МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Комп'ютерних наук, управління та адміністрування
Кафедра Інформаційних технологій

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка мобільного застосування для контролю надактивних органічних з'єднань

Виконав студент 4 курсу групи К-41
Спеціальність 122 Комп'ютерні науки
Кириловський Олександр Олександрович

Керівник ст.викл.
Рольщиков Вадим Борисович

Консультант к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Рецензент к.геогр.н., доцент
Лужбін Анатолій Михайлович

Одеса 2020

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

ПП – програмний продукт

ПХДД – поліхлоровані дибензодіоксини

ПХДФ – поліхлоровані дибензофурани

СКБД – система керування базами даних

СОЗ – стійкі органічні забруднювачі

APK (Android Package) – формат архівних файлів-додатків для «Android»

IDE (Integrated Development Environment) – інтегроване середовище розробки

JVM (Java Virtual Machine) – віртуальна машина Java

SQL (Structured query language) – мова структурованих запитів

UML (Unified Modeling Language) – уніфікована мова моделювання

ВСТУП

Протягом останніх 25 років через збільшення відходів значну небезпеку для навколишнього середовище та здоров’я людини становлять стійкі органічні забруднювачі (СОЗ). В основному це галогенові органічні сполуки синтезовані в промислових цілях. Вони характеризуються низькою розчинністю у воді та здатністю накопичуватися в живих організмах[1][1)].

СОЗ переносяться повітрям та, не розкладаючись, впливають на всі живі організми.

Найбільш токсичними є ПХДД (поліхлоровані дибензодіоксини) і ПХДФ (поліхлоровані дибензофурани), також відомі як діоксини і фуранами. Ці речовини утворюються при високотемпературних процесах. Останнім часом діоксини широко поширилися в усьому світі й виявляються в тканинах людей і тварин у будь-якій частині світу.

Виходячи із загального положення велика токсичність СОЗ є невирішеною, тому їх відносять до глобальних екологічних проблем.

Метою дипломної роботи є створення мобільного додатку, який зможе проводити розрахунки ПХДД і ПХДФ.

Представлений дипломний проект містить 53 сторінки, 30 рисунків, 18 посилань та 17 листів додатків.

[1)][1] Федоров Л.А. Диоксины как экологическая опасность: ретроспектива и перспективы. М. : Наука. 1993. 266с. ISBN 5-02-001674-8.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В результаті розробки отримаємо програму для допомоги у розрахунку півроспаду СОЗ за різний період часу.

Актуальність розробки програми полягає у проханні кафедри екології та охорони довкілля Одеського державного екологічного університету створити мобільний додаток для розрахунку ПХДД і ПХДФ.

Готовий програмний продукт має забезпечувати наступні функціональні можливості:

− розраховувати ПХДД і ПХДФ за різні періоди часу;

− мати графічну оболонку;

− креслити графік, який показує зміни ПХДД і ПХДФ протягом часу;

− повинен мати базу даних, для зберігання результатів.

Згідно з джерела[2][1)] у список Стокгольмської конвенції про СОЗ у 2001 році було занесено наступні сполуки.

- дихлордифеніл-трихлоретан (інсектицид, токсичний для багатьох організмів, накопичується в харчовому ланцюгу);
- альдрин (пестицид-інсектицид, що є дуже токсичним для риб, птахів і людини);
- дильдрин;
- ендрін (пестицид-інсектицид, який є високотоксичним для риб);
- хлордан (інсектицид проти термітів, який опинився токсичним для риб, птахів; у людини впливає на імунну систему, потенційний канцероген);
- мірекс (інсектицид проти мурашок, також є потенційним канцерогеном);
- токсафен (інсектицид проти кліщів, є потенційним канцерогеном);

[1)][2] Стокгольмская конвенция о стойких органических загрязнителях – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Стокгольмская_конвенция_о_стойких_органических_загрязнителях (дата звернення 15.02.2020).

- гептахлор (інсектицид, проти ґрунтових комах, токсичний для птахів);
- поліхлоровані дифеніли (ПХД);
- гексахлорбензол (ГХБ) (пестицид-фунгіцид, впливає на репродуктивні органи);
- поліхлордібензодіоксіни (ПХДД);
- поліхлордібензофурани (ПХДФ).

Стокгольмська конвенція націлена на скорочення використання, припинення виробництва і подальшу повну ліквідацію СОЗ.

Поступово список розширюється. Конвенція вимагає від сторін ліквідувати ті речовини, які навмисно виробляються, знаходяться в торговельному обороті і використовуються та зменшувати викиди тих з, які утворюються ненавмисно в результаті діяльності людини.

Відповідно до статті 8 Стокгольмської конвенції, список можливо збільшити шляхом додавання нових з'єднань.

У 2009 році, після четвертого з'їзду сторін конвенції до списку додали ще 9 органічних сполук, а саме:

- альфа гексахлорциклогексан;
- бета гексахлорциклогексан;
- хлордекан;
- гексабромбіфеніл;
- гекса- і гептахлорбіфеніловий ефір;
- ліндан;
- пентахлорбензол;
- перфтороктановой сульфонат;
- тетрабромдіфеніловий ефір і пентабромдіфеніловий ефір.

Більшість з цих сполук заборонені для виробництва. Проте є і ті, що утворюються ненавмисно. Вони найбільш токсичні і є побічним продуктом абсолютно різних виробництв з різних галузей.

Моделювання процесу розробки

Процес розробки мобільного додатку змодельовано за домопогою методології графічного опису систем і процесів IDEF0.

Методологія IDEF0 складається з контекстної діаграми та діаграм декомпозиції. В контекстній діаграмі всього одна робота, яку виконує система в цілому – розробка мобільного додатку.

Для зв'язків об'єкта моделювання з навколишнім середовищем використовують граничні стрілки.

Є 5 типів граничних стрілок:

- вхід (зліва від блоку)
- вихід (виходить з правої сторони блоку);
- управління (зверху);
- механізм (знизу);
- виклик (виходить знизу).

Блоки-роботи діаграми декомпозиції зв’язані між собою внутрішніми стрілками.

Існує 5 типів внутрішніх стрілок:

- зв'язок по входу, зв'язок з управління (показує домінування вище стоячої роботи);
- зворотній зв'язок по входу;
- зворотній зв'язок з управління;
- зв'язок вихід-механізм (показує, що ресурси однієї роботи необхідні для іншої).

На рис. 1.1 показана контекстна діаграма нульового рівня.

Рисунок 1.1 – Контекстна діаграма нульового рівня

На рис. 1.2 показана діаграма декомпозиції першого рівня.

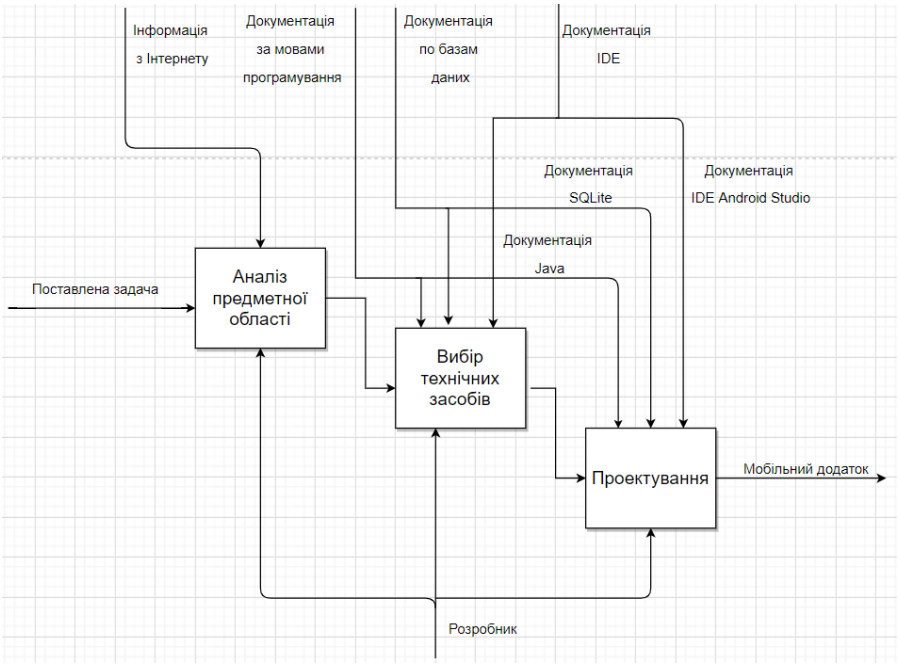

Рисунок 1.2 – Діаграма декомпозиції першого рівня

На рис. 1.3 показана діаграма декомпозиції першого рівня.

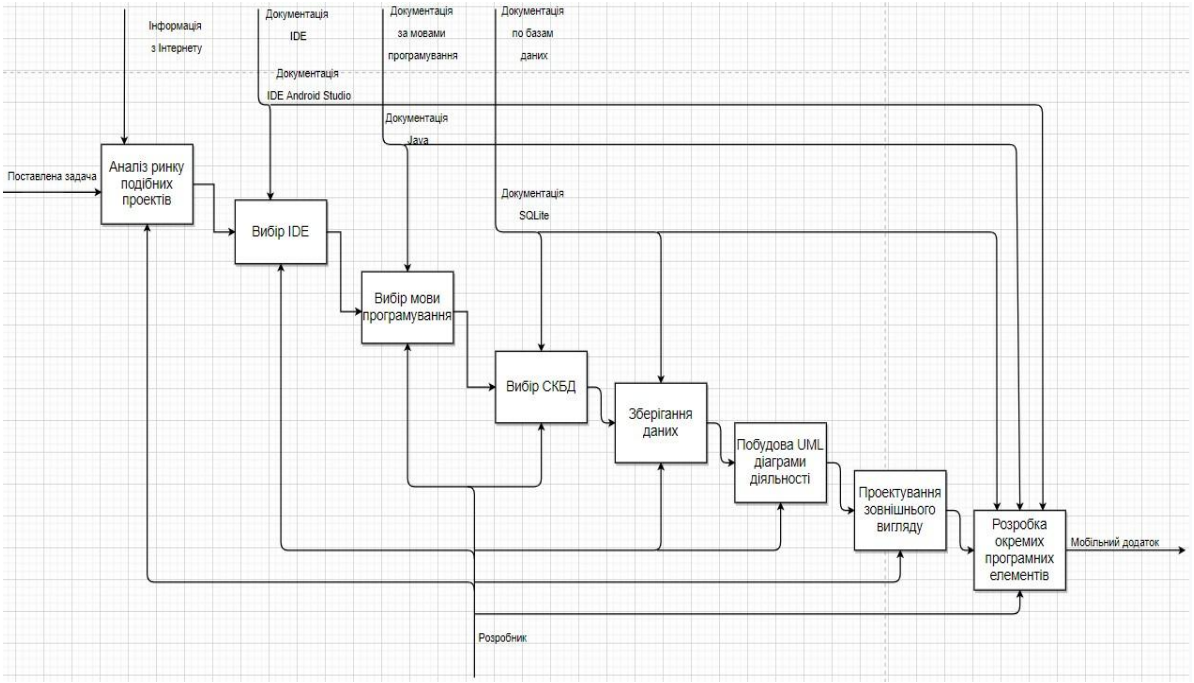

Рисунок 1.3 – Діаграма декомпозиції другого рівня

Аналіз ринку подібних проектів

Серед існуючого програмного забезпечення, спрямованого на вирішення екологічних проблем є програми EWG's Healthy Living[3][1)], Earth-Now[4][2)], PCDD_PCDF_Calcualte_and_Draw[5][3)] та Loss of the night[6][4)].

[1)][3] Приложения в Google Play – EWG's Healthy Living. URL: https://play.google.com/store/apps/details?id=com.skindeep.mobile&hl=uk (дата звернення 17.02.2020).

[2)][4] Приложения в Google Play – Earth-Now. URL: https://play.google.com/store/apps/details?id=gov.nasa.jpl.earthnow.activity&hl=ru (дата звернення 19.02.2020).

[3)][5] Нємцев В.Є. Створення програми для часової інвентаризації накопичення стійких органічних забруднюючих речовин у довкіллі. – URL: http://eprints.library.odeku.edu.ua/5940/1/Niemtsev_Rozrobka_inventarizacii_B_2019.pdf (дата звернення 22.02.2020).

[4)][6] Приложения в Google Play – Loss of the night. URL: https://play.google.com/store/apps/details?id=com.cosalux.welovestars&hl=ru (дата звернення 24.02.2020).

Розгляд застосування EWG’s Healthy Living

Від їжі до засобів гігієни: ми піддаємось дії хімічних речовин щосекунди, але про них ми знаємо дуже мало.

EWG’s Healthy Living – це мобільний додаток, орієнтований для допомоги у виборі продуктів з найменшим вмістом синтетичних хімікатів.

EWG об'єднала два найпопулярніші ресурси, бази даних Skin Deep та Food Scores, щоб сформувати додаток здорового життя. Тепер надійні рейтинги EWG для понад 120 000 продуктів харчування та косметики є у вас під рукою.

На рис. 1.4 наведений загальний вигляд інтерфейсу застосування.

Рисунок 1.4 – Головна сторінка додатку

Після сканування штрих-коду, пошуку по імені або перегляду за категоріями показує список продуктів (див. рис. 1.5).

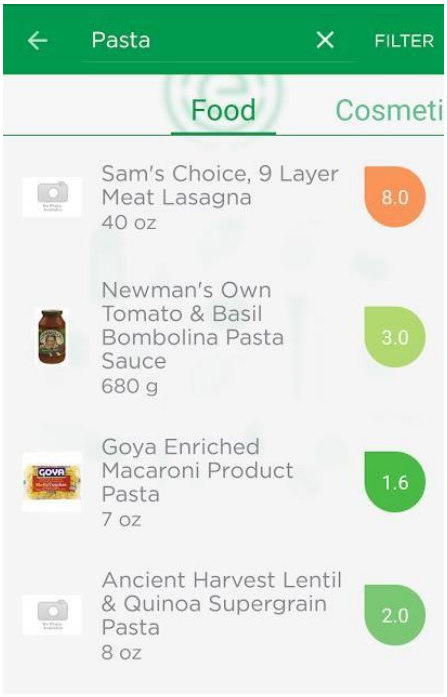

Рисунок 1.5 – Список продуктів

На рис. 1.6 зображено інформацію про обраний продукт. Також EWG показує вміст хімікатів в балах від 1 до 10, де 1 найкращий.

Рисунок 1.6 – Інформація про продукт

На жаль, додаток орієнтований тільки на ринок продуктів США.

Розгляд застосування Earth-Now

Earth-Now – це додаток, що візуалізує останні глобальні кліматичні дані із супутників Землі. Він був розроблений командами із застосування програм та технологій візуалізації технологій зв'язку та візуалізації Землі в лабораторії реактивного руху НАСА за підтримки штабу НАСА

На рис. 1.7 наведено 3D модель землі та кнопки вибору необхідної інформації.

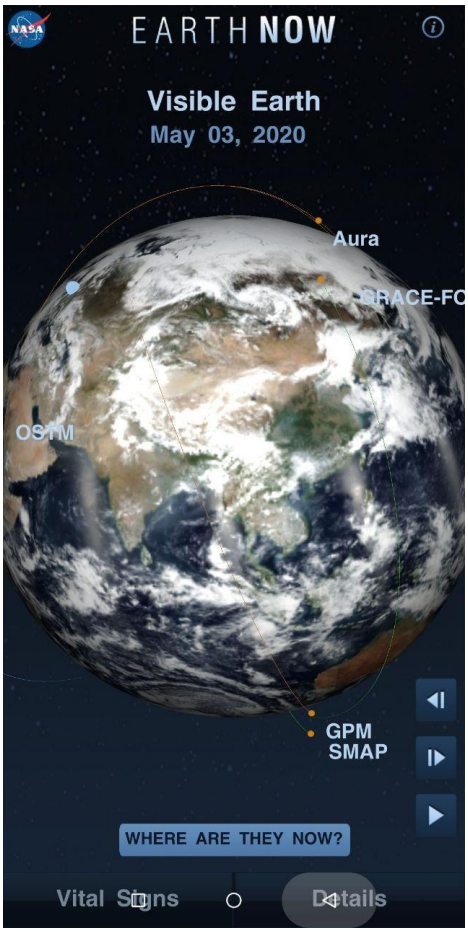

Рисунок 1.7 – Головний екран

На рис. 1.8 показано категорії кліматичних даних.

Рисунок 1.8 – Меню вибору інформації для відображення

Після вибору категорії відображається модель Землі з відповідною інформацією, наприклад окис вуглецю (див. рис. 1.9).

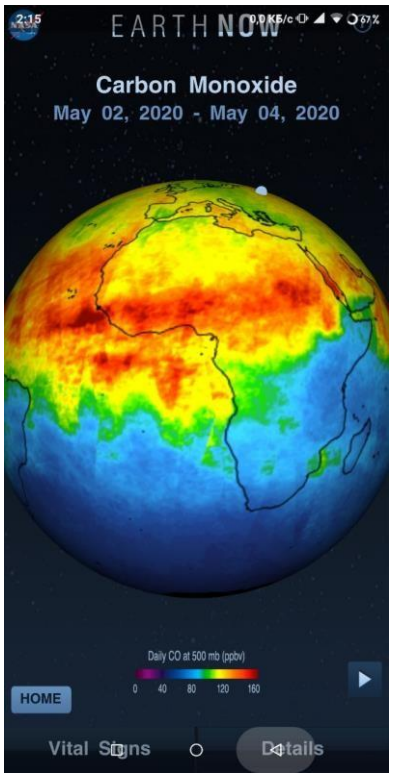

Рисунок 1.9 – Модель землі з показниками окису вуглецю

Розгляд застосування PCDD_PCDF_Calcualte_and_Draw

PCDD_PCDF_Calcualte_and_Draw – це застосування для розрахунку ПХДД і ПХДФ за різний період часу.

На першій вкладці (див. рис. 1.10) спочатку потрібно додати місто, а потім вписати значення у відповідні поля.

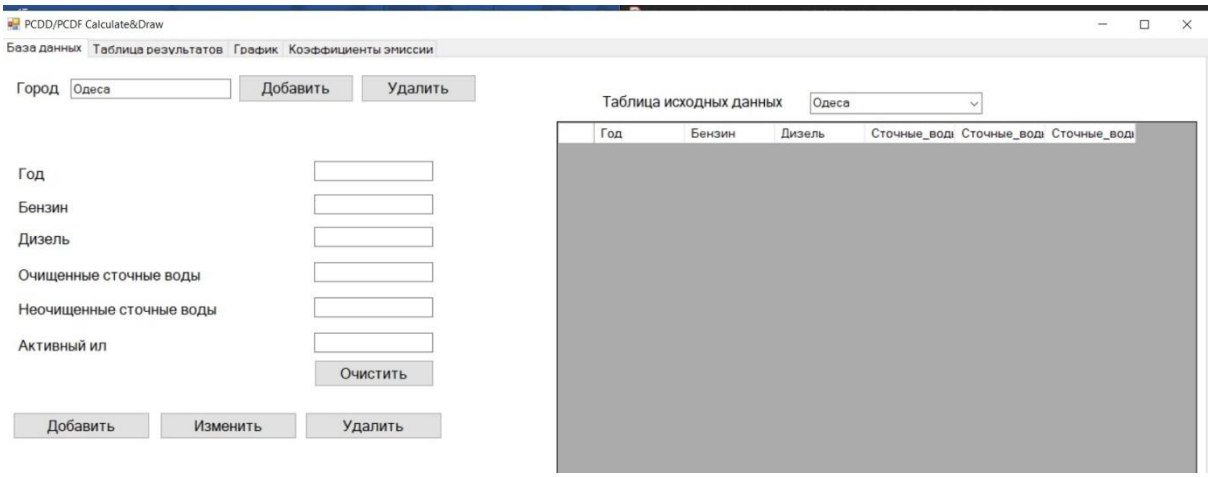

Рисунок 1.10 – Головна сторінка

Перейшовши до вкладки «Таблица результатов», можна побачити розрахунки ПХДД і ПХДФ для різних типів забруднення щодо обраного міста (див.рис.1.11).

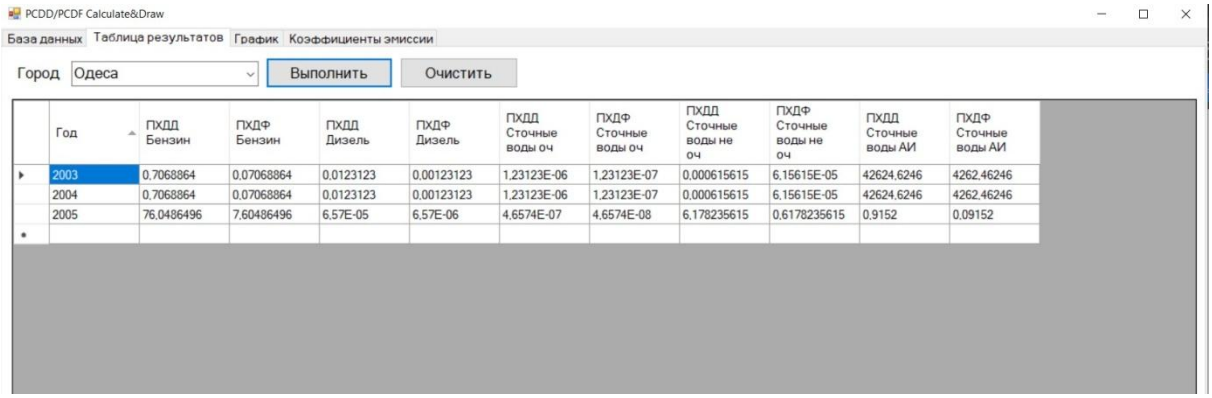

Рисунок 1.11 – Вкладка «Таблица результатов»

Також у користувача є можливість побачити графік піврозпаду ПХДД та ПХДФ протягом часу для різних типів забруднення (див.рис.1.12).

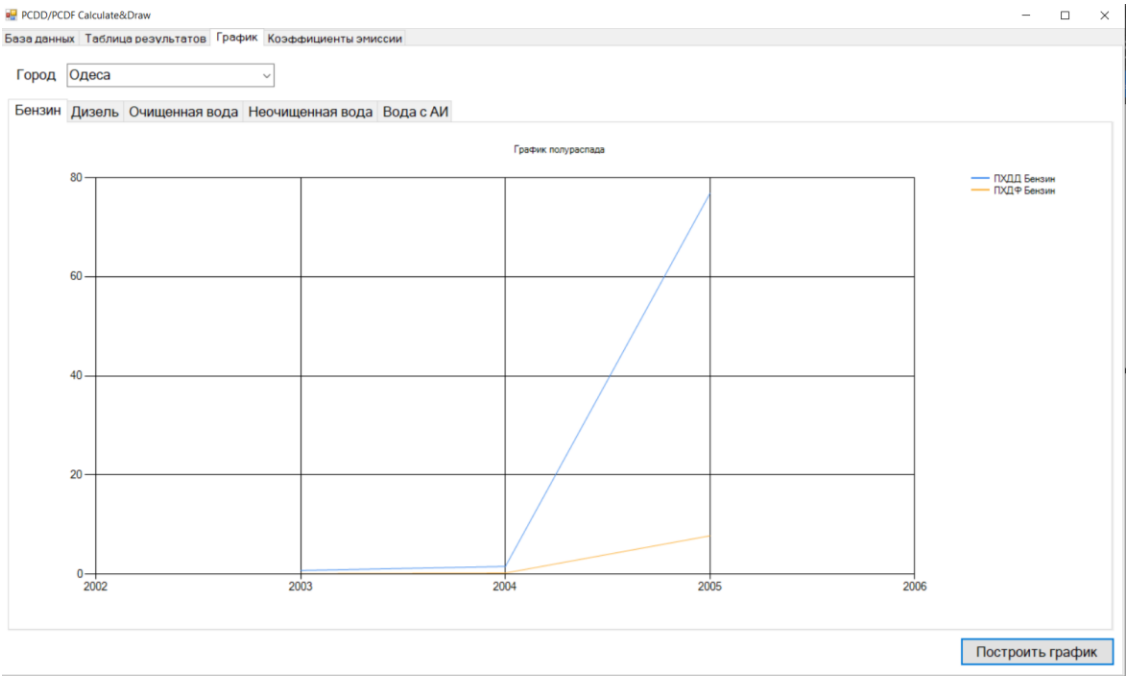

Рисунок 1.12 – Графік піврозпаду

На останній вкладці є можливість редагувати коефіцієнти за якими виконуються розрахунки (див.рис.1.13).

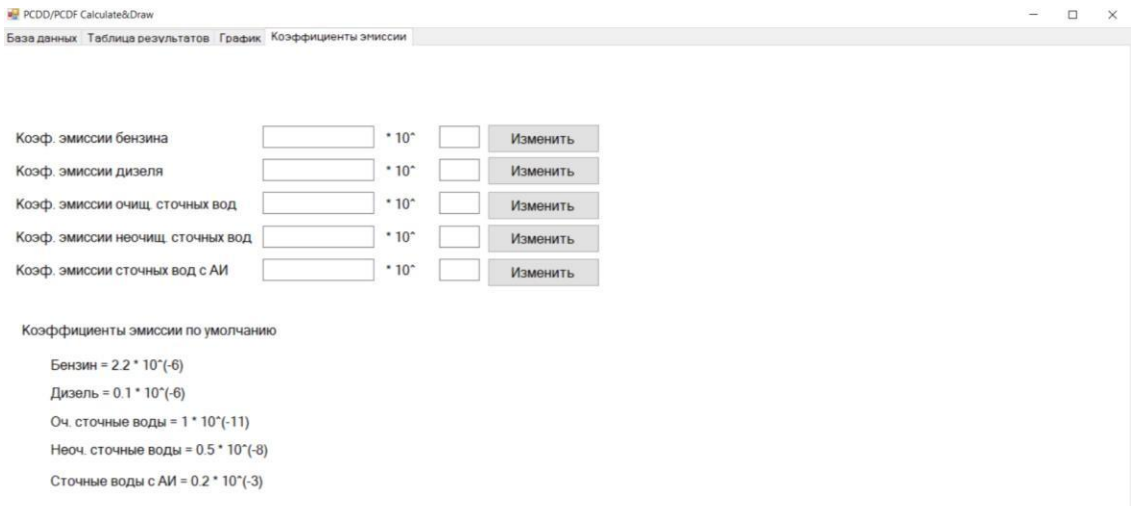

Рисунок 1.13 – Коефіцієнти

До плюсів програми можна віднести те, що серед існуючого ПЗ аналогів саме для розрахунку ПХДД і ПХДФ немає.

Розгляд застосування Loss of the night

Loss of the night – це застосування, засноване на Google в Sky Map для вимірювання забруднення нічного неба. Дозволяє за рахунок розгляду оцінити рівень забруднення. Це одночасно пізнавальна і важлива наукова інформація, яка може допомогти захистити навколишнє середовище в майбутньому.

На рис. 1.14 зображено головний екран, на якому показано сузір’я і кнопку для початку тесту.

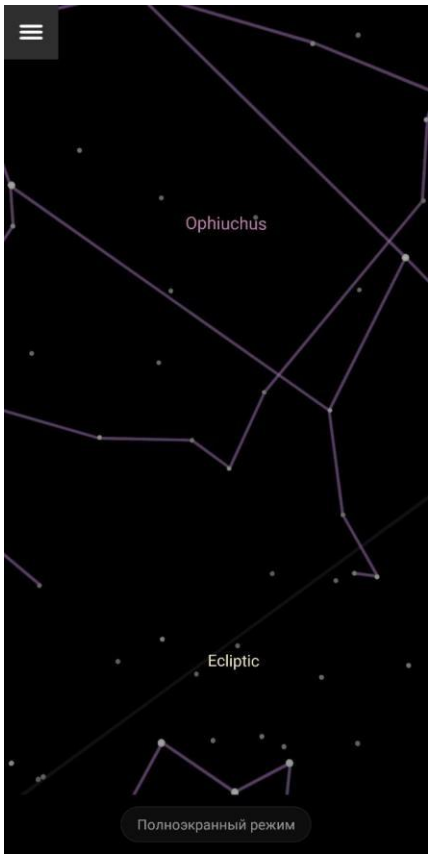

Рисунок 1.14 – Стартовий екран застосування

Після того, як тест розпочато (див. рис. 1.15) застосування наводить користувача до необхідної зірки.

Рисунок 1.15 – Екран пошуку зірки

На наступній вкладці (див. рис. 1.16) вже знайдена зірка, і застосування опитує користувача наскільки її видно.

Рисунок 1.16 – Екран опитування щодо видимості зірки для користувача

На останній вкладці (див. рис. 1.17) зображено меню завершення роботи застосування та питання чи хоче користувач відправити дані до вчених або продовжити тест.

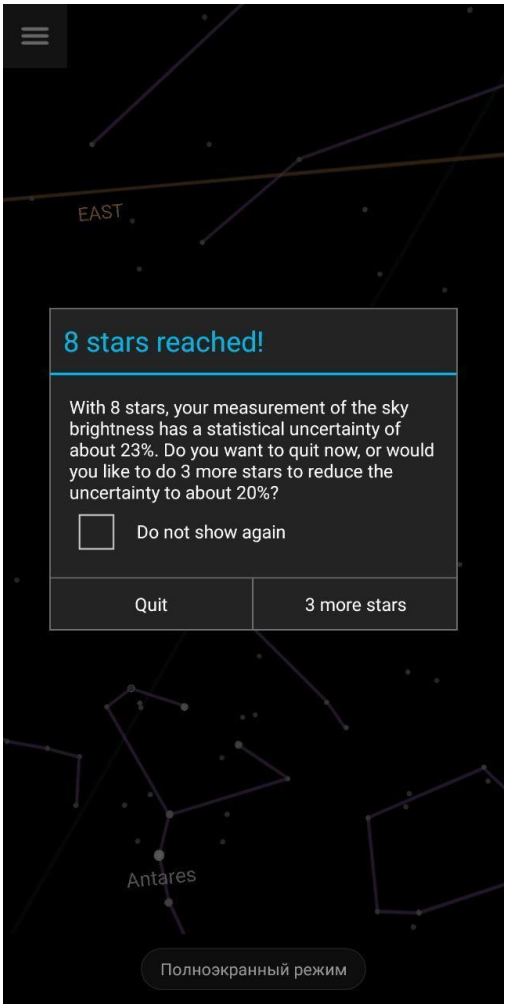

Рисунок 1.17 – Екран завершення тесту

Всі ці застосування є яскравими прикладами необхідності аналогічного ПЗ для різних галузей екології. Але жодне з них не надає можливості контролювати і прогнозувати накопичення СОЗ у довкіллі того чи іншого регіону.

2 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНІЧНИХ ЗАСОБІВ

Вибір середовища розробки для програмування – відповідальне завдання. IDE, на якому ви пишете свій код, визначає не тільки його якість, але навіть структуру і стиль написання. Тому між розробниками не вщухають суперечки про те, яке середовище розробки краще.

Вибір інтегрованого середовища розробки

Для створення програми на ОС Android найбільшою популярністю користуються такі середовища розробки:

– «Android Studio»
– «Eclipse»
– «Xamarin Studio»

Розгляд Android Studio

Android Studio – це універсальне середовище розробки, так як дозволяє оптимізувати роботу майбутніх додатків для роботи не тільки на смартфонах, але і на планшетах, портативних ПК, які працюють на основі даної операційної системи. Фактично Android Studio – це відома Java IDE IntelliJ IDEA з плагінами, що відрізняє від IDEA дрібницями, але ці дрібниці значно полегшують розробку додатків під Android.

У програму також вбудований емулятор, що дозволяє перевірити коректну роботу програми на пристроях з різними екранами, з різними співвідношеннями сторін[7][1)].

[1)][7] Android Studio: среда разработки мобильных приложений. URL: https://arduinoplus.ru/android-studio/ (дата звернення 03.03.2020).

Середа Android Studio призначена як для невеликих команд розробників мобільних додатків (навіть в кількості однієї людини), або ж великих міжнародних організацій. Додатки розробляються в Android Studio з використанням Java, C ++ або Kotlin , яку Google анонсував 17 травня 2017, як офіційну мову програмування. В основі робочого процесу Android Studio закладений концепт безперервної інтеграції, що дозволяє відразу ж виявляти наявні проблеми[8][1)].

Також в Android Studio реалізовані:

- підтримка системи автоматичного складання Gradle;
- унікальна система переробки коду;
- інструменти для пошуку та усунення різних проблем;
- вікно попереднього перегляду.

В ході вивчення на практиці IDE Android Studio були визначені наступні недоліки:

- необхідно мати базовий рівень програмування на мові Java і знання англійської мови;
- високі системні вимоги для ПК;
- стандартні емулятори вимогливі по відношенню до системних ресурсів, довго запускаються та не мають усіх можливостей, що є у реальних смартфонах.

Переваги Android Studio:

- зручний дизайн.
- зручний конструктор інтерфейсів, що дозволяє переглядати відображення екрану на будь-якому пристрої.
- зручна структура проекту.
- наявність логів для відстеження помилок, процесів і потоків.

[1)][8] Android Studio IDE от Google. URL: http://wnfx.ru/android-studio-ide-ot-google/ (дата звернення 03.03.2020).

Розгляд Eclipse

Eclipse – це безкоштовне середовище розробки від некомерційної організації Eclipse Foundation.

Завдяки активному розвитку, а також підтримки з боку компанії і сторонніх розробників, на даний момент у цій IDE є наступні переваги:

- офіційна русифікація інтерфейсу і документації
- відмінна продуктивність на слабких ПК
- велике число доповнень (наприклад, для роботи з сервером, базою даних)
- можливість підключення модулів
- можливість групової розробки

Eclipse була дуже популярна кілька років тому. Однак у зв'язку з виходом Android Studio, в 2014 році Google перестала підтримувати Eclipse як основне середовище для розробки додатків під ОС Android[9][1)].

2.1.2 Розгляд Xamarin Studio

Xamarin Studio – це середовище розробки додатків для Windows, Android і iOS. IDE має досить простий інтерфейс. Для написання коду використовуються мови програмування C#, Visual Basic.NET, C/C++, Vala.

Розробники мають можливість для робити віддаленим налагодженням на пристроях Android, IOS та Windows.

Недоліки Xamarin Studio є: регулярні помилки, як безпосередньо в самій IDE, так і в вихідному коді, іноді проект перестає компілюватиметься. Також, перенести вже готові програми на Xamarin досить важко[10][2)].

[1)][9] StarDroid - уроки по Android. URL: https://venomwind.wixsite.com/stardroid/blank-thepk (дата звернення 03.03.2020).

[2)][10] Подробно о Xamarin. URL: https://habr.com/ru/post/188130/ (дата звернення 03.03.2020).

Через невелику популярність цієї IDE та регулярні помилки від неї довелося відмовитися.

У результаті, в якості середи розробки була вибрана IDE Android Studio. Причиною вибору є досить зручний інтерфейс та легкість написання коду.

Вибір мови програмування

При виборі мови програмування були розглянуті Java та Kotlin.

Розгляд Java

Java – об'єктно-орієнтована мова програмування, яку розробила компанія «Sun Microsystems», яка зараз належить компанії «Oracle». Основним мотивом для створення Java була потреба в мові програмування, яку можна було б використовувати для створення ПЗ, що вбудовується в різноманітні побутові електронні прилади[11][1)].

Більшість синтаксису для Java взято із C і C++. Процес розробки значно полегшено, через виконання деяких дій віртуальною машиною, а не розробниками.

В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи і устаткування , що дозволяє виконувати Java-додатки на всіх пристроях, на яких встановлено JVM. Через гнучку систему безпеки, виконання програми повністю контролюється віртуальною машиною. Операції, які перевищують встановлені повноваження програми, викликають негайне переривання.

[1)][11] Java – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Java (дата звернення 06.03.2020).

Недоліком JVM є зниження продуктивності. Проте, ряд удосконалень збільшив швидкість виконання програм на Java:

- застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми з можливістю збереження версій класу в машинному коді;
- широке застосування платформено-орієнтованого коду в стандартних бібліотеках;
- апаратні засоби, що забезпечують прискорену обробку байт-коду.

Java в порівнянні з C++ виконує програми в середньому в півтора-два рази повільніше. Також, споживання пам'яті JVM в декілька разів більше, ніж програмою на C ++[12][1)].

Отже, до переваг Java можна віднести наступне:

- поєднання простого синтаксису з зручним в роботі середовищем розробки дозволяє швидше створювати нові програми;
- велику кількість бібліотек для вирішення різних задач;
- програми, написані на Java, працюють на всіх пристроях на яких встановлена JVM.

Розгляд Kotlin

Kotlin – це статично типізована мова програмування, розроблена компанією JetBrains. Подібно до мови Java, Kotlin став відмінним вибором для розробки додатків на Android. Це можна побачити навіть з того факту, що Android Studio поставляється з вбудованою підтримкою Kotlin, як і з підтримкою Java.

Метою для створення Kotlin було зробити такий продукт, який можна буде використовувати в змішаних проектах, які створюються на декількох мовах, а також він повинен працювати всюди, де працює Java.

[1)][12] Java – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Java#Основные_особенности_языка (дата звернення 06.03.2020).

Уже в 2017 році Kotlin став офіційною мовою для розробки застосунків для платформи Android. Він має підвищену продуктивність: програмний код на ньому виходить в середньому на 40% коротше, ніж на інших мовах, а також він сумісний з Java.

Відомо, що програмісти витрачають більше часу на перевірку коду, ніж на його написання, тому важливо, щоб конструкції, доступні в мові програмування, дозволяли писати програми коротко і зрозуміло. Завдання Kotlin – поліпшити цю ситуацію. На жаль, строгі методи оцінювання мов з точки зору їх лаконічності розвинені досить слабо, але є непрямі критерії. Один з них – можливість створення бібліотек, робота з якими близька до використання предметно-орієнтованих мов. Для цього необхідна певна гнучкість синтаксису в сукупності з функціями вищих порядків, тобто ті, що приймають інші функції як аргументи чи повертає іншу функцію як результат.

Тепер, коли програмісти починають створювати новий додаток в офіційному середовищі розробки Android Studio, вони відразу можуть включити плагін «підтримка Kotlin».

До плюсів Kotlin можна віднести можливість конвертувати вже створені рядки коду на інших мовах в мову Kotlin та вставляти блоки на інших мовах в рядки коду на Kotlin. Під час компіляції коду відбувається безліч перевірок, покликаних гарантувати, що ті чи інші помилки не відбудуться під час виконання[13][1)].

Вибір системи керування базами даних

Розгляд MySQL

MySQL – система керування реляційними базами даних з відкритим кодом.

[1)][13] Kotlin – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Kotlin (дата звернення 08.03.2020).

Для підвищення швидкодії обробки великих баз даних, компанія «TcX» розробила MySQL. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL. Походження назви «MySQL» – досі не відомо.

Sun Microsystems придбала розробника СКБД MySQL у 2008 році. Зараз ця БД є власністю компанії Oracle Corporation[14][1)].

MySQL є однією з найпопулярніших за декількох причин:

- ліцензійна політика MySQL більш гнучка в порівнянні з іншими серверами БД. Вона розповсюджується безкоштовно за винятком продажі послуг, що створені з її допомогою;
- також є можливість змінити вихідний код у відповідності до ваших потреб;
- широкий спектр кластерних серверів підтримує MySQL;
- використання MySQL протягом багатьох років, дає розробникам велику кількість ресурсів. Тому користувачі можуть розраховувати на швидку розробку ПЗ;
- за допомогою системи доступу і керування обліковими записами MySQL встановлює високий рівень безпеки. Доступна перевірка на основі хоста і шифрування пароля.

До переваг можна віднести наступне.

По-перше, простота: MySQL легко встановлюється.

По-друге, велика кількість функцій: MySQL підтримує більшу частину функціоналу SQL.

По-третє, безпека: в MySQL вбудовано багато функцій безпеки.

По-четверте, потужність і масштабованість: MySQL може працювати з великими обсягами даних[15][2)].

[1)][14] MySQL – Вікіпедія. URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення 11.03.2020).

[2)][15] Что Такое MySQL: Объяснение MySQL Для Начинающих. URL: https://www.hostinger.ru/rukovodstva/shto-takoje-mysql/ (дата звернення 11.03.2020).

2.3.1 Розгляд SQLite

SQLite є безкоштовною бібліотекою, яка реалізує автономні бази даних, що не вимогливі до ресурсів, компактні і надійні. Робота з БД SQLite виконується за допомогою команд мови SQL. SQLite читає і пише дані в звичайні файли на диску.

Особливістю цієї БД є те, що вона не потребує серверів. Доступ до БД відбувається через «підключення» до БД. Дозволено відкривати безліч підключень до однієї і теж БД в одному або різних додатках. Система використовує механізми блокування доступу до файлу на рівні ОС, щоб це все працювало[16]1).

До плюсів БД SQLite для зберігання файлів можна віднести наступне:

- SQLite дозволяє зберігати дані структурованим чином.
- SQLite має більш високу продуктивність.
- бази даних SQLite також можуть бути запитані, і пошук даних є набагато більш надійним.
- вміст можна переглянути за допомогою сторонніх інструментів.
- файл додатка переноситься у всіх операційних системах, 32-бітних і 64-бітних, а також у великих і малих версіях.
- додаток має завантажувати стільки даних, скільки потрібно, замість того, щоб читати весь файл програми та проводити повний аналіз в пам'яті. Скорочується час запуску і споживання пам'яті.
- кілька процесів можуть підключатися до одного файлу програми і можуть читати і писати, не заважаючи один одному.

За замовчуванням в Android Studio використовується SQLite – популярна і проста в освоєнні реляційна база даних. Тож для ПП було обрано саме цю БД.

[1)] [16] В чем преимущество использования SQLite, а не файла. URL: http://www.ohandroid.com/30896.html (дата звернення 12.03.2020).

3 ПРОЕКТНА ЧАСТИНА

Зберігання даних в SQLite

База даних складається з трьох таблиць:

– cities

– data_city

– coefficient

Таблиця cities має два стовпці:

– city_id, що є унікальним ідентифікатором

– city – зберігає назву міста

– Таблиця data_city зберігає обраховані дані для кожного міста.

Нижче представлені стовпці таблиці data_city:

– data_id – унікальний ідентифікатор

– data_city_id – є зовнішнім ключем і посилається на таблицю cities.

Значення в полі data_city_id відповідає первинному ключу в таблиці cities для того міста, до якого відносяться дані.

На рис. 3.1 представлена таблиця data_city в графічному виді із зазначенням типів даних які зберігаються в кожному стовпці.

Таблиця coefficient слугує лише для зберігання коефіцієнтів.

Нижче представлені стовпці таблиці coefficient:

– coef_id – унікальний ідентифікатор

– coef_pet – коефіцієнт утворення забруднюючих речовин для бензину

– coef_dies – коефіцієнт утворення забруднюючих речовин для дизеля

– coef_treat – коефіцієнт утворення забруднюючих речовин для очищених стічних вод

– coef_untreat – коефіцієнт утворення забруднюючих речовин для неочищених стічних вод

– coef_activ – коефіцієнт утворення забруднюючих речовин для АМ

Cid	Name	Type
0	data_id	INTEGER
1	data_city_id	INTEGER
2	year	INTEGER
3	petrol	REAL
4	diesel	REAL
5	treated_waste_water	REAL
6	untreated_waste_water	REAL
7	activated_sludge	REAL
8	petrol_dd	REAL
9	petrol_dph	REAL
10	diesel_dd	REAL
11	diesel_dph	REAL
12	treated_waste_water_dd	REAL
13	treated_waste_water_dph	REAL
14	untreated_waste_water_dd	REAL
15	untreated_waste_water_dph	REAL
16	activated_sludge_dd	REAL
17	activated_sludge_dph	REAL

Рисунок 3.1 – Таблиця data_city

Побудова UML діаграми діяльності

Діаграма діяльності є діаграмою поведінки UML, яка представляє робочий процес поетапної діяльності системи. Ці діаграми широко використовуються в описі поведінки, що включає велику кількість паралельних процесів. Кожний стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід в наступний стан виконується тільки після завершення цієї операції.

Така операція є фундаментальною одиницею визначення поведінки в специфікації. Операція, отримуючи один або декілька вхідних сигналів, перетворює їх на вихідні сигнали.

Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності також може дозволяти виконання декількох потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку[17][1)].

Діаграми діяльності побудовані з обмеженої кількості форм, пов'язаних стрілками. Найважливіші типи форм:

- округлені прямокутники позначають дії;
- ромби позначають рішення;
- риски позначають порядок дій;
- чорний кружок позначає старт (початковий стан) процесу;
- чорний кружок в колі позначає кінець (кінцевий стан).

Діаграми діяльності можна розглядати як форму структурованої блок-схеми у поєднанні з традиційною схемою потоку даних.

На рис. 3.2 зображено діаграма діяльності для ПП.

[1)][17] Діаграма діяльності – Вікіпедія.
URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності (дата звернення 13.03.2020).

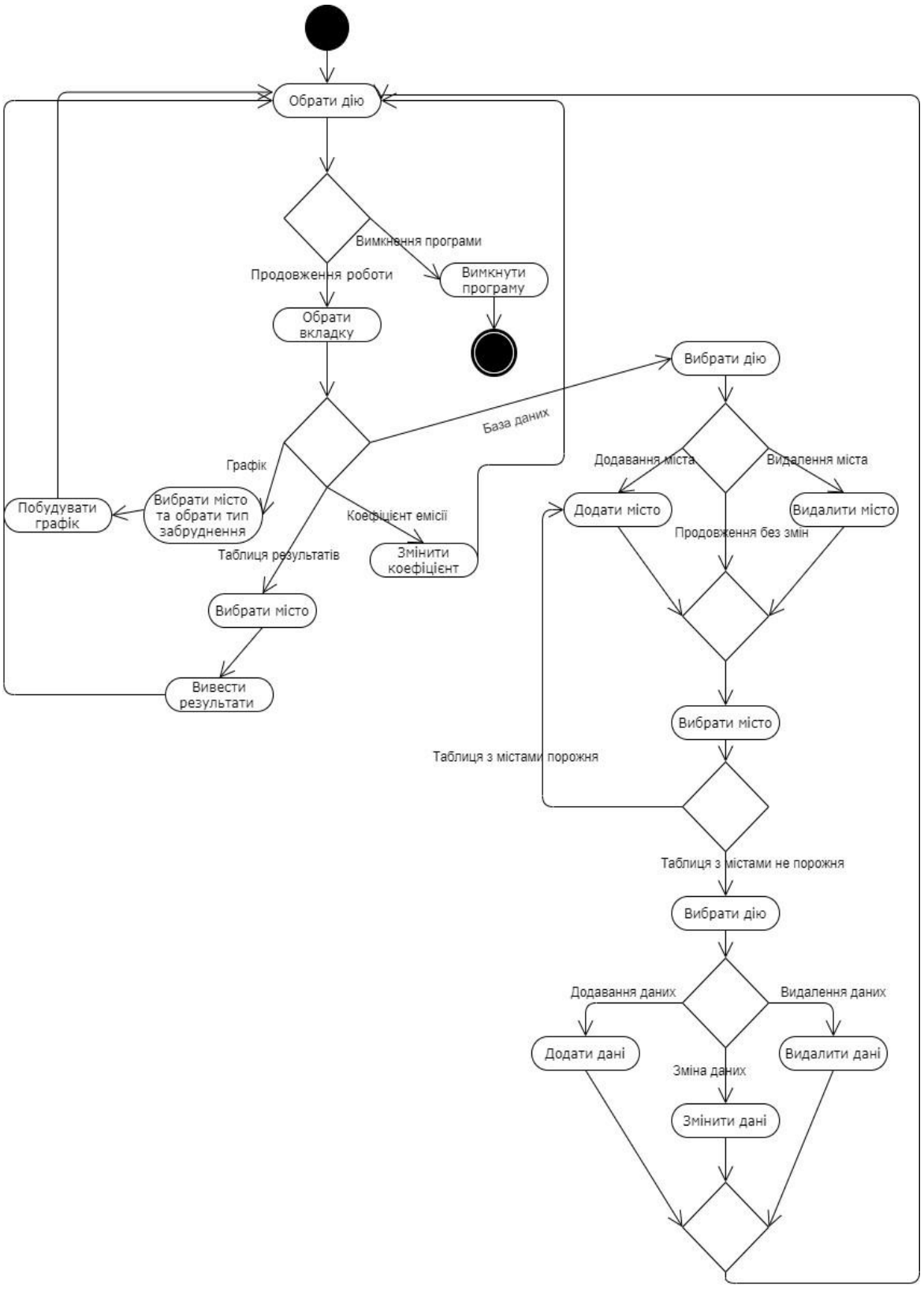

Рисунок 3.2 – UML діаграма діяльності

Діаграма, яка показана на рис. 3.2 відображає послідовність дій у мобільному додатку для допомоги у розрахунку піврозпаду СОЗ за різний період часу.

Після запуску програми користувач може продовжити роботу або зупинити роботу застосування. Якщо ж роботу все таки продовжено, то потрібно обрати одну з чотирьох вкладок, а саме:

− «База даних»;

− «Таблиця результатів»;

− «Графік»;

− «Коефіцієнти емісії».

При виборі «База даних», користувачу надається вибір додати нове місто, видалити існуюче місто або продовжити роботу з тими містами, які вже є в БД. Далі потрібно обрати місто зі списку і обрати дію:

− додати дані забруднень для обраного міста у БД;

− видалити дані забруднень для обраного міста з БД;

− змінити дані забруднень для обраного міста у БД.

При виборі вкладки «таблиця результатів» потрібно обрати місто, після чого застосування покаже результати в таблиці.

На вкладці «Графік» користувачу потрібно обрати місто та тип забруднення, щоб побудувати графік.

При виборі вкладки «Коефіцієнти емісії» є можливість змінити коефіцієнти емісії.

Проектування зовнішнього вигляду

Першим, що буде відображатися після запуску додатку це вкладка «База даних»(див.рис.3.3). На цій вкладці користувач буде мати змогу керувати даними, а саме додавати та видаляти міста та проводити такі дії для з інформацією:

– додавати дані до БД;

– видаляти дані с БД;

– або змінювати їх.

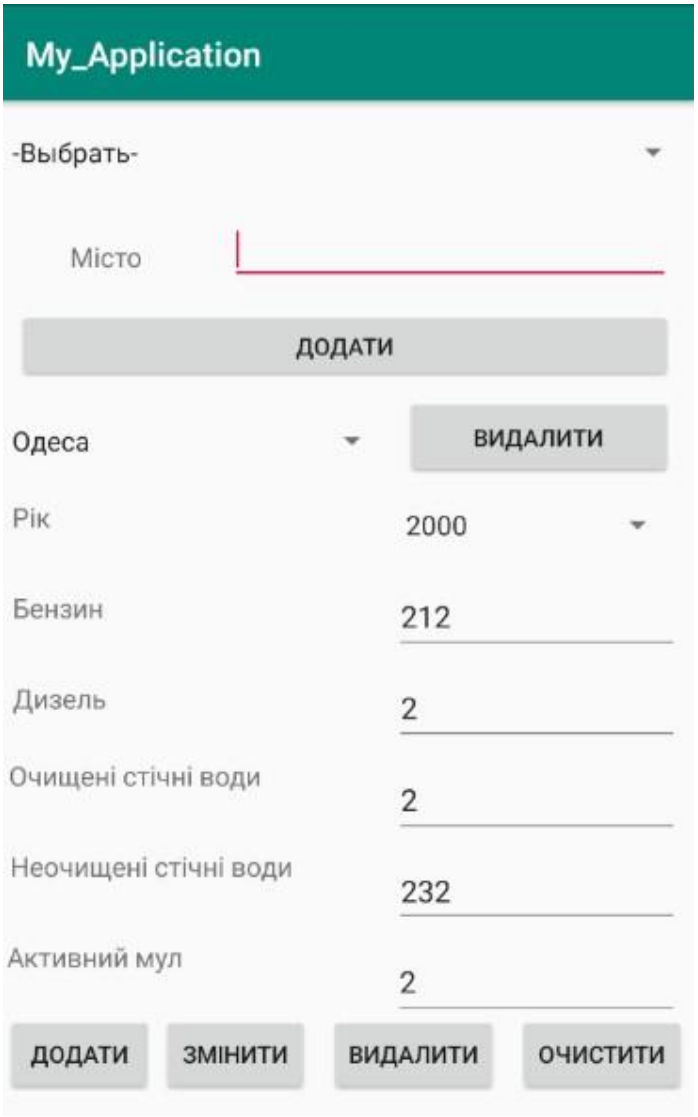

Рисунок 3.3 – Вкладка для роботи з БД

На вкладці «Таблиця результатів» (див.рис.3.4).буде відображена таблиця з результатами розрахунку та їх вхідними значеннями для обраного міста. Для того, щоб відобразити таблицю, користувачу потрібно буде обрати відповідні дані з двох випадаючих списків.

Рисунок 3.4 – Вкладка з таблицею результатів

За графічне відображення напіврозпаду ПХДД та ПХДФ відносно часу буде відповідати вкладка «Графік» (див.рис.3.5).

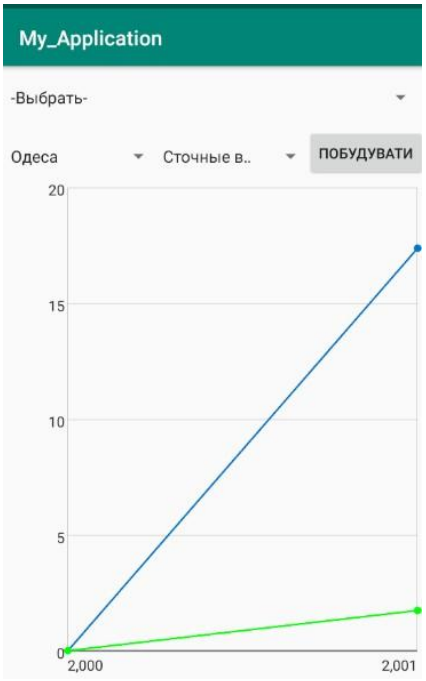

Рисунок 3.5 – Вкладка «Графік»

Значення коефіцієнтів встановлені за замовченням, але їх завжди можна змінити на вкладці «Коефіцієнти емісії» (див.рис.3.6).

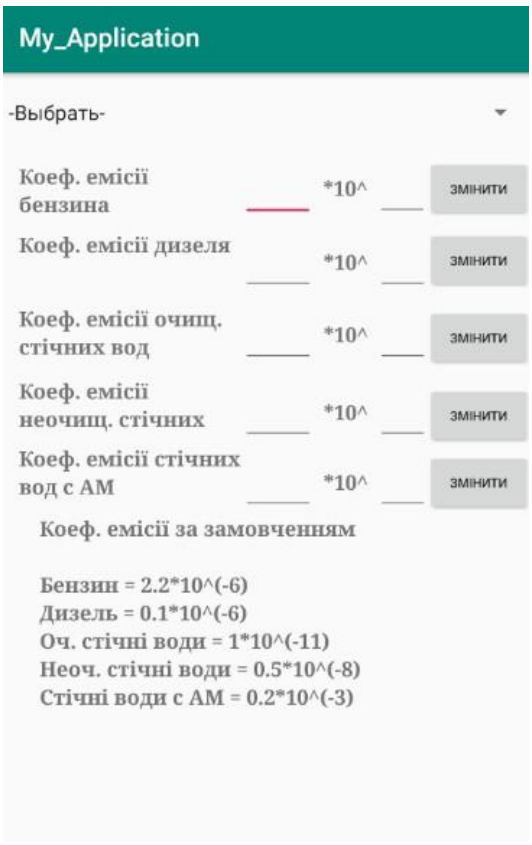

Рисунок 3.6 – Зовнішній вигляд вкладки «Коефіцієнти емісії»

Розробка окремих програмних елементів

Розробка інструментів для взаємодії с БД

Усі методи для взаємодії с базою даних, які нам будуть потрібні при розробці мобільного додатка присутні в бібліотеці android.database.sqlite.

Для роботи з БД ми використовували два класи:

– DBHelper, що успадковує SQLiteOpenHelper (допоміжний клас для управління створенням бази даних і управлінням версіями). Метод getWritableDatabase виконує підключення до бази даних і повертає нам об'єкт SQLiteDatabase для роботи з нею. Метод close закриває підключення до БД. У разі, коли БД відсутня або застаріла, клас надає

нам самим реалізувати створення або оновлення в методах onCreate і onUpgrate.

– SQLiteDatabase, що містить методи для роботи з даними – тобто вставка, оновлення, видалення та читання та виконання команд SQL[18][1)].

Для вирішення цих питань оголошено декілька змінних.
DBHelper dbHelper – це об'єкт класу DBHelper для управління БД.

SQLiteDatabase database – це об'єкт класу SQLiteDatabase для роботи с БД.

Наступним рядком виводимо таблиці за допомогою методу rawQuery, який приймає на вхід SQL-запит і список аргументів для умови WHERE (якщо необхідно) та передаємо результат query_table_data – об’єкту класу Cursor.

```
query_table_data=database.rawQuery("SELECT * FROM data_city WHERE
data_city_id= " + get_city(get_cursor()) + ";", null);
```

В SQL-запиті використовуємо наступні методи:

```
public Cursor get_cursor(){
    Cursor cursor;
    cursor = database.rawQuery("SELECT city_id FROM cities WHERE
city = '" + citySpinner.getSelectedItem().toString() + "';",
null);
    return cursor;
}
```

Цей метод читає дані з БД та повертає значення типу Cursor в наступний метод, що повертає ідентифікатор обраного міста с БД.

```
public int get_city(Cursor cursor){
    int city_id=0;
    if(cursor.moveToFirst()){
        do{
            city_id = cursor.getInt(0);
        } while (cursor.moveToNext());
    }
    return city_id;
}
```

[1)][18] Урок 34. Хранение данных. SQLite.
URL: https://startandroid.ru/ru/uroki/vse-uroki-spiskom/74-urok-34-hranenie-dannyh-sqlite.html (дата звернення 15.03.2020).

Виконання наступних команд додає місто до бази даних.

```
contentValues.put(DBHelper.NAME,
mEnterCity.getText().toString());
database.insert(DBHelper.TABLE_CITY,null,contentValues);
```

contentValues – це об’єкт класу ContentValues, використовується для вказівки полів таблиці і значень, які ми в ці поля будемо вставляти.

Далі реалізовано команду для видалення даних з БД шляхом виконання SQL-запиту.

```
database.execSQL("DELETE FROM cities WHERE city_id="
+get_city(get_cursor()));
```

Назва потрібного для видалення міста обирається з випадаючого списку.

Далі зображено реалізацію дій для взаємодії з даними, що доступні користувачу:

– додавати дані

– змінювати дані

– видаляти дані

Наступні команди виконуються при натисненні відповідних кнопок.

Команди додавання:

```
contentValues.put(DBHelper.CITY_ID, get_city(get_cursor()));
contentValues.put(DBHelper.YEAR,
Integer.parseInt(spinner_year.getSelectedItem().toString()));
contentValues.put(DBHelper.PETROL,
Double.valueOf(mPetrol.getText().toString()));
contentValues.put(DBHelper.PETROL_DD,petrol_dd );
contentValues.put(DBHelper.PETROL_DPH, petrol_dph);
contentValues.put(DBHelper.DIESEL,
Double.valueOf(mDiesel.getText().toString()));
contentValues.put(DBHelper.DIESEL_DD,diesel_dd );
contentValues.put(DBHelper.DIESEL_DPH, diesel_dph);
contentValues.put(DBHelper.TREATED_WASTEWATER,
Double.valueOf(mTreated.getText().toString()));
contentValues.put(DBHelper.TREATED_WASTEWATER_DD,treated_dd );
contentValues.put(DBHelper.TREATED_WASTEWATER_DPH, treated_dph);
```

```
contentValues.put(DBHelper.UNTREATED_WASTEWATER,
Double.valueOf(mUntreated.getText().toString()));
contentValues.put(DBHelper.UNTREATED_WASTEWATER_DD,untreated_dd
);
contentValues.put(DBHelper.UNTREATED_WASTEWATER_DPH,
untreated_dph);
contentValues.put(DBHelper.ACTIVATED_SLUDGE,
Double.valueOf(mActivated.getText().toString()));
contentValues.put(DBHelper.ACTIVATED_SLUDGE_DD,activated_dd );
contentValues.put(DBHelper.ACTIVATED_SLUDGE_DPH, activated_dph);
database.insert(DBHelper.TABLE_DATA,null,contentValues);
```

Команди для зміни:

```
database.execSQL("UPDATE data_city SET petrol
="+Double.valueOf(mPetrol.getText().toString())
        + ",
diesel="+Double.valueOf(mDiesel.getText().toString())
        +",
treated_waste_water="+Double.valueOf(mTreated.getText().toString
())
        +",
untreated_waste_water="+Double.valueOf(mUntreated.getText().toSt
ring())
        +",
activated_sludge="+Double.valueOf(mActivated.getText().toString(
))
        +", petrol_dd="+petrol_dd
        +", petrol_dph="+petrol_dph
        +", diesel_dd="+diesel_dd
        +", diesel_dph="+diesel_dph
        +", treated_waste_water_dd="+treated_dd
        +", treated_waste_water_dph="+treated_dph
        +", untreated_waste_water_dd="+untreated_dd
        +", untreated_waste_water_dph="+untreated_dph
        +", activated_sludge_dd="+activated_dd
        +", activated_sludge_dph="+activated_dph
        +" WHERE data_city_id=" +get_city(get_cursor())+" AND
year="+
Integer.parseInt(spinner_year.getSelectedItem().toString()));
```

Команди для видалення:

```
database.execSQL("DELETE FROM data_city WHERE data_city_id="
+get_city(get_cursor())+" AND year="+
Integer.parseInt(spinner_year.getSelectedItem().toString()));
```

Розробка інструментів розрахунку по вхідним даним

Для розрахунку будемо використовувати методи класу Math. Коефіцієнти, надані кафедрою екології та охорони довкілля Одеського державного екологічного університету, знаходяться в таблиці coefficient нашої БД. За замовчуванням вони мають такі значення (див. рис. 3.7).

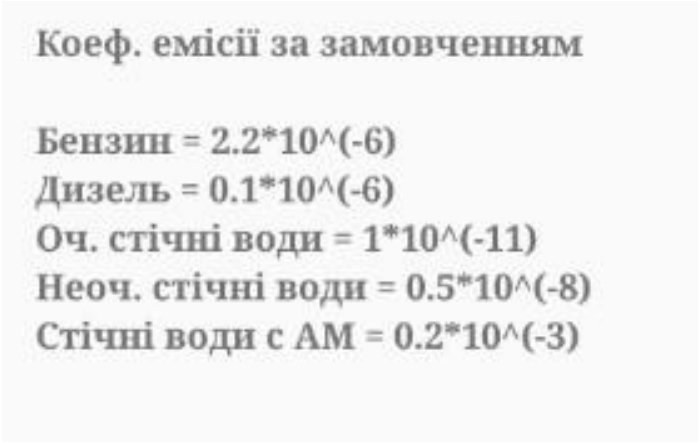

Рисунок 3.7 – коефіцієнти за замовченням

Шляхом виконання наступних команд виконується розрахунок ПХДД.

```
if (cursor.moveToFirst()) {
    do {

petrol_dd=cursor.getDouble(1)*Double.parseDouble(mPetrol.getText
().toString());

diesel_dd=cursor.getDouble(2)*Double.parseDouble(mDiesel.getText
().toString());

treated_dd=cursor.getDouble(3)*Double.parseDouble(mTreated.getTe
xt().toString());

untreated_dd=cursor.getDouble(4)*Double.parseDouble(mUntreated.g
etText().toString());

activated_dd=cursor.getDouble(5)*Double.parseDouble(mActivated.g
etText().toString());
    }
    while (cursor.moveToNext());
}
```

Значення ПХДФ отримуємо після перемноження значень ПХДД на змінну dph, яка дорівнює 0.1.

```
petrol_dph=petrol_dd*dph;
diesel_dph=diesel_dd*dph;
treated_dph=treated_dd*dph;
untreated_dph=untreated_dd*dph;
activated_dph=activated_dd*dph;
```

У разі потреби, значення можна змінити за допомогою наступного методу.

```
public void onClick(View view){
    switch (view.getId()) {
        case R.id.ref_base_button_petrol:
            database.execSQL("UPDATE coefficient SET coef_pet
="+(Double.parseDouble(Petrol.getText().toString())*Math.pow(10,
Double.parseDouble(Petrol1.getText().toString()))));
            break;
        case R.id.ref_base_button_diesel:
            database.execSQL("UPDATE coefficient SET coef_dies
="+(Double.parseDouble(Diesel.getText().toString())*Math.pow(10,
Double.parseDouble(Diesel1.getText().toString()))));
            break;
        case R.id.ref_base_button_treat:
            database.execSQL("UPDATE coefficient SET coef_treat
="+(Double.parseDouble(Treated.getText().toString())*Math.pow(10
,Double.parseDouble(Treated1.getText().toString()))));
            break;
        case R.id.ref_base_button_untreat:
            database.execSQL("UPDATE coefficient SET
coef_untreat
="+(Double.parseDouble(Untreated.getText().toString())*Math.pow(
10,Double.parseDouble(Untreated1.getText().toString()))));
            break;
        case R.id.ref_base_button_activ:
            database.execSQL("UPDATE coefficient SET coef_activ
="+(Double.parseDouble(Activated.getText().toString())*Math.pow(
10,Double.parseDouble(Activated1.getText().toString()))));
            break;
}
  }
```

Цей метод змінює значення коефіцієнту в БД на вказані користувачем у полі введення для типу забруднення відповідно до нажатої кнопки.

Для розрахунку напіврозпаду використовуються 5 масивів, які заповнюються даними с БД значеннями ПХДД і ПХДФ.

```
             double[] mas_pet = new
double[query_table_data.getCount()];
        double[] mas_dies = new
double[query_table_data.getCount()];
        double[] mas_treat = new
double[query_table_data.getCount()];
        double[] mas_untreat = new
double[query_table_data.getCount()];
        double[] mas_activ = new
double[query_table_data.getCount()];
        if(query_table_data.moveToFirst()){
            int i=0;
            do{
                mas_pet[i] = (query_table_data.getDouble(8));
                mas_dies[i]
=Double.parseDouble(query_table_data.getString(10));
                mas_treat[i] =
Double.parseDouble(query_table_data.getString(12));
                mas_untreat[i] =
Double.parseDouble(query_table_data.getString(14));
                mas_activ[i] =
Double.parseDouble(query_table_data.getString(16));
                i++;
            }
            while (query_table_data.moveToNext());
        }
```

Наступний метод виконує розрахунок напіврозпаду ПХДД та ПХДФ для різних типів забруднення.

```
     public double[] result(double[] mas, Cursor
query_table_data){
        for(int i=0; i<query_table_data.getCount(); i++) {
            double z=0;
            for(int y=0; y<=i;y++){
                z+=mas[i-y]*Math.exp((-1)*y/14.5);
            }
            mas[i]=z;
        }
        return mas;
    }
```

Наступні команди заповнюють масиви.

```
     result(mas_pet,query_table_data);
        result(mas_dies,query_table_data);
        result(mas_treat,query_table_data);
        result(mas_untreat,query_table_data);
        result(mas_activ,query_table_data);
        double[][] array = new
double[5][query_table_data.getCount()];
            for (int j = 0; j < query_table_data.getCount();
j++) {
```

```
                array[0][j]=mas_pet[j];
                array[1][j]=mas_dies[j];
                array[2][j]=mas_treat[j];
                array[3][j]=mas_untreat[j];
                array[4][j]=mas_activ[j];
            }
```

Розробка елементів відображення

Для побудови графіку використовується спеціально призначена для цього бібліотека com.jjoe64.graphview.GraphView.

Розрахунок точок графіка:

```
if(query.moveToFirst()){
                    int i=0;
                    do{
                        DP[i] = new
DataPoint(Integer.parseInt(query.getString(2)),
Double.parseDouble(query.getString(col_dd)));
                        DP1[i] = new
DataPoint(Integer.parseInt(query.getString(2)),
Double.parseDouble(query.getString(col_dph)));
                        i++;
                    }
                    while (query.moveToNext());
                }
```

Реалізація побудови графіка:

```
                series = new LineGraphSeries<>(DP);
                series1 = new LineGraphSeries<>(DP1);
                GraphView graph = (GraphView)
findViewById(R.id.graph1);
                graph.setVisibility(View.VISIBLE);
                graph.removeAllSeries();
                series.setDrawDataPoints(true);
                series.setDataPointsRadius(10);
                series.setAnimated(true);
        series1.setColor(GREEN);
        series1.setDrawDataPoints(true);
        series1.setDataPointsRadius(10);
        series1.setAnimated(true);
graph.getGridLabelRenderer().setNumHorizontalLabels(query.getCou
nt());                  graph.addSeries(series);
        graph.addSeries(series1);
```

4 ОПИС РОБОТИ З ПРОГРАМОЮ

Установка мобільного додатка

Установка програми виконується шляхом скачування APK – файлу, який необхідно завантажити в пам'ять пристрою та, використовуючи файловий менеджер, відкрити цей файл для установки.

Функціонал програми

Після запуску програми відкривається вкладка «База даних». Дане вікно слугує для роботи с БД. В ньому користувач може додати нове місто, ввівши назву в поле «Місто» та натиснувши кнопку «Додати» (див. рис. 4.1).

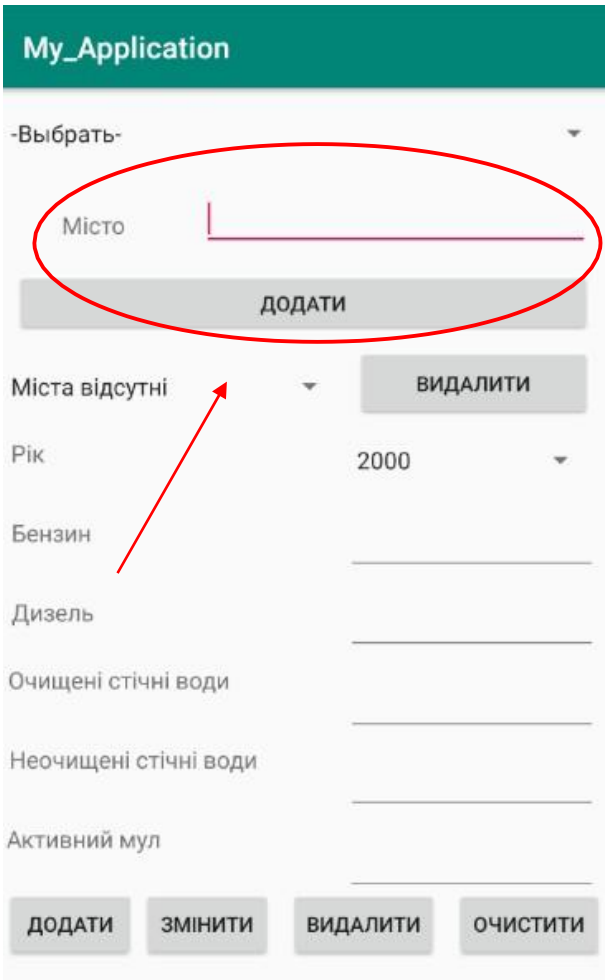

Рисунок 4.1 – Додати нове місто до БД

Також користувач може обрати місто вже з існуючих в БД або видалити його (див. рис. 4.2). Щоб це зробити потрібно вибрати місто з випадаючого списку і, при потребі, видалити, натиснувши кнопку «Видалити».

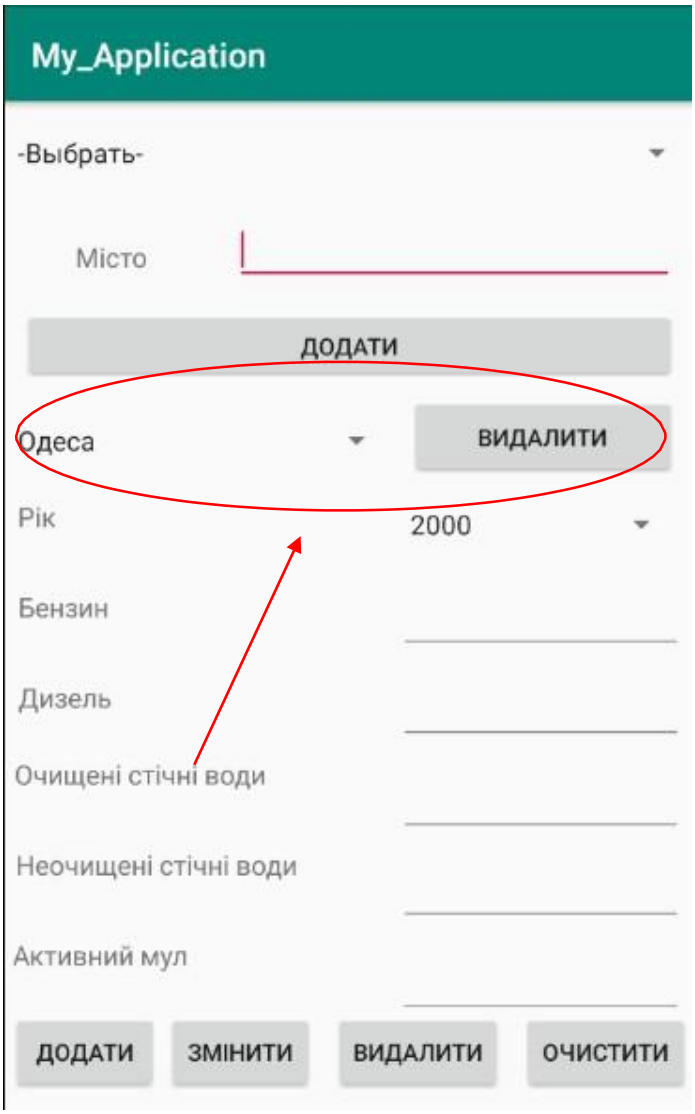

Рисунок 4.2 – Можливість обрати місто та видалити

Після того як користувач обрав місто, він може додати дані, змінити їх або зовсім видалити (див. рис. 4.3). Після натиснення кнопки «Додати» виконується розрахунок ПХДД та ПХДФ. Також, якщо в БД присутні дані для обраного міста с певним роком, то при виборі вносяться дані с бази даних у відповідні поля.

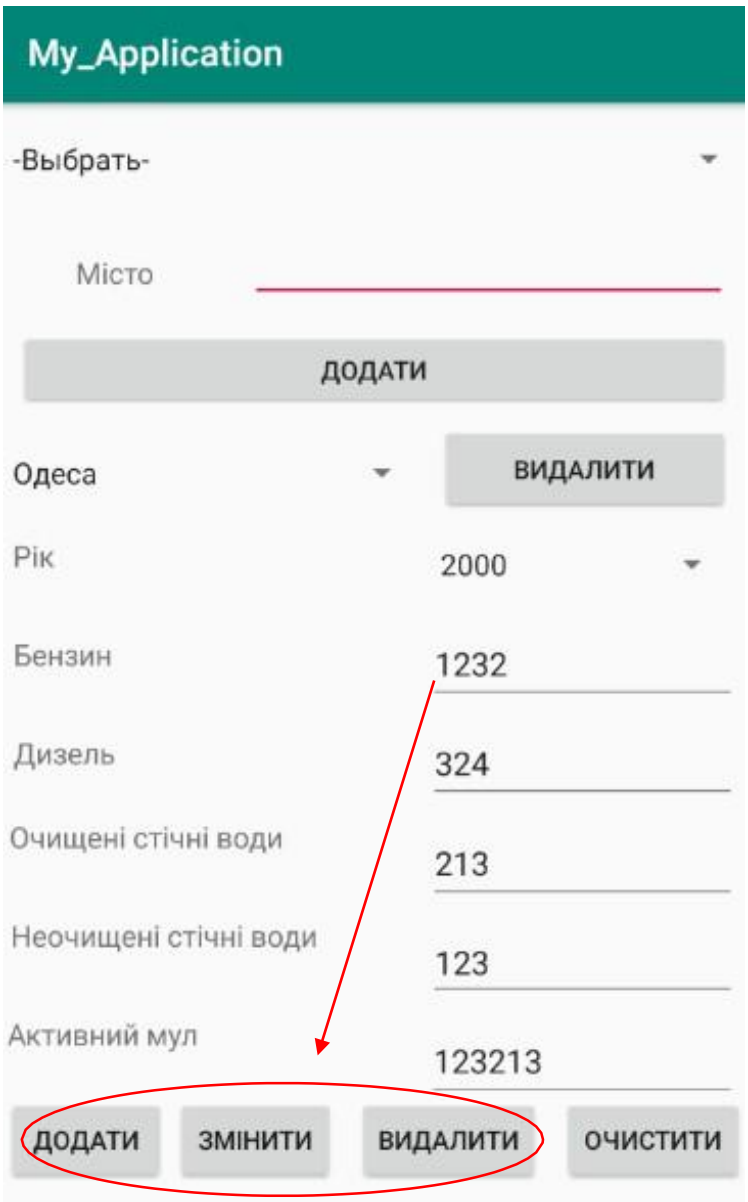

Рисунок 4.3 – Операції з даними

Після заповнення всіх полів необхідними даними, користувач може перейти до вкладки «Таблиця результатів» (див. рис. 4.4).

В цьому вікні йому необхідно вибрати потрібне місто та натиснути на кнопку «Обрати», після чого виконуються розрахунки напіврозпаду ПХДД і ПХДФ для різних типів забруднення.

У результаті, на екрані з’явиться таблиця з усіма введеними та обрахованими даними.

Рисунок 4.4 – Таблиця даних

Вкладка «Графік» візуально у вигляді графіка представляє піврозпад ПХДД та ПХДФ різних типів забруднень для кожного року, що є в БД. Для того, щоб побачити графік користувачу потрібно обрати місто і тип забруднення, а потім натиснути кнопку «Побудувати» (див. рис. 4.5).

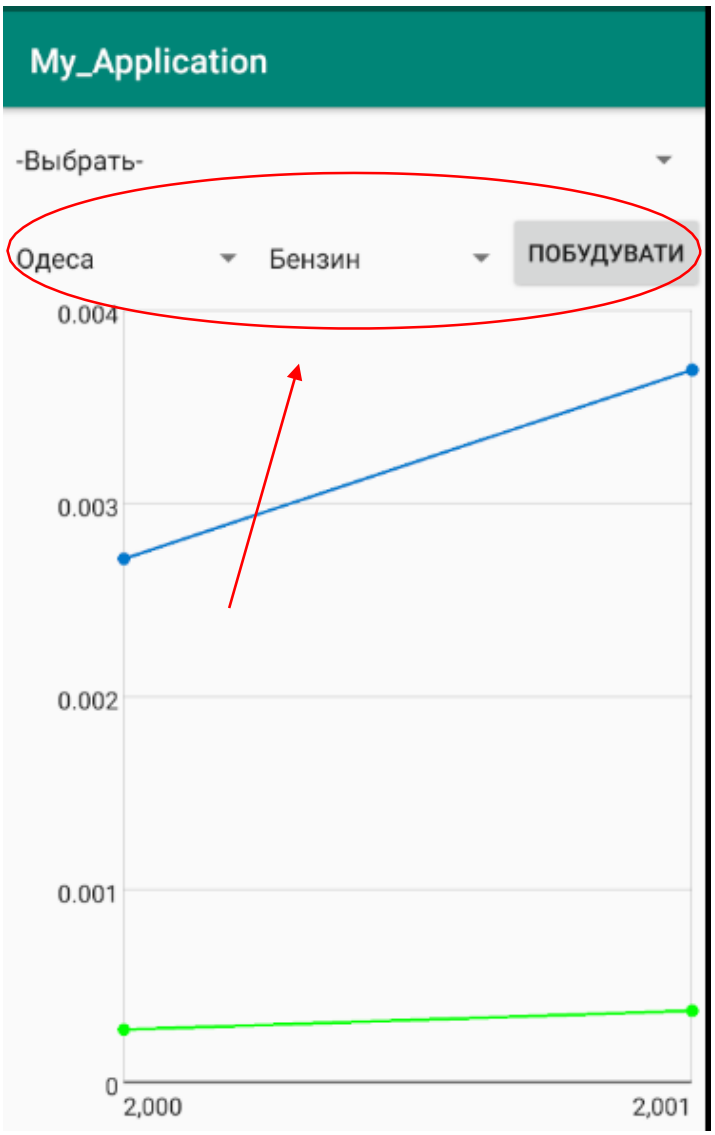

Рисунок 4.5 – Побудова графіків

Коефіцієнти, по яким виконуються розрахунки знаходяться в окремій таблиці БД. Їх значення за замовченням були надані кафедрою екології та охорони довкілля Одеського державного екологічного університету.

При потребі їх можна змінити на вкладці «Коефіцієнти емісії» (див. рис. 4.6). Після вводу нового значення та натиснення кнопки «Змінити», коефіцієнт змінюється в базі даних.

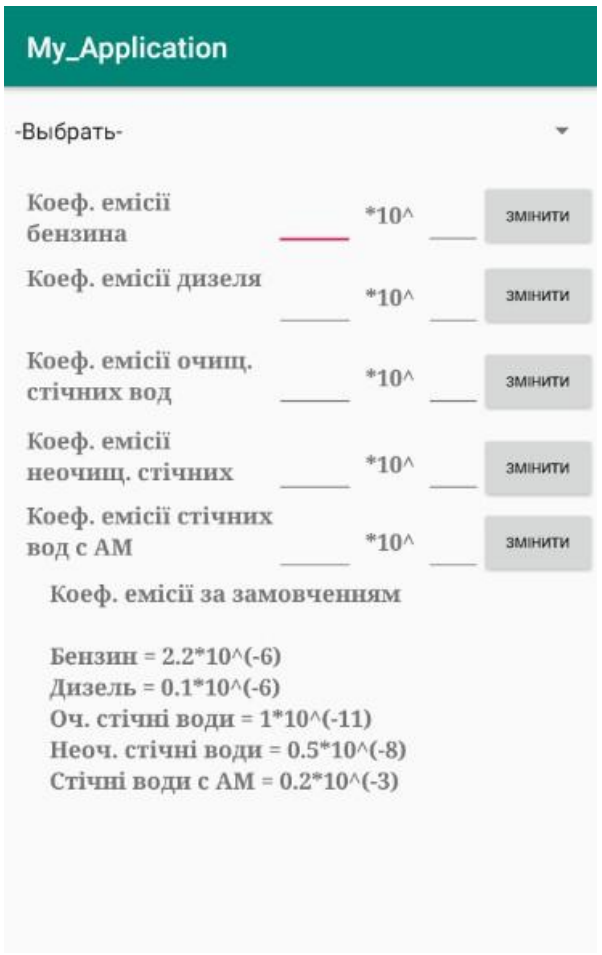

Рисунок 4.6 – Коефіцієнти емісії

Тестування ПП

В процесі розробки мобільного додатку вироблялося поетапне тестування з метою виявлення програмних помилок. Для цього були створені емулятори смартфона різних версій Android.

Додаток було запущено на пристроях, що працюють під управлінням різних версій Android з метою виявлення особливостей роботи додатку.

Після завершення циклу розробки, ПП тестувався на реальних пристроях.

ВИСНОВКИ

У результаті виконання бакалаврської роботи було проведено аналіз предметної галузі та подібних ПП.

Було обрано IDE Android Studio та мова програмування Java.

Проведено моделювання бази даних.

Було розроблено програмний продукт, який реалізує наступні функції:

– проводить облік ПХДД та ПХДФ;

– виводить дані у таблицю;

– креслить графік.

У подальшому є можливості до розвинення проекту, а також покращення дизайну.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Федоров Л.А. Диоксины как экологическая опасность: ретроспектива и перспективы. М. : Наука. 1993. 266с. ISBN 5-02-001674-8.
2. Стокгольмская конвенция о стойких органических загрязнителях – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Стокгольмская_конвенция_о_стойких_органических_загрязнителях (дата звернення 15.02.2020).
3. Приложения в Google Play – EWG's Healthy Living. URL: https://play.google.com/store/apps/details?id=com.skindeep.mobile&hl=uk (дата звернення 17.02.2020).
4. Приложения в Google Play – Earth-Now. URL: https://play.google.com/store/apps/details?id=gov.nasa.jpl.earthnow.activity&hl=ru (дата звернення 19.02.2020).
5. Нємцев В.Є. Створення програми для часової інвентаризації накопичення стійких органічних забруднюючих речовин у довкіллі. – URL: http://eprints.library.odeku.edu.ua/5940/1/Niemtsev_Rozrobka_inventarizacii_B_2019.pdf (дата звернення 22.02.2020).
6. Приложения в Google Play – Loss of the night. URL: https://play.google.com/store/apps/details?id=com.cosalux.welovestars&hl= ru (дата звернення 24.02.2020).
7. Android Studio: среда разработки мобильных приложений. URL: https://arduinoplus.ru/android-studio/ (дата звернення 03.03.2020).
8. Android Studio IDE от Google. URL: http://wnfx.ru/android-studio-ide-ot-google/ (дата звернення 03.03.2020).
9. StarDroid - уроки по Android. URL: https://venomwind.wixsite.com/stardroid/blank-thepk (дата звернення 03.03.2020).
10. Подробно о Xamarin. URL: https://habr.com/ru/post/188130/ (дата звернення 03.03.2020).

11. Java – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Java (дата звернення 06.03.2020).
12. Java – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Java#Основные_особенности_языка (дата звернення 06.03.2020).
13. Kotlin – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Kotlin (дата звернення 08.03.2020).
14. MySQL – Вікіпедія. URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення 11.03.2020).
15. Что Такое MySQL: Объяснение MySQL Для Начинающих. URL: https://www.hostinger.ru/rukovodstva/shto-takoje-mysql/ (дата звернення 11.03.2020).
16. В чем преимущество использования SQLite, а не файла. URL: http://www.ohandroid.com/30896.html (дата звернення 12.03.2020).
17. Діаграма діяльності – Вікіпедія. URL:https://uk.wikipedia.org/wiki/Діаграма_діяльності (дата звернення 13.03.2020).
18. Урок 34. Хранение данных. SQLite. URL: https://startandroid.ru/ru/uroki/vse-uroki-spiskom/74-urok-34-hranenie-dannyh-sqlite.html (дата звернення 15.03.2020).

ДОДАТОК А

Лістинг класів

```
package com.example.my_application;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import androidx.annotation.Nullable;
public class DBHelper extends SQLiteOpenHelper{
private  static final int DATABASE_VERSION=1;
    public static final String DATABASE_NAME="cityDB";
    static final String TABLE_CITY="cities";
    public static final String TABLE_DATA="data_city";
    public static final String TABLE_COEFFICIENT="coefficient";
    public static final String NAME = "city";
    public static final String NAME_ID = "city_id";
    public static final String DATA_ID="data_id";
    public static final String CITY_ID="data_city_id";
    public static final String YEAR="year";
    public static final String PETROL="petrol";
    public static final String PETROL_DD="petrol_dd";
    public static final String PETROL_DPH="petrol_dph";
    public static final String DIESEL="diesel";
    public static final String DIESEL_DD="diesel_dd";
    public static final String DIESEL_DPH="diesel_dph";
    public static final String
TREATED_WASTEWATER="treated_waste_water";
    public static final String
TREATED_WASTEWATER_DD="treated_waste_water_dd";
    public static final String
TREATED_WASTEWATER_DPH="treated_waste_water_dph";
    public static final String
UNTREATED_WASTEWATER="untreated_waste_water";
    public static final String
UNTREATED_WASTEWATER_DD="untreated_waste_water_dd";
    public static final String
UNTREATED_WASTEWATER_DPH="untreated_waste_water_dph";
    public static final String
ACTIVATED_SLUDGE="activated_sludge";
    public static final String
ACTIVATED_SLUDGE_DD="activated_sludge_dd";
    public static final String
ACTIVATED_SLUDGE_DPH="activated_sludge_dph";
    public static final String COEF_ID = "coef_id";
    public static final String COEF_PET = "coef_pet";
    public static final String COEF_DIES = "coef_dies";
    public static final String COEF_TREAT = "coef_treat";
    public static final String COEF_UNTREAT = "coef_untreat";
    public static final String COEF_ACTIV = "coef_activ";
    public static final String CREATE_TABLE_CITY = "CREATE TABLE
IF NOT EXISTS " + TABLE_CITY
            + " ( " +NAME_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT UNIQUE, " + NAME
            + " TEXT NOT NULL )";
    public static final String CREATE_TABLE_DATA = "CREATE TABLE
IF NOT EXISTS " +
```

```
            TABLE_DATA + " (" +
            DATA_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            CITY_ID + " INTEGER, " +
            YEAR + " INTEGER, " +
            PETROL + " REAL, " +
            DIESEL + " REAL, " +
            TREATED_WASTEWATER + " REAL, " +
            UNTREATED_WASTEWATER + " REAL, " +
            ACTIVATED_SLUDGE + " REAL, " +
            PETROL_DD + " REAL, " +
            PETROL_DPH + " REAL, " +
            DIESEL_DD + " REAL, " +
            DIESEL_DPH + " REAL, " +
            TREATED_WASTEWATER_DD + " REAL, " +
            TREATED_WASTEWATER_DPH + " REAL, " +
            UNTREATED_WASTEWATER_DD + " REAL, " +
            UNTREATED_WASTEWATER_DPH + " REAL, " +
            ACTIVATED_SLUDGE_DD + " REAL, " +
            ACTIVATED_SLUDGE_DPH + " REAL, " +
            "FOREIGN KEY(" + CITY_ID + ") REFERENCES " +
            TABLE_CITY + " (" + NAME_ID + ") " + ")";
    public static final String CREATE_TABLE_COEFFICIENT =
"CREATE TABLE IF NOT EXISTS "
            + TABLE_COEFFICIENT
            + " ( " +COEF_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT, "
            + COEF_PET + " REAL,"
            + COEF_DIES + " REAL,"
            + COEF_TREAT + " REAL,"
            + COEF_UNTREAT + " REAL,"
            + COEF_ACTIV + " REAL"
            +" )";
    public DBHelper(@Nullable Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_TABLE_CITY);
    db.execSQL(CREATE_TABLE_DATA);
        db.execSQL(CREATE_TABLE_COEFFICIENT);
        db.execSQL("INSERT INTO coefficient
(coef_pet,coef_dies,coef_treat,coef_untreat,coef_activ) VALUES
(" + (2.2*Math.pow(10,-6)) + "," +(0.1*Math.pow(10,-6))+ "," +
(1*Math.pow(10,-11)) + ","+(0.5*Math.pow(10,-8)) + "," +
(0.2*Math.pow(10,-3)) +")");
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
    }
}


package com.example.my_application;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
```

```
    static String[] data = {"-Выбрать-", "База даних", "Таблиця
результатів", "Графік", "Коефіцієнти емісії"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Intent intent = new
Intent(MainActivity.this,BaseActivity.class);
                startActivity(intent);
    }
}
```

```
package com.example.my_application;
import androidx.appcompat.app.AppCompatActivity;
import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import java.util.Calendar;
import static com.example.my_application.MainActivity.data;
public class BaseActivity extends AppCompatActivity {
    public void restart(){
        Intent intent = getIntent();
            finish();
        startActivity(intent);
    }
    public int get_city(Cursor cursor){
        int city_id=0;
        if(cursor.moveToFirst()){
            do{
                city_id = cursor.getInt(0);
            } while (cursor.moveToNext());
        }
        return city_id;
    }
    public Cursor get_cursor(){
        Cursor cursor;
        cursor = database.rawQuery("SELECT city_id FROM cities
WHERE city = '" + citySpinner.getSelectedItem().toString() +
"';", null);
        return cursor;
    }
    public Cursor get_cursor_coef(){
        Cursor cursor;
        cursor  = database.rawQuery("SELECT * FROM
coefficient;", null);
        return cursor;
    }
    public void Select(){
        query_spin1 = database.rawQuery("SELECT * FROM data_city
WHERE data_city_id=" + get_city(get_cursor()) + " AND year=" +
Integer.parseInt(spinner_year.getSelectedItem().toString()) +
";", null);
        if(query_spin1.getCount()!=0) {
            if (query_spin1.moveToFirst()) {
```

```
                do {
                    mPetrol.setText(query_spin1.getString(3));
                    mDiesel.setText(query_spin1.getString(4));
                    mTreated.setText(query_spin1.getString(5));

mUntreated.setText(query_spin1.getString(6));

mActivated.setText(query_spin1.getString(7));
                } while (query_spin1.moveToNext());
            }
        }
        else {
            mPetrol.setText(null);
            mDiesel.setText(null);
            mTreated.setText(null);
            mUntreated.setText(null);
            mActivated.setText(null);
        }
    }
private EditText mEnterCity;
    private EditText mPetrol;
    private EditText mDiesel;
    private EditText mTreated;
    private EditText mUntreated;
    private EditText mActivated;
DBHelper dbHelper;
    ArrayAdapter<String> adapter;
    ArrayAdapter<String> adapter_city;
    ArrayAdapter<String> adapter_year;
    SQLiteDatabase database;
    Spinner spinner;
    Spinner citySpinner;
    Spinner spinner_year;
    static String[] city_mass;
    static String[] date_mas;
    Cursor query1;
    Cursor query_spin1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_base);
        mEnterCity =  findViewById(R.id.enter_city);
        mPetrol =  findViewById(R.id.petrol_edit);
        mDiesel =  findViewById(R.id.diesel_edit);
        mTreated =  findViewById(R.id.treated_wastewater_edit);
        mUntreated =
findViewById(R.id.untreated_waste_water_edit);
        mActivated =  findViewById(R.id.activated_sludge_edit);
        dbHelper= new DBHelper(this);
        date_mas = new
String[Calendar.getInstance().get(Calendar.YEAR)-1999];
int y=2000;
for(int i=0; i<date_mas.length;i++){
date_mas[i]=Integer.toString(y++);
}
        adapter_year = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, date_mas);

adapter_year.setDropDownViewResource(android.R.layout.simple_spi
nner_dropdown_item);
```

```
         spinner_year = findViewById(R.id.spinner_year);
        spinner_year.setAdapter(adapter_year);
        spinner_year.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
               // query_spin=get_cursor();
                if(get_cursor().getCount()!=0) {
                Select();
                }
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent)
{
            }
        });
        adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, data);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
         spinner = findViewById(R.id.spinner);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
                if(position==1){
                    Intent intent = new
Intent(BaseActivity.this,BaseActivity.class);
                    startActivity(intent);
                }
                if(position==2){
                    Intent intent = new
Intent(BaseActivity.this,TableOfResults.class);
                    startActivity(intent);
                }
                if(position==3){
                    Intent intent = new
Intent(BaseActivity.this,ScheduleActivity.class);
                    startActivity(intent);
                }
                if(position==4){
                    Intent intent = new
Intent(BaseActivity.this,EmissionFactorActivity.class);
                    startActivity(intent);
                }
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent)
{
            }
        });
        database = dbHelper.getWritableDatabase();
         query1  = database.rawQuery("SELECT * FROM cities;",
null);
if(query1.getCount()==0) {
    city_mass = new String[1];
```

```
    city_mass[0]= "Міста відсутні";
}
else  if(query1.getCount()!=0){
    city_mass = new String[query1.getCount()];
    if (query1.moveToFirst()) {
        int i = 0;
        do {
            city_mass[i++] = (query1.getString(1));
        }
        while (query1.moveToNext());
    }
}
adapter_city = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, city_mass);

adapter_city.setDropDownViewResource(android.R.layout.simple_spi
nner_dropdown_item);
    citySpinner = findViewById(R.id.spinner_city);
    citySpinner.setAdapter(adapter_city);
        citySpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
                if(get_cursor().getCount()!=0) {
                   Select();
                }
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent)
{
            }
        });
    query1.close();
    }
    public void onClick(View view){
        double petrol_dd = 0, petrol_dph, diesel_dd=0,
diesel_dph, treated_dd=0, treated_dph, untreated_dd=0,
untreated_dph, activated_dd=0, activated_dph, dph=0.1;
        database = dbHelper.getWritableDatabase();
        Cursor cursor;
        cursor=get_cursor_coef();

        ContentValues contentValues = new ContentValues();
        switch (view.getId()){
            case R.id.ad_city_button:
                contentValues.put(DBHelper.NAME,
mEnterCity.getText().toString());

database.insert(DBHelper.TABLE_CITY,null,contentValues);
                restart();
                break;
            case R.id.del_city_button:
                database.execSQL("DELETE FROM cities WHERE
city_id=" +get_city(get_cursor()));
                restart();
              break;
            case R.id.add_base_button:
                if (cursor.moveToFirst()) {
                    do {
```

```
petrol_dd=cursor.getDouble(1)*Double.parseDouble(mPetrol.getText
().toString());
diesel_dd=cursor.getDouble(2)*Double.parseDouble(mDiesel.getText
().toString());
treated_dd=cursor.getDouble(3)*Double.parseDouble(mTreated.getTe
xt().toString());
untreated_dd=cursor.getDouble(4)*Double.parseDouble(mUntreated.g
etText().toString());
activated_dd=cursor.getDouble(5)*Double.parseDouble(mActivated.g
etText().toString());
                    }
                    while (cursor.moveToNext());
                }
                petrol_dph=petrol_dd*dph;
                diesel_dph=diesel_dd*dph;
                treated_dph=treated_dd*dph;
                untreated_dph=untreated_dd*dph;
                activated_dph=activated_dd*dph;
                    contentValues.put(DBHelper.CITY_ID,
get_city(get_cursor()));
                contentValues.put(DBHelper.YEAR,
Integer.parseInt(spinner_year.getSelectedItem().toString()));
                contentValues.put(DBHelper.PETROL,
Double.valueOf(mPetrol.getText().toString()));
                contentValues.put(DBHelper.PETROL_DD,petrol_dd
);
                contentValues.put(DBHelper.PETROL_DPH,
petrol_dph);
                contentValues.put(DBHelper.DIESEL,
Double.valueOf(mDiesel.getText().toString()));
                contentValues.put(DBHelper.DIESEL_DD,diesel_dd
);
                contentValues.put(DBHelper.DIESEL_DPH,
diesel_dph);
                contentValues.put(DBHelper.TREATED_WASTEWATER,
Double.valueOf(mTreated.getText().toString()));

contentValues.put(DBHelper.TREATED_WASTEWATER_DD,treated_dd );

contentValues.put(DBHelper.TREATED_WASTEWATER_DPH, treated_dph);
                contentValues.put(DBHelper.UNTREATED_WASTEWATER,
Double.valueOf(mUntreated.getText().toString()));
contentValues.put(DBHelper.UNTREATED_WASTEWATER_DD,untreated_dd
);
contentValues.put(DBHelper.UNTREATED_WASTEWATER_DPH,
untreated_dph);
                contentValues.put(DBHelper.ACTIVATED_SLUDGE,
Double.valueOf(mActivated.getText().toString()));
contentValues.put(DBHelper.ACTIVATED_SLUDGE_DD,activated_dd );
                contentValues.put(DBHelper.ACTIVATED_SLUDGE_DPH,
activated_dph);
database.insert(DBHelper.TABLE_DATA,null,contentValues);
                restart();
                break;
                case R.id.ref_base_button:
                    cursor = database.rawQuery("SELECT * FROM
coefficient;", null);
                    if (cursor.moveToFirst()) {
                        do {
petrol_dd=cursor.getDouble(1)*Double.parseDouble(mPetrol.getText
```

```
().toString());
diesel_dd=cursor.getDouble(2)*Double.parseDouble(mDiesel.getText
().toString());
treated_dd=cursor.getDouble(3)*Double.parseDouble(mTreated.getTe
xt().toString());
untreated_dd=cursor.getDouble(4)*Double.parseDouble(mUntreated.g
etText().toString());
activated_dd=cursor.getDouble(5)*Double.parseDouble(mActivated.g
etText().toString());
                        }
                        while (cursor.moveToNext());
                    }
                    petrol_dph=petrol_dd*dph;
                    diesel_dph=diesel_dd*dph;
                    treated_dph=treated_dd*dph;
                    untreated_dph=untreated_dd*dph;
                    activated_dph=activated_dd*dph;
                    database.execSQL("UPDATE data_city SET
petrol ="+Double.valueOf(mPetrol.getText().toString())
+ ", diesel="+Double.valueOf(mDiesel.getText().toString())
+",
treated_waste_water="+Double.valueOf(mTreated.getText().toString
())                          +",
untreated_waste_water="+Double.valueOf(mUntreated.getText().toSt
ring())                                 +",
activated_sludge="+Double.valueOf(mActivated.getText().toString(
))
                            +", petrol_dd="+petrol_dd
                            +", petrol_dph="+petrol_dph
                            +", diesel_dd="+diesel_dd
                            +", diesel_dph="+diesel_dph
                            +",
treated_waste_water_dd="+treated_dd
                            +",
treated_waste_water_dph="+treated_dph
                            +",
untreated_waste_water_dd="+untreated_dd
                            +",
untreated_waste_water_dph="+untreated_dph
                            +",
activated_sludge_dd="+activated_dd
                            +",
activated_sludge_dph="+activated_dph
                            +" WHERE data_city_id="
+get_city(get_cursor())+" AND year="+
Integer.parseInt(spinner_year.getSelectedItem().toString()));
                        restart();
                    break;
            case R.id.del_base_button:
                database.execSQL("DELETE FROM data_city WHERE
data_city_id=" +get_city(get_cursor())+" AND year="+
Integer.parseInt(spinner_year.getSelectedItem().toString()));
                restart();
                break;
            case R.id.null_button:
                mPetrol.setText(null);
                mDiesel.setText(null);
                mTreated.setText(null);
                mUntreated.setText(null);
                mActivated.setText(null);
```

```
                break;
        }
        dbHelper.close();
    }
}
```

```
package com.example.my_application;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import static com.example.my_application.MainActivity.data;
public class TableOfResults extends AppCompatActivity {
    public int get_city(Cursor cursor){
        int city_id=0;
        if(cursor.moveToFirst()){
            do{
                city_id = cursor.getInt(0);
            } while (cursor.moveToNext());
        }
        return city_id;
    }
    public Cursor get_cursor(){
        Cursor cursor;
        cursor = database.rawQuery("SELECT city_id FROM cities
WHERE city = '" + citySpinner.getSelectedItem().toString() +
"';", null);
        return cursor;
    }
    public double[] result(double[] mas, Cursor
query_table_data){
        for(int i=0; i<query_table_data.getCount(); i++) {
            double z=0;
            for(int y=0; y<=i;y++){
                z+=mas[i-y]*Math.exp((-1)*y/14.5);
            }
            mas[i]=z;
        }
        return mas;
    }
    ArrayAdapter<String> adapter;
    ArrayAdapter<String> adapter_city;
    DBHelper dbHelper;
    SQLiteDatabase database;
    Spinner citySpinner;
    static String[] city_mass_select;
    Cursor query_select;
    Cursor query_table_data;
    TableLayout table_dataCity;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_table_of_results);
        adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, data);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
        Spinner spinner = findViewById(R.id.spinner);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
                if(position==1){
                    Intent intent = new
Intent(TableOfResults.this,BaseActivity.class);
                    startActivity(intent);
                }
                if(position==2){
                    Intent intent = new
Intent(TableOfResults.this,TableOfResults.class);
                    startActivity(intent);
                }
                if(position==3){
                    Intent intent = new
Intent(TableOfResults.this,ScheduleActivity.class);
                    startActivity(intent);
                }
                if(position==4){
                    Intent intent = new
Intent(TableOfResults.this,EmissionFactorActivity.class);
                    startActivity(intent);
                }
            }
           @Override
            public void onNothingSelected(AdapterView<?> parent)
{
            }
        });
        dbHelper= new DBHelper(this);
        database = dbHelper.getWritableDatabase();
        query_select = database.rawQuery("SELECT * FROM
cities;", null);
        if(query_select.getCount()==0) {
            city_mass_select = new String[1];
            city_mass_select[0]= "Очень пусто";
        }
        else  if(query_select.getCount()!=0){
            city_mass_select = new
String[query_select.getCount()];
            if (query_select.moveToFirst()) {
                int i = 0;
                do {
                    city_mass_select[i++] =
(query_select.getString(1));
                }
                while (query_select.moveToNext());
            }
        }
```

```
        adapter_city = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, city_mass_select);

adapter_city.setDropDownViewResource(android.R.layout.simple_spi
nner_dropdown_item);
        citySpinner = findViewById(R.id.spinner_city2);
        citySpinner.setAdapter(adapter_city);
    }
    public void onClick(View view){
        database = dbHelper.getWritableDatabase();
         table_dataCity =findViewById(R.id.table_base);
        String[] titles = {"Год", "Бензин", "Дизель",
                "Сточные воды оч.", "Сточные воды не оч.","Сточ-
ные воды АИ" ,
                "ПХДД Бензин", "ПХДФ Бензин",
                "ПХДД Дизель", "ПХДФ Дизель",
                "ПХДД Сточные воды оч.", "ПХДФ Сточные воды
оч.",
                "ПХДД Сточные воды не оч.","ПХДФ Сточные воды не
оч.",
 "ПХДД Сточные воды АИ", "ПХДФ Сточные воды АИ",
                "НР ПХДД Бензин", "НР ПХДФ Бензин",
                "НР ПХДД Дизель", "НР ПХДФ Дизель",
                "НР ПХДД Сточные воды оч.", "НР ПХДФ Сточные
воды оч.",
                "НР ПХДД Сточные воды не оч.","НР ПХДФ Сточные
воды не оч." ,
                "НР ПХДД Сточные воды АИ", "НР ПХДФ Сточные воды
АИ"
        };
        table_dataCity.removeAllViews();
        TableRow title = new TableRow(this);
        for(int i=0; i<titles.length; i++){
            TextView yearLabel1 = new TextView(this);
            yearLabel1.setText(titles[i]);
            yearLabel1.setTypeface(Typeface.DEFAULT_BOLD);
            yearLabel1.setGravity(Gravity.CENTER);
            yearLabel1.setBackgroundResource(R.drawable.border);
            yearLabel1.setPadding(8,0,8,0);
            title.addView(yearLabel1);
        }
            table_dataCity.addView(title);
        query_table_data=database.rawQuery("SELECT * FROM
data_city WHERE data_city_id= " + get_city(get_cursor()) + "
ORDER BY year ", null);
        double[] mas_pet = new
double[query_table_data.getCount()];
        double[] mas_dies = new
double[query_table_data.getCount()];
        double[] mas_treat = new
double[query_table_data.getCount()];
        double[] mas_untreat = new
double[query_table_data.getCount()];
        double[] mas_activ = new
double[query_table_data.getCount()];
        if(query_table_data.moveToFirst()){
            int i=0;
            do{
                mas_pet[i] = (query_table_data.getDouble(8));
                mas_dies[i]
```

```
=Double.parseDouble(query_table_data.getString(10));
                mas_treat[i] =
Double.parseDouble(query_table_data.getString(12));
                mas_untreat[i] =
Double.parseDouble(query_table_data.getString(14));
                mas_activ[i] =
Double.parseDouble(query_table_data.getString(16));
                i++;
            }
            while (query_table_data.moveToNext());
        }
result(mas_pet,query_table_data);
        result(mas_dies,query_table_data);
        result(mas_treat,query_table_data);
        result(mas_untreat,query_table_data);
        result(mas_activ,query_table_data);
        double[][] array = new
double[5][query_table_data.getCount()];
            for (int j = 0; j < query_table_data.getCount();
j++) {
                array[0][j]=mas_pet[j];
                array[1][j]=mas_dies[j];
                array[2][j]=mas_treat[j];
                array[3][j]=mas_untreat[j];
                array[4][j]=mas_activ[j];
            }
        if(query_table_data.moveToFirst()){
            int count=0;
            do{
                TableRow rowYearLabels = new TableRow(this);
                for(int i=2; i<8; i++) {
                    TextView year_ = new TextView(this);

year_.setText(query_table_data.getString(i));
                    year_.setTypeface(Typeface.DEFAULT_BOLD);
                    year_.setGravity(Gravity.CENTER);

year_.setBackgroundResource(R.drawable.border);
                    rowYearLabels.addView(year_);
                }
for(int i=8; i<=17; i++) {
    TextView year = new TextView(this);

year.setText(String.format("%.10f",query_table_data.getDouble(i)
));
    year.setTypeface(Typeface.DEFAULT_BOLD);
    year.setGravity(Gravity.CENTER);
    year.setBackgroundResource(R.drawable.border);
    rowYearLabels.addView(year);
}
                for(int j=0; j<array.length; j++){
                TextView year_half_life1 = new TextView(this);

year_half_life1.setText(String.format("%.15f",array[j][count]));

year_half_life1.setTypeface(Typeface.DEFAULT_BOLD);
                    year_half_life1.setGravity(Gravity.CENTER);

year_half_life1.setBackgroundResource(R.drawable.border);
                    rowYearLabels.addView(year_half_life1);
```

```
                    TextView year_half_life = new
TextView(this);
year_half_life.setText(String.format("%.15f",array[j][count] *
0.1));
year_half_life.setTypeface(Typeface.DEFAULT_BOLD);
                    year_half_life.setGravity(Gravity.CENTER);

year_half_life.setBackgroundResource(R.drawable.border);
                    rowYearLabels.addView(year_half_life);
                }
count++;
                table_dataCity.addView(rowYearLabels);

            } while (query_table_data.moveToNext());
        }
            table_dataCity.setVisibility(View.VISIBLE);
    }
}
```

```
package com.example.my_application;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import  com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;
import static android.graphics.Color.GREEN;
import static com.example.my_application.MainActivity.data;
public class ScheduleActivity extends AppCompatActivity {
    public int get_city(Cursor cursor){
        int city_id=0;
        if(cursor.moveToFirst()){
            do{
                city_id = cursor.getInt(0);
            } while (cursor.moveToNext());
        }
        return city_id;
    }
    public Cursor get_cursor(){
        Cursor cursor;
        cursor = database.rawQuery("SELECT city_id FROM cities
WHERE city = '" + citySpinner.getSelectedItem().toString() +
"';", null);
        return cursor;
    }
    ArrayAdapter<String> adapter;
    DBHelper dbHelper;
    LineGraphSeries<DataPoint> series;
    LineGraphSeries<DataPoint> series1;
    SQLiteDatabase database;
    ArrayAdapter<String> adapter_city;
    Spinner citySpinner;
    static String[] city_mass;
    Cursor query;
    Cursor query1;
```

```
    String[] variant = {"Бензин", "Дизель", "Сточные воды оч.",
"Сточные воды не оч." , "Сточные воды АИ"};
    ArrayAdapter<String> adapter_variant;
    Spinner variantSpinner;
TableOfResults table = new TableOfResults();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_schedule);
        dbHelper= new DBHelper(this);
        adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, data);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
        Spinner spinner = findViewById(R.id.spinner);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
                if(position==1){
                    Intent intent = new
Intent(ScheduleActivity.this,BaseActivity.class);
                    startActivity(intent);
                }
                if(position==2){
                    Intent intent = new
Intent(ScheduleActivity.this,TableOfResults.class);
                    startActivity(intent);
                }
                if(position==3){
                    Intent intent = new
Intent(ScheduleActivity.this,ScheduleActivity.class);
                    startActivity(intent);
                }
                if(position==4){
                    Intent intent = new
Intent(ScheduleActivity.this,EmissionFactorActivity.class);
                    startActivity(intent);
                }
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent)
{
            }
        });
        database = dbHelper.getWritableDatabase();
        query1  = database.rawQuery("SELECT * FROM cities;",
null);
        if(query1.getCount()==0) {
            city_mass = new String[1];
            city_mass[0]= "Очень пусто";
        }
        else  if(query1.getCount()!=0){
            city_mass = new String[query1.getCount()];
            if (query1.moveToFirst()) {
                int i = 0;
                do {
                    city_mass[i++] = (query1.getString(1));
```

```
                }
                while (query1.moveToNext());
            }
        }
        adapter_city = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, city_mass);

adapter_city.setDropDownViewResource(android.R.layout.simple_spi
nner_dropdown_item);
        citySpinner = findViewById(R.id.spinner_city3);
        citySpinner.setAdapter(adapter_city);
        adapter_variant = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, variant);

adapter_variant.setDropDownViewResource(android.R.layout.simple_
spinner_dropdown_item);
        variantSpinner = findViewById(R.id.spinner_variant);
        variantSpinner.setAdapter(adapter_variant);
    }
    public void onClick(View view){
        database = dbHelper.getWritableDatabase();
int count=0;
                query = database.rawQuery("SELECT * FROM
data_city WHERE data_city_id= " + get_city(get_cursor()) + "
ORDER BY year ", null);
                series = new LineGraphSeries<DataPoint>();
        double[] mas_pet = new double[query.getCount()];
        double[] mas_dies = new double[query.getCount()];
        double[] mas_treat = new double[query.getCount()];
        double[] mas_untreat = new double[query.getCount()];
        double[] mas_activ = new double[query.getCount()];
        if(query.moveToFirst()){
            int i=0;
            do{
                mas_pet[i] = (query.getDouble(8));
                mas_dies[i]
=Double.parseDouble(query.getString(10));
                mas_treat[i] =
Double.parseDouble(query.getString(12));
                mas_untreat[i] =
Double.parseDouble(query.getString(14));
                mas_activ[i] =
Double.parseDouble(query.getString(16));
                i++;
            }
            while (query.moveToNext());
        }
        table.result(mas_pet,query);
        table.result(mas_dies,query);
        table.result(mas_treat,query);
        table.result(mas_untreat,query);
        table.result(mas_activ,query);
        double[][] array = new double[5][query.getCount()];
        for (int j = 0; j < query.getCount(); j++) {
            array[0][j]=mas_pet[j];
            array[1][j]=mas_dies[j];
            array[2][j]=mas_treat[j];
            array[3][j]=mas_untreat[j];
            array[4][j]=mas_activ[j];
        }
```

```
        DataPoint[] DP = new DataPoint[query.getCount()];
        DataPoint[] DP1 = new DataPoint[query.getCount()];
        switch (variantSpinner.getSelectedItemPosition()) {
            case 0:
               count=0;
                break;
            case 1:
                count=1;
                break;
            case 2:
                count=2;
                break;
            case 3:
                count=3;
                break;
            case 4:
                count=4;
                break;
        }
                if(query.moveToFirst()){
                    int i=0;
                    int column=0;
                    do{
                        DP[i] = new
DataPoint(Integer.parseInt(query.getString(2)),
array[count][column]);
                        DP1[i] = new
DataPoint(Integer.parseInt(query.getString(2)),
array[count][column]*0.1);
                        i++;
                        column++;
                    }
                    while (query.moveToNext());
                }
                series = new LineGraphSeries<>(DP);
                series1 = new LineGraphSeries<>(DP1);
                GraphView graph = (GraphView)
findViewById(R.id.graph1);
                graph.setVisibility(View.VISIBLE);
                graph.removeAllSeries();
                series.setDrawDataPoints(true);
                series.setDataPointsRadius(10);
                series.setAnimated(true);
        series1.setColor(GREEN);
        series1.setDrawDataPoints(true);
        series1.setDataPointsRadius(10);
        series1.setAnimated(true);

graph.getGridLabelRenderer().setNumHorizontalLabels(query.getCou
nt());
                graph.addSeries(series);
        graph.addSeries(series1);
                query.close();
        dbHelper.close();
    }
}
```

```
package com.example.my_application;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.database.Cursor;
```

```
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import static com.example.my_application.MainActivity.data;
public class EmissionFactorActivity extends AppCompatActivity {
    private EditText Petrol;
    private EditText Diesel;
    private EditText Treated;
    private EditText Untreated;
    private EditText Activated;
    private EditText Petrol1;
    private EditText Diesel1;
    private EditText Treated1;
    private EditText Untreated1;
    private EditText Activated1;
    DBHelper dbHelper;
    SQLiteDatabase database;
    ArrayAdapter<String> adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        dbHelper= new DBHelper(this);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_emission_factor);
        Petrol =  findViewById(R.id.petrol_koph);
        Diesel =  findViewById(R.id.diesel_koph);
        Treated =  findViewById(R.id.treat_koph);
        Untreated =  findViewById(R.id.untreat_koph);
        Activated =  findViewById(R.id.activ_koph);
        Petrol1 =  findViewById(R.id.petrol_koph1);
        Diesel1 =  findViewById(R.id.diesel_koph1);
        Treated1 = findViewById(R.id.treat_koph1);
        Untreated1 =  findViewById(R.id.untreat_koph1);
        Activated1 =  findViewById(R.id.activ_koph1);
        adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, data);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
        Spinner spinner = findViewById(R.id.spinner);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
                if(position==1){
                    Intent intent = new
Intent(EmissionFactorActivity.this,BaseActivity.class);
                    startActivity(intent);
                }
                if(position==2){
                    Intent intent = new
Intent(EmissionFactorActivity.this,TableOfResults.class);
                    startActivity(intent);
                }
                if(position==3){
                    Intent intent = new
```

```
Intent(EmissionFactorActivity.this,ScheduleActivity.class);
                    startActivity(intent);
                }
                if(position==4){
                    Intent intent = new
Intent(EmissionFactorActivity.this,EmissionFactorActivity.class)
;
                    startActivity(intent);
                }
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent)
{
            }
        });
    }
    public void onClick(View view){
        switch (view.getId()) {
            case R.id.ref_base_button_petrol:
                database.execSQL("UPDATE coefficient SET
coef_pet
="+(Double.parseDouble(Petrol.getText().toString())*Math.pow(10,
Double.parseDouble(Petrol1.getText().toString()))));
                break;
            case R.id.ref_base_button_diesel:
                database.execSQL("UPDATE coefficient SET
coef_dies
="+(Double.parseDouble(Diesel.getText().toString())*Math.pow(10,
Double.parseDouble(Diesel1.getText().toString()))));
                break;
            case R.id.ref_base_button_treat:
                database.execSQL("UPDATE coefficient SET
coef_treat
="+(Double.parseDouble(Treated.getText().toString())*Math.pow(10
,Double.parseDouble(Treated1.getText().toString()))));
                break;
            case R.id.ref_base_button_untreat:
                database.execSQL("UPDATE coefficient SET
coef_untreat
="+(Double.parseDouble(Untreated.getText().toString())*Math.pow(
10,Double.parseDouble(Untreated1.getText().toString()))));
                break;
            case R.id.ref_base_button_activ:
                database.execSQL("UPDATE coefficient SET
coef_activ
="+(Double.parseDouble(Activated.getText().toString())*Math.pow(
10,Double.parseDouble(Activated1.getText().toString()))));
                break;
        }
    }
}
```