                String rain = "";
                if (rainObj != null) {
                    rain = getRainString(rainObj);
                } else {
                    JSONObject snowObj = listItem.optJSONObject("snow");
                    if (snowObj != null) {
                        rain = getRainString(snowObj);
                    } else {
                        rain = "0";
                    }
                }
                weather.setRain(rain);

                final String idString =
listItem.optJSONArray("weather").getJSONObject(0).getString("id");
                weather.setId(idString);

                final String dateMsString = listItem.getString("dt") + "000";
                Calendar cal = Calendar.getInstance();
                cal.setTimeInMillis(Long.parseLong(dateMsString));
                weather.setIcon(setWeatherIcon(Integer.parseInt(idString),
cal.get(Calendar.HOUR_OF_DAY)));

                Calendar today = Calendar.getInstance();
                if (cal.get(Calendar.DAY_OF_YEAR) ==
today.get(Calendar.DAY OF YEAR)) {
                    longTermTodayWeather.add(weather);
                } else if (cal.get(Calendar.DAY_OF_YEAR) ==
today.get(Calendar.DAY_OF_YEAR) + 1) {
                    longTermTomorrowWeather.add(weather);
                } else {
                    longTermWeather.add(weather);
                }
            }
            SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(MainActivity.this).edit();
            editor.putString("lastLongterm", result);
            editor.commit();
        } catch (JSONException e) {
            Log.e("JSONException Data", result);
            e.printStackTrace();
            return ParseResult.JSON_EXCEPTION;
        }

        return ParseResult.OK;
    }

    private void updateLongTermWeatherUI() {
        if (destroyed) {
            return;
        }

        ViewPagerAdapter viewPagerAdapter = new
ViewPagerAdapter(getSupportFragmentManager());

        Bundle bundleToday = new Bundle();
```

```
        bundleToday.putInt("day", 0);
        RecyclerViewFragment recyclerViewFragmentToday = new
RecyclerViewFragment();
        recyclerViewFragmentToday.setArguments(bundleToday);
        viewPagerAdapter.addFragment(recyclerViewFragmentToday,
getString(R.string.today));

        Bundle bundleTomorrow = new Bundle();
        bundleTomorrow.putInt("day", 1);
        RecyclerViewFragment recyclerViewFragmentTomorrow = new
RecyclerViewFragment();
        recyclerViewFragmentTomorrow.setArguments(bundleTomorrow);
        viewPagerAdapter.addFragment(recyclerViewFragmentTomorrow,
getString(R.string.tomorrow));

        Bundle bundle = new Bundle();
        bundle.putInt("day", 2);
        RecyclerViewFragment recyclerViewFragment = new RecyclerViewFragment();
        recyclerViewFragment.setArguments(bundle);
        viewPagerAdapter.addFragment(recyclerViewFragment,
getString(R.string.later));

        int currentPage = viewPager.getCurrentItem();

        viewPagerAdapter.notifyDataSetChanged();
        viewPager.setAdapter(viewPagerAdapter);
        tabLayout.setupWithViewPager(viewPager);

        if (currentPage == 0 && longTermTodayWeather.isEmpty()) {
            currentPage = 1;
        }
        viewPager.setCurrentItem(currentPage, false);
    }

    private boolean isNetworkAvailable() {
        ConnectivityManager connectivityManager = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo activeNetworkInfo =
connectivityManager.getActiveNetworkInfo();
        return activeNetworkInfo != null && activeNetworkInfo.isConnected();
    }

    private boolean shouldUpdate() {
        long lastUpdate =
PreferenceManager.getDefaultSharedPreferences(this).getLong("lastUpdate", -1);
        boolean cityChanged =
PreferenceManager.getDefaultSharedPreferences(this).getBoolean("cityChanged",
false);
        // Update if never checked or last update is longer ago than specified
threshold
        return cityChanged || lastUpdate < 0 ||
(Calendar.getInstance().getTimeInMillis() - lastUpdate) >
NO_UPDATE_REQUIRED_THRESHOLD;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
```

```
        int id = item.getItemId();

        if (id == R.id.action_refresh) {
            if (isNetworkAvailable()) {
                getTodayWeather();
                getLongTermWeather();
            } else {
                Snackbar.make(appView,
getString(R.string.msg_connection_not_available), Snackbar.LENGTH_LONG).show();
            }
            return true;
        }
        if (id == R.id.action_map) {
            Intent intent = new Intent(MainActivity.this, MapActivity.class);
            startActivity(intent);
        }
        if (id == R.id.action_graphs) {
            Intent intent = new Intent(MainActivity.this, GraphActivity.class);
            startActivity(intent);
        }
        if (id == R.id.action_search) {
            searchCities();
            return true;
        }
        if (id == R.id.action_location) {
            getCityByLocation();
            return true;
        }
        if (id == R.id.action_settings) {
            Intent intent = new Intent(MainActivity.this,
SettingsActivity.class);
            startActivity(intent);
        }
        if (id == R.id.action_about) {
            aboutDialog();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    public static void initMappings() {
        if (mappingsInitialised)
            return;
        mappingsInitialised = true;
        speedUnits.put("m/s", R.string.speed_unit_mps);
        speedUnits.put("kph", R.string.speed_unit_kph);
        speedUnits.put("mph", R.string.speed_unit_mph);
        speedUnits.put("kn", R.string.speed_unit_kn);

        pressUnits.put("hPa", R.string.pressure_unit_hpa);
        pressUnits.put("kPa", R.string.pressure_unit_kpa);
        pressUnits.put("mm Hg", R.string.pressure_unit_mmhg);
    }

    private String localize(SharedPreferences sp, String preferenceKey, String
defaultValueKey) {
        return localize(sp, this, preferenceKey, defaultValueKey);
    }

    public static String localize(SharedPreferences sp, Context context, String
preferenceKey, String defaultValueKey) {
        String preferenceValue = sp.getString(preferenceKey, defaultValueKey);
        String result = preferenceValue;
```

```
        if ("speedUnit".equals(preferenceKey)) {
            if (speedUnits.containsKey(preferenceValue)) {
                result = context.getString(speedUnits.get(preferenceValue));
            }
        } else if ("pressureUnit".equals(preferenceKey)) {
            if (pressUnits.containsKey(preferenceValue)) {
                result = context.getString(pressUnits.get(preferenceValue));
            }
        }
        return result;
    }

    public static String getWindDirectionString(SharedPreferences sp, Context
context, Weather weather) {
        try {
            if (Double.parseDouble(weather.getWind()) != 0) {
                String pref = sp.getString("windDirectionFormat", null);
                if ("arrow".equals(pref)) {
                    return weather.getWindDirection(8).getArrow(context);
                } else if ("abbr".equals(pref)) {
                    return
weather.getWindDirection().getLocalizedString(context);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return "";
    }

    void getCityByLocation() {
        locationManager = (LocationManager) getSystemService(LOCATION SERVICE);

        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED)
{
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)) {
                // Explanation not needed, since user requests this themmself

            } else {
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_ACCESS_FINE_LOCATION);
            }

        } else if
(locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER) ||
                locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER))
{
            progressDialog = new ProgressDialog(this);
            progressDialog.setMessage(getString(R.string.getting_location));
            progressDialog.setCancelable(false);
            progressDialog.setButton(DialogInterface.BUTTON_NEGATIVE,
getString(R.string.dialog_cancel), new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    try {
                        locationManager.removeUpdates(MainActivity.this);
                    } catch (SecurityException e) {
                        e.printStackTrace();
                    }
```

```
                }
            });
            progressDialog.show();
            if
(locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {

locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0,
this);
            }
            if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER))
{

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
this);
            }
        } else {
            showLocationSettingsDialog();
        }
    }

    private void showLocationSettingsDialog() {
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
        alertDialog.setTitle(R.string.location_settings);
        alertDialog.setMessage(R.string.location_settings_message);
        alertDialog.setPositiveButton(R.string.location_settings_button, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                Intent intent = new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                startActivity(intent);
            }
        });
        alertDialog.setNegativeButton(R.string.dialog_cancel, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });
        alertDialog.show();
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String
permissions[], int[] grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_ACCESS_FINE_LOCATION: {
                // If request is cancelled, the result arrays are empty.
                if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    getCityByLocation();
                }
                return;
            }
        }
    }

    @Override
    public void onLocationChanged(Location location) {
        progressDialog.hide();
        try {
            locationManager.removeUpdates(this);
        } catch (SecurityException e) {
            Log.e("LocationManager", "Error while trying to stop listening for
```

```
location updates. This is probably a permissions issue", e);
        }
        Log.i("LOCATION (" + location.getProvider().toUpperCase() + ")",
location.getLatitude() + ", " + location.getLongitude());
        double latitude = location.getLatitude();
        double longitude = location.getLongitude();
        new ProvideCityNameTask(this, this, progressDialog).execute("coords",
Double.toString(latitude), Double.toString(longitude));
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {

    }

    @Override
    public void onProviderEnabled(String provider) {

    }

    @Override
    public void onProviderDisabled(String provider) {

    }


    class TodayWeatherTask extends GenericRequestTask {
        public TodayWeatherTask(Context context, MainActivity activity,
ProgressDialog progressDialog) {
            super(context, activity, progressDialog);
        }

        @Override
        protected void onPreExecute() {
            loading = 0;
            super.onPreExecute();
        }

        @Override
        protected void onPostExecute(TaskOutput output) {
            super.onPostExecute(output);
            // Update widgets
            AbstractWidgetProvider.updateWidgets(MainActivity.this);
            DashClockWeatherExtension.updateDashClock(MainActivity.this);
        }

        @Override
        protected ParseResult parseResponse(String response) {
            return parseTodayJson(response);
        }

        @Override
        protected String getAPIName() {
            return "weather";
        }

        @Override
        protected void updateMainUI() {
            updateTodayWeatherUI();
            updateLastUpdateTime();
        }
    }
```

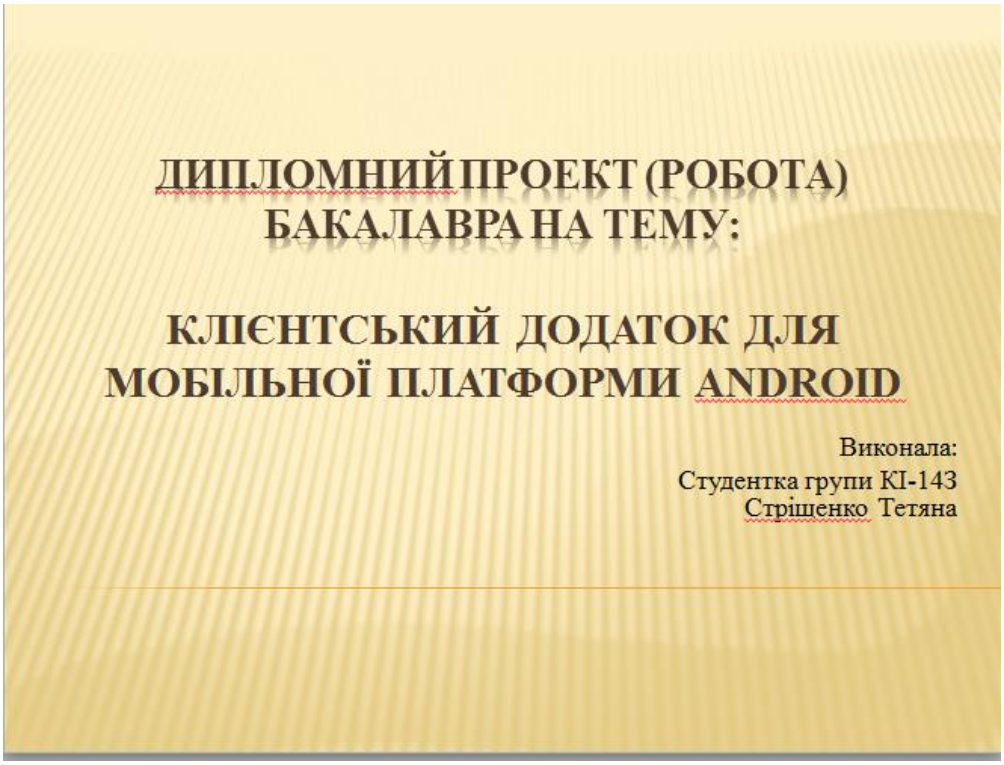
```
    class LongTermWeatherTask extends GenericRequestTask {
        public LongTermWeatherTask(Context context, MainActivity activity,
ProgressDialog progressDialog) {
            super(context, activity, progressDialog);
        }

        @Override
        protected ParseResult parseResponse(String response) {
            return parseLongTermJson(response);
        }

        @Override
        protected String getAPIName() {
            return "forecast";
        }

        @Override
        protected void updateMainUI() {
            updateLongTermWeatherUI();
        }
    }

    class ProvideCityNameTask extends GenericRequestTask {

        public ProvideCityNameTask(Context context, MainActivity activity,
ProgressDialog progressDialog) {
            super(context, activity, progressDialog);
        }

        @Override
        protected void onPreExecute() { /*Nothing*/ }

        @Override
        protected String getAPIName() {
            return "weather";
        }

        @Override
        protected ParseResult parseResponse(String response) {
            Log.i("RESULT", response.toString());
            try {
                JSONObject reader = new JSONObject(response);

                final String code = reader.optString("cod");
                if ("404".equals(code)) {
                    Log.e("Geolocation", "No city found");
                    return ParseResult.CITY_NOT_FOUND;
                }

                String city = reader.getString("name");
                String country = "";
                JSONObject countryObj = reader.optJSONObject("sys");
                if (countryObj != null) {
                    country = ", " + countryObj.getString("country");
                }

                saveLocation(city + country);

            } catch (JSONException e) {
                Log.e("JSONException Data", response);
                e.printStackTrace();
                return ParseResult.JSON_EXCEPTION;
            }
```

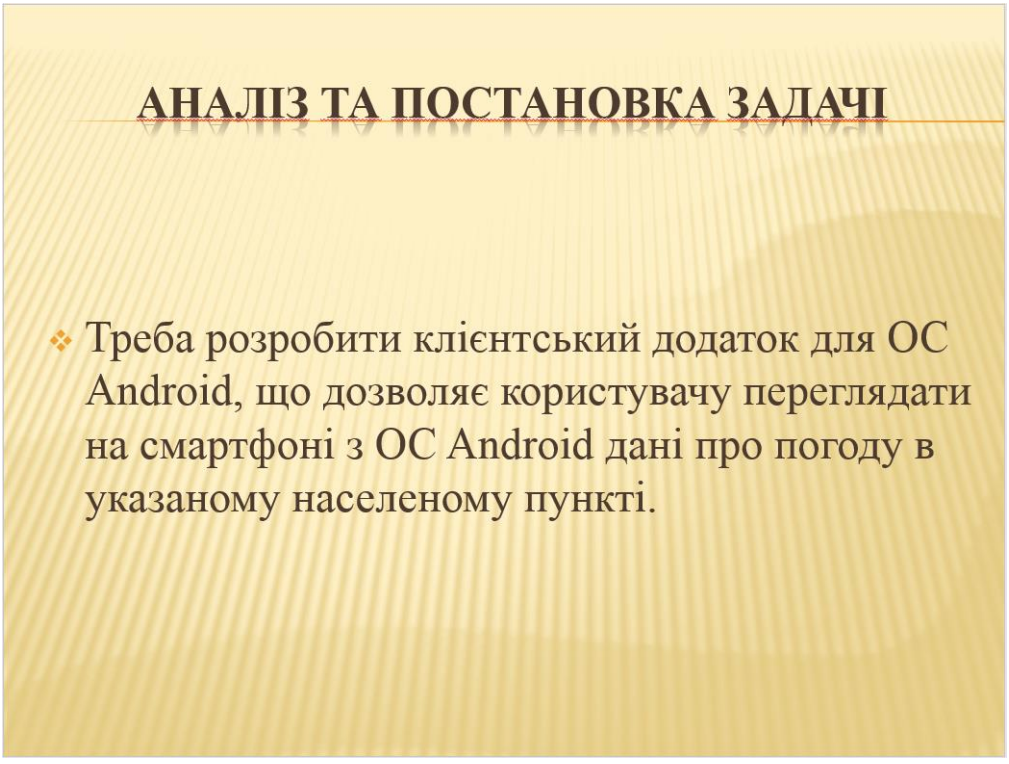
```
            return ParseResult.OK;
        }

        @Override
        protected void onPostExecute(TaskOutput output) {
            /* Handle possible errors only */
            handleTaskOutput(output);
        }
    }

    public static long saveLastUpdateTime(SharedPreferences sp) {
        Calendar now = Calendar.getInstance();
        sp.edit().putLong("lastUpdate", now.getTimeInMillis()).apply();
        return now.getTimeInMillis();
    }

    private void updateLastUpdateTime() {
        updateLastUpdateTime(

PreferenceManager.getDefaultSharedPreferences(this).getLong("lastUpdate", -1)
        );
    }

    private void updateLastUpdateTime(long timeInMillis) {
        if (timeInMillis < 0) {
            // No time
            lastUpdate.setText("");
        } else {
            lastUpdate.setText(getString(R.string.last_update,
formatTimeWithDayIfNotToday(this, timeInMillis)));
        }
    }

    public static String formatTimeWithDayIfNotToday(Context context, long
timeInMillis) {
        Calendar now = Calendar.getInstance();
        Calendar lastCheckedCal = new GregorianCalendar();
        lastCheckedCal.setTimeInMillis(timeInMillis);
        Date lastCheckedDate = new Date(timeInMillis);
        String timeFormat =
android.text.format.DateFormat.getTimeFormat(context).format(lastCheckedDate);
        if (now.get(Calendar.YEAR) == lastCheckedCal.get(Calendar.YEAR) &&
                now.get(Calendar.DAY_OF_YEAR) ==
lastCheckedCal.get(Calendar.DAY_OF_YEAR)) {
            // Same day, only show time
            return timeFormat;
        } else {
            return
android.text.format.DateFormat.getDateFormat(context).format(lastCheckedDate) +
" " + timeFormat;
        }
    }

    private int getTheme(String themePref) {
        switch (themePref) {
            case "dark":
                return R.style.AppTheme_NoActionBar_Dark;
            case "black":
                return R.style.AppTheme_NoActionBar_Black;
            case "classic":
                return R.style.AppTheme_NoActionBar_Classic;
            case "classicdark":
```

```
                return R.style.AppTheme_NoActionBar_Classic_Dark;
            case "classicblack":
                return R.style.AppTheme_NoActionBar_Classic_Black;
            default:
                return R.style.AppTheme_NoActionBar;
        }
    }
}
```

Додаток Б

Комп’ютерна презентація





ОБГРУНТУВАННЯ НЕОБХІДНОСТІ ВИКОРИСТАННЯ ОС ANDROID.

- Мобільний смартфон - це копія комп'ютера, яку завжди можна мати при собі. В наш час, найпоширенішою є операційна система (ОС) Android. Головна причина поширення ОС Android - безкоштовні засоби розробки, в той час як розробка під систему IOS вимагає високих початкових витрат.

РОЗРОБКА ПРОГРАМИ

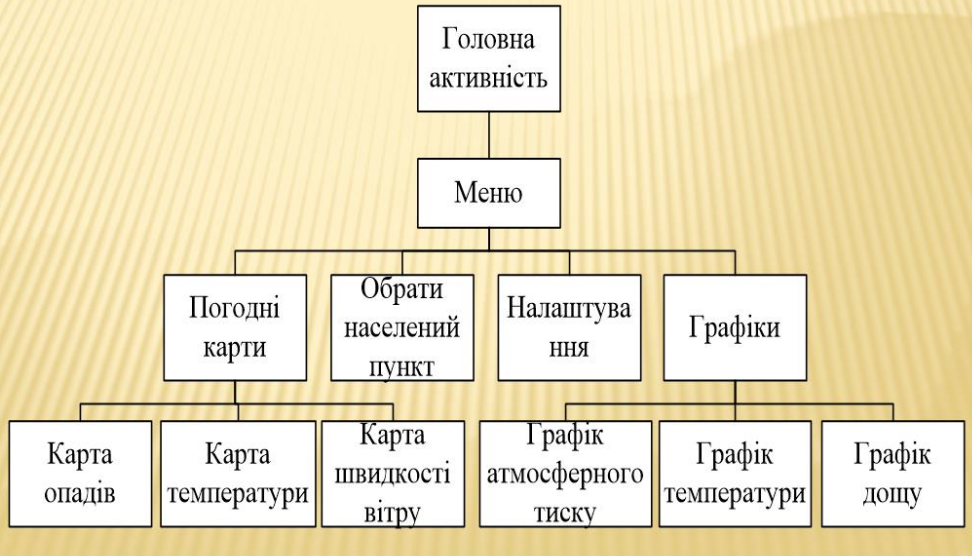
- Клієнт–серверний додаток на базі ОС Android відображує поточні погодні умови у вказаному населеному пункті.

Основні функції додатка:

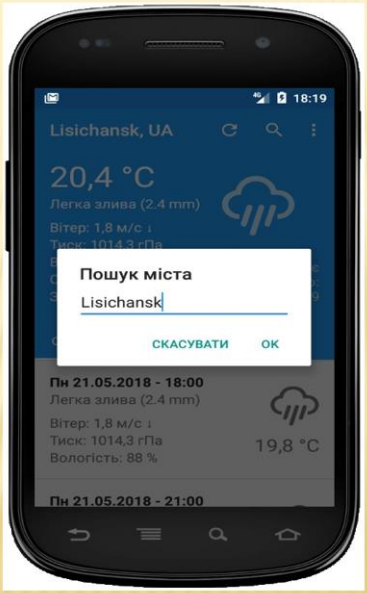
- відображення у зручному форматі температури, атмосферного тиску, опадів, тощо;
- можливість пошуку населеного пункту за його назвою
- зберігання інформації про останній населений пункт до наступного запуску програми
- отримання даних і відображення погодних умов на наступні декілька діб;
- відображення карт температури, швидкості вітру, опадів.

ІНТЕРФЕЙС ПРОГРАМИ

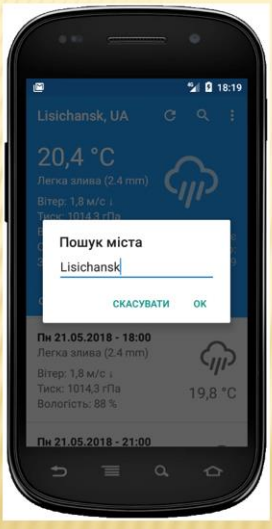
СТРУКТУРНА СХЕМА ДОДАТКУ



Пошук відомостей про населений пункт

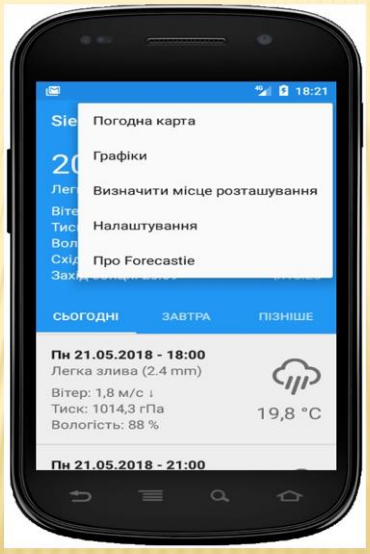


ПОШУК ВІДОМОСТЕЙ ПРО НАСЕЛЕНИЙ ПУНКТ



- Якщо в меню обраний значок пошуку, то створюється і виводиться на екран діалогове вікно AlertDialog. Воно є розширенням класу Dialog і використовується, коли потрібен діалог з кнопками «Так» і «Ні»

ОСНОВНЕ МЕНЮ ПРОГРАМИ



СТРУКТУРА КЛАСУ WEATHER

- Клас Weather отримує інформацію з сервісу OpenWeatherMap. Ця інформація згодом відображається активностями клієнтського додатку

ОХОРОНА ПРАЦІ

Організація робочого місця з ПК повинна враховувати вимоги безпеки, зручність положення, рухів і дій працівника:

- Робочий стіл з урахуванням характеру виконуваної роботи повинен мати достатній розмір для раціонального розміщення монітора (дисплея), клавіатури, іншого використовуваного обладнання і документів.
- Клавіатура розташовується на поверхні столу таким чином, щоб простір перед клавіатурою було достатнім для опори рук працівника.
- Площина екрану монітора розташовується нижче рівня очей працівника переважно перпендикулярно до нормальної лінії погляду працівника.
- Робочий стілець повинен бути стійким, місце сидіння має регулюватися по висоті, а спинка сидіння - по висоті, кутах нахилу, а також відстані спинки від переднього краю сидіння. Регулювання повинно бути легко здійснюваним плюс надійно фіксованим.

ВИСНОВКИ

Тільки додатки можуть зробити будь-яку операційну систему придатною для роботи, розваги, виходу в Інтернет, перегляду веб-сторінок і багато чого іншого, що перетворює звичайний телефон в маленький кишеньковий комп'ютер з повним набором функціональних можливостей.

В результаті виконання даного дипломного проекту вирішені завдання, які були поставлені на початку роботи. Булв розроблен додаток основні можливості якого: відображення погодних умов на декілька днів у вказаному населеному пункті, побудова графіків зміни погодних показників, отримання від серверу і перегляд карт погоди.