Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет радіоелектроніки та телекомунікацій
(повне найменування інституту, назва факультету)
Кафедра інформаційних технологій електронних засобів
(повна назва кафедри)

Пояснювальна записка

до магістерської роботи

Магістр
(ступінь вищої освіти (освітній ступінь))

на тему Система комплексного збору метеоданих з урахуванням пожеж

Виконав: студент VI курсу, групи РТ-513м
спеціальності (спеціалізації)
172 "Телекомунікації та радіотехніка"
(код і назва спеціалізації, спеціальності)

Калоєв К.С.
(прізвище та ініціали)

Керівник Шило І.М.
(прізвище та ініціали)

Рецензент Зеленьова І.Я.
(прізвище та ініціали)

м.Запоріжжя
2018 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет радіотехніки та телекомунікацій
Кафедра інформаційних технологій видавничих робіт
Спеціальність 172 "Телекомунікації та радіотехніка"

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТВР
к.т.н. доц. Шило Г.М.
«11» грудня 2018 р.

ЗАВДАННЯ

На дипломний проект (роботу) студента групи РТ-513м ОКР _____

Колобову Костянтину Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Система комплексного збору метеоданих з прогнозуванням погоди,
затверджена наказом ЗНТУ від «7» листопада 2018 р. № 338.

2. Термін подання студентом закінченого проекту (роботи)_____

3. Вихідні дані для проекту (роботи) сервер з даними погодних умов openweathermap.org, сервер з даними екологічних і погодних умов i-ted.tk

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити) Вибір обладнання, розробка і постановка завдання, розробка ієрархічної взаємодії системи з серверами зі збору екологічних умов, розробка програмного забезпечення, економічна ефективність науково-технічного проекту, охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу Плакат 1 Склад алгоритмів;
Плакат 2 Блок-схема алгоритму програми;
Плакат 3 Інтерфейс розробленої програми;
Плакат 4 Склад таблиць бази даних.

6. Перелік програмних продуктів, які належить використати в процесі розроблення проекту (роботи) login activity, register activity, weather activity, maps activity, statistic activity, database activity

7. Консультування проекту (роботи), із зазначенням розділів

Розділ	Консультант	Завдання видав		Завдання прийн	
		підпис	дата	підпис	да
1-2	Шило Т.М.				
3	Шило Т.М.				
4	Остапенко В.В.				
5	Коробко О.В.				
Нкон	Васильєв Т.О.				19.1

8. Дата, коли видано завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів проекту (роботи)	Термін виконання етапів проекту (роботи)	Прим
1	Огляд обмежень, розробка і постановка завдань	1 тиждень	
2	Розробка і опис перевірок вхідних даних з урахуванням обмежень	3 тижні	
3	Розробка програмного забезпечення	3 тижні	
4	Тестування отриманого програмного продукту	2 тижні	
5	Вихідна кваліфікаційна робота в надзвичайній ситуації	2 тижні	

Завідувач кафедри _____
(підпис)

Керівник _____
(підпис)

Студент _____
(підпис)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту: 125 сторінок, 15 рисунків, 13 таблиць, 14 джерел, 3 додатки.

Об’єкт досліджень: Система комплексного збору метеоданих з прогнозуванням погоди.

У першому розділі проводиться огляд області розробки та постановка завдань дипломного проекту, дослідження необхідності моніторингу погодної та екологічної обстановки.

У другому розділі проводиться розробка і опис переваг взаємодії системи з сервісами зі збору кліматичних даних, розгляд протоколів обміну даними з метеорологічними серверами, обґрунтування використання даних про екологічний фон в системі прогнозування погоди.

У третьому розділі обґрунтовуються вибір платформи для розробки програмного забезпечення, розробляється інтерфейс та робочі модулі програми.

У четвертому розділі проводиться розрахунок економічної ефективності та теоретичної окупності науково-дослідницького проекту.

У п'ятому розділі розглядаються питання з охорони праці та безпеки в надзвичайних ситуаціях.

ПОГОДА, ЕКОЛОГІЧНИЙ ФОН, БЛОК-СХЕМА, АЛГОРИТМ, ПРОГРАМА, БАЗА ДАНИХ, СЕРВЕР, ANDROID

ЗМІСТ

ВСТУП

Однією із проблем індустріальних міст України та світу є високий рівень забруднення атмосфери викидами від промислових підприємств.

Екологічні проблеми Запоріжжя:

- 150 тисяч тон шкідливих речовин від промислових підприємств;
- техногенне навантаження на одного жителя міста становить 191 кг викидів забруднюючих речовин;
- перевищення рівня марганцевих сполук, оксиду алюмінію, міді та хлору.

У дипломному проекті пропонується розробка термінального програмного додатку для ОС Android для висвітлення погодного стану на визначеній території та аналізу екологічного стану на основі даних отриманих з датчиків екологічного фону.

Проект здатний задовольнити потреби як простого користувача, давши йому всю необхідну йому інформацію про навколишній екологічний фон, так і підприємств, дозволяючи відокремити їх датчики за допомогою системи аккаунтів, проте при цьому об'єднати їх дані з даними пристроїв у відкритому доступі, і отримати при цьому ще більш точну усереднену картину поточної екологічної обстановки.

Також програма здатна надати не тільки загальну картину, але і допомогти проаналізувати тенденцію екологічного фону за допомогою побудови графіків, і допуску користувача до записів в базі даних.

1 ОГЛЯД ОБЛАСТІ РОЗРОБКИ І ПОСТАНОВКА ЗАВДАНЬ

1.1 Застосування систем віддаленого моніторингу датчиків кліматичних умов і екологічного фону.

Згідно карти викидів забруднюючих речовин в атмосферне повітря, яку можна знайти на рис. 1.1, проблема забруднення повітря, особливо на сході є дуже гострою, тому розробка методів реєстрації та відстежування викидів шкідливих речовин дуже важлива та актуальна.

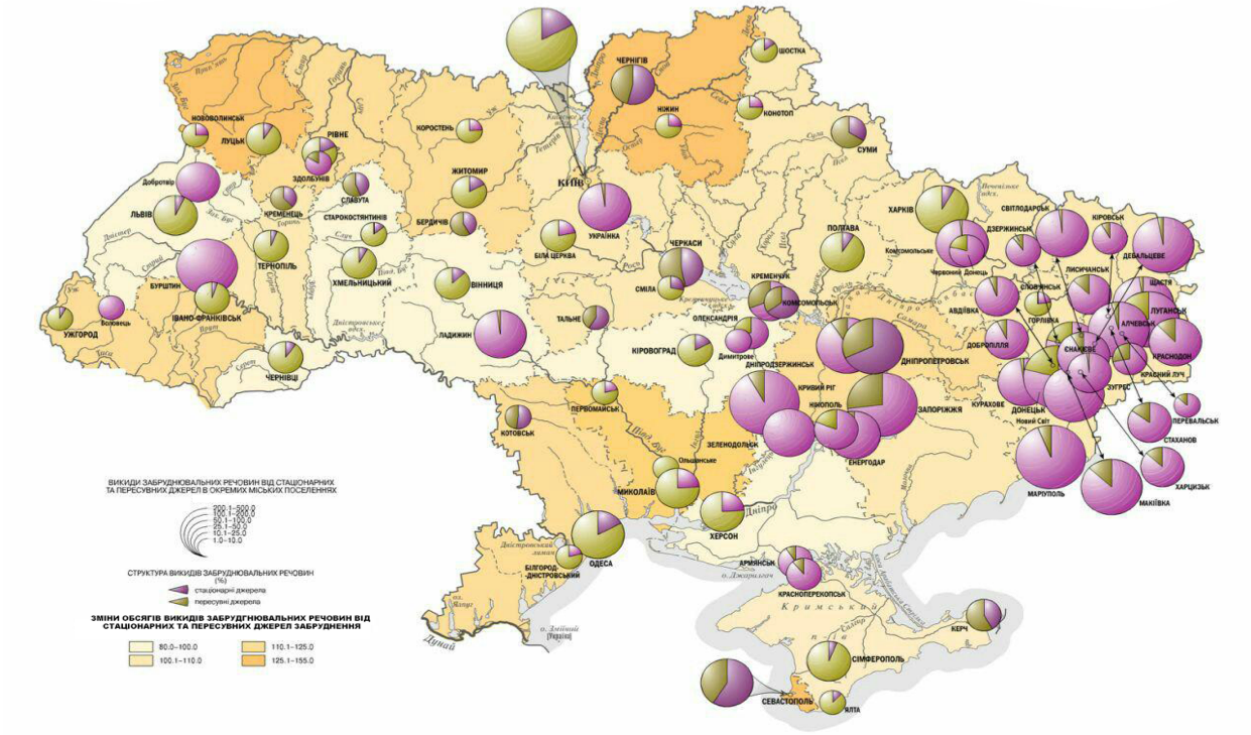

Рисунок 1.1 – Карта викидів забруднюючих речовин в атмосферне повітря. Україна.

Також про поганий екологічний фон свідчать приведені в таблиці 1.1 дані про викиди забруднюючих речовин від стаціонарних джерел.

Будь-який прогноз погоди складається одночасно для величезної території: міста, області, краю і т. П. Сучасні метеостанції оснащені хорошим обладнанням, але розташовані на дуже великій відстані один одного, що істотно впливає на загальну точність прогнозування. Збільшувати щільність метеостанцій занадто дорого, ось і виходить, що нам доводиться задовольнятися приблизними, усередненими даними. І в черговий раз раптово змінювати свої рішення.

Таблиця 1.1 - Викиди забруднюючих речовин від стаціонарних джерел.

	2013	2014	2015	2016	2017
Усього	90.5	104.9	103.7	103.0	117.0
У тому числі:					
Тверді речовини	39.8	42.2	37.9	39.9	38.3
Газоподібні та рідкі речовини	50.7	62.7	65.8	63.1	78.7
З них:					
Діоксид сери	20.0	18.4	19.1	16.8	21.1
Оксид азоту	4.5	7.1	7.1	7.3	7.8
Оксид вуглецю	24.1	34.4	37.6	36.0	46
Вуглеводні	0.2	0	0.3	0.3	0.8
Летючі органічні сполуки	0.1	0.2	0.2	0.2	0.3

Приватна, локальна метеостанція може становити прогноз погоди конкретно для вашої території. Така станція оцінює стан середовища на кілька днів вперед. Автономний екологічний метеокомплекс стежить за екологічними і метеорологічними показниками і відображає цю інформацію на екрані Вашого комп'ютера, планшета або мобільного телефону Web-інтерфейс дозволяє спостерігати поточний стан метеорологічних і екологічних показників, формувати звіти і графіки по різним групам параметрів за вибраний проміжок часу, проводити аналіз статистичної інформації та багато іншого. Має функцію прогнозу погоди, яка враховує координати місця розташування комплексу, час року, поточне значення атмосферного тиску і його зміна, швидкість і напрям вітру, температуру і вологість повітря, кількість опадів, стану ґрунту, забруднення повітря та радіаційного фону.

Інформація, регулярно надходить на основі показань метеокомплекс, служить одним з важливих чинників забезпечення безпеки і запобігання нещасним випадкам та аварійних ситуацій.

Метеостанції активно використовуються науковими співробітниками і студентами галузевих вузів і факультетів, як геологорозвідувальні, географічні, екологічні, океанологічні, спеціалізованими коледжами та школами.

Метеокомплекси застосовуються у всіх галузях сільського господарства - для обладнання теплиць, тваринницьких ферм, сільськогосподарських угідь, садівничих господарств, плодово-овочевих баз.

Установка метеостанції в вашому котеджі або заміському будинку - це особистий і абсолютно точний прогноз погоди. Система обміну метеоінформацією через Internet дозволить вам бути в курсі погоди в вашому домі в будь-якій точці земної кулі! Метеокомплекс умовно розділений на 3 рівні: нижній середній і верхній:

1) Нижній (польовий) рівень являє собою апаратний комплекс екологічного та метеорологічного спостереження. У нього входить метеостанція погодна, система контролю стану ґрунту, набір датчиків забруднення повітря, датчики радіаційного фону; автономний телеметричний контролер шафового типу який здійснює отримання даних з усіх вимірників, датчиків і передачу даних по бездротовому каналу GSM / GPRS на віддалений сервер збору і обробки даних; система автономного живлення на сонячних батареях; акумуляторний відсік і контролер заряду АКБ.

2) Середній рівень являє собою «хмарні» сервери прийому даних з віддалених об'єктів. Встановлене на них програмне забезпечення відкриває Інтернет-з'єднання з контролером телеметрії та отримує дані з метеокомплекса в зашифрованому форматі. Отримані свідчення з комплексу обробляються і поміщаються в базу даних для тривалого зберігання і надання в інтерфейс користувача.

3) Верхній рівень складається з Інтернет-додатки «Web-сервер Монітор телеметрії», який надає доступ до Web-інтерфейсу комплексу на будь-якому Інтернет-браузері ПК або мобільного пристрою типу IPad. Користувач може бачити поточний стан метеорологічних і екологічних показників, формувати звіти за різними групами параметрів за вибраний проміжок часу, проводити аналіз статистичної інформації та багато іншого.

В даний час в промисловості, транспорті, енергетиці та інших сферах діяльності людини існує потреба в контролі за станом різних складних технічних систем, в тому числі і для забезпечення їх екологічної безпеки. При цьому в залежності від сфери, в якій працює те чи інше підприємство або організація, можуть висуватися різні вимоги до засобів моніторингу та забезпечення комплексної безпеки. Одна з найважливіших причин створення універсальної платформи комплексного моніторингу та забезпечення комплексної безпеки, це необхідність створити систему здатну без

істотних змін здійснювати централізований контроль стану різних технічних і екосистем, а також здійснювати збір, аналіз інформації і видачу рекомендацій експлуатує її персоналу.

Однак, можливості застосування подібної системи не обмежуються тільки будинками або підприємствами. З урахуванням модульності системи і взаємозамінності елементів, одним з основних призначень системи комплексної безпеки (СКБ) може стати моніторинг лісових пожеж. Адже екологічні наслідки від лісових пожеж досить значні: відбувається забруднення атмосферного повітря вуглекислим газом і продуктами піролізу лісових горючих матеріалів, вигоряння кисню. З лісовими пожежами в повітря потрапляють частинки сажі, що складаються з вуглецю і продуктів неповного згоряння деревини. Задимлення повітря призводить до погіршення мікроклімату землі; збільшення числа туманних днів, зменшення прозорості атмосфери і зумовленого ним зниження видимості, освітленості, ультрафіолетової радіації. І навіть дуже малі концентрації деяких речовин є досить небезпечними.

1.2 Огляд існуючих аналогів.

Жоден з відомих сервісів з надання даної в даному дипломному проекті не надає в повній мірі ту інформаці, яку здатний зібрати розроблений раніше датчик, який використовується в магістерській робота.

У таблиці 1.2 наведені деякі аналоги сенсорів, які виробляють виміри метеорологічних і / або екологічних показників.

Таблиця 1.2 - Аналоги сенсорів, які виробляють виміри метеорологічних і / або екологічних показників.

Назва	Вимірювані параметри	Тип датчиків	Зв'язок з ПК	Віддалена передача даних	Ціна, у.о.
ADTEK HTO	CO, CO_2, t°, вологість	недисперсійний інфрачервоний аналізатор (NDIR сенсор)	RS-485	-	300-320
RFTF-CO2	CO_2, t°, вологість	NDIR сенсор	-	-	380-400
PolyGard® IAQ-S	CO_2	Біо-напівпровідник	RS-485	-	200-220
Lifecontrol	CO, летючі органічні речовини, t°, вологість	Невідомо (в документації не вказано)	Віддалено	+ мобільний додаток або сайт	100
КОМЕТА-М	CH_3OH, CO_2, O_2, CH_4, CO, C_2H_5OH, H_2S, NH_3, Cl_2, H_2, C_3H_8, NO_2, SO_2, HCl (різні комплектації по 5 газів)	Побудовані на зміні опору термокаталітичного або напівпровідникового сенсора	З'ємна карта пам'яті	-	2000-3000

Розглядаючи таблицю 1.2 можна зробити висновок про дорожнечу і обмеженості функціонала вищеописаних датчиків, що виділяє розроблений і використовується в магістерській роботі датчик з позитивного боку. Найочевиднішим гідністю є використання надійних і зручних методів бездротового зв'язку і додатки під рамую розповсюдженими мобільну систему.

Однак додаток з цієї магістерської роботи не унікальна, і є сервіси мають в своєму розпорядженні схожим функціоналом. Приклади сервісів можна знайти в таблиці 1.3.

Таблиця 1.2 - аналоги сервісів, що надають іформацію про метеорологічних і / або екологічних показниках.

Назва	Вимірювані параметри	Країни	Платформа
AirVisual	SO_2, CO, NO_2	Весь світ	Google Android, Apple IOS, Web
Kanarek	SO_2, CO, C_6H_6	Польща	Google Android, Web
Plume Air Report - Live and forecast smog reports	CO_2 ,t^o	Якість повітря лише у США	Google Android, Apple IOS

Перераховані вище проекти мають невеликі охоплення територій навіть в країнах, де їх розробили, и тім більше їх вплив на розвиток наших систем аналізу екологічної обстановки практично відсутній.

На підставі цього можна зробити висновок, що в поточному своєму вигляді смороду практично повністю даремні для жителів України, та розробка свого сервісу доцільна.

1.3 Постановка завдань магістерської роботи.

Під час виконання дипломного проекту планується створення програмного комплексу з наступним функціоналом:

- реєстрації нового аккаунта користувача;
- авторизація в системі, використовуючи існуючі в базі даних записи;
- огляд поточної погодної та екологічної ситуації використовуючи дані з сервера, який акумулює інформацію з датчиків;
- будівництво графіків екологічних обставин;
- перегляд бази даних;
- вивід позицій датчиків на мапу.

Створення програми для смартфона є користувальницької частиною великого проекту по створенню ефективної, зручної і гнучкої системи з відстеження точних даних метеорологічної та екологічної обстановки.

Завданням магістерської роботи стала розробка мобільного додатка, здатного як в простій і зрозумілій формі відобразити наявні на сервері дані, донісши до користувача інформацію про погодні та екологічних умовах в місті його проживання, так і дати можливість вивчити ці дані в максимально докладному вигляді.

У програмі повинен бути присутнім функціонал, що дозволяє користувачам реєструвати власний аккаунт, використовуючи його безпосередньо в програмі, а так же набір засобів для надійного зберігання призначених для користувача даних.

Додаток має надавати загальне зведення погодних і екологічних даних у зручному привабливому форматі.

Для тих, хто хоче зібрати більш докладні дані, додаток повинен вміти будувати графіки, надаючи можливість оцінити динаміку зміни екологічної ситуації під час проведення вимірювань, а так само дати можливість вручну переглянути дані будь-якого запису будь-якого сенсора в базі даних програми.

Для наочності відображати працюють датчики на карті, використовуючи API одного з картографічних сервісів.

2 РОЗРОБКА І ОПИС ПЕРЕВАГ ВЗАЄМОДІЇ СИСТЕМИ З СЕРВІСАМИ ЗІ ЗБОРУ КЛІМАТИЧНИХ ДАНИХ.

2.1 Протоколи обміну даними з метеорологічними серверами.

Основним джерелом даних для визначення погодних і екологічних умов є розроблений раніше датчик екологічної обстановки, розміщений в одному з центральних районів міста.

На сервері дані зберігаються у вигляді MySQL таблиці, і мають такий набір комірок:

Id – номер запису по-порядку. Дозволяє сортувати записи в таблиці за номером, і вибудовувати структуру черговості. Дані зберігаються у вигляді INTEGER.

dev_id – унікальний номер датчика. Дозволяє виділяти дані з кожного окремого датчика, і висилати дані відповідно до номером запитаного датчика, відсікаючи необхідність пересилання всієї бази даних, а так само явно вводити нові датчики в систему «на льоту», без необхідності перебудови системи, і повністю інтегруючи його як в базу даних, так і безпосередньо в робочі алгоритми сервера. Дані зберігаються у вигляді INTEGER.

smoke – показник датчика запиленості і диму. Так, як датчик працює з прозорістю повітря, його показники можуть вказувати як на перший вид забруднення, так і на другий. Дані зберігаються у вигляді INTEGER.

methane1 – тут зберігаються показання першого датчика реєстрації природних газів. Залежно від налаштувань, датчик здатний вимірювати широкий спектр природних газів. Дані зберігаються у вигляді INTEGER.

methane2 – місце зберігання показників другого, аналогічного датчика реєстрації природних газів. Залежно від налаштувань, датчик здатний вимірювати широкий спектр природних газів. Дані зберігаються у вигляді INTEGER.

Carbon – місце зберігання показань з датчика вуглецю. Дані зберігаються у вигляді INTEGER.

LPG - місце зберігання показань з датчика пропан-бутану. Дані зберігаються у вигляді INTEGER.

humidity – в даному осередку зберігаються дані з датчика вологості. Дані зберігаються у вигляді INTEGER.

hydrogen – комірка для зберігання даних з датчика водню. Дані зберігаються у вигляді INTEGER.

temp - осередок для зберігання даних з датчика температури навколишнього середовища. Дані зберігаються у вигляді INTEGER.

datetime – дата і час отримань даних з сервера. Дані зберігаються у вигляді TEXT.

Для отримання даних з сервера необхідно сформувати запит по одному з шаблонів:

http://api.ited.tk/?end=INTEGER.

Де INTEGER замінюється за потрібне нам кінцевим номером запису в серверній базі даних. До цьому випадку ми отримуємо заздалегідь відформатований файл формату .json масив записів в форматі JSONArray, в якому фігуруватиме набір даних, який можна розпарсити в формат JSONObject, і підбити під потрібну нам в додатку структуру даних.

Другою особливістю подібного запиту є те, що якщо поставити INTEGER1 вище, ніж номер останнього запису в базі даних, то сервер зрозуміє це як необхідність локалізувати дані з id = 0 до id = id останнього запису в базі даних.

http://api.ited.tk/?start= INTEGER 1&end= INTEGER2

Де INTEGER1 замінюється за потрібне початковим номером запису в серверній базі даних, а INTEGER2 - за потрібне кінцевим номером запису в північній базі даних. Особливість такого запиту лежить в тому, що ми можемо отримати тільки потрібну нам частину бази даних, локалізувати винятковий потрібний відрізок часу і заощадивши на обсягах отримуваного файлу.

Другою особливістю подібного запиту є те, що як і по-аналогії з попереднім, якщо поставити INTEGER2 вище, ніж номер останнього запису в базі даних, то сервер зрозуміє це як необхідність локалізувати дані з id = INTEGER1 до id = id останнього запису в базі даних.

http://api.ited.tk/?start=INTEGER1&end=INTEGER2&dev=INTEGER3

Де INTEGER1 замінюється за потрібне початковим номером запису в серверній базі даних, а INTEGER2 - за потрібне кінцевим номером запису в північній базі даних.

Особливість такого запиту лежить в тому, що ми можемо отримати тільки потрібну нам частину бази даних, локалізувати винятковий потрібний відрізок часу і заощадивши на обсягах отримуваного файлу.

Другою особливістю подібного запиту є те, що як і по-аналогії з попереднім, якщо поставити INTEGER2 вище, ніж номер останнього запису в базі даних, то сервер зрозуміє це як необхідність локалізувати дані з id = INTEGER1 до id = id останнього запису в базі даних .

Третьою особливістю цього шаблону запиту є те, що ми маємо можливість вибрати виключно потрібний нам датчик, запровадивши його id замість INTEGER 3. Це дозволяє істотно скоротити розмір підвантажуваних даних, сильно збільшити швидкість їх інтерпретацію і перетворення в підготовлену в програмі структуру даних.

http://api.ited.tk/max.php

Відправивши даний запит, у відповідь ми отримуємо номер останнього запису в серверній базі даних, що дозволяє нам без зайвих розрахунків і підрахунків визначити розміри лише у віддаленій базі даних, і після цього відправити запит в потрібних нам рамках.

Основною особливістю цього методу також є те, що ми отримуємо комплексну величину записів, тобто - відразу з усіх датчиків, що дозволяє нам при проведенні сумарних порівнянь провести порівняння локальної бази даних і бази даних на сервері.

Відправивши один з перерахованих вище запитів, виключаючи останній, ми отримуємо масив даних в форматі .json, який представляє з себе набір об'єктів JSONArray.

Приклад запиту:

http://api.ited.tk/?start=1783&end=5000&dev=1

Наведений вище запит - симуляція того, що на нашому пристрої вже є отриманий масив даних з датчика під номером «1», і ми просто хочемо дізнатися, чи є новіші записи. Під номер останнього запису на пристрої беремо "один тисячу сімсот вісімдесят два". Тому щоб перевірити наявність нових, більш свіжих записів, необхідно відправити в якості стартового номера значення, на 1 вище того, що вже є в базі даних.

Приклад відповіді:

[{"id":"1783","dev_id":"1","datetime":"2018-03-18 16:06:03","temp":"29","humidity":"19","methane1":"0","methane2":"10","LPG":"7","Smoke":"50","Hydrogen":"25","Carbon":"0"}]

Як можна побачити, в базі даних сервера дійсно є один новий запис, що має порядковий номер "1783". Для зручності відформатуємо надану вище інформацію:

[
{
"id":"1783",
"dev_id":"1",
"datetime":"2018-03-18 16:06:03",
"temp":"29",
"humidity":"19",
"methane1":"0",
"methane2":"10",
"LPG":"7",
"Smoke":"50",
"Hydrogen":"25",
"Carbon":"0"
}
]

Квадратні дужки ("[]") означають масив даних, і що знаходиться між ними інформація представляє з себе масив, виду JSONArray, що дає нам можливість:

- отримати кількість елементів масиву;

- отримати та інтерпретувати дані в потрібний нам формат як по номеру елемента масиву, так і по черзі пропустивши їх через цикл.

Друге джерело отримання даних - openweathermap.org. Є відкритим майданчиком як для користувачів, так і для розробника, і має велику кількість погодної інформації, якою без проблем ділиться з розробниками.

Для отримання даних про погоду, додаток формує наступний запит:

"Http://api.openweathermap.org/data/2.5/weather?q="+Constants.city+"&APPID=972b2b5d81e3c5e03a141d8568ca8028".

де:

- 2.5 - версія API;

- Constants.city - місто, для якого потрібно отримати прогноз погоди;

- 972b2b5d81e3c5e03a141d8568ca8028 - унікальний ключ розробника, отримати який можна в своєму особистому кабінеті на сайті openweathermap.org.

Ключ являє собою зашифровану послідовність унікального id користувача на сайті, і номера токена, який надає інформацію про цілі отримання даних з сервера погодних умов.

Відповідь, що отримується за запитом http://api.openweathermap.org/data/2.5/weather?q=Zaporizhzhia&APPID=972b2b5d81e3c5e03a141d8568ca8028:

{"coord":{"lon":35.12,"lat":47.85},"weather":[{"id":701,"main":"Mist","description":"mist","icon":"50n"}],"base":"stations","main":{"temp":272.7,"pressure":1028,"humidity":92,"temp_min":272.15,"temp_max":273.15},"visibility":2600,"wind":{"speed":2,"deg":310},"clouds":{"all":40},"dt":1544882400,"sys":{"type":1,"id":8902,"message":0.0018,"country":"UA","sunrise":1544851337,"sunset":1544881643},"id":687700,"name":"Zaporizhzhia","cod":200}

Для зручності відформатуємо отриману відповідь:

{"coord":{"lon":35.12,"lat":47.85},
"weather":[{"id":701,"main":"Mist","description":"mist","icon":"50n"}],
"base":"stations",
"main":{"temp":272.7,"pressure":1028,"humidity":92,"temp_min":272.15,"temp_max":273.15},
"visibility":2600,
"wind":{"speed":2,"deg":310},
"clouds":{"all":40},
"dt":1544882400,
"sys":{"type":1,"id":8902,"message":0.0018,"country":"UA","sunrise":1544851337,"sunset":1544881643},
"id":687700,"name":"Zaporizhzhia",
"cod":200}

Як можна помітити, отриманий масив даних не одновимірний, і вимагає більш опрацьованих методів вилучення даних.

2.2 Принципи та методи прогнозування погоди.

Передбачення погоди з наукової точки зору - одна з найскладніших завдань фізики атмосфери. Існують різні методи для прогнозування метеорологічних явищ і їх величин, але в повному обсязі жоден метод не забезпечує поки точного прогнозу. Є пряма залежність між завчасністю прогнозів і зростанням їх помилок.

Які вирішуються при прогнозі погоди рівняння відносяться до найбільш складним в гідродинаміки. У комплекс прогностичних рівнянь входять рівняння, що описують закони збереження маси, енергії і кількості руху, т. Е. Рівняння нерозривності, рівняння руху, рівняння збереження енергії. Для повного опису реальної атмосфери тут бракує обліку зволоженості, запиленості, турбулентності і ін. При вирішенні системи рівнянь доводиться вдаватися до спрощень. Зокрема, в рівнянні припливу тепла не враховують приплив тепла ззовні, а обмежуються припущенням, що в короткі відрізки часу процеси розвиваються по адіабатичному закону і т. д.

Для кожного конкретного прогнозу записується система рівнянь, причому виробляються деякі спрощення стосовно до поставленого завдання. У рівняннях залишаються тільки головні члени, а другорядні з розгляду виключаються. Рішенням рівнянь знаходяться такі невідомі, як майбутні значення полів тиску, температури, швидкості вітру. Для цього задається початковий стан атмосфери, т. Е. Поля тих же елементів (тиску, температури, швидкості).

Рішення системи прогностичних рівнянь до появи обчислювальної техніки практично було неможливо через неймовірно великої кількості обчислень. Наприклад, для прогнозу погоди на найближчу добу потрібна була б робота декількох тисяч обчислювачів протягом доби.

Зародження сучасних принципів чисельного методу прогнозу погоди можна віднести до початку 20-х років. У 1922 р англійським математиком П, Річардсон була зроблена перша спроба розрахунку зміни атмосферного тиску за шестигодинний період. Ідея полягала в застосуванні астрономічних методів розрахунку до метеорології, хоча було очевидно, що астрономії притаманні прості механічні залежності, а метеорології - закони динаміки і термодинаміки. І не дивно, що розрахований з великими труднощами приклад виявився невдалим. Але прийнятий принцип розрахунку зміни поля тиску по кроках, т. Е. Послідовно через малі проміжки часу, ліг в основу сучасних чисельних методів прогнозу погоди.

Сенс розрахунку по кроках полягає в наступному. Необхідно, наприклад, розрахувати поле тиску на 24 години вперед. Для цього є дані про вихідний поле тиску і про барическу тенденцію, т. Е. Про зміни тиску в різних частинах взятого поля за останні 3 години. Якщо, виходячи з величини тенденції, розрахувати майбутнє поле

тиску відразу на 24 години, то можна зробити грубі помилки, так як зміна поля тиску протікає не з постійною швидкістю. Тому проміжок 24 години розбивається на малі відрізки часу (наприклад, по 2 години) і послідовно проводиться розрахунок очікуваного поля тиску спочатку на перші 2 години; потім за отриманою прогностичної мапі тиску розрахунок ведеться на наступні 2 години і т. д. В результаті на останній прогностичної мапі тиску (після 12 передвичисленням проміжних карт) виявляються врахованими всі можливі зміни баричного поля. Хоча ці зміни враховуються з наближенням, все ж при розрахунку проміжних карт виключаються грубі помилки.

Що застосовуються в даний час швидкодіючі електронні обчислювальні машини, використовуючи допоміжні дані з датчиків виробляють мільйони необхідних арифметичних дій за кілька хвилин. Приклад набору професійних датчиків можна побачити на рис. 2.1

Рисунок 2.1 – старий, але актуальний донині набір датчиків - датчик швидкості вітру, напрямку вітру, температури, вологості, атмосферного тиску, сонячного випромінювання, дощу, коливань.

Синоптичний метод складання прогнозів погоди заснований на аналізі карт погоди. Сутність цього методу полягає в одночасному огляді стану атмосфери на великій території, що дозволяє визначити характер розвитку атмосферних процесів і подальше найбільш ймовірна зміна погодних умов в цікавому районі. Здійснюється такий огляд за допомогою карт погоди, на які наносяться дані метеорологічних спостережень на різних висотах, а також у поверхні землі, вироблених одночасно по одній програмі в різних точках земної кулі. На основі детального аналізу цих карт синоптик визначає подальші умови розвитку атмосферних процесів в певний період

часу і розраховує характеристики метеоелементів - температуру, вітер, хмарність, опади і т.д. Приклад мапи погоди проекту Ventusky можна побачити на рис 2.2.

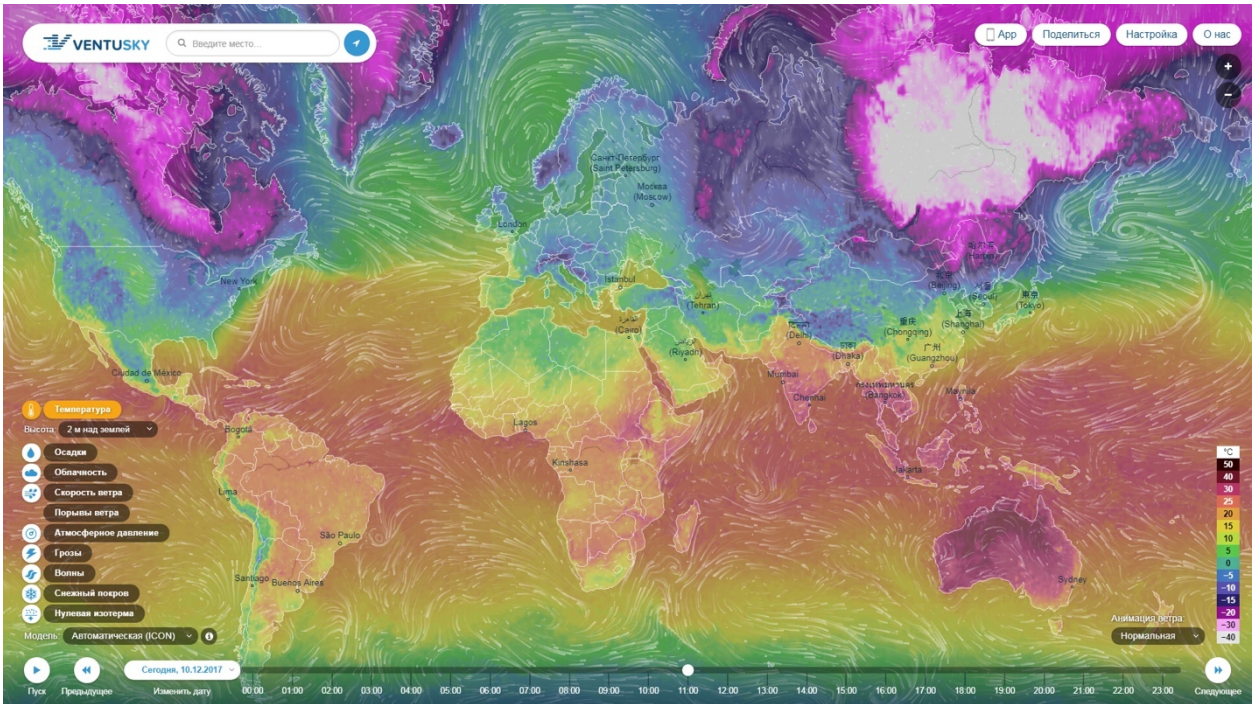

Рисунок 2.2 – Мапа погоди.

Чисельні (гідродинамічні) методи прогнозу погоди засновані на математичному рішенні системи повних рівнянь гідродинаміки і отримання прогностичних полів тиску, температури на певні проміжки часу. Обчислювальні центри використовують різні чисельні схеми розвитку великомасштабних атмосферних процесів. Точність чисельних прогнозів залежить від швидкості розрахунку обчислювальних систем, від кількості та якості інформації, що надходить з метеостанцій. Чим більше даних, тим точніше розрахунок. Якщо технічна і математична сторона методу щорічно поліпшується, то, на жаль, в останні роки на території нашої країни значно зменшилася кількість метеорологічних і аерологічних станцій, що істотно впливає на кінцевий результат.

Статистичні методи прогнозу дозволяють по минулого і сучасного стану атмосфери спрогнозувати на певний майбутній період часу стан погоди, тобто передбачити зміни різних метеоелементів в майбутньому. В оперативній практиці синоптики використовують кілька методів, іноді незбіжних по ряду параметрів. Тому останнє слово завжди залишається за прогнозистом, які вибирають з його точки зору кращий метод прогнозування. Часто вибирається комплексний підхід - використання

відразу декількох приватних методів прогнозу однієї і тієї ж характеристики стану атмосфери з метою вибору остаточного формулювання прогнозу. Приморські синоптики, використовуючи всі передові методи і технології при складанні прогнозів погоди, знання регіональних особливостей розвитку синоптичних процесів, з огляду на унікальність географічного положення краю і спираючись на багаторічний досвід, передбачають погоду з високим ступенем справджуваності.

2.3 Використання даних про екологічний тлі в системі прогнозування погоди.

Однією з головних особливостей розробленої в ході магістерської роботи програми є те, що крім погодних зведень вона надає перелік даних про поточну екологічну обстановку в деякому радіусі навколо себе. Робиться це за допомогою декількох підключених до пристрою датчиків.

Експериментальний пристрій для магістерської роботи має наступних набір датчиків:

- датчик запиленості і диму, який працює з прозорістю повітря, його показники можуть вказувати як на перший вид забруднення, так і на другий;
- два датчика реєстрації природних газів, який залежно від налаштувань, датчик здатний вимірювати широкий спектр природних газів;
- датчик реєстрації оксидів вуглецю;
- датчик реєстрації пропан-бутану;
- цифровий датчик температури і вологості DHT22;
- датчик водню;

Завдяки використанню цієї апаратури, і аналізу надходять з неї даних, ми можемо не тільки мати загальну картину екологічної обстановки в даний момент, але і аналізувати її розвиток, прогнозувати області, перебування на яких буде потенційно шкідливо для здоров'я.

Ці дані використовуються в двох напрямках.

Побудова на їх основі графіків. Мобільний додаток підтримує одночасне відображення до десяти записів, і дозволяє управляти рамками, в яких проводиться побудова вершин.

Відображення на карті актуальною екологічної ситуації у вигляді міток

на координатах датчиків, і спрайтових плям там, де імовірно буде основне вогнище забруднення найближчим часом. Прогнозування проводиться при комбінації даних з погодних серверів.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка блок-схеми алгоритму.

Щоб пояснити загальні принципи функціонування програми, використовується блок-схема алгоритму. Ця схема не містить в собі детального опису роботи кожного сценарію роботи програми, проте структурно представляє загальну схему алгоритму основної частини програми, на яку буде звертати увагу основна маса користувачів.

Блок-схема алгоритму програми зображена на рис. 3.1.

Початок – вхідна точка в програму.

Авторизація – частина, в якій користувач вводить свої дані, або реєструє собі новий акаунт.

Установлення з'єднання з сервером – підтвердження дієздатності робочого сервера датчиків. Якщо сервер не доступний – виводиться повідомлення про помилку

З'єднання з сервером погоди – отримання даних погоди з відкритих джерел.

Запрос координат – підтвердження координат користувача для коректного визначення даних у місті користувача.

Запит температури та рози вітрів – статистичні дані для побудови статистики та прогнозів.

З'єднання з сервером екологічних датчиків та цикл далі - щоби продовжити роботу, програма має получити дані з робочого сервера. Отримання йде циклом, для кожного датчика в списку окремо, після чого основуючись на спільних даних йде розрахунок середніх показників погодних та екологічних умов.

Занесення в даних в базу даних – щоб не робити повторних загрузок та уникнути зайвої трати інтернет трафіка та ресурсів телефону, програма має власну базу даних, в яку заносяться записи даних з датчиків.

Данні погодних умов та погодного фона згідно координат – вивід даних на інтерфейс користувача.

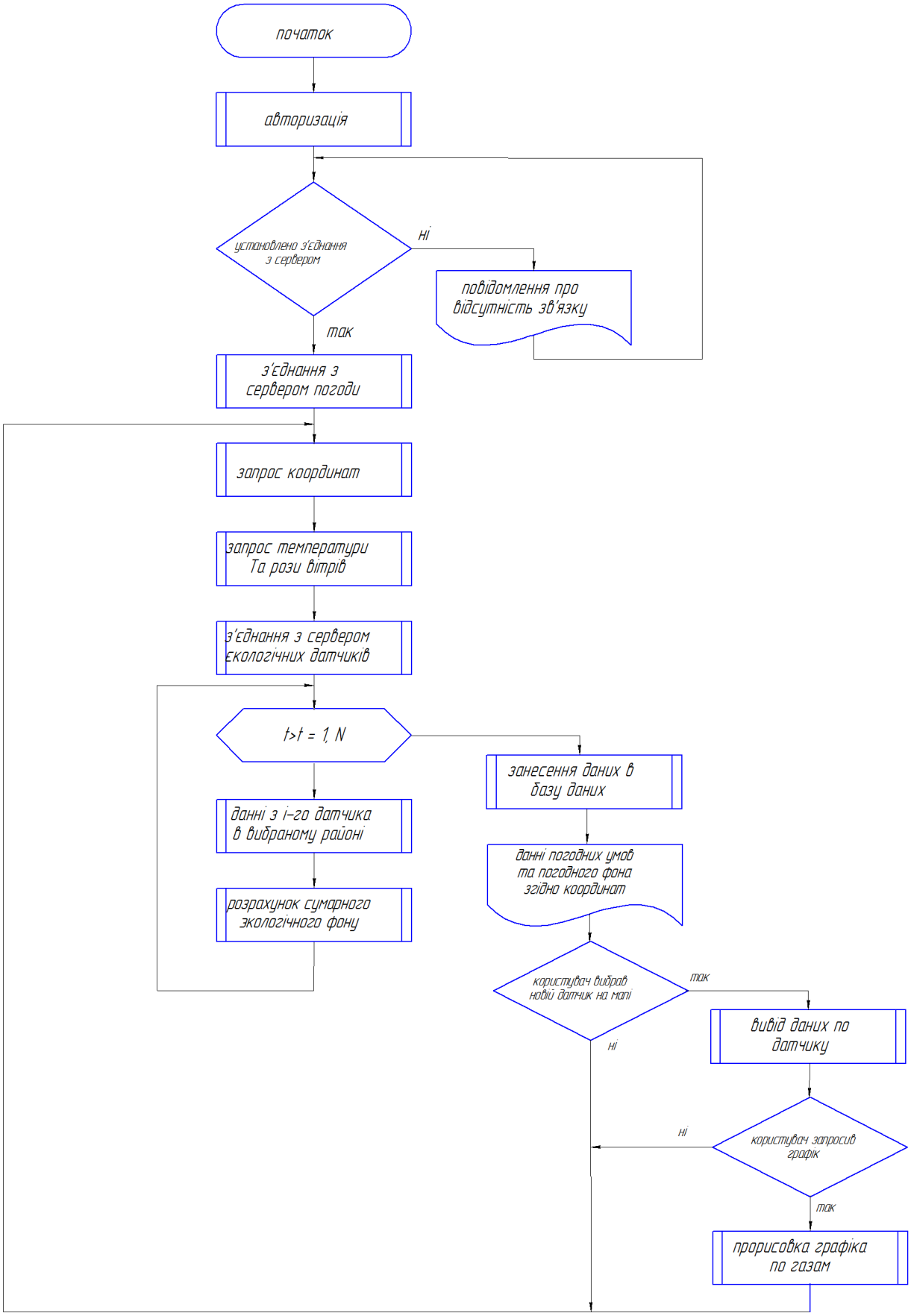

Рисунок 3.1 – Блок-схема алгоритму програми.

Якщо користувач запросив дані у вигляді графіка, або списку, цикл повторюється з моменту запиту координат, а користувач отримує дані у вибраному вигляді.

3.2 Вибір засобів розробки.

В якості основної робочої платформи була обрана операційна система Google Android.

Обрана платформа дає нам переваги наведені нижче.

Простота освоєння "рідного" мови програмування java.

Доступ до широкого діапазону пристроїв (більше 86% смартфонів для України працюють під управлінням OS Google Android версії 4.4 і вище).

Низький поріг входження. Пристрій під керуванням Google Android з адекватним "залізним" оснащенням коштують від 100 доларів, проти 450 у смартфонів конкурента - Apple. До цього списку не були включені смартфони під управлінням Windows в силу того, що система паче не підтримується офіційним розробником і вважається мертвою. Її охоплення на стільки малий, що закупівля пристроїв під управлінням цієї OS не має ніякого сенсу.

Дешевизна аккаунта розробника. 25 доларів довічно проти 100 доларів в рік у Apple.

Можливість розробки програмного забезпечення під будь-який популярної настільної платформою - Windows, Linux, MacOS X. У конкурента в особі Apple офіційної платформою залишається тільки MaxOS X. І не дивлячись на наявність можливості використовувати таке стороннє IDE, як Xamarin, додатки підписуються тільки на комп'ютерах Apple, вартість яких починається від 800 доларів.

Для розробки програми була обрана наступна середовище розробки – Android Studio версії 3.2.

Рішення це мотивується наступними перевагами:

- Android Studio — інтегроване середовище розробки виробництва Google, за допомогою якої розробникам стають доступні інструменти для створення додатків на платформі Android OS;

- Android Studio можна встановити на Windows, Mac и Linux;

- після установки Android Studio можна відразу перейти безпосередньо до розробки програми, уникнувши зайвих налаштувань і скачавши всі необхідні файли в автоматичному режимі;

- у ній присутні макети для створення UI, з чого зазвичай починається робота над додатком. Це дозволяє істотно заощадити час і уникнути помилок, не створюючи свої милиці в уже налагодженому механізмі;

- інтеграція з GIT, що дозволяє швидко і якісно відстежувати зміни в коді, і створювати резервні копії;

- в основі робочого процесу Android Studio закладений концепт безперервної інтеграції, що дозволяє відразу ж виявляти наявні проблеми;

- додаток здатний підписувати додатки перед публікацією його в Google Play, що дозволяє економити час;

- інструменти для аналізу батареї показують, яке навантаження доводиться на пристрій і як високий витрата батареї, правда, в зразковому співвідношенні;

- можливість використовувати підсвічування синтаксису, передбачену творцями операційної системи, що дозволяє уникнути зайвих помилок.

3.3 Розробка інтерфейсу системи.

Весь інтерфейс Android додатків, розроблених в Android Studio складається з так званих "макетів", ілюстрацію яких можна побачити на рис 3.2.

Щоб оголосити свій макет, можна створити екземпляри об'єктів View в коді і запустити побудова дерева, але найпростіший і найбільш ефективний спосіб - визначення макета за допомогою файлу XML. XML дозволяє створювати легку для читання структуру макета, подібно HTML.

Ім'я елемента XML для виду відповідає класу Android, до якого від відноситься. Так, елемент <TextView> створює віджет TextView в інтерфейсі, а елемент <LinearLayout> створює групу перегляду LinearLayout.

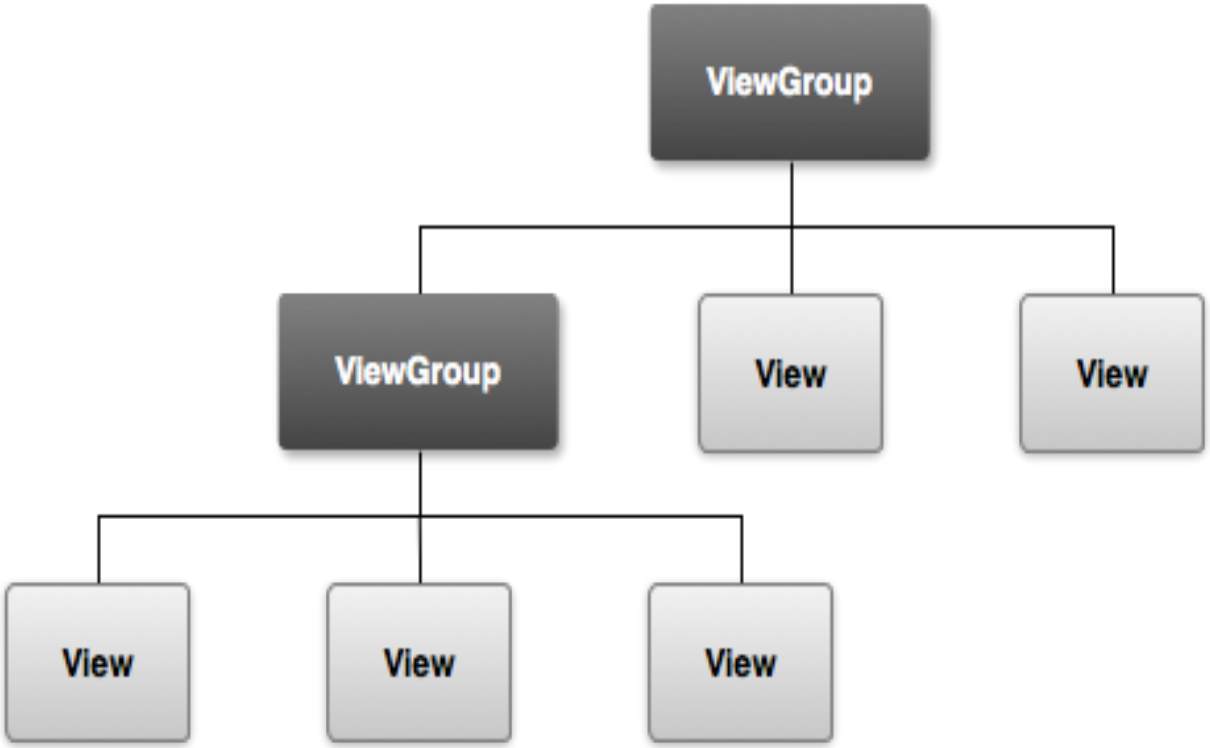

Рисунок 3.1 – Ілюстрація ієрархії макета интерфейса Android додатку.

login_activity – стартова activity, яка відповідає за ідентифікацію користувача, містить поле введення логіна і пароля, а так-же наступний набір кнопок:

Авторизуватися. При натисканні система зчитує дані з полів введення логіна і пароля, і порівнює їх з тим, що є у відповідній вкладці вбудованої SQLite бази даних в додатку. Якщо відповідна введених даних запис знайдений, відбувається перехід на main_menu_activity з передачею даних про користувача.

Увійти без аккаунта. Користувачеві не обов'язково мати акаунт щоб запустити програму, і він може скористатися базовим функціоналом і не вводячи своїх даних. Після натискання відбувається перехід на main_menu_activity без передачі будь-яких додаткових даних.

Реєстрація. Дозволяє зареєструвати новий акаунт. Перенаправляє на register_activity.

Зображення результату інтерпретації макета login_activity показано на рис 3.1

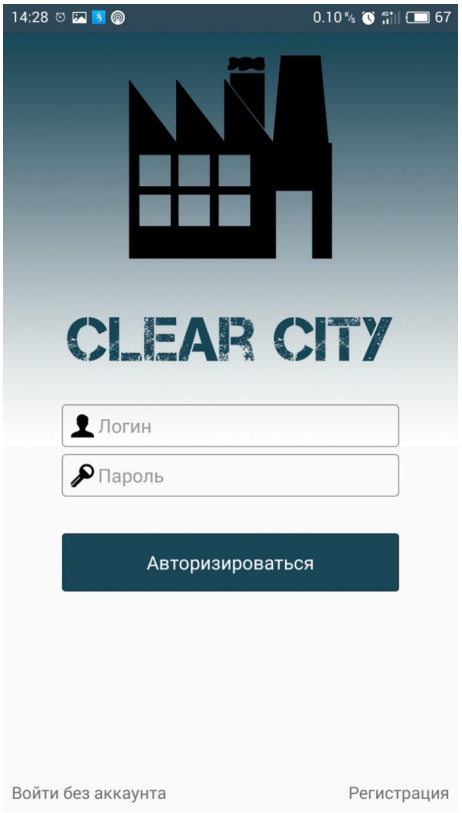

Рисунок 3.2 – Відображення інтерпретації макета інтерфейсу activity_login.xml.

Вихідний код макета інтерфейсу activity_login.xml можна знайти в додатку Б.

register_activity – сторінка реєстрації. Має наступні поля введення для заповнення даними користувачів:

- ім'я;
- логін, що вводиться при авторизації на login_activity;
- пароль, що вводиться при авторизації на login_activity;
- підтвердження паролю, що вводиться при авторизації на login_activity.

А також кнопку – «Зареєструватися». Після перевірки на коректність, дані заносяться в базу даних після процедури шифрування і генерації MD5 контрольної суми. Якщо дані не коректні, висвічується повідомлення про помилку.

Після вдалої реєстрації користувач переходить на main_menu_activity, на яку разом з цим передаються його дані.

Зображення інтерпретації інтерфейсу login_activity показано на рис 3.3

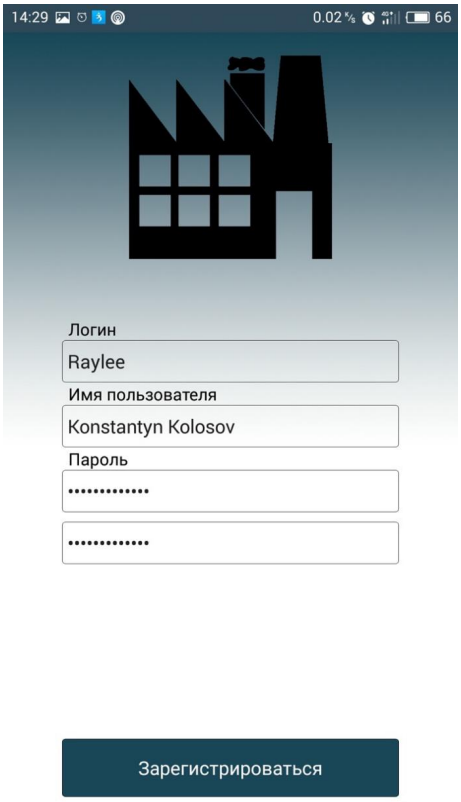

Рисунок 3.3 – Відображення інтерпретації макета інтерфейсуactivity_register.xml.

Вихідний код макета інтерфейсу activity_register.xml можна знайти в додатку Б.

main_menu_activity – є центром програми, що відображає загальну картину екологічної ситуації.

Після завантаження Activity відбувається спроба довантажити погодні умови з одного з відкритих джерел. Для отримання даних використовується http://openweathermap.org/. Якщо завантажити прогноз погоди не вдалося, програма використовує дані, отримані в попередній раз, довантажуючи їх з бази даних, і повідомляє про це користувача.

Наступним кроком стає спроба оновити дані з датчиків екологічної обстановки. Працює це за аналогією з роботою погодного сервісу і можливістю використовувати більш старі дані.

Після подгрузки вищезгаданої інформації програма проводить розрахунки і видає оцінку відхилень показників від норми від 0 (дуже погано) до 10 (дуже добре), після чого виводить коментарі відхилень і погодні дані.

Містить виїжджає меню, в якому відображені ім'я і пошта користувача, і яке містить наступні кнопки:

- кнопка «статистика» – підвантажує statictic_activity після натискання;
- кнопка «дані» – підвантажує data_activity після натискання;
- кнопка «мапа» – підвантажує map_activity після натискання;
- кнопка «налаштування» – підвантажує settings_activity після натискання.

Зображення інтерпретації макета activity_main_menu.xml показано на рис 3.3 и 3.4

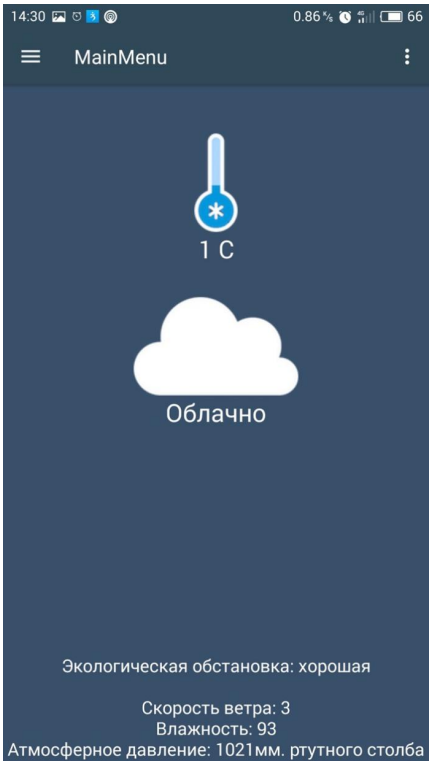

Рисунок 3.4 – Відображення інтерпретації макета інтерфейсуactivity_main_menu.xml.

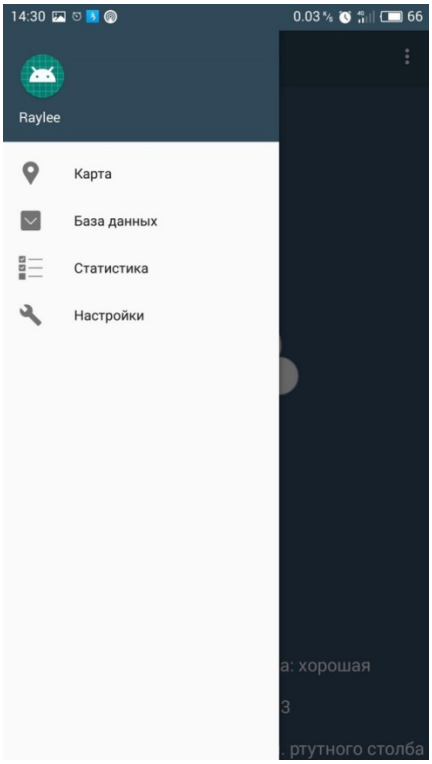

Рисунок 3.3 – Відображення інтерпретації макета інтерфейсу activity_main_menu.xml. Вид з відкритим боковим меню.

Поточний макет комплексний, і складається з декількох файлів:

1) content_main_menu.xml – основне вікно програми.

2) app_bar_main_menu.xml – верхня частина інтерфейсу з назвою Activity та кнопкою бокового меню

3) nav_header_main_menu.xml – шаблон для верстки бокового меню.

4) activity_main_menu.xml – фінальный макет, що компілює все наведене вище.

Вихідний код цих макетів можна знайти в додатку Б.

statictic_activity – Ця частина програми відповідає за відстеження прогресії екологічної ситуації і виводить на екран графіки, ґрунтуючись на показниках датчиків, і дати отримання цих показників. Графік дозволяє вмикати або вимикати деякі показники, а так-же вибирати початкову і кінцеву дату для побудови. За замовчуванням показує графік, побудований на даних за останній тиждень, беручи середній результат за кожен з 7 днів. Дані для побудови графіків беруться з бази даних.

Зображення інтерпретації интерфейса activity_ statictic.xml показано на рис. 3.5 и 3.6

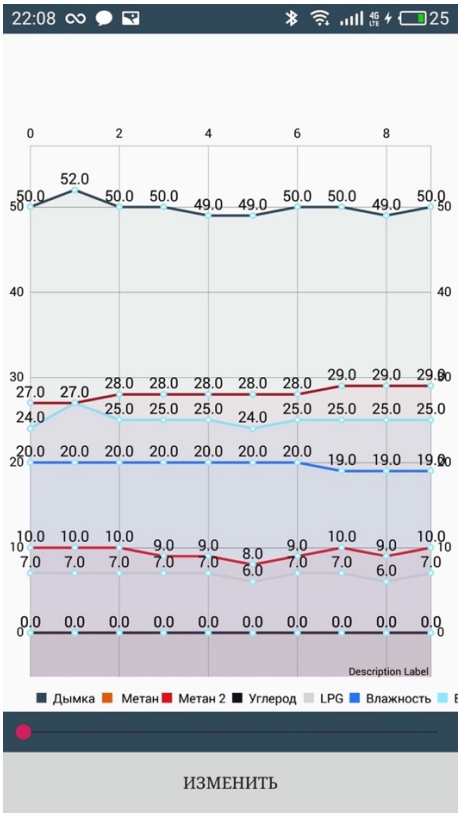

Рисунок 3.5 – Відображення інтерпретації макета інтерфейсу activity_ statictic.xml.

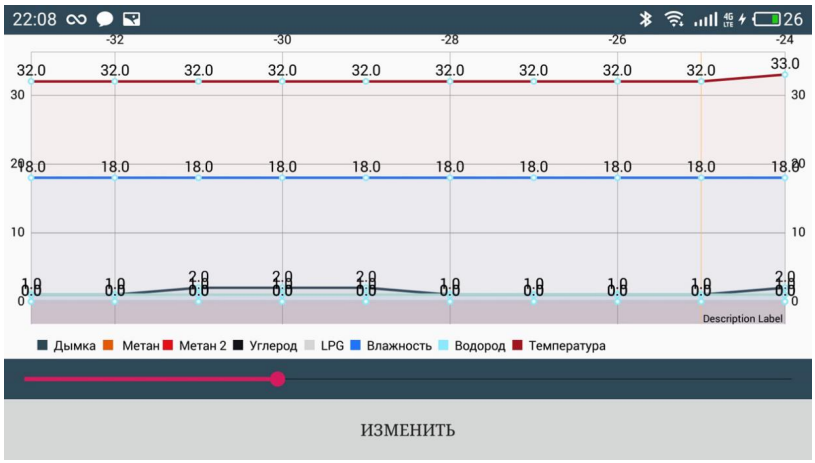

Рисунок 3.6 – Відображення інтерпретації макета інтерфейсу activity_ statictic.xml. Вид з горизонтальною орієнтацією.

data_activity – Дозволяє вивести дані екологічної ситуації у вигляді впорядкованого списку, сортуючи за датою отримання даних, або ж за конкретними показниками, дозволяючи швидко вловити дати найбільш або найменш критичних показників. Спочатку виводить дату і обраний показник, проте при натисканні на будь-яку клітинку списку виводиться діалог з повним переліком інформації. Інформація виходить з бази даних.

Зображення інтерпретації интерфейса activity_ statictic.xml показано на рис. 3.7 и 3.8

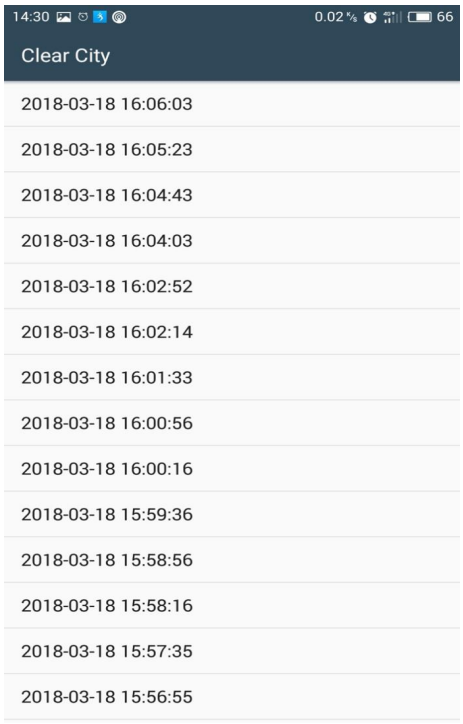

Рисунок 3.7 – Відображення інтерпретації макета інтерфейсу activity_ data.xml.

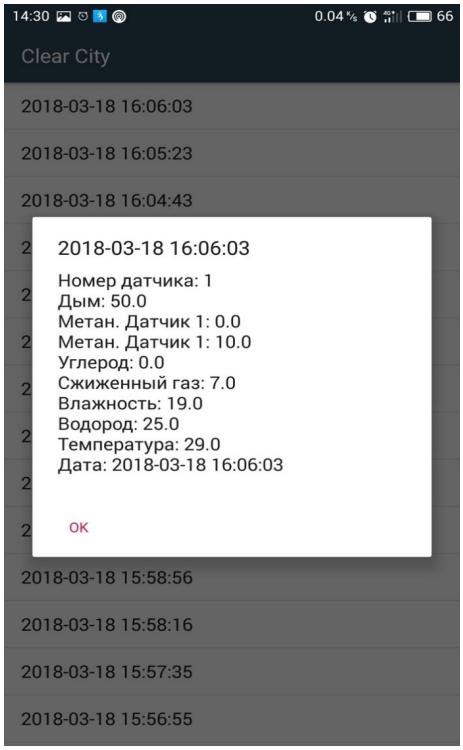

Рисунок 3.8 – Відображення інтерпретації макета інтерфейсу activity_ data.xml. Вибір одного з елементів списку.

map_activity – Карта, на якій завдяки інформації з бази даних наносяться області викидів і схематичні напрямки, в якому буде йти хмара. Під картою знаходиться

повзунок, переміщаючи який можна відображати прогнозовані області грунтуючись на прогнозі, який буде проведений додатком на основі даних про погоду і екологічну обстановку, і дозволить побачити передбачувану картину забруднення в промежут від однієї години до однієї доби.

Зображення інтерпретації интерфейса activity_ map.xml показано на рис. 3.9 и 3.10

Рисунок 3.9 – Відображення інтерпретації макета інтерфейсуactivity_ map.xml.

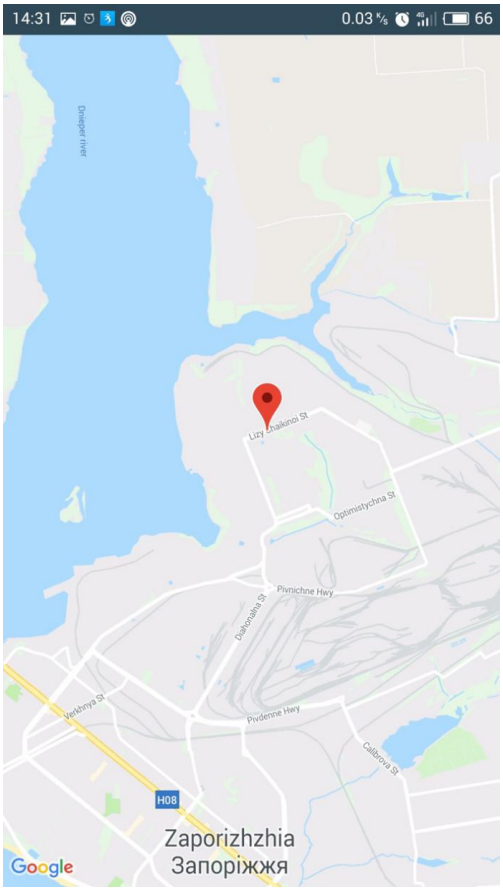

Рисунок 3.10 – Відображення інтерпретації макета інтерфейсуactivity_ map.xml. Результат наближення карти подвійним натисканням.

3.4 Написання коду програми.

Першою частиною роботи з програмою була розробка бази даних, яка в підсумку придбала 3 таблиці і має такий вигляд:

Таблиця 3.1 – Таблиця БД з інформацією про користувачів users

№	Назва поля	Вміст поля	Тип даних	Посилання на таблицю
1	id_user	Номер запису	INTEGER	
2	user_name	Ім’я користувача	TEXT	
3	user_login	Логін користувача	TEXT	
4	user_pass	Пароль користувача в MD5	TEXT	
5	access_lead	Рівень доступу користувача	INTEGER	
6	server_id	Номер згідно з сервером	INTEGER	

Таблиця 3.2 – Таблиця БД з інформацією з сенсорів sensors

№	Назва поля	Вміст поля	Тип даних	Посилання на таблицю
1	id_number	Номер запису	INTEGER	
2	coord_x	Координати по осі x	INTEGER	
3	coord_y	Координати по осі y	INTEGER	
4	co_concentration	Концентрація углероду	INTEGER	
5	no_concentration	Концентрація азоту	INTEGER	
6	ch4_concentration	Концентрація метану	INTEGER	
7	duct_concentration	Концентрація пилу	INTEGER	

Таблиця 3.3 – Таблиця БД з інформацією про погоду weather

№	Назва поля	Вміст поля	Тип даних	Посилання на таблицю
1	id_number	Номер запису	INTEGER	
2	date_time	Дата запису	TEXT	
3	clouds	Хмарливість	INTEGER	
4	temperature	Температура	INTEGER	
5	temperature_min	Мінімальна температура	INTEGER	
6	temperature_max	Максимальна температура	INTEGER	
7	wind_deg	Напрям вітру	INTEGER	
8	wind_speed	Швидкість вітру	INTEGER	
9	humidity	Вологість	INTEGER	
10	pressure	Атмосферний тиск	INTEGER	

Для спрощення стартової розробки, розмежування користувачів з прав і більш зручної роботи з даними, спочатку була створена система реєстрації. Система тісно пов'язана з базою даних SQLite, і збирає наступну інформацію про користувача:

- логін;
- ім'я;
- пароль.

Після чого ця інформація проходить перевірку на унікальність і помилки при заповненні полів для введення, після чого при успішності перевірки заноситься в базу даних.

Код відповідає за внесення запису в базу даних:

```
public static Boolean addUser(Context context, int id_user, String user_name, String user_login, String user_pass, int access_lead, int server_id){
    try{
      database.execSQL("INSERT INTO users VALUES ( " + id_user + ",
       '" + user_name +"',
       '"+ user_login + "',
       '" + Encryption.MD5(user_pass) + "',
        + access_lead + ",
       " + server_id + ");");
      Log.d("Database","user " + user_name + " added!");
      return true;
    }
    catch (Throwable t){
      Log.d("Database","user " + user_name + " is NOT added!");
      return false;
    }
  }
```

Як можна помітити, система отримує всі дані про користувача і створює відповідний запис в таблиці "users". Особливістю записи пароля є MD5 шифрування.

Код відповідає за шифрування паролів:

```
public static String MD5(String md5) {
      try {
         java.security.MessageDigest md = java.security.MessageDigest.getInstance("MD5");
         byte[] array = md.digest(md5.getBytes());
         StringBuffer sb = new StringBuffer();
         for (int i = 0; i < array.length; ++i) {
            sb.append(Integer.toHexString((array[i] & 0xFF) | 0x100).substring(1,3));
         }
         return sb.toString();
      } catch (java.security.NoSuchAlgorithmException e) {
      }
      return null;
    }
```

Повний код можна знайти в додатку А.

Далі був розроблений метод авторизації користувача в додатку, для чого був розроблений метод вилучення списку користувачів з бази даних, який отримав назву getUserInfo і він віддає набір екземплярів класу userInfo.

Структура класса userInfo:

```
class userInfo{
```

```
        int id_user;
        String user_name;
        String user_login;
        String user_pass;
        int access_lead;
         int server_id;
}
```

Структура метода getUserInfo:

```
public static userInfo[] getUserInfo(Context context){
     int count = usersCount(context);
     userInfo[] toReturn = new userInfo[count];
    if(database == null || query == null){
    Initiate();
    }
     for(int i = 0;i<count;i++){
       toReturn[i] = new userInfo();
    }
    query = database.rawQuery("SELECT * FROM users;", null);
    if(query.moveToFirst()){
       int number = 0;
       do{
          toReturn[number].id_user = query.getInt(0);
         toReturn[number].user_name = query.getString(1);
         toReturn[number].user_login = query.getString(2);
         toReturn[number].user_pass = query.getString(3);
        toReturn[number].access_lead = query.getInt(4);
         toReturn[number].server_id = query.getInt(5);

        number++;
     }
    while(query.moveToNext());
    }
    query.close();
     database.close();
        return toReturn; }
```

Отримані дані використовуються для перевірки введених даних за допомогою циклу:

```
public void LoginWithAccountOnClick(View v){
     boolean Status = false;
    for(int x = 0; x < Users.length ; x++){
       Log.d("Login", Users[x].user_login.toString() + " - " + Login.getText().toString());
```

```
 Log.d("Login", Users[x].user_pass.toString() + " - " + Pass.getText().toString());
 if(
      Users[x].user_login.toString().equals(Login.getText().toString()) &&
       Users[x].user_pass.toString().equals(Encryption.MD5(Pass.getText().toString())))
 {
     Status = true;
     Intent myIntent = new Intent(LoginActivity.this, MainMenu.class);
     myIntent.putExtra("user_name", Users[x].user_name);
     myIntent.putExtra("user_login", Users[x].user_login);
     LoginActivity.this.startActivity(myIntent);
      finish();
  }
 }
 if(!Status){
     Toast.makeText(getBaseContext(), "Неверное имя или пароль!",
          Toast.LENGTH_LONG).show();
 }
}
```

Якщо користувач ввів в поле для введення правильний логін і пароль, програма переводить його на основне activity додатки, яке носить ім'я MainMenu.

MainMenu є одним з головних місць докладання. Саме на ньому можна переглянути прогноз погоди, дізнатися загальну оцінку поточної екологічної ситуації. І з нього можна потрапити в інші частини програми - аналізатор бази даних, карту, де вказано положення датчиків і екологічна обстановка в їх області, activity статистики, де знаходяться графіки.

При переході на MainActivity додаток робить спробу отримати дані погоди і дані з сервера сенсорів екологічної обстановки.

Отримавши дані з сервера сенсорів, процес отримання яких розписаний в пункті 2.1, програма відтворює наступний цикл:

```
for(int x = 0; x< data.length();x++)
{
      SensorsCore.SensorsInfo[x] = new SensorsParameters(data.getJSONObject(x));
    Database.writeSensors(SensorsCore.SensorsInfo[x])
 }
```

Цей код ділить масив JSONArray під ім'ям data на елементи JSONObject, які після цього передаються на масив SensorsInfo класу SensorsCore для зберігання і

отримання даних іншими елементами програми. Самі записи являють собою елементи класу SensorsParameters, об'єднання в масив.

Для роботи з наданими сервером даними був розроблений наступний клас, що включає в себе перелік необхідних для роботи даних, одержуваних з сервера:

```
class SensorsParameters {
        public int ID; //id записи
        public int ID_dev; //id датчика
        public float Smoke; //Температура
        public float Methane_1; //Данные с сенсора 1
        public float Methane_2; //Данные с сенсора 2
        public float Carbon; //Описание
        public float LPG; //Описание
        public float Humidity; //Влажность
        public float Hydrogen; //Гидроген
        public float Temp; //Температура
         public String Date; //Дата записи
//Конструктор. Получает JSONObject, и извлекает из него данные по ключам.
   public SensorsParameters(JSONObject data){
      Log.d("WeatherParametrs", "Initiated");
      Log.d("SensorsParam.ID", "================================");
      try {
         ID = data.getInt("id");
         Log.d("SensorsParam.ID", String.valueOf(ID) + "");
         ID_dev = data.getInt("dev_id");
         Log.d("SensorsParam.ID_dev", String.valueOf(ID_dev) + "");
         Smoke = data.getInt("Smoke");
         Log.d("SensorsParam.Smoke", String.valueOf(Smoke) + "");
         Methane_1 = data.getInt("methane1");
         Log.d("SensorsParam.Methane_1", String.valueOf(Methane_1) + "");
         Methane_2 = data.getInt("methane2");
         Log.d("SensorsParam.Methane_2", String.valueOf(Methane_2) + "");
         Carbon = data.getInt("Carbon");
         Log.d("SensorsParam.Carbon", String.valueOf(Carbon) + "");
         LPG = data.getInt("LPG");
         Log.d("SensorsParam.LPG", String.valueOf(LPG) + "");
         Humidity = (float)data.getDouble("humidity");
         Log.d("SensorsParam.Humidity", String.valueOf(Humidity) + "");
         Hydrogen = data.getInt("Hydrogen");
         Log.d("SensorsParam.Hudrogen", String.valueOf(Hydrogen) + "");
         Temp = (float)data.getDouble("temp");
         Log.d("SensorsParam.Temp", String.valueOf(Temp) + "");
         Date = data.getString("datetime");
```

```
            Log.d("SensorsParam.Date", String.valueOf(Date) + "");
        }catch (Exception e){
            Log.e("SensorsParameters", e.toString());
        }
    }
}
```

Клас SensorsParameters складається з набору змінних, в яких зберігаються дані з сенсорів і конструктор, який отримує дані у вигляді об'єкта JSONObject, з якого ці дані і витягуються, щоб встати на зберігання.

Отримані таким чином дані готові для використання всередині програми, і можуть бути отримані з будь-якого іншого класу.

Цілісний модуль з отримання та обробки погодних умов в додатку можна найти в Додатках А.

Щоб уникнути зберігання зайвих даних, був проведений відбір на їхню актуальність для роботи програмних алгоритмів, на основі якого був написаний наступний клас:

```
class WeatherParametrs{
    public float Date = 0f;
    public int Clouds = 0;//Облачность в процентах
    public int WindSpeed = 0;//Скорость ветра
    public int WindDeg = 0;//Направление ветра
    public int TempMin = 0;//Прогноз. Минимальная температура
    public int TempMax = 0;//Прогноз. Максимальная температура
    public int Temp = 0;//Текущая температура
    public int Humidity = 0;//Влажность
    public int Pressure = 0;//Атмосферное давление
}
```

Клас зберігає такі дані:

Дату і час - для визначення актуальності даних.

Хмарність - для складання завершеною картини прогнозу погоди, так, як розроблений пристрій немає методами визначення хмарності.

Швидкість вітру - як і в попередньому випадку - для складання завершеною картини прогнозу погоди. Так само ці дані беруть участь у формуванні прогнозування екологічної обстановки в якості дальності можливого екологічного забруднення.

Напрямок вітру - аналогічно швидкості вітру, з різницею в тому, що замість дальності допомагає визначити сторону, в якій буде розвиватися хмара газів.

Мінімальна температура - прогнозований температурний мінімум на окремо взятий день. Використовується для складання прогнозу погоди і виведення цієї інформації користувачеві.

Максимальна температура - прогнозований температурний максимум на окремо взятий день. Використовується для складання прогнозу погоди і виведення цієї інформації користувачеві.

Температура - температура в даний момент часу. Призначена для звірки показань датчика з показаннями з відкритих джерел.

Вологість - вологість в даний момент часу. Призначена для звірки показань датчика з показаннями з відкритих джерел.

Для обробки вхідних даних в класі передбачений конструктор:

```
public WeatherParametrs(JSONObject data, float date){
    Log.d("WeatherParametrs", "Initiated");
    try {
      Date = date;
      Log.d("WeatherPar.Date", String.valueOf(date) + "");
      Clouds = data.getJSONObject("clouds").getInt("all");
      Log.d("WeatherPar.Clouds", Clouds + "");
      WindSpeed = data.getJSONObject("wind").getInt("speed");
      Log.d("WeatherPar.WindSpeed", WindSpeed + "");
      WindDeg = data.getJSONObject("wind").getInt("deg");
      Log.d("WeatherPar.Clouds", WindDeg + "");
      TempMin = data.getJSONObject("main").getInt("temp_min") - 273;
      Log.d("WeatherPar.TempMin", TempMin + "");
      TempMax = data.getJSONObject("main").getInt("temp_max") - 273;
      Log.d("WeatherPar.TempMax", TempMax + "");
      Temp = data.getJSONObject("main").getInt("temp") - 273;
      Log.d("WeatherPar.Temp", Temp + "");
      Humidity = data.getJSONObject("main").getInt("humidity");
      Log.d("WeatherPar.Humidity", Humidity + "");
      Pressure = data.getJSONObject("main").getInt("pressure");
      Log.d("WeatherPar.Pressure", Pressure + "");
    }catch (Exception e){
      Log.e("WeatherParametrs", e.toString());
    }
  }
```

При створенні нового екземпляра класу на нього передається вихідний JSONObject, отриманий з сервера, і поточна дата на пристрої. Особливість методів вилучення даних погоди полягає в тому, що отриманий з сервера JSONObject містить кілька вкладених аналогічних по класу об'єктів, що змушує діставати їх окремим викликом методу .getJSONObject ().

Вихідний код частини, що відповідає за отримання погодних умов з сервера можна найти в Додатку А.

Отримавши дані, програма виводить їх на інтерфейс користувача, позначаючи поточні погодні умови іконками, супроводжувані подробицями в текстовому вигляді.

З MainActivity можна вільно перейти в наступні Activity.

MapsActivity – вдає із себе інтерактивну карту, яка використовує Google Maps API, і що зберігає місце розташування датчиків і інформацію про екологічну обстановку.

StatisticActivity – частина програми, відповідальна за аналіз даних і побудови на їх основі графіків. Примітна тим, що дозволяє вільно маніпулювати з тимчасовими відрізками аналізованих даних, і будувати графіки "на льоту".

DatabaseActivity – activity, що дозволяє вивести всі наявні в базі дані у вигляді списку, натискання на будь-який елемент якої призведе до виклику діалогу з відображенням повної інформації.

MapsActivity був розроблений Google, а тому всередині програми він був просто інтерпретований і поставлений на необхідну позицію в загальній архітектурі програми. При вході в Activity відбувається виклик на отримання і відтворення карти, а так само нанесення положення датчиків на координатну сітку:

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
    private GoogleMap mMap;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
```

```
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        // Add a marker in Sydney and move the camera
        LatLng Sensor_1 = new LatLng(47.893647, 35.146677);
        mMap.addMarker(new MarkerOptions().position(Sensor_1).title("Marker in Sensor 1"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(Sensor_1));
    }
}
```

StatisticActivity є Вільна інтерпритація відкритого API MPAndroidChart. Для його введення було імпортовано наступні бібліотеки:

com.github.mikephil.charting.charts.Chart

com.github.mikephil.charting.charts.LineChart

com.github.mikephil.charting.data.Entry

com.github.mikephil.charting.data.LineData

com.github.mikephil.charting.data.LineDataSet

com.github.mikephil.charting.interfaces.datasets.ILineDataSet

А в xml файл макета інтерфейсу був внесений наступний елемент:

```
  <com.github.mikephil.charting.charts.LineChart
    android:id="@+id/linechart"
    android:layout_width="match_parent"
    android:layout_height="462dp"
    android:layout_above="@+id/linearLayout2"
    android:layout_alignParentStart="true"
    android:layout_marginStart="0dp"
    android:layout_marginBottom="1dp"></com.github.mikephil.charting.charts.LineChart>
```

Зазначений макет дозволяє будувати лінійні графіки з використанням призначених для користувача даних, а так само підтримує гарячу заміну даних без необхідності перезавантажувати макет.

Дані заносяться з використанням наступного коду:

```
private void UpdateInfo(){
    //Туман
    ArrayList<Entry> SmokeValues = new ArrayList<>();
    Log.d("Shift", Shift + "");
    for(int x = Start - Shift; x < Finish - Shift; x++){
        SmokeValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Smoke));
```

```
}
LineDataSet SmokeSet = new LineDataSet(SmokeValues,"Дымка");
SmokeSet.setFillAlpha(FillAlpha);
SmokeSet.setColor(Color.parseColor("#304959"));
SmokeSet.setLineWidth(LineWidth);
SmokeSet.setValueTextSize(12);
SmokeSet.setDrawFilled(FillStatus);
SmokeSet.setDrawCircles(true);
SmokeSet.setFillColor(getResources().getColor(R.color.colorPrimary));
//Methane_1
ArrayList<Entry> Methane1Values = new ArrayList<>();
for(int x = Start - Shift; x < Finish - Shift; x++){
   Methane1Values.add(new Entry(x,Sensor[Sensor.length - 10 + x].Methane_1));
}
LineDataSet Methan1 = new LineDataSet(Methane1Values," Метан ");
Methan1.setFillAlpha(FillAlpha);
Methan1.setColor(Color.parseColor("#E65A08"));
Methan1.setLineWidth(LineWidth);
Methan1.setValueTextSize(12);
Methan1.setDrawFilled(FillStatus);
Methan1.setDrawCircles(true);
Methan1.setFillColor(getResources().getColor(R.color.Methan1Fill));
//Methane_2
ArrayList<Entry> Methane2Values = new ArrayList<>();
for(int x = Start - Shift; x < Finish - Shift; x++){
   Methane2Values.add(new Entry(x,Sensor[Sensor.length - 10 + x].Methane_2));
}
LineDataSet Methan2 = new LineDataSet(Methane2Values,"Метан 2 ");
Methan2.setFillAlpha(FillAlpha);
Methan2.setColor(Color.parseColor("#E50F16"));
Methan2.setLineWidth(LineWidth);
Methan2.setValueTextSize(12);
Methan2.setDrawFilled(FillStatus);
Methan2.setDrawCircles(true);
Methan2.setFillColor(getResources().getColor(R.color.Methan2Line));
//Carbon
ArrayList<Entry> CarbonValues = new ArrayList<>();
for(int x = Start - Shift; x < Finish - Shift; x++){
   CarbonValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Carbon));
}
LineDataSet Carbon = new LineDataSet(CarbonValues,"Углерод ");
Carbon.setFillAlpha(FillAlpha);
Carbon.setColor(Color.parseColor("#12131A"));
Carbon.setLineWidth(LineWidth);
Carbon.setValueTextSize(12);
```

```
Carbon.setDrawFilled(FillStatus);
Carbon.setDrawCircles(true);
Carbon.setFillColor(getResources().getColor(R.color.CarbonFill));
//LPG
ArrayList<Entry> LPGValues = new ArrayList<>();
for(int x = Start - Shift; x < Finish - Shift; x++){
   LPGValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].LPG));
}
LineDataSet LPG = new LineDataSet(LPGValues,"LPG ");
LPG.setFillAlpha(FillAlpha);
LPG.setColor(Color.parseColor("#D5D5D5"));
LPG.setLineWidth(LineWidth);
LPG.setValueTextSize(12);
LPG.setDrawFilled(FillStatus);
LPG.setDrawCircles(true);
LPG.setFillColor(getResources().getColor(R.color.LPGFill));
//Humidity
ArrayList<Entry> HumidityValues = new ArrayList<>();
for(int x = Start - Shift; x < Finish - Shift; x++){
   HumidityValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Humidity));
}
LineDataSet Humidity = new LineDataSet(HumidityValues,"Влажность");
Humidity.setFillAlpha(FillAlpha);
Humidity.setColor(Color.parseColor("#33A1FD"));
Humidity.setLineWidth(LineWidth);
Humidity.setValueTextSize(12);
Humidity.setDrawFilled(FillStatus);
Humidity.setDrawCircles(true);
Humidity.setFillColor(getResources().getColor(R.color.HumidityFill));
//Hydrogen
ArrayList<Entry> HydrogenValues = new ArrayList<>();
for(int x = Start - Shift; x < Finish - Shift; x++){
   HydrogenValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Hydrogen));
}
LineDataSet Hydrogen = new LineDataSet(HydrogenValues,"Водород");
Hydrogen.setFillAlpha(FillAlpha);
Humidity.setColor(Color.parseColor("#2176FF"));
Hydrogen.setLineWidth(LineWidth);
Hydrogen.setValueTextSize(12);
Hydrogen.setDrawFilled(FillStatus);
Hydrogen.setDrawCircles(true);
Hydrogen.setFillColor(getResources().getColor(R.color.HydrogenFill));
//Temp
ArrayList<Entry> TempValues = new ArrayList<>();
for(int x = Start - Shift; x < Finish - Shift; x++){
```

```
            TempValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Temp));
        }
        LineDataSet Temp = new LineDataSet(TempValues,"Температура");
        Temp.setFillAlpha(FillAlpha);
        Temp.setColor(Color.parseColor("#A31621"));
        Temp.setLineWidth(LineWidth);
        Temp.setValueTextSize(12);
        Temp.setDrawFilled(FillStatus);
        Temp.setDrawCircles(true);
        Temp.setFillColor(getResources().getColor(R.color.TempFill));
        //Добавление на график
        ArrayList<ILineDataSet> dataSets = new ArrayList<>();
        dataSets.add(SmokeSet);
        dataSets.add(Methan1);
        dataSets.add(Methan2);
        dataSets.add(Carbon);
        dataSets.add(LPG);
        dataSets.add(Humidity);
        dataSets.add(Hydrogen);
        dataSets.add(Temp);
        LineData data = new LineData(dataSets);
        Chart.setData(data);
        Chart.invalidate();
    }
```

Як можна помітити, код для додавання даних повністю ідентичний для кожної нової лінії, розрізняючи лише значеннями змінних, що дозволяє в майбутньому модифікувати код програми, включивши громіздке повторення в цикл.

DatabaseActivity призначене для зручного перегляду окремих елементів бази даних. Побудова списку відбувається наступним чином:

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_database);
        Sensor = Database.GetSensorsFromDatabase();
        for(int i=0; i<Sensor.length/2; i++){
            SensorsParameters temp = Sensor[i];
            Sensor[i] = Sensor[Sensor.length -i -1];
            Sensor[Sensor.length -i -1] = temp;
        }
        AdapterTitles = new String[Sensor.length];
        for(int x = 0; x < AdapterTitles.length; x++){
            AdapterTitles[x] = Sensor[x].Date;
        }
```

```
        //Присваиваем ListView
        DataList = (ListView) findViewById(R.id.DataListView);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
            android.R.layout.simple_list_item_1, AdapterTitles);
        DataList.setAdapter(adapter);
    }
```

При погляді на код, можна зробити висновок, що він є досить ефективним методом побудови списку, адже дозволяє перебудовувати його на ходу, просто присвоївши йому новий adapter.

При натискання на будь-який елемент списку відбувається відображення всієї інформації за обраним номером:

```
    DataList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View itemClicked, int position,
                                long id) {
            AlertDialog alertDialog = new AlertDialog.Builder(DatabaseActivity.this).create();
            alertDialog.setTitle(AdapterTitles[position]);
            alertDialog.setMessage(
                    "Номер датчика: " + Sensor[position].ID_dev + "\n" +
                    "Дым: " + Sensor[position].Smoke + "\n" +
                    "Метан. Датчик 1: " + Sensor[position].Methane_1 + "\n" +
                    "Метан. Датчик 1: " + Sensor[position].Methane_2 + "\n" +
                    "Углерод: " + Sensor[position].Carbon + "\n" +
                    "Сжиженный газ: " + Sensor[position].LPG + "\n" +
                    "Влажность: " + Sensor[position].Humidity + "\n" +
                    "Водород: " + Sensor[position].Hydrogen + "\n" +
                    "Температура: " + Sensor[position].Temp + "\n" +
                    "Дата: " + Sensor[position].Date + "\n"
            );
            alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                            dialog.dismiss();
                        }
                    });
            alertDialog.show();
        }
    });
```

4 СИСТЕМА КОМПЛЕКСНОГО ЗБОРУ МЕТЕОДАНИХ З ПРОГНОЗУВАННЯМ ПОГОДИ

Розроблена в магістерській роботі программа для операційної системи Android заснована на ідеї створення простих і зручних методів відстеження поточного рівня екологічної обстановки на окремо взятій території. Додаток призначений для:

– отримання інформації з розроблених на кафедрі датчиків відстеження наявності і точної концентрації деяких домішок в повітрі;

– отримання інформації про погодні умови поруч з датчиками з відкритих інтернет-джерел;

– аналізу отриманої інформації, і побудови на її основі прогнозу поширення шкідливих домішок в повітрі;

– нанесення прогнозу на інтерактивну карту.

Концепція даної розробки не нова, і на ринку вже існує готова продукція в цьому напрямку, однак фірми в більшості країн не пропонують кінцевої вартості, заздалегідь не вислявши команду технічного обслуговування для замірів, а після оплати - монтажу і калібрування. Також не рідкісним є випадок виготовлення датчика під замовлення, що негативно позначається на кінцевій вартості.

З більш доступного сегмента з відкритими цінами подібні датчики надають компанії, перераховані в таблиці 4.1.

Розроблені датчики і програмне забезпечення мають наступні переваги перед найближчими аналогами:

Простота збірки. Мінімальна кількість пайки, можливість використовувати роз'єми, і обійтися без неї.

Універсальність. Можливість зібрати аналогічний датчик навколо будь-якої з популярних платформ. Наприклад: Pic Micro, Arduino, Raspberry, OrangePC і будь-яка інша, сумісна з дротового або бездротового передачею даних. Те ж саме стосується датчиків, дозволяючи робити кінцевий продукт дешевше, або дорожче, виключивши деякі з них, або навпаки - включивши в комплект більш точний і швидкий датчик.

Таблиця 4.1 – Компанія, що виробляє схожі по функціоналу з розробленими для магістерської роботи.

Компанія	Країна	Назва датчика	Ціна
Greentech Instruments.	Індія	GTI-241 A	980 доларів США. = 27314 грн за один датчик, та 200 доларів щороку на обслуговування. Так само, як і для розробки магістерської роботи прогнозована купівля 3 одиниць смартфонів, ціною біля 3000 гривень кожен. В результаті ми маємо ціну в 96518 грн без врахування оподаткування та 115861 грн з урахуванням.

Розроблене додаток універсально, і робить розмежування між конкретними корпоративними користувачами, а не пристроями, що дозволяє використовувати класичну схему логін / пароль, і не налаштовувати датчик і додаток окремо. Подібний підхід також дозволив використовувати так звані «незалежні» датчики, інформація з яких відкрита для всіх користувачів програми, навіть не мають рахунок.

Спрощене обслуговування. Розроблена схема пристрою невибаглива до погодних умов, а датчики і мікроконтролер легко замінюються.

Плаваюча ціна. Завжди можна замовити пристрій, ідеально вписується в технічне завдання.

Однак подібна схема не позбавлена недоліків:

Необхідність підзарядки, або наявності джерела живлення.

Необхідність ручної перезавантаження датчика при збої.

Недолік точності в порівнянні з дорогими аналогами.

Необхідність діагностики, порівняння значень датчика з еталонними з частотою, яка залежить від теплових умов.

Ручне складання на увазі можливу наявність дефектів в корпусі, що може привести до утворення роси.

Пластиковий корпус з часом зношується і приходить в непридатність, особливо під прямими сонячними променями.

У зв'язку з тим, що собівартість датчика в більшості можливих комплектацій не перевищує 1526 гривень, планується реалізація трьох датчиків, об'єднаних в одну мережу по 3000 гривень за штуку.

4.1 Планування розрахунків проекту

Весь процес дослідження можна розділити на етапи. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт. У моделюванні проекту приймають участь фахівець протягом двох місяців і консультант протягом 0,5 місяця. Дослідження починаються першого жовтня і повинні бути виконані до тридцятого листопада 2018 року.

Тривалість робіт визначають за формулою 4.1:

$$T_{\text{ц}} = \frac{Q}{R} \qquad (4.1)$$

де $T_ц$ - тривалість циклу, днів;

Q - трудомісткість, людино-днів;

R - кількість виконавців, чол.

Отримана інформація зведена в табл. 4.1

За даними таблиці 4.1 складається зведений стрічковий графік планування розробки проекту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення термінів початку виконання всі види роботи, а навпаки - календарний період їх виконання.

Таблиця 4.1 – Характеристика робіт по розробки проекту

Найменування роботи	Трудомісткість		Виконавці	Тривалість, днів
	люд.-дні	%к підсумку		
Аналіз предметної області (ПО)	10	11.11	Консультант Фахівець	5
Визначення вимог до програмному продукту	5	11.11	Фахівець	5
Проектування структури програми.	10	22.22	Фахівець	10
Створення програми.	15	33.33	Фахівець	15
Налагодження програми.	5	11.11	Фахівець	5
Складання програмної документації	10	11.11	Консультант Фахівець	5
Разом	55	100		45

Зведений графік планування розробки програмного продукту наведено в таблиці 4.2.

Таблиця 4.2 – Зведений графік планування розробки програмного продукту

Найменування роботи	Календарний період, дні								
	01.10-5.10	08.10-12.10	15.10-19.10	22. 10-26. 10	29.10-2.11	5.11-9.11	12.11-16.11	19.11-23.11	26.11-30.11
Аналіз предметної області (ПО)									
Визначення вимог до програмного продукту									
Проектування структури програми.									
Створення програми.									
Налагодження програми.									
Складання програмної документації									

фахівець консультант

4.2 Визначення витрат на моделювання моделі

Для визначення витрат на моделювання моделі складається калькуляція вартісної вартості робіт, яка включає наступні статті:

- основна заробітна плата;
- додаткова заробітна плата;
- єдиний соціальний внесок (ЄСВ);
- витрати на спеціальне обладнання;
- матеріали і комплектуючі вироби;
- накладні витрати;
- податки.

4.2.1 Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих в моделюванні моделі. Розрахунок зарплати ведеться на підставі даних про трудомісткості, представлених в табл. 4.3.

Таблиця 4.3 - Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн.	Кількість місяців роботи	Сума ЗП, грн
Фахівець	1	6000	2	12000
Консультант	1	9340	0,5	4670
Разом	2			16670

4.2.2 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою 4.2:

$$ЗП_{доп} = ЗП_{осн} \cdot 0{,}1 \quad (4.2)$$

Підставивши величину основної заробітної плати в формулу 4.2, отримуємо:

$$ЗП_{доп} = 16670 \times 0{,}1 = 1667 \text{ грн.}$$

4.2.3 Відрахування на єдиний соціальний внесок

Вони становлять 22% і беруться від основної та додаткової заробітної плати.

$$ОТ = (ЗП_{осн} + ЗП_{доп}) \times 0{,}22 \qquad (4.3)$$

$$ОТ = (16670 + 1667) \times 0{,}22 = 4034.14 \text{ грн.}$$

4.2.4 Визначення затрат на матеріали

Використовується 3 найменування матеріалів:
Датчик збору метеоданих – 4578 грн. (3 одиниці).
Смартфони на ОС Google Android 4.4+ - 9000 (3 одиниці).
Витрати на комплектуючі розраховують за формулою 4.4:

$$M = \sum_{i=1}^{n}(Ц_i \times N_i \times (1 + K_{т.з}) - Ц_{io} \times N_{io}) \qquad (4.4)$$

де M - витрати на покупні напівфабрикати і комплектуючі вироби, грн.;
$K_{т.з.}$ - коефіцієнт, що враховує транспортно-заготівельні витрати;
$Ц_i$ - ціна і-го найменування напівфабрикату і комплектуючого, грн.;
N_i - потреба в і-му напівфабрикаті і комплектуючому;
$Ц_{io}$ - вартість зворотних відходів і-го найменування комплектуючого, грн.;
N_{io} - кількість зворотних відходів і-го найменування;
n - кількість найменувань напівфабрикатів і комплектуючих.

$$Ц_{io} = 0;\ N_{io} = 0;\ K_{т.з.} = 0{,}05;$$

$$M = (1 + 0{,}05) \times (9000 + 4578) = 13578 \text{грн.}$$

Разом, витрати на комплектуючі становлять 13587 грн.

4.2.5 Витрати на спеціальне обладнання

У цю статтю входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання.

Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для проведення розрахунків теплових режимів блоків РЕЗ. Для чого необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування» виходячи з якої визначиться вартість одного машино-години роботи ПК, після множення якої на машинний час пішло на процес моделювання отримаємо витрати на оплату машинного часу.

Амортизаційні відрахування визначають за формулою 4.5:

$$A = Ф_б \cdot \frac{H_a}{100}, \tag{4.5}$$

де $Ф_б$ - балансова вартість обчислювальної техніки, грн .;

H_a - норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, для ПК 25%.

Балансова вартість обчислювальної техніки становить 5000 грн.

Отримуємо:

$$A = 27000 \times 0{,}25 = 6750 \text{грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію і допоміжні комплектуючі.

$$C_э = N_н \times Ф_{эф} \times K_{зв} \times K_{зм} \times Ц_в \tag{4.6}$$

де $N_н$ - номінальна потужність ЕОМ, кВт;

$Ф_{еф}$ - річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{зв}$ - середній коефіцієнт завантаження за часом;

$К_ш$ - коефіцієнт завантаження по потужності;

$Ц_е$ - ціна одного кВт-год електроенергії, грн./(кВт-ч).

Номінальна потужність ЕОМ - 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і за потужністю рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,11 грн.

Отримуємо:

$$С_э = 0{,}2 \times 1800 \times 0{,}9 \times 0{,}6 \times 2{,}11 = 410{,}184 \text{грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою 4.7:

$$ЗП_{обсл} = ФЗП_г \times (1 + К_{отн}) \times \frac{t_{обсл}}{Ф_{эф.обсл.}} \qquad (4.7)$$

де $ФЗП_г$ - річний фонд заробітної плати (основної і додаткової) обслуговуючих робітників, грн.;

$К_{отч}$- коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{обсл}$ - час протягом року, необхідне на технічне обслуговування ЕОМ, ч/рік;

$Ф_{еф.обсл}$ - річний ефективний фонд часу обслуговуючого персоналу, ч/рік.

Місячна заробітна плата обслуговуючого персоналу становить 3723 грн., а річний фонд заробітної плати відповідно дорівнює 44676 грн. Річний ефективний фонд робочого часу обслуговуючого ПК працівника дорівнює 1750 год / рік. На обслуговування одного ПК витрачається по 1 годині на місяць, що в рік становить 12 годин.

$$ЗП_{одсл} = 44676 \times (1 + 0{,}22) \times \frac{12}{1750} = 373{,}747 \text{грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і складає 202,5 грн.

Стаття «Інші витрати» приймається рівною п'яти відсоткам від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 7736,431 грн., 5% від суми складають 386,82грн.

Розраховані статті витрат на утримання і експлуатацію устаткування внесені в таблицю 4.4

Таблиця 4.4 - Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	6750,00
Експлуатація обладнання (крім витрат на поточний ремонт)	410,18
Заробітна плата основна і додаткова обслуговуючих робітників з ЄСВ	373,747
Поточний ремонт обладнання	202,5
Інші витрати	386,82
Разом	8123,247

Витрати на оплату машинного часу ЕОМ для розробки моделей і налагодження програмних засобів визначаються за формулою 4.8:

$$C_{\text{мо}} = P_{\text{экс}} \times t_{\text{мо}} \qquad (4.8)$$

де $C_{мо}$ - витрати на оплату машинного часу, грн .;

$P_{екс}$ - експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино-год .;

$t_{мо}$ - машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино-год.

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат за кошторисом «Витрати на утримання та експлуатацію обладнання (ЕОМ)» (табл. 4.4) на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. В результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$P_{\text{екс}} = \frac{8123{,}247}{1800} = 4{,}51 \text{ грн/машин} - \text{год}$$

ЕОМ експлуатується 45 днів в одну зміну, що становить в сумі 360 годин. Таким чином, витрати на оплату машинного часу складуть:

$$C_{MO} = 4{,}51 \times 624 = 1623{,}60 \text{грн.}$$

4.2.6 Інші прямі витрати

В інші прямі витрати включаються витрати на яке використовується при розробці системи комерційне програмне забезпечення:

- дольове ПЗ, що використовується постійно при роботі ПК (Windows 10 Professional) - 3900 грн. з НДС;

- Акаунт розробника Google Play – 850.10 грн з НДС.

$$S_{\text{доп.ПЗ}} = \frac{\text{Ц}_{\text{ПЗ } Windows} \times T_{\text{КТС}}}{\Phi_{\text{еф.КТС}} \times T_{\text{с ПЗ}}} \qquad (4.9)$$

$$S_{\text{діл ПЗ}} = \text{Ц}_{\text{ПЗ А}}$$

де $S_{\text{доп.ПЗ}}$ - витрати на дольове ПЗ при моделюванні розробляється в розрахунку ПЗ, грн .;

$S_{\text{діл ПЗ}}$ - витрати на цільове ПЗ, що купується виключно для моделювання в розрахунку ПЗ, грн .;

$\text{Ц}_{\text{ПЗ } Windows}$ - ціна ПЗ Windows, грн;

$\text{Ц}_{\text{ПЗ А}}$ - акаунта розробника Google Play, грн;

$T_{\text{КТС}}$ - машинний час КТС, необхідне користувачеві для моделювання моделі, машино-год / рік;

$\Phi_{\text{еф.КТС}}$ - річний ефективний фонд часу роботи КТС, машино-год / рік;

$T_{\text{с ПЗ}}$ - термін служби дольової ПЗ, років.

$$S_{\text{доп ПЗ}} = \frac{3900 \times 697}{1800 \times 5} = 302.03 \text{ грн.}$$

$$S_{\text{діл ПЗ}} = 850.10 \text{ грн.}$$

$$S_{\sum} = 302{,}3 + 850.10 = 1162{,}04 \text{ грн.}$$

4.2.7 Розрахунок накладних витрат

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т.д.), утримання та експлуатацію будівель. Накладні витрати включаються до вартості моделювання моделі непрямим шляхом - у відсотках до основної заробітної плати розробників. В даному випадку накладні витрати становлять 40% до основної заробітної плати розробників, що складає 6668 грн.

Результати визначення витрат на моделювання моделі у вигляді калькуляції кошторисної вартості робіт наведені в табл. 4.5.

Таблиця 4.5 - Калькуляція кошторисної вартості робіт з розрахунків

№	Найменування статей витрат	Сума, грн	Питома вага до підсумку,%
1	Основна заробітна плата	16670	30.59
2	Додаткова заробітна плата	1667	3.06
3	ЄСВ	4034.14	7.40
4	Матеріали	13587	24.93
5	Витрати на спец. обладнання	1623,60	2.98
6	Інші прямі витрати	1162,04	2.13
7	Накладні витрати	6668	12.23
8	ПДВ(20%)	9082.36	16,67
9	Разом (S_{pn})	54494.14	100

Провівши необхідні розрахунки, ми бачимо, що система аналогічного класу більш, ніж у два рази (116 тисяч грн. проти 56) перевищує ціну нашого обладнання разом з усіма витратами на його розробку. Що в сумі з можливістю мати поруч спеціаліста з ремонту та настройки цього датчика, та його передбаченій простоті, модульності та дешевизні дозволяє зекономити як на придбанні, обслуговуванні та ремонті, так і на штрафах за відсутню систему відстежування забруднення повітря і за сам факт його забруднення.

Також майже всі компанії, що виробляють подібні пристрої мають закриту цінову політику що ускладнює взаємодію з ними та дозволяє їм диктувати свої

правила обслуговування та більшості інших питань при підписанні контракту, чого не можна сказати про предмет моєї магістерської роботи.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В ході магістерської роботи на тему "Система комплексного збору метеоданних з прогнозування погоди" розроблено програмне забезпечення для операційної системи Android заснована на ідеї створення простих і зручних методів відстеження поточного рівня екологічної обстановки на окремо взятій території.

Основною метою розділу є розробка попереджувальних заходів запобігання виробничого травматизму, професійних захворювань, шкідливого впливу на навколишнє середовище, які є специфічними при виконанні майбутніх службових обов’язків

5.1 Аналіз потенційних небезпек

Небезпечні та шкідливі виробничі чинники визначені в ГОСТ 12.0.003-74 (1999) «ССБТ. Опасные и вредные производственные факторы. Классификация».

Згідно класифікації ці чинники поділяються на:

- фізичні - будь-які фізичні об’єкти , фізичні явища, фізичні процеси, які здатні нанести шкоду здоров’ю або життю людини;

- психофізіологічні - чинники, причинами яких є важкість та напруженість праці, емоційні перевантаження, тощо;

- санітарно-гігієнічні - параметри повітряного середовища виробничих приміщень, рівень освітлення робочого місця або робочої зони, шум, електромагнітні поля та випромінювання; іонізуючі та оптичні випромінювання тощо.

- біологічні – включають наступні біологічні об'єкти: патогенні мікроорганізми (бактерії, віруси, спірохети, грибки, найпростіші) і продукти їх життєдіяльності

У контексті роботи розробника програмного забезпечення, можна виділити такі основні потенційні небезпеки:

Потенційні небезпеки фізичного характеру:

- підвищена або низька температура повітря робочої зони - може призвести до виникнення захворювань або підвищеної втомлюваності;

- підвищений рівень шуму на робочому місці - може призвести до захворювань органів слуху або до впливати на нервову систему;

- підвищена або низька вологість повітря - може призвести до зниження працездатності та загальних захворювань;

- підвищена або низька рухливість повітря;

- підвищене значення напруги електричної мережі - може стати причиною ураження електричним струмом;

- відсутність або недостатність природного освітлення - може призвести до перенапруження та захворювань органів зору.

Потенційні небезпеки психофізіологічного характеру - фактори, які призводять до порушень нервової системи та зниження працездатності:

- розумове перенапруження;

- перенапруження аналізаторів;

- монотонність труда;

- емоціональні перенавантаження;

- потенційні небезпеки санітарно-гігієнічного характеру;

- потенційні небезпеки, пов’язані з порушеннями правил пожежної безпеки, внаслідок яких може виникнути пожежа;

- потенційні небезпеки, пов’язані з проявом наслідків надзвичайних ситуацій.

5.2 Заходи по забезпеченню техніки безпеки

При роботі роботи розробника програмного забезпечення можливими фізичними факторами ураження є :

- ураження електричним струмом;

- механічне травмування.

Для запобігання ураження електричним струмом встановлено електроустаткування, яке відповідає вимогам: «Правил устрою електроустановок» (далі «ПУЕ-2014») і ГОСТ 12.1.030-81 (2001) «ССБТ. Электробезопасность. Защитное заземление, зануление», величина опору захисного заземлення електрообладнання приміщення - 4 Ом.

Для запобігання ураженню електричним струмом передбачено наступні заходи:

- використання прихованого типу прокладання провідників струму, використання в зовнішніх провідниках струму виключно проводки із подвійною ізоляцією.

- встановлення автоматичної системи онлайн сповіщення задимлення, вогню, затоплення. Інформація про це приходить у виді SMS на мобільний телефон.

- для створення стабільного потоку струму до комп'ютерної техніки кожен комп'ютер обладнано системою безперебійної подачі живлення, що мають вбудовану акумуляторну батарею, яка дозволить продовжити роботу ще деякий час після відключення електричного струму, не дасть спалити техніку, створить для неї необхідні умови для стабільної та довговічної роботи

- встановлення електричного обладнання згідно норм ДСТУ IEC 60884-1:2007 Вилки та розетки побутової та аналогічної призначеності.

- для підвищення рівню безпеки у разі короткого замикання, між розетками та блоками безперервного живлення було поставлено мережеві фільтри, що мають функцію автоматично відключатися у разі КЗ.

Встановлено інтелектуальне обладнання, яке у постійному режимі видає звіти про поточний стан вхідної напруги, та дозволяє автоматично обірвати живлення у разі її виходу за робочі рамки.

У зв'язку із стресовими ситуаціями та нервово-емоційними навантаженнями у працівників може виникнути ймовірність захворювань загально-невротичного характеру. З метою зниження нервово-емоційного напруження, стомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втоми, передбачені перерви у роботі – 15 хвилин кожні дві години

Проведення навчання з правил електробезпеки, перевірка знань та атестація персоналу на кваліфікаційну групу з електробезпеки, згідно НПАОП 0.00-4.12-05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці».

Для запобігання порушень системи органів травлення і ушкодження хребта і м'язів спини у зв'язку з тривалим перебуванням у положенні сидячи впродовж робочого дня необхідно виконувати фізичні вправи 2-3 рази протягом робочого дня

(повороти і нахили тулуба, присідання), а також більше рухатись при найменшій нагоді.

Ймовірність механічного травмування може виникнути внаслідок не раціонального розташування робочих місць, або у зв’язку з недбалістю та неуважністю обслуговуючого персоналу. Для запобігання травмуванню персоналу проведено планування ергономіки приміщення, розташування офісних меблів і виробничого обладнання у приміщенні. Розводка крученої пари виконана згідно стандарту EIA/TIA 568 (Американський стандарт проводки у комерційних приміщеннях), а також міжнародному стандарту ISO 11801. Приміщення оснащене офісними меблями, що відповідають ДСТУ EN 527-2:2007 «Меблі для адміністративних приміщень. Робочі столи. Частина 2. Вимоги механічної безпеки».

5.3 Заходи з забезпечення виробничої санітарії та гігієни праці

Заходи з забезпечення виробничої санітарії та гігієни праці мають на меті нівелювання санітарно-гігієнічних та психофізіологічних факторів потенційних небезпек.

Обрані заходи наведено в таблиці 5.1.

До заходів для забезпечення виробничої санітарії та гігієни праці можна віднести наступні:

- оптимальне розташування робочих місць;
- облаштування систем опалення та кондиціонування згідно СНиП 2.04.05-91*У «Отопление, вентиляция и кондиционирование»;
- обладнання приміщення джерелами штучного освітлення у відповідності до ДБН В.2.5-28-2006 "Природне і штучне освітлення";
- обладнання допоміжних приміщень санітарного значення згідно СНиП 2.09.04-87 . "Административные и бытовые здания"

Таблиця 5.1 – Заходи з забезпечення виробничої санітарії та гігієни праці

Вид потенційної небезпеки	Технічні заходи	Організаційні заходи
Підвищена або низька температура повітря робочої зони; підвищена або низька вологість повітря; підвищена або низька рухливість повітря	Обладнання приміщення системами опалення та кондиціонування згідно СНиП 2.04.05-91*У «Отопление, вентиляция и кондиционирование»	
Відсутність або недостатність природного освітлення	Обладнання приміщення джерелами штучного освітлення у відповідності до ДБН В.2.5-28-2006 "Природне і штучне освітлення"	
Розумове перенапруження, перенапруження аналізаторів, монотонність труда, емоціональні перенавантаження	Облаштування допоміжних приміщень згідно СНиП 2.09.04-87 . "Административные и бытовые здания", ДСанПІН 3.3.2.007-98 " Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин"	Побудова розкладу робочого дня згідно ДСанПіН 3.2.007-98 "Державні санітарні правила і норми роботи з візуальними терміналами електронно-обчислювальних машин "

Робочі місця працівників, обладнані персональними комп’ютерами (далі – робочі місця), повинні відповідати вимогам «Правил охорони праці під час експлуатації електронно-обчислювальних машин», затверджених Наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 року № 65 (Правила), та «Державних санітарних правил і норм

роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», затверджених постановою Головного державного санітарного лікаря України (ДСанПіН 3.3.2-007-98). Правила поширюються на всіх суб’єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов’язану з персональними комп’ютерами, у тому числі на тих, які мають робочі місця, обладнані персональними комп’ютерами і периферійними пристроями. Зазначені нормативно-правові акти встановлюють санітарно-гігієнічні вимоги до приміщення, в якому розташоване робоче місце, власне до робочого місця, освітлення, рівнів вібрації і шуму, мікроклімату в приміщенні тощо.

Приміщення, в яких встановлені персональні комп’ютери, повинні мати природне та штучне освітлення відповідно до ДБН В.2.5-28-2006 "Природне і штучне освітлення".

Природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт природною освітленості (КПО) не нижче ніж 1,5%.

Штучне освітлення в приміщеннях з робочими місцями має здійснюватись системою загального рівномірного освітлення. У разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім системи загального освітлення додатково встановлюються світильники місцевого освітлення).

Для розрахунку штучного освітлення необхідно визначити клас робіт згідно ДБН В.2.5-28-2006 "Природне і штучне освітлення". Роботу оператора ПЕОМ можна віднести до класу зорових робіт високої точності (РЗР III), що пов'язано з необхідністю опрацювання документів.

У якості джерел світла обрано люмінесцентні лампи типу ЛБ.

Для розрахунку штучного освітлення використовують, в основному, три методи: світлового потоку (коефіцієнта використання), точковий та питомої потужності.

Призначений для розрахунку загального рівномірного освітлення горизонтальних поверхонь. Цей метод дозволяє врахувати як прямий світловий потік, так і відбитий від стін та стелі. Світловий потік лампи $Ф_л$ визначають за формулою (5.1) :

$$\Phi_{л} = \frac{E_{н} \cdot S \cdot k_{з} \cdot z}{N \cdot n \cdot \eta}, \quad (5.1)$$

де $\Phi_{л}$ - розрахункове значення світлового потоку однієї лампи в кожному світильнику, лм;

$E_{н}$ - нормоване значення освітленості, лк;

S - площа освітлюваної поверхні, м^2;

$k_{з}$ - коефіцієнт запасу;

z - коефіцієнт мінімальної освітленості;

N -загальна кількість світильників;

n - кількість ламп у світильнику;

η - коефіцієнт використання світлового потоку.

Рівень нормованої освітленості $E_{н}$ за ДБН В.2.5-28-2006 для РЗР ІІІ складає 400 лк. Площа приміщення S складає 56 м^2.

Коефіцієнт запасу $k_{з}$ = 1.4 враховує запиленість приміщення, зниження світлового потоку ламп в процесі експлуатації.

Коефіцієнт нерівномірності освітлення z зазвичай дорівнює:

Для оцінки коефіцієнтів відбиття поверхонь приміщення (від стелі $\rho_{с}$, стін $\rho_{ст}$, підлоги $\rho_{п}$) приймаються наступні значення:

$$\rho_{с} = 70\%, \rho_{ст} = 50\%, \rho_{п} = 30\%$$

Чисельне значення індексу приміщення визначають за формулою

$$i = \frac{AB}{h(A+B)} \quad (5.2)$$

де A - довжина приміщення, м;

B - ширина приміщення,м;

h - висота розміщення світильників над робочою поверхнею.

Висота розміщення світильників над робочою поверхнею h визначається за формулою 5.3.

$$h = H - h_p - h_з\,, \qquad (5.3)$$

де H - висота приміщення, м;

h_p - висота робочої поверхні над рівнем полу,м;

$h_з$ - висота звисання світильника від стелі,м

$$h = 3 - 0{,}8 - 0{,}1 = 2{,}1 \text{ м}$$

$$i = \frac{(7 \cdot 8)}{2{,}1 \cdot (7 + 8)} = 1{,}78 \approx 2{,}0$$

Використовуючи наведені розрахунки, для світильника типу ЛПО отримуємо з довідкової літератури значення $\eta = 54$

розраховуємо сумарний світловий потік світлової установки за формулою 5.4:

$$\Phi_\Sigma = \frac{\mathrm{E_н} \cdot S \cdot k_з \cdot z}{\eta}, \qquad (5.4)$$

де ϕ_Σ - розрахункове значення сумарного світлового потоку в приміщенні, лм;

$\mathrm{E_н}$ - нормоване значення освітленості, лк;

S - площа освітлюваної поверхні, м^2;

$k_з$ - коефіцієнт запасу;

z - коефіцієнт мінімальної освітленості;

η - коефіцієнт використання світлового потоку.

$$\phi_\Sigma = \frac{400 \cdot 56 \cdot 1{,}4 \cdot 1{,}1}{0{,}54} = 63881 \text{ лк}$$

Визначаємо максимальну відстань між рядами світильників та світильниками у ряду за формулою 5.5 :

$$L_{max} = [L/h] \cdot h \qquad (5.5)$$

Відповідно,

$$L_{max} = 1{,}4 \cdot 2{,}1 = 2{,}94 \text{ м}$$

Визначимо кількість рядів світильників у приміщенні за формулою:

$$N_{\text{р}} = \frac{B}{L_{max}} \qquad (5.6)$$

$$N_{\text{р}} = \frac{8}{2{,}94} = 2{,}72 \approx 3 \text{ ряди}$$

З огляду на те, що пропонується використовувати лампи типу ЛБ, пропонується організовувати розташування світильників методом "світлового потоку".

При використанні 5 світильників у ряду по 2 лампи у кожному загальна кількість ламп складе 30 шт.

Світловий потік однієї лампи повинен складати:

$$\phi_{\text{л}} = \frac{\phi_{\Sigma}}{N_{\text{л}}} = \frac{63881}{30} = 2129{,}37 \text{ лк}$$

Найближчою зі стандартних ламп за характеристиками потоку є лампа ЛБ потужністю 30Вт та світловим потоком 2180 лк

Визначаємо загальну розрахункову освітленість у приміщенні за формулою 5.7:

$$E_{\text{р}} = \frac{\phi_{\text{л}} \cdot N_{\text{л}} \cdot \eta}{S \cdot k_{\text{з}} \cdot z} \qquad (5.7)$$

$$E_{\text{р}} = \frac{2180 \cdot 30 \cdot 0{,}54}{56 \cdot 1{,}4 \cdot 1{,}1} = 409{,}5 \text{ лк}$$

При правильному виборі типу і кількості стандартних ламп повинна виконуватись умова 5.8:

$$\mathrm{E_р} = (-10\% \ldots + 20\%) \cdot E_{\text{н}} \qquad (5.8)$$

$$\frac{\mathrm{E_р}}{\mathrm{E_н}} = \frac{409{,}5}{400} = 1{,}02$$

Це означає, що умова 5.8 виконується і тип світильників обрано вірно.

Загальна потужність світлової установки розраховується за формулою 5.9:

$$P_{\Sigma} = N_{\text{л}} \cdot P_{\text{л}}\text{, Вт} \qquad (5.9)$$

де $\mathrm{P_л}$ - потужність обраної стандартної лампи.

Згідно 5.9,

$$P_{\Sigma} = 30 \cdot 30 = 900 \text{ Вт}$$

5.4 Заходи з пожежної безпеки

Планування та впровадження заходів пожежної безпеки здійснено у відповідності до Закону України Про пожежну безпеку, Правил пожежної безпеки в Україні та вимогам відповідних нормативних актів.

При плануванні заходів з пожежної безпеки враховано, що обчислювальний центр, згідно НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» відноситься до категорії «Д», а клас можливої пожежі, згідно ДБН В.1.1.7-2016 «Пожежна безпека об’єктів будівництва», визначається, як «А». Враховуючи площу приміщення, що складає 56 м^2 , згідно НАПБ Б.03.001-2004 «Типові норми належності вогнегасників», приміщення обладнане двома порошковими вогнегасниками ВП-5(з) (ОП-5(3)).

Розроблено графік перевірки вогнегасників і призначено відповідальну особу. Вогнегасники в приміщенні розташовані таким чином, щоб відстань від найвіддаленішого місця приміщення до найближчого вогнегасника не перевищувала 20 метрів.

Окрім цього приміщення обладнане пожежною сигналізацією, що використовує комбіновані теплові мультисенсорні датчики та димові іонізаційні датчики, підключені до сигналізаційної димової пожежної установки типу СДПУ-1.

Робочі місця розміщенні в приміщенні таким чином, щоб відстань від найбільш віддаленого робочого місця до найближчого евакуаційного виходу становила не більше ніж 100 метрів (згідно СНиП II-90- 81). Меблі та устаткування розташоване таким чином, щоб забезпечити вільний прохід у разі термінової евакуації завширшки не менш 1 метру.

Для обробки стін, підлоги та стелі приміщення використано негорючі полімерні матеріали, що зменшує ризики швидкого розповсюдження вогню у разі виникнення пожежі.

З організаційних заходів щодо забезпечення пожежної безпеки розроблено план евакуації персоналу в разі виникнення пожежі і розміщено в доступному для перегляду місці, проведено інструктаж з персоналом щодо дотримання правил пожежної безпеки у приміщенні і вміння користуватися наявними засобами пожежогасіння, розроблено план навчань персоналу (з використанням учбових пожежних тривог).

5.5 Заходи безпеки у надзвичайних ситуаціях

Цивільний захист (ЦЗ) - це складова частина системи загальнодержавних оборонних заходів, що проводяться в мирний і воєнний час.

Основними завданнями ЦЗ є:

- захист населення від зброї масового ураження противника;
- забезпечення стійкої роботи підприємства в умовах воєнного часу;
- проведення рятувальних і невідкладних аварійно-відновлювальних робіт в осередках ураження.

Забезпечення виконання заходів цивільного захисту на підприємстві покладено на службу цивільного захисту. Штаб цивільного захисту, якій очолює начальник цивільного захисту, організовує роботу в цьому напрямку. Відповідальною особою за

організацію заходів цивільного захисту, забезпечення відповідного стану засобів цивільного захисту на підприємстві є керівник підприємства.

Одним з найважливіших елементів системи цивільного захисту є організація системи сповіщення служби цивільного захисту і персоналу підприємства про загрозу нападу і застосування зброї масового ураження (ядерної, хімічної, біологічної, тощо). Враховуючи раптовий характер виникнення таких загроз і обмеженість реального часу для попередження, ефективність системи сповіщення є вкрай важливою.

З метою своєчасного попередження працівників підприємства про виникнення безпосередньої небезпеки застосування супротивником зброї масового ураження і про необхідності застосування заходів захисту встановлені наступні сигнали оповіщення цивільного захисту:

- "Повітряна тривога";
- "Відбій повітряної тривоги";
- "Радіаційна небезпека";
- "Хімічна тривога".

Сигнал "Повітряна тривога" попереджає про безпосередню небезпеку ураження супротивником даної місцевості. За допомогою комунікаційних мереж широкомовлення передається повідомлення наступного змісту: "Увага! Увага! Громадяни! Повітряна тривога! Повітряна тривога!". За цим сигналом виробничі об’єкти припиняють роботу, транспорт зупиняється і все населення ховається в захисних спорудах.

Сигнал "Відбій повітряної тривоги" передається органами цивільної оборони. За радіотрансляційної мережі передається текст наступного змісту: "Увага! Увага! Громадяни! Відбій повітряної тривоги! Відбій повітряної тривоги!". За цим сигналом населення, з дозволу комендантів укриттів, залишає їх.

Сигнал "Радіаційна небезпека" подається за допомогою всіх місцевих технічних засобів зв’язку та оповіщення та дублюється звуковими і світловими засобами в населених пунктах і районах, у напрямку до яких рухається радіаційна хмара.

За сигналом "Радіаційна небезпека" необхідно одягнути респіратор, ватяно-марлеву пов’язку, узяти підготовлений запас продуктів, індивідуальні засоби медичного захисту, предмети першої необхідності і піти в найближче протирадіаційне (чи найпростіше) укриття.

Сигнал "Хімічна тривога" подається при загрозі або безпосередньому виявленні хімічного або бактеріологічного нападу (зараження). Він передається по радіотрансляційної мережі: "Увага! Увага! Громадяни! Хімічна тривога! Хімічна тривога!". За цим сигналом необхідно швидко надіти протигаз, а в разі необхідності - і засоби захисту шкіри і при першій же можливості сховатися в захисній споруді.

Таким чином, за допомогою розглянутих сигналів система оповіщення дозволяє своєчасно оповістити працівників підприємства про загрозу виникнення надзвичайної ситуації.

Враховуючи нормативні документи та наведені розрахунки, для забезпечення безпечних умов праці при виконанні персоналом своїх службових зобов’язань, було вжито низку суттєвих заходів, а саме:

- використання захисного заземлення (з електричним опором 4 Ом) згідно ГОСТ 12.1.019-79 «Электробезопасность. Общие требования и номенклатура видов защиты» та «Правил устрою електроустановок» («ПУЕ»);
- для підтримання оптимального рівня освітлення приміщення обладнано 15 світильниками типу ЛПО 2х30, розташованими у 3 ряди по що забезпечують питомий світловий потік 409 лк . Це відповідає вимогам ДБН В.2.5-28-2006 "Природне і штучне освітлення".
- для забезпечення пожежної безпеки приміщення обладнане двома порошковими вогнегасниками ВП-5(3) згідно з НАПБ Б.03.001-2004 «Типові норми належності вогнегасників»,

ВИСНОВКИ

Розроблена в ході магістерської роботи програма не має аналогів, досконально відтворюють або перевершують її за кількістю наданих даних.

Проект здатний задовольнити потреби як простого користувача, давши йому всю необхідну йому інформацію про навколишній екологічний фон, так і підприємств, дозволяючи відокремити їх датчики за допомогою системи акаунтів, проте при цьому об'єднати їх дані з даними пристроїв у відкритому доступі, і отримати при цьому ще більш точну усереднену картину поточної екологічної обстановки.

Також програма здатна надати не тільки загальну картину, але і допомогти проаналізувати тенденцію екологічного фону за допомогою побудови графіків, і допуску користувача до записів в базі даних.

У проекту є всі дані для успішного росту, особливо, якщо випустити сумісні з програмою датчики на вільний ринок, де можливість їх купити буде у кожного бажаючого.

ПЕРЕЛІК ПОСИЛАНЬ

1. Огляд про стан забруднення навколишнього природного середовища на території України за даними спостережень гідрометеорологічних організацій у 2014, 2015, 2016 році (за даними мережі спостережень національної гідрометслужби україни) URL: http://www.cgo.kiev.ua/index.php?fn=u_zabrud&f=ukraine&p=1

2. Данлоп. С. А. Атлас погоды. Атмосферные явления и прогнозы. Санкт Петербург: Амфора. 1999. – 192 с.

3. Другов Ю.С., Беликов А.Б. Методы анализа загрязнений воздуха. Москва: Химия. 1984. – 384 с.

4. Романов. В.И. Дышите мной и наслаждайтесь. Информационно-познавательное пособие. Москва: Химия. 2010. – 402 с.

5. Романов. В.И. Прикладные аспекты аварийных выбросов в атмосферу. Справочное пособие. Москва: Химия. 2006. – 489 с.

6. Байтелова А.И. Источники загрязнения среды обитания. учеб. пособие / Гарицкая М. Ю., Куксанов В. Ф., А.И. Байтелова .— Оренбург : ГОУ ОГУ, 2009 .— 191 с.

7. Квашнин И.М. Промышленные выбросы в атмосферу. Инженерные расчеты и инвентаризация. - М.:АВОК-Г1РЕСС, 2005. - 392 с.

8. Хотунцев Ю.Л. Экология и экологическая безопасность: Учеб. пособие: Рек. УМО по спец. / Ю.Л. Хотунцев. - М. : Академия, 2004. - 480 с.

9. Ветошкин А.Г. Аппаратурное оформление процессов защиты атмосферы от газовых выбросов. Пенза: Изд-во Пенз. гос. ун-та, 2004. - 271 с.

10. Бретшнайдер Б., Курфюрст И. Охрана воздушного бассейна от загрязнений: технология и контроль. Пер. с англ. / Под ред. А.Ф. Туболкина. – Л.: Химия, 1989 – 288 с.

11. Васильев А.А. Вильфанд Р.М. Прогноз погоды. Москва: Химия. 2008. – 62 с.

12. Руководство по краткосрочным прогнозам погоды. Л.:Гидрометеоиздат, 1986.- 502 с.

13. Воробьев В.И. Синоптическая метеорология.-Л.:Гидрометеоиэдат, 1994.- 716 с.

14. Матвеев. Е.С. Android для разработчиков. 3-е изд. СПб.: Питер, 2016. — 512 с.

ДОДАТОК А
Вихідний код програми

Вихідний код LoginActivity:

```
package com.technomagicfox.rayllienstery.clearcity;

import android.content.Intent;
import android.os.Debug;
import android.provider.ContactsContract;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class LoginActivity extends AppCompatActivity {
   userInfo[] Users;
   EditText Login, Pass;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_login);
      PermissionsControl.CheckPermissions(getApplicationContext(), LoginActivity.this);

      Login = (EditText) findViewById(R.id.loginText);
      Pass = (EditText) findViewById(R.id.passwordText);


      Database.Initiate();

      if(Database.usersCount(getBaseContext()) > 0){
         Users = Database.getUserInfo(getBaseContext());
      }else {
         Intent myIntent = new Intent(LoginActivity.this, RegisterActivity.class);
         LoginActivity.this.startActivity(myIntent);
      }
   }

   public void LoginWithoutAccountOnClick(View v){
      Intent myIntent = new Intent(LoginActivity.this, MainMenu.class);
      myIntent.putExtra("user_name", "Без имени");
      myIntent.putExtra("user_login", "Без логина");
      LoginActivity.this.startActivity(myIntent);
   }

   public void LoginWithAccountOnClick(View v){
      boolean Status = false;
      for(int x = 0; x < Users.length ; x++){
         Log.d("Login", Users[x].user_login.toString() + " - " + Login.getText().toString());
         Log.d("Login", Users[x].user_pass.toString() + " - " + Pass.getText().toString());
         if(
               Users[x].user_login.toString().equals(Login.getText().toString()) &&
               Users[x].user_pass.toString().equals(Encryption.MD5(Pass.getText().toString())))
         {
```

```
            Status = true;
            Intent myIntent = new Intent(LoginActivity.this, MainMenu.class);
            myIntent.putExtra("user_name", Users[x].user_name);
            myIntent.putExtra("user_login", Users[x].user_login);
            LoginActivity.this.startActivity(myIntent);
            finish();
        }
      }
      if(!Status){
        Toast.makeText(getBaseContext(), "Неверное имя или пароль!",
            Toast.LENGTH_LONG).show();
      }
    }

    public void RegisterOnClick(View v){
      Intent myIntent = new Intent(LoginActivity.this, RegisterActivity.class);
      LoginActivity.this.startActivity(myIntent);
      finish();
    }
}
```

Вихідний код RegisterActivity:

```
package com.technomagicfox.rayllienstery.clearcity;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class RegisterActivity extends AppCompatActivity {

    EditText Login;
    EditText Username;
    EditText Password1;
    EditText Password2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_register);

      Login = (EditText) findViewById(R.id.loginText);
      Username = (EditText) findViewById(R.id.usernameText);
      Password1 = (EditText) findViewById(R.id.passwordText);
      Password2 = (EditText) findViewById(R.id.passwordText2);
    }

    public void RegisterOnClick(View v){
      if(
            Username.getText().length() >= 6 && //Имя пользователя больше 5 символов
            Login.getText().length() >= 6 && //Логин больше 5 символов
            Password1.getText().length() >= 6 && //Пароль больше 5 символов
            Password1.getText().toString().equals(Password2.getText().toString())//И они сходятся
            )
      {
        Toast.makeText(getBaseContext(), "Регистрация одобрена",
            Toast.LENGTH_LONG).show();

Database.addUser(getBaseContext(),Database.usersCount(getBaseContext()),Username.getText().toString(),Login.getText().to
String(),Password1.getText().toString(),0,0);

        Intent myIntent = new Intent(RegisterActivity.this, MainMenu.class);
```

```
            myIntent.putExtra("user_name", Username.getText().toString());
            myIntent.putExtra("user_login", Login.getText().toString());
            RegisterActivity.this.startActivity(myIntent);
            finish();

        }else {
            Toast.makeText(getBaseContext(), "Регистрация не одобрена",
                    Toast.LENGTH_SHORT).show();
        }
        //DEBUG
        Log.d("Register", Login.getText() + " " + String.valueOf(Login.getText().length()));
        Log.d("Register", Username.getText() + " " + String.valueOf(Username.getText().length()));
        Log.d("Register", Password1.getText() + " " + String.valueOf(Password1.getText().length()));
        Log.d("Register", Password2.getText() + " " + String.valueOf(Password2.getText().length()));
        Log.d("Register", String.valueOf(Password1.getText().toString().equals(Password2.getText().toString())));

    }
}
```

Вихідний код MainMenu:

```
package com.technomagicfox.raylllienstery.clearcity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.concurrent.TimeUnit;

public class MainMenu extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    TextView Info, Username, Temp, Clouds;
    Intent intent;

    ImageView CloudsImage;

    ImageView TempImage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_menu);
        InterfaceInit();

        intent = getIntent();

        CheckWeatherUpdateStatus();
        GetSensorsFromWeb();

        UpdateInfo();

    }
```

```
@SuppressLint("WrongViewCast")
private void InterfaceInit(){
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    //CloudsImage = (ImageView) findViewById(R.id.Clouds);
    //TempImage = (ImageView) findViewById(R.id.TempImage);

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
    drawer.addDrawerListener(toggle);
    toggle.syncState();

    NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);
    View headerView = navigationView.getHeaderView(0);

    Info = (TextView) findViewById(R.id.Info);
    Clouds = (TextView) findViewById(R.id.Clouds);
    Temp = (TextView) findViewById(R.id.TempInfo);
    Username = (TextView) headerView.findViewById(R.id.usernameText);
}//Отвечает за инициализацию интерфейсов

@Override
public void onBackPressed() {
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}//Кнопка "Назад"

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();

    if (id == R.id.map) {
        Intent myIntent = new Intent(MainMenu.this, MapsActivity.class);
        MainMenu.this.startActivity(myIntent);
    } else if (id == R.id.database) {
        Intent myIntent = new Intent(MainMenu.this, DatabaseActivity.class);
```

```
            MainMenu.this.startActivity(myIntent);
        } else if (id == R.id.statistic) {
            Intent myIntent = new Intent(MainMenu.this, StatisticActivity.class);
            MainMenu.this.startActivity(myIntent);
        } else if (id == R.id.settings) {
            Intent myIntent = new Intent(MainMenu.this, Settings.class);
            MainMenu.this.startActivity(myIntent);
        }

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    public void CheckWeatherUpdateStatus() {
        if(!Core.WeatherUpdateStatus){
            Core.WeatherUpdateStatus = true;

            Toast.makeText(getBaseContext(), "Обновление прогноза погоды...",
                    Toast.LENGTH_SHORT).show();

            try {
                new GetWeatherData().execute().get(10, TimeUnit.SECONDS);
                if(WeatherCore.Weather != null) {
                    Toast.makeText(getBaseContext(), "Погода получена...",
                            Toast.LENGTH_SHORT).show();
                }else {
                    Toast.makeText(getBaseContext(), "Погода не получена...",
                            Toast.LENGTH_SHORT).show();
                }

            }catch (Exception e){
                Log.e("Weather Exception", e.toString());
            }

        }
    }//Автоматически обновляем прогноз погоды при входе в главное меню

    private void UpdateInfo(){
        try{
            Username.setText(intent.getStringExtra("user_login"));
        }catch (Exception e){
            Username.setText("Error!");
        }

        if(WeatherCore.Weather == null) {
            Info.setText("Empty");
        }else {
            Temp.setText(WeatherCore.Weather.Temp + " C");
            if(WeatherCore.Weather.Temp > 5){
                // TempImage.setImageResource(R.drawable.weather1);
            }else {
                // TempImage.setImageResource(R.drawable.weather2);
            }


            if(WeatherCore.Weather.Clouds < 10) {
                Clouds.setText("Ясно");
                // CloudsImage.setImageResource(R.drawable.sun);
            }else {
                Clouds.setText("Облачно");
                // CloudsImage.setImageResource(R.drawable.cloud);
            }

            Info.setText(
                    //"Облака: " + WeatherCore.Weather.Clouds + "\n" +
                    "Экологическая обстановка: хорошая\n\n" +
```

```
                "Скорость ветра: " + WeatherCore.Weather.WindSpeed + "\n" +
                //"Направление ветра: " + WeatherCore.Weather.WindDeg + "\n" +
                //"Текущая температура: " + WeatherCore.Weather.Temp + "\n" +
                "Влажность: " + WeatherCore.Weather.Humidity + "\n" +
                "Атмосферное давление: " + WeatherCore.Weather.Pressure + "мм. ртутного столба"
        );
    }
  }

  private void GetSensorsFromWeb(){
    try {
      new GetSensorsData().execute().get(10,TimeUnit.SECONDS);

    } catch (Exception e) {
      Log.e("GetSensorsFromWeb", e.toString());
    }
  }

  public void ToSettingsOnClick(View v){
    Intent myIntent = new Intent(MainMenu.this, Settings.class);
    MainMenu.this.startActivity(myIntent);
  }
}
```

Вихідний код MapsActivity:

```
package com.technomagicfox.rayllienstery.clearcity;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

  private GoogleMap mMap;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified when the map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
  }


  /**
   * Manipulates the map once available.
   * This callback is triggered when the map is ready to be used.
   * This is where we can add markers or lines, add listeners or move the camera. In this case,
   * we just add a marker near Sydney, Australia.
   * If Google Play services is not installed on the device, the user will be prompted to install
   * it inside the SupportMapFragment. This method will only be triggered once the user has
   * installed Google Play services and returned to the app.
   */
  @Override
  public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
```

```
        // Add a marker in Sydney and move the camera
        LatLng Sensor_1 = new LatLng(47.893647, 35.146677);
        mMap.addMarker(new MarkerOptions().position(Sensor_1).title("Marker in Sensor 1"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(Sensor_1));
    }
}
```

Вихідний код DatabaseActivity:

```
package com.technomagicfox.rayllienstery.clearcity;

import android.content.DialogInterface;
import android.content.Intent;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;


public class DatabaseActivity extends AppCompatActivity {
    private SensorsParameters[] Sensor;
    String[] AdapterTitles;

    ListView DataList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_database);

        Sensor = Database.GetSensorsFromDatabase();
        for(int i=0; i<Sensor.length/2; i++){
            SensorsParameters temp = Sensor[i];
            Sensor[i] = Sensor[Sensor.length -i -1];
            Sensor[Sensor.length -i -1] = temp;
        }

        AdapterTitles = new String[Sensor.length];
        for(int x = 0; x < AdapterTitles.length; x++){
            AdapterTitles[x] = Sensor[x].Date;
        }

        //Присваиваем ListView
        DataList = (ListView) findViewById(R.id.DataListView);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
                android.R.layout.simple_list_item_1, AdapterTitles);

        DataList.setAdapter(adapter);

        DataList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View itemClicked, int position,
                                    long id) {
                AlertDialog alertDialog = new AlertDialog.Builder(DatabaseActivity.this).create();
                alertDialog.setTitle(AdapterTitles[position]);
                alertDialog.setMessage(
                        "Номер датчика: " + Sensor[position].ID_dev + "\n" +
                        "Дым: " + Sensor[position].Smoke + "\n" +
                        "Метан. Датчик 1: " + Sensor[position].Methane_1 + "\n" +
                        "Метан. Датчик 1: " + Sensor[position].Methane_2 + "\n" +
                        "Углерод: " + Sensor[position].Carbon + "\n" +
```

```
                    "Сжиженный газ: " + Sensor[position].LPG + "\n" +
                    "Влажность: " + Sensor[position].Humidity + "\n" +
                    "Водород: " + Sensor[position].Hydrogen + "\n" +
                    "Температура: " + Sensor[position].Temp + "\n" +
                    "Дата: " + Sensor[position].Date + "\n"
            );
            alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                            dialog.dismiss();
                        }
                    });
            alertDialog.show();
        }
    });

  }


}
```

Вихідний код StatisticActivity:

```
package com.technomagicfox.rayllienstery.clearcity;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.LinearGradient;
import android.graphics.Shader;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.SeekBar;

import com.github.mikephil.charting.charts.Chart;
import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;

import java.util.ArrayList;

public class StatisticActivity extends AppCompatActivity {
    LineChart Chart;
    int Start = 0;
    int Finish = 10;
    int Shift = 0;

    boolean FillStatus = true;
    int LineWidth = 2;
    int FillAlpha = 16;

    SeekBar ShiftBar;
    private SensorsParameters[] Sensor;//Данные с сенсора

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_statistic);
```

```
    //Задаем базовые переменные
    Chart = (LineChart) findViewById(R.id.linechart);
    Chart.setDragEnabled(true);
    Chart.setScaleEnabled(false);

    //Получаем данные о датчиках из базы данных
    Sensor = Database.GetSensorsFromDatabase();

    SeekInitiate();

    UpdateInfo();
}

@Override
public void onBackPressed() {
    Intent myIntent = new Intent(StatisticActivity.this, MainMenu.class);
    StatisticActivity.this.startActivity(myIntent);
}

private void SeekInitiate(){
    ShiftBar = (SeekBar) findViewById(R.id.ShiftBar);
    ShiftBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
            Shift = progress;
            UpdateInfo();
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
            UpdateInfo();
        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
            UpdateInfo();
        }
    });
}

private void UpdateInfo(){

    //Туман
    ArrayList<Entry> SmokeValues = new ArrayList<>();
    Log.d("Shift", Shift + "");

    for(int x = Start - Shift; x < Finish - Shift; x++){
        SmokeValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Smoke));
    }

    LineDataSet SmokeSet = new LineDataSet(SmokeValues,"Дымка");
    SmokeSet.setFillAlpha(FillAlpha);
    SmokeSet.setColor(Color.parseColor("#304959"));
    SmokeSet.setLineWidth(LineWidth);
    SmokeSet.setValueTextSize(12);
    SmokeSet.setDrawFilled(FillStatus);
    SmokeSet.setDrawCircles(true);
    SmokeSet.setFillColor(getResources().getColor(R.color.colorPrimary));

    //Methane_1
    ArrayList<Entry> Methane1Values = new ArrayList<>();

    for(int x = Start - Shift; x < Finish - Shift; x++){
        Methane1Values.add(new Entry(x,Sensor[Sensor.length - 10 + x].Methane_1));
    }
```

```
LineDataSet Methan1 = new LineDataSet(Methane1Values," Метан ");
Methan1.setFillAlpha(FillAlpha);
Methan1.setColor(Color.parseColor("#E65A08"));
Methan1.setLineWidth(LineWidth);
Methan1.setValueTextSize(12);
Methan1.setDrawFilled(FillStatus);
Methan1.setDrawCircles(true);
Methan1.setFillColor(getResources().getColor(R.color.Methan1Fill));

//Methane_2
ArrayList<Entry> Methane2Values = new ArrayList<>();

for(int x = Start - Shift; x < Finish - Shift; x++){
   Methane2Values.add(new Entry(x,Sensor[Sensor.length - 10 + x].Methane_2));
}

LineDataSet Methan2 = new LineDataSet(Methane2Values,"Метан 2 ");
Methan2.setFillAlpha(FillAlpha);
Methan2.setColor(Color.parseColor("#E50F16"));
Methan2.setLineWidth(LineWidth);
Methan2.setValueTextSize(12);
Methan2.setDrawFilled(FillStatus);
Methan2.setDrawCircles(true);
Methan2.setFillColor(getResources().getColor(R.color.Methan2Line));

//Carbon
ArrayList<Entry> CarbonValues = new ArrayList<>();

for(int x = Start - Shift; x < Finish - Shift; x++){
   CarbonValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Carbon));
}

LineDataSet Carbon = new LineDataSet(CarbonValues,"Углерод ");
Carbon.setFillAlpha(FillAlpha);
Carbon.setColor(Color.parseColor("#12131A"));
Carbon.setLineWidth(LineWidth);
Carbon.setValueTextSize(12);
Carbon.setDrawFilled(FillStatus);
Carbon.setDrawCircles(true);
Carbon.setFillColor(getResources().getColor(R.color.CarbonFill));

//LPG
ArrayList<Entry> LPGValues = new ArrayList<>();

for(int x = Start - Shift; x < Finish - Shift; x++){
   LPGValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].LPG));
}

LineDataSet LPG = new LineDataSet(LPGValues,"LPG ");
LPG.setFillAlpha(FillAlpha);
LPG.setColor(Color.parseColor("#D5D5D5"));
LPG.setLineWidth(LineWidth);
LPG.setValueTextSize(12);
LPG.setDrawFilled(FillStatus);
LPG.setDrawCircles(true);
LPG.setFillColor(getResources().getColor(R.color.LPGFill));

//Humidity
ArrayList<Entry> HumidityValues = new ArrayList<>();

for(int x = Start - Shift; x < Finish - Shift; x++){
   HumidityValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Humidity));
}

LineDataSet Humidity = new LineDataSet(HumidityValues,"Влажность");
Humidity.setFillAlpha(FillAlpha);
Humidity.setColor(Color.parseColor("#33A1FD"));
```

```
    Humidity.setLineWidth(LineWidth);
    Humidity.setValueTextSize(12);
    Humidity.setDrawFilled(FillStatus);
    Humidity.setDrawCircles(true);
    Humidity.setFillColor(getResources().getColor(R.color.HumidityFill));

    //Hydrogen
    ArrayList<Entry> HydrogenValues = new ArrayList<>();

    for(int x = Start - Shift; x < Finish - Shift; x++){
       HydrogenValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Hydrogen));
    }

    LineDataSet Hydrogen = new LineDataSet(HydrogenValues,"Водород");
    Hydrogen.setFillAlpha(FillAlpha);
    Humidity.setColor(Color.parseColor("#2176FF"));
    Hydrogen.setLineWidth(LineWidth);
    Hydrogen.setValueTextSize(12);
    Hydrogen.setDrawFilled(FillStatus);
    Hydrogen.setDrawCircles(true);
    Hydrogen.setFillColor(getResources().getColor(R.color.HydrogenFill));

    //Temp
    ArrayList<Entry> TempValues = new ArrayList<>();

    for(int x = Start - Shift; x < Finish - Shift; x++){
       TempValues.add(new Entry(x,Sensor[Sensor.length - 10 + x].Temp));
    }

    LineDataSet Temp = new LineDataSet(TempValues,"Температура");
    Temp.setFillAlpha(FillAlpha);
    Temp.setColor(Color.parseColor("#A31621"));
    Temp.setLineWidth(LineWidth);
    Temp.setValueTextSize(12);
    Temp.setDrawFilled(FillStatus);
    Temp.setDrawCircles(true);
    Temp.setFillColor(getResources().getColor(R.color.TempFill));

    //Добавление на график
    ArrayList<ILineDataSet> dataSets = new ArrayList<>();
    dataSets.add(SmokeSet);
    dataSets.add(Methan1);
    dataSets.add(Methan2);
    dataSets.add(Carbon);
    dataSets.add(LPG);
    dataSets.add(Humidity);
    dataSets.add(Hydrogen);
    dataSets.add(Temp);

    LineData data = new LineData(dataSets);

    Chart.setData(data);
    Chart.invalidate();
}

public void ChangeOnClick(View v){
    AlertDialog alertDialog = new AlertDialog.Builder(StatisticActivity.this).create();
    alertDialog.setTitle("Изменить");
    alertDialog.setMessage("Доступен всего один датчик"
    );
    alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();
                }
            });
    alertDialog.show();
```

```
    }
}
```

Вихідний код WeatherPart:

```
package com.technomagicfox.rayllienstery.clearcity;

import android.content.Context;
import android.os.AsyncTask;
import android.provider.ContactsContract;
import android.util.Log;
import android.widget.Toast;

import org.json.JSONObject;

import java.lang.*;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.concurrent.ExecutionException;

public class WeatherPart{

}//Эта часть кода отвечает за получения данных погоды из открытых источников

class WeatherCore{
    public static WeatherParametrs Weather;
}

class WeatherParametrs{
    public float Date = 0f;
    public int Clouds = 0;//Облачность в процентах
    public int WindSpeed = 0;//Скорость ветра
    public int WindDeg = 0;//Направление ветра
    public int TempMin = 0;//Прогноз. Минимальная температура
    public int TempMax = 0;//Прогноз. Максимальная температура
    public int Temp = 0;//Текущая температура
    public int Humidity = 0;//Влажность
    public int Pressure = 0;//Атмосферное давление

    public WeatherParametrs(JSONObject data, float date){
        Log.d("WeatherParametrs", "Initiated");

        try {
            Date = date;
            Log.d("WeatherPar.Date", String.valueOf(date) + "");

            Clouds = data.getJSONObject("clouds").getInt("all");
            Log.d("WeatherPar.Clouds", Clouds + "");

            WindSpeed = data.getJSONObject("wind").getInt("speed");
            Log.d("WeatherPar.WindSpeed", WindSpeed + "");

            WindDeg = data.getJSONObject("wind").getInt("deg");
            Log.d("WeatherPar.Clouds", WindDeg + "");

            TempMin = data.getJSONObject("main").getInt("temp_min") - 273;
            Log.d("WeatherPar.TempMin", TempMin + "");

            TempMax = data.getJSONObject("main").getInt("temp_max") - 273;
            Log.d("WeatherPar.TempMax", TempMax + "");

            Temp = data.getJSONObject("main").getInt("temp") - 273;
```

```
            Log.d("WeatherPar.Temp", Temp + "");

            Humidity = data.getJSONObject("main").getInt("humidity");
            Log.d("WeatherPar.Humidity", Humidity + "");

            Pressure = data.getJSONObject("main").getInt("pressure");
            Log.d("WeatherPar.Pressure", Pressure + "");

        }catch (Exception e){
            Log.e("WeatherParametrs", e.toString());
        }
    }

}//Класс, который содержит результаты парсинга погодных условий.

class GetWeatherData extends AsyncTask <Void,Void,String> {
    static JSONObject data = null;

    @Override
    protected String doInBackground(Void... params) {
        try {
            URL url = new
URL("http://api.openweathermap.org/data/2.5/weather?q="+Constants.city+"&APPID=972b2b5d81e3c5e03a141d8568ca8028
");

            HttpURLConnection connection = (HttpURLConnection) url.openConnection();

            BufferedReader reader =
                    new BufferedReader(new InputStreamReader(connection.getInputStream()));

            StringBuffer json = new StringBuffer(1024);
            String tmp = "";

            while((tmp = reader.readLine()) != null)
                json.append(tmp).append("\n");
            reader.close();

            data = new JSONObject(json.toString());

            if(data.getInt("cod") != 200) {
                System.out.println("Cancelled");
                return null;
            }

            if(data!= null){
                WeatherCore.Weather = new WeatherParametrs(data, Float.parseFloat(new
SimpleDateFormat("yyyyMMddHHmmss").format(Calendar.getInstance().getTime())));
                Database.writeWeather(WeatherCore.Weather);
            }


        } catch (Exception e) {
            System.out.println("Weather Exception "+ e.getMessage());
            return null;
        }

        return null;
    }

    @Override
    protected void onPostExecute(String result) {
        // TODO Auto-generated method stub
        super.onPostExecute(result);
    }

}
```

Вихідний код PermissionsControl:

```
package com.technomagicfox.rayllienstery.clearcity;

import android.Manifest;
import android.app.Activity;
import android.content.Context;
import android.content.pm.PackageManager;
import android.os.Build;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.util.Log;

public class PermissionsControl {

    public static void CheckPermissions(Context context, Activity activity){
        if (Build.VERSION.SDK_INT >= 23) {
            //динамическое получение прав на INTERNET
            if (ContextCompat.checkSelfPermission(context, android.Manifest.permission.INTERNET)
                    == PackageManager.PERMISSION_GRANTED) {
                Log.d("INTERNET", "Permission is granted");

            } else {
                Log.d("INTERNET", "Permission is revoked");
                //запрашиваем разрешение
                ActivityCompat.requestPermissions(activity, new String[]{Manifest.permission.INTERNET}, 1);
            }

            //динамическое получение прав на READ_EXTERNAL_STORAGE
            if (ContextCompat.checkSelfPermission(context, android.Manifest.permission.READ_EXTERNAL_STORAGE)
                    == PackageManager.PERMISSION_GRANTED) {
                Log.d("READ_EXTERNAL_STORAGE", "Permission is granted");

            } else {
                Log.d("READ_EXTERNAL_STORAGE", "Permission is revoked");
                //запрашиваем разрешение
                ActivityCompat.requestPermissions(activity, new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
1);
            }

            //динамическое получение прав на WRITE_EXTERNAL_STORAGE
            if (ContextCompat.checkSelfPermission(context, android.Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    == PackageManager.PERMISSION_GRANTED) {
                Log.d("WRITE_EXTERNAL_STORAGE", "Permission is granted");

            } else {
                Log.d("WRITE_EXTERNAL_STORAGE", "Permission is revoked");
                //запрашиваем разрешение
                ActivityCompat.requestPermissions(activity, new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
1);
            }
        }
    }
}
```

Вихідний код Encryption:

```
package com.technomagicfox.rayllienstery.clearcity;

public class Encryption {

    public static String MD5(String md5) {
        try {
            java.security.MessageDigest md = java.security.MessageDigest.getInstance("MD5");
            byte[] array = md.digest(md5.getBytes());
```

```
            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < array.length; ++i) {
                sb.append(Integer.toHexString((array[i] & 0xFF) | 0x100).substring(1,3));
            }
            return sb.toString();
        } catch (java.security.NoSuchAlgorithmException e) {
        }
        return null;
    }
    //Принимает строку, отдает ее в зашифрованном виде. MD5

}
```

Вихідний код LoginActivity:

```
package com.technomagicfox.rayllienstery.clearcity;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;

import java.io.File;

public class Database {
    public static SQLiteDatabase database = null;
    static Cursor query;

    public static void Initiate() {
        Log.d("Database", "default path: " + Core.PathDatabase);
        try {
            if (database == null && CheckDatabaseStatus()) {
                database = SQLiteDatabase.openOrCreateDatabase(Core.PathDatabase, null);
                Log.d("Database", "найдена и была присвоен");
            } else {
                database = SQLiteDatabase.openOrCreateDatabase(Core.PathDatabase, null);
                database.execSQL("CREATE TABLE IF NOT EXISTS users (" +
                        "id_user INTEGER, " +
                        "user_name TEXT, " +
                        "user_login TEXT, " +
                        "user_pass TEXT, " +
                        "access_lead INTEGER, " +
                        "server_id INTEGER)");
                //Пользователи

                database.execSQL("CREATE TABLE IF NOT EXISTS weather (" +
                        "data_time REAL, " +
                        "clouds INTEGER, " +
                        "temperature INTEGER, " +
                        "temperature_min INTEGER, " +
                        "temperature_max INTEGER, " +
                        "wind_speed INTEGER, " +
                        "wind_deg INTEGER, " +
                        "humidity INTEGER, " +
                        "pressure INTEGER)");
                //Погода

                database.execSQL("CREATE TABLE IF NOT EXISTS sensor (" +
                        "id INTEGER, " +
                        "dev_id INTEGER, " +
                        "smoke INTEGER, " +
                        "methane1 INTEGER, " +
                        "methane2 INTEGER, " +
                        "Carbon INTEGER, " +
                        "LPG INTEGER, " +
                        "humidity INTEGER, " +
```

```
                "hudrogen INTEGER, " +
                "temperapure INTEGER, " +
                "datetime TEXT)");
            //Погода

            Log.d("Database", "была создана");
        }

    }catch (Exception e){
        Log.d("Database", e.toString());
    }
}
//Основной метод для работы с БД - проверяет наличие, создает/открывает БД

public static Boolean addUser(Context context, int id_user, String user_name, String user_login, String user_pass, int
access_lead, int server_id){
    try{
        database.execSQL("INSERT INTO users VALUES ( " + id_user + ", '" + user_name +"','"+ user_login + "','" +
Encryption.MD5(user_pass) + "'," + access_lead + ", " + server_id + ");");
        Log.d("Database","user " + user_name + " added!");
        return true;
    }
    catch (Throwable t){
        Log.d("Database","user " + user_name + " is NOT added!");
        return false;
    }
}
//добавление записи пользователя в бд

public static Boolean writeWeather(WeatherParametrs WP){
    try{
        database.delete("weather", null, null);
        Initiate();
        database.execSQL("INSERT INTO weather VALUES ( " + WP.Date + ", " + WP.Clouds +", "+ WP.Temp + ", " +
WP.TempMin + ", " + WP.TempMax + ", " + WP.WindSpeed + ", " + WP.WindDeg + ", " + WP.Humidity + ", " +
WP.Pressure + ");");
        Log.d("Database","Weather added!");
        return true;
    }
    catch (Throwable t){
        Log.d("Database","Weather is NOT added!");
        return false;
    }
}
//добавление записи погоды в бд

public static Boolean writeSensors(SensorsParameters SP){
    try{
        Initiate();
        database.execSQL("INSERT INTO sensor VALUES ( " +
                + SP.ID + ", "
                + SP.ID_dev +", "
                + SP.Smoke + ", "
                + SP.Methane_1 + ", "
                + SP.Methane_2 + ", "
                + SP.Carbon + ", "
                + SP.LPG + ", "
                + SP.Humidity + ", "
                + SP.Hydrogen + ", "
                + SP.Temp + ", '"
                + SP.Date + "');");
        Log.d("Database","Sensor added!");
        return true;
    }
    catch (Throwable t){
        Log.d("Database","Sensor is NOT added!");
        return false;
```

```
    }
}
//добавление записи показаний сенсоров в бд

public static int usersCount(Context context){
    int count = 0;
    if(database == null || query == null){
        Initiate();
    }
    query = database.rawQuery("SELECT * FROM users;", null);
    if(query.moveToFirst()){
        do{
            count++;
            Log.d("Database.User",query.getString(2));
        }
        while(query.moveToNext());
    }
    Log.d("Database.User","найдено " + count + " записей");
    return count;
}
//Считает количество записей пользователей в БД

public static int sensorsInfoCount(){
    int count = 0;
    Initiate();
    query = database.rawQuery("SELECT * FROM sensor;", null);
    if(query.moveToFirst()){
        do{
            count++;
            //Log.d("Database.sensorNumber",String.valueOf(query.getInt(0)));
        }
        while(query.moveToNext());
    }
    Log.d("Database.sensor","найдено " + count + " записей");
    return count;
}
//Считает количество записей пользователей в БД

public static int sensorsLastNumber(){
    Initiate();
    database.query("sensor", null, null, null, null, null, "id");
    query = database.rawQuery("SELECT * FROM sensor;", null);
    query.moveToLast();
    Log.d("Database.sensorLast",String.valueOf(query.getInt(0)));
    return query.getInt(0);
}
//Считает количество записей пользователей в БД

public static SensorsParameters[] GetSensorsFromDatabase(){
    SensorsParameters[] toReturn = new SensorsParameters[sensorsInfoCount()];
    int number = 0;
    Initiate();
    database.query("sensor", null, null, null, null, null, "id");
    query = database.rawQuery("SELECT * FROM sensor;", null);

    if(query.moveToFirst()){
        do{
            toReturn[number] = new SensorsParameters(
                    query.getInt(0),
                    query.getInt(1),
                    query.getInt(2),
                    query.getInt(3),
                    query.getInt(4),
                    query.getInt(5),
                    query.getInt(6),
                    query.getInt(7),
                    query.getInt(8),
```

```
                    query.getInt(9),
                    query.getString(10)
                );
                number++;
            }
            while(query.moveToNext());
        }
        return toReturn;
    }
    //Считает количество записей пользователей в БД

    static boolean CheckDatabaseStatus(){
        File file = new File(Core.PathDatabase);

        if(file.exists()){
            Log.d("Database","существует");
            return true;
        }
        else{
            Log.d("Database","не существует");
            return false;
        }
    }
    //Проверяет наличие файла базы данных. Возвращает булево значение

    public static userInfo[] getUserInfo(Context context){
        int count = usersCount(context);
        userInfo[] toReturn = new userInfo[count];
        if(database == null || query == null){
            Initiate();
        }
        for(int i = 0;i<count;i++){
            toReturn[i] = new userInfo();
        }
        query = database.rawQuery("SELECT * FROM users;", null);
        if(query.moveToFirst()){
            int number = 0;
            do{
                toReturn[number].id_user = query.getInt(0);
                toReturn[number].user_name = query.getString(1);
                toReturn[number].user_login = query.getString(2);
                toReturn[number].user_pass = query.getString(3);
                toReturn[number].access_lead = query.getInt(4);
                toReturn[number].server_id = query.getInt(5);

                number++;

            }
            while(query.moveToNext());
        }
        query.close();
        database.close();
        return toReturn;
    }
    //Возвращает массив экземпляров класса userInfo со считанной информацией пользователей с БД database

}

class userInfo{
    int id_user;
    String user_name;
    String user_login;
    String user_pass;
    int access_lead;
    int server_id;
}
```

Вихідний код Core:

```
package com.technomagicfox.raylllienstery.clearcity;

import android.os.Environment;

public class Core {
    //Состояние обновления прогноза погоды. Предназначено для проверки при новом входе в программу
    public static boolean WeatherUpdateStatus = false;

    //Состояние обновления Данных с датчика. Предназначено для проверки при новом входе в программу
    public static boolean SensorUpdateStatus = false;
    //SQLite database
    public static String PathDatabase = "/data/data/com.technomagicfox.raylllienstery.clearcity/database.db";
}
```

Вихідний код Constants:

```
package com.technomagicfox.raylllienstery.clearcity;

public class Constants {
    public final static String city = "Zaporizhzhya";//Город. За ненадобностью его менять в рамках дипломной работы отнесен сюда.
}
```

Вихідний код AppCompatPreferenceActivity:

```
package com.technomagicfox.raylllienstery.clearcity;

import android.content.res.Configuration;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.support.annotation.LayoutRes;
import android.support.annotation.Nullable;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatDelegate;
import android.support.v7.widget.Toolbar;
import android.view.MenuInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * A {@link android.preference.PreferenceActivity} which implements and proxies the necessary calls
 * to be used with AppCompat.
 */
public abstract class AppCompatPreferenceActivity extends PreferenceActivity {

    private AppCompatDelegate mDelegate;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        getDelegate().installViewFactory();
        getDelegate().onCreate(savedInstanceState);
        super.onCreate(savedInstanceState);
    }

    @Override
```

```
protected void onPostCreate(Bundle savedInstanceState) {
    super.onPostCreate(savedInstanceState);
    getDelegate().onPostCreate(savedInstanceState);
}

public ActionBar getSupportActionBar() {
    return getDelegate().getSupportActionBar();
}

public void setSupportActionBar(@Nullable Toolbar toolbar) {
    getDelegate().setSupportActionBar(toolbar);
}

@Override
public MenuInflater getMenuInflater() {
    return getDelegate().getMenuInflater();
}

@Override
public void setContentView(@LayoutRes int layoutResID) {
    getDelegate().setContentView(layoutResID);
}

@Override
public void setContentView(View view) {
    getDelegate().setContentView(view);
}

@Override
public void setContentView(View view, ViewGroup.LayoutParams params) {
    getDelegate().setContentView(view, params);
}

@Override
public void addContentView(View view, ViewGroup.LayoutParams params) {
    getDelegate().addContentView(view, params);
}

@Override
protected void onPostResume() {
    super.onPostResume();
    getDelegate().onPostResume();
}

@Override
protected void onTitleChanged(CharSequence title, int color) {
    super.onTitleChanged(title, color);
    getDelegate().setTitle(title);
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    getDelegate().onConfigurationChanged(newConfig);
}
```

```
    @Override
    protected void onStop() {
        super.onStop();
        getDelegate().onStop();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        getDelegate().onDestroy();
    }

    public void invalidateOptionsMenu() {
        getDelegate().invalidateOptionsMenu();
    }

    private AppCompatDelegate getDelegate() {
        if (mDelegate == null) {
            mDelegate = AppCompatDelegate.create(this, null);
        }
        return mDelegate;
    }
}
```

ДОДАТОК Б
Вихідний код макетів інтерфейсів

Вихідний код макета інтерфейса activity_login.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LoginActivity">

    <ImageView
        android:id="@+id/background"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:adjustViewBounds="true"
        android:cropToPadding="true"
        android:scaleType="fitCenter"
        app:srcCompat="@drawable/top_part" />

    <EditText
        android:id="@+id/loginText"
        android:layout_width="320dp"
        android:layout_height="40dp"
        android:layout_alignBottom="@+id/background"
        android:layout_centerHorizontal="true"
        android:background="@drawable/login_part"
        android:hint="@string/login"
        android:inputType="textPersonName"
        android:paddingLeft="34dp"
        android:singleLine="true"
        android:textSize="18sp"
        android:visibility="visible" />

    <EditText
        android:id="@+id/passwordText"
        android:layout_width="320dp"
        android:layout_height="40dp"
        android:layout_alignStart="@+id/loginText"
        android:layout_below="@+id/background"
        android:layout_marginTop="7dp"
        android:background="@drawable/password_part"
```

```
    android:hint="@string/password"
    android:inputType="textPassword"
    android:paddingLeft="34dp"
    android:singleLine="true"
    android:textSize="18sp"
    android:visibility="visible" />

<Button
    android:id="@+id/autorizeButton"
    android:layout_width="320dp"
    android:layout_height="55dp"
    android:layout_below="@+id/passwordText"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="35dp"
    android:background="@drawable/login_button_part"
    android:text="Авторизироваться"
    android:onClick="LoginWithAccountOnClick"
    android:textAllCaps="false"
    android:textColor="#ffffff"
    android:textSize="18sp" />

<TextView
    android:id="@+id/registerButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentEnd="true"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:onClick="RegisterOnClick"
    android:clickable="true"
    android:text="@string/registration"
    android:textSize="16sp" />

<TextView
    android:id="@+id/loginWithoutAccoun"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/registerButton"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="8dp"
    android:layout_marginBottom="8dp"
    android:clickable="true"
    android:onClick="LoginWithoutAccountOnClick"
```

```
            android:text="@string/non_account_login"
            android:textSize="16sp" />

    </RelativeLayout>
```

Вихідний код макета інтерфейса activity_register.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical"
    tools:context="com.technomagicfox.rayllienstery.clearcity.RegisterActivity">

    <ImageView
        android:id="@+id/background"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:adjustViewBounds="true"
        android:cropToPadding="true"
        android:scaleType="fitCenter"
        app:srcCompat="@drawable/top_part_register" />

    <TextView
        android:id="@+id/loginLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/loginText"
        android:layout_alignStart="@+id/loginText"
        android:paddingLeft="6dp"
        android:text="@string/login"
        android:textColor="#000000"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/loginText"
        android:layout_width="320dp"
        android:layout_height="40dp"
        android:layout_above="@+id/usernameLabel"
        android:layout_alignStart="@+id/usernameLabel"
        android:background="@drawable/empty_part"
        android:inputType="textPersonName"
        android:paddingLeft="5dp"
        android:singleLine="true"
        android:textSize="18sp"
        android:visibility="visible" />

    <TextView
        android:id="@+id/usernameLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/usernameText"
        android:layout_alignStart="@+id/usernameText"
        android:paddingLeft="6dp"
        android:text="@string/username"
        android:textColor="#000000"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/usernameText"
```

```
        android:layout_width="320dp"
        android:layout_height="40dp"
        android:layout_above="@+id/passwordLabel"
        android:layout_centerHorizontal="true"
        android:background="@drawable/empty_part"
        android:inputType="textPersonName"
        android:paddingLeft="5dp"
        android:singleLine="true"
        android:textSize="18sp"
        android:visibility="visible" />

    <TextView
        android:id="@+id/passwordLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignStart="@+id/passwordText"
        android:layout_below="@+id/background"
        android:text="@string/password"
        android:textColor="#000000"
        android:paddingLeft="6dp"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/passwordText"
        android:layout_width="320dp"
        android:layout_height="40dp"
        android:layout_alignStart="@+id/loginText"
        android:layout_below="@+id/passwordLabel"
        android:background="@drawable/empty_part"
        android:hint="@string/register_pass"
        android:inputType="textPassword"
        android:paddingLeft="5dp"
        android:singleLine="true"
        android:textSize="18sp"
        android:visibility="visible" />

    <EditText
        android:id="@+id/passwordText2"
        android:layout_width="320dp"
        android:layout_height="40dp"
        android:layout_alignStart="@+id/passwordText"
        android:layout_below="@+id/passwordText"
        android:layout_marginTop="9dp"
        android:background="@drawable/empty_part"
        android:hint="@string/register_pass_repeat"
        android:inputType="textPassword"
        android:paddingLeft="5dp"
        android:singleLine="true"
        android:textSize="18sp"
        android:visibility="visible" />

    <Button
        android:id="@+id/registerButton"
        android:layout_width="320dp"
        android:layout_height="55dp"
        android:layout_alignParentBottom="true"
        android:layout_alignStart="@+id/passwordText"
        android:layout_marginBottom="15dp"
        android:background="@drawable/login_button_part"
        android:text="@string/register"
        android:textAllCaps="false"
        android:textColor="#ffffff"
        android:onClick="RegisterOnClick"
        android:textSize="18sp" />


</RelativeLayout>
```

Вихідний код макета інтерфейса content_main_menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/colorPrimaryLight"
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   tools:context=".MainMenu"
   tools:showIn="@layout/app_bar_main_menu">

   <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent">

      <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:layout_marginTop="50dp"
         android:orientation="vertical">

         <ImageView
            android:id="@+id/TempImage"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:srcCompat="@drawable/weather2" />

         <TextView
            android:id="@+id/TempInfo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="25 C"
            android:textAlignment="center"
            android:textColor="@color/White"
            android:textSize="25sp" />
      </LinearLayout>

      <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="370dp"
         android:layout_alignParentStart="true"
         android:layout_alignParentTop="true"
         android:layout_marginStart="0dp"
         android:layout_marginTop="216dp"
         android:orientation="vertical">

         <ImageView
            android:id="@+id/imageView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignStart="@+id/TempImage"
            app:srcCompat="@drawable/cloud" />


         <TextView
            android:id="@+id/Clouds"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="25 C"
            android:textAlignment="center"
            android:textColor="@color/White"
            android:textSize="25sp" />
```

```
    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:gravity="end"
        android:orientation="vertical"
        tools:layout_editor_absoluteX="10dp"
        tools:layout_editor_absoluteY="363dp">


        <TextView
            android:id="@+id/Info"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="25 C"
            android:textAlignment="center"
            android:textColor="@color/White"
            android:textSize="18sp" />
    </LinearLayout>

  </RelativeLayout>

</LinearLayout>
```

Вихідний код макета інтерфейса activity_main_menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main_menu"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main_menu"
        app:menu="@menu/activity_main_menu_drawer" />

</android.support.v4.widget.DrawerLayout>
```

Вихідний код макета інтерфейса nav_header_main_menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="@dimen/nav_header_height"
    android:background="@color/colorPrimaryDark"
    android:gravity="bottom"
```

```
    android:orientation="vertical"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/nav_header_desc"
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        app:srcCompat="@mipmap/ic_launcher_round" />

    <TextView
        android:id="@+id/usernameText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:text="@string/nav_header_title"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

</LinearLayout>
```

Вихідний код макета інтерфейса app_bar_main_menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainMenu">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main_menu" />

</android.support.design.widget.CoordinatorLayout>
```

Вихідний код макета інтерфейса activity_database.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    tools:context="com.example.tf.inquest.InquestActivity">


    <RelativeLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent">


        <LinearLayout
            android:id="@+id/linearLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/colorPrimary"
            android:orientation="vertical">


        </LinearLayout>

        <ListView
            android:id="@+id/DataListView"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_above="@+id/linearLayout2"
            android:layout_below="@+id/linearLayout"
            android:layout_alignParentEnd="true"
            android:headerDividersEnabled="false"
            tools:layout_editor_absoluteX="8dp"
            tools:layout_editor_absoluteY="8dp" />

    </RelativeLayout>

</LinearLayout>
```

Вихідний код макета інтерфейса activity_maps.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity" />
```

Вихідний код макета інтерфейса activity_statistic.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"><![CDATA[
    tools:context=".StatisticActivity">

    ]]>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.github.mikephil.charting.charts.LineChart
            android:id="@+id/linechart"
            android:layout_width="match_parent"
            android:layout_height="462dp"
            android:layout_above="@+id/linearLayout2"
            android:layout_alignParentStart="true"
            android:layout_marginStart="0dp"
            android:layout_marginBottom="1dp"></com.github.mikephil.charting.charts.LineChart>
```

```
    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:background="@color/colorPrimary"
        android:gravity="end"
        android:orientation="vertical"
        tools:layout_editor_absoluteX="10dp"
        tools:layout_editor_absoluteY="363dp">


        <SeekBar
            android:id="@+id/ShiftBar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1" />

        <Button
            android:id="@+id/nextButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_above="@+id/OpenFile"
            android:layout_alignTop="@+id/linearLayout2"
            android:layout_alignParentStart="true"
            android:layout_alignParentEnd="true"
            android:layout_weight="1"

            android:fontFamily="serif"
            android:onClick="ChangeOnClick"
            android:text="Изменить" />

    </LinearLayout>

  </RelativeLayout>
</LinearLayout>
```