МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук управ-
ління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Створення програми для часової інвентаризації накопичення стійких органічних забруднюючих речовин у довкіллі

Виконав студент 4 курсу групи К-42
Напрямок 6.05.01.01 Комп'ютерні науки

Нємцев Владислав Євгенович

Керівник ст.викл.
Рольщиков Вадим Борисович

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

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

Одеса 2019

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

БД – база даних

ПП – програмний продукт

ПК – персональний комп’ютер

ПЗ – програмне забезпечення

СКБД – система керування базами даних

ПХДД – поліхлоровані дибензо-п-діоксини

ПХДФ – поліхлоровані дибензофурани

СОЗ – стійкі органічні забруднювачі

ТЗ – технічне завдання

ТСН – технічні засоби навчання

IDE – Integrated Development Environment, інтегроване середовище розробки

UML – unified modeling language, уніфікована мова моделювання

ВСТУП

З початку індустріальної епохи у XVIII ст. споживання ресурсів людиною значно збільшилось. Що привело до збільшення відходів з різних сфер виробництва, та заповнення навколишнього середовища новими забруднювачами.

Найбільшу небезпеку представляють стійкі органічні забруднюючі речовини (СОЗ). Ці стійкі органічні забруднюючі речовини негативно впливають на здоров’я людини та навколишню середовище. Вони поширюються у повітрі та воді, та можуть впливати на людину та довкілля на значній відстані від того місця, де їх застосовували.

СОЗ можуть бути як навмисно вироблені (хімічні речовини для використання у сільському господарстві і т.д.), так і ненавмисно (виділяються при деяких виробничих процесах).

Нормального обліку СОЗ не ведеться, отже і невідомо наскільки сильно забруднене довкілля і наскільки серйозним може бути вплив на людину.

Актуальність роботи полягає у прохані в наданні кафедрі екології та охорони довкілля Одеського державного екологічного університету програми для обліку стійких органічних забруднюючих речовин. Оскільки ця проблема є маловивчена, та завдяки щорічному викиду мільйонів тон різних забруднюючих речовин. Також дослідження вчених які показують, що збільшення антропогенних навантажень на природні системи перевищують допустимі рівні.

Отже велика токсичність СОЗ є невирішеною і з кожним роком тільки погіршується.

Саме тому вченим необхідне застосування для обліку стійких органічних забруднюючих речовин в навколишньому середовищі та бази даних (БД) до неї.

Метою дипломної роботи є створення системи, яка зможе проводити облік цих речовин та обробляти дані по ним. А також допоможе вченим визначати рівень забруднення, та допомогти вченим запобігти подальшому забрудненню навколишнього середовища.

Отже задачами для виконання дипломної роботи є:

- провести фізичне та логічне моделювання бази даних;
- обрати СКБД;
- обрати систему розробки програми;
- виконати програмування;
- розробити інструкцію користувача.

Представлений дипломний проект містить 57 сторінок,24 рисунки, 16 посилань, та 1 таблицю.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз предметної області

Кінцевий програмний продукт – це програмний комплекс, який допомагає екологам відстежувати кількість СОЗ в довкіллі і розраховувати їх піврозпад за період часу.

ПП включатиме в себе місце збереження даних і програми настільного застосування(desktop application), котре буде відображати ці дані, давати змогу їх обробки та визначати кількість поліхлоровані дибензо-п-діоксини(ПХДД) і поліхлоровані дибензофурани(ПХДФ) за період часу у довкіллі.

Підставою для проведення розробки було прохання кафедри екології та охорони довкілля Одеського державного екологічного університету про створення програми, яка б допомогла у розрахунку ПХДД і ПХДФ з різних джерел.

Під час розробки ПП повинні виконуватися вимоги:

- ПП повинен розраховувати ПХДД і ПХДФ за різні періоди часу;
- ПП повинен мати графічний інтерфейс;
- ПП повинен креслити графік, який показує зміни ПХДД і ПХДФ протягом даного періоду часу;
- ПП повинен мати БД, яка зберігає вхідні дані для розрахунку.

Актуальність цієї роботу полягає в тому, що проблема викидів СОЗ є маловивчена. Також СОЗ мають серйозні наслідки навіть при недовгому впливу на організм. Отже інформація щодо розміру забруднення є дуже необхідною для захисту довкілля, і для виявлення «гарячих точок» у місцях, де через забруднення відбуваються негативні зміни в здоров'ї населення.

Джерело [1][1)]вказує, що з XVIII ст. через промислову, а потім і науково-технічну революцій на зміну доіндустріальній епосі приходить індустріальна. У результаті за останні 100 років споживання людиною ресурсів виросло у 100

[1)] [1] Солдатов А. И. Источники загрязнения среды обитания. Часть 1: Стойкие органические загрязнители: конспект лекций – Челябинск: Издательский центр ЮУрГУ, 2015. 157с.

разів. А науково-технічний прогрес у першій половині XX століття привів до бурхливого розвитку хімії та хімічної технології, яке супроводжувалося отриманням у лабораторіях тисячі нових синтетичних органічних речовин. Поява великої кількості речовин, призвело до появи ще одної кризи – кризи редуцентів. Редуценти не встигають очищати біосферу від забруднення, часто вони на це не здатні біологічно. І це призводить до порушення кругообігу речовин у біосфері.

Згідно з джерела [2][1)] СОЗ бувають:

- навмисно вироблені;
- ненавмисно вироблені.

Навмисно вироблені СОЗ – це хімічні речовини вироблені для використання в сільському господарстві, виробництві або в технологічних процесах.

Ненавмисно вироблені СОЗ – це хімічні речовини які виділяються при деяких виробничих процесах та при згоранні.

Усі відомі СОЗ мають чотири спільних, характерних для них особливостей:

- висока токсичність, яка проявляється вже при надзвичайно малих дозах;
- висока стійкість в довкіллі під впливом природних факторів;
- здатність накопичуватись в живих організмах;
- здатність до транскордонного перенесення на велику відстань.

Завдяки цим особливостям проблему СОЗ вважають однією з глобальних екологічних проблем, таких як руйнування озонового шару.

Механізми впливу СОЗ дуже складні та являються рядом послідовних подій на молекулярному рівні, який призводить до змін в регуляції роботи генів та життєдіяльності клітин.

Вплив СОЗ приводять до різних наслідків (див. табл. 1).

[1)] [2] Стойкие органические загрязнители: глобальная проблема, глобальное решение. URL: http://www.unido-russia.ru/archive/num_14/art14_5 (дата звернення 10.06.2019).

Таблиця 1 – Деякі основні ефекти СОЗ на здоров'я людини та біосферу [3][1)]

Речовини	Вплив
ДДТ	Тварини: пошкодження репродуктивної функції, в т.ч. потоншення яєчної шкаралупи у птахів в різних регіонах світу і фемінізація чайок в Каліфорнії, різке падіння чисельності популяції алігаторів у Флориді, зниження у них рівня тестостерона в крові самців і зменшення розміру статевого члена; пороки розвитку скелета у сірого тюленя.
	Придушення синтезу зелених водоростей.
	Зникнення перелітних дроздів в США в результаті отруєння ДДТ.
	Можливий канцероген для людини, можливо один з факторів ризику розвитку раку молочної залози.
	Високі дози впливу призводять до порушень нервової системи (конвульсій, тремору, м'язової слабкості).
Гептахлор	Тварини: вплив на рівні прогестерону і естрогену у лабораторних щурів.
	Порушення нервової системи і функції печінки.
Поліхлоровані дибензо-п-діоксини – ПХДД і поліхлоровані дибензофурани – ПХДФ	Тварини – політропність впливу, в т.ч. на імунну систему, ураження печінки, синдром виснаження, порушення порфіринового обміну, вплив на ферментні системи, придушення процесу кровотворення, набряк підшкірної клітковини у курей при впливі забруднених кормів. Порушення репродуктивної функції, трансплацентарний ефект, зменшення кількості сперми, нездатність завагітніти або доносити плід у мавп, вроджені Тератогенний ефект – вроджені вади розвитку у щурів, мишей, мавп. ТХДД – канцерогенний ефект – пухлини печінки, легені та інших локалізацій у щурів і мишей.

[1)] [3] Ревич Б. А. Основные сведения о стойких органических загрязнителях (СОЗ).: [б. м.].

Продовження таблиці 1

Речовини	Вплив
	Людина – ТХДД, можливий канцероген для людини; можливо є фактором ризику розвитку раку молочної залози, легенів, шлунку, печінки і жовчних шляхів, неходжкінської лімфоми. Шкірні прояви – хлоракне. Неврологічні ефекти – порушення зору, невропатії та ін. Вплив на репродуктивне здоров'я, ендокринну та імунну систему; в т.ч. затримка статевого розвитку, сексуальні розлади, збільшення частоти безпліддя, спонтанних абортів, ендометріозу, вроджених вад розвитку, новонароджених з малою масою тіла, зміни співвідношення статей новонароджених, порушення гормонального статусу, в т.ч. зниження рівня тестостерону і інші прояви фемінізації, медико-генетичні порушення – збільшення числа хромосомних аберацій.
Поліхлоровані біфеніли (ПХБ)	Тварини - порушення відтворення (репродуктивної функції) у норки, птахів (скопа, орел), тюленя, видри.
	Можливий канцероген для людини; порушення репродуктивного здоров'я (безпліддя, самовільні аборти, мала вага у новонароджених); вплив на нервово-психічний розвиток дітей.

1.2 Аналіз ринку подібних проектів

Був проведений моніторинг існуючого програмного забезпечення (ПЗ), призначеного для допомоги у вирішенні екологічних проблем. В результаті, були виявлені такі програми як Dirty Dozen[4][1)], Earth-Now[5][2)] та Loss of the night[6][3)].

[1)] [4] Dirty Dozen for Android – APK Download. URL: https://apkpure.com/dirty-dozen/com.ewg.dirtydozen(дата звернення 10.06.19).

[2)] [5] Приложения в Google Play – Earth-Now. URL: https://play.google.com/store/apps/details?id=gov.nasa.jpl.earthnow.activity&hl=ru(дата звернення 10.06.19).

[3)] [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 10.06.19).

1.2.1 Розгляд застосування Dirty Dozen

Dirty Dozen – це сучасне застосування, для людей які бажають їжу вільну від синтетичних хімікатів. Тому як за даними міністерства сільського господарства США було знайдено 178 видів пестицидів та продуктів розпаду на тисячах проаналізованих зразків продукції. Пестициди зберігалися на фруктах і овочах, навіть коли вони були вимиті і в деяких випадках очищені.

На рисунку 1.1 наведений загальний вигляд інтерфейсу застосування.

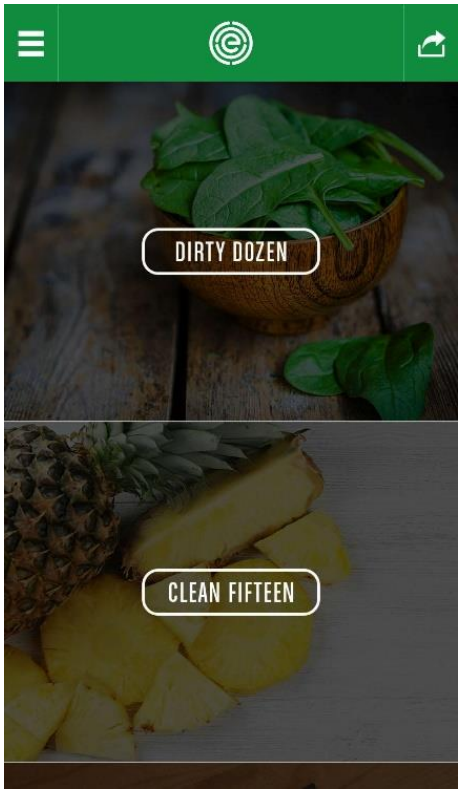

Рисунок 1.1 – Вибір списку чистих або брудних фруктів

Після натиснення на той чи інший заголовок списку застосування показує список овочів і фруктів в обраному списку (див. рис. 1.2).

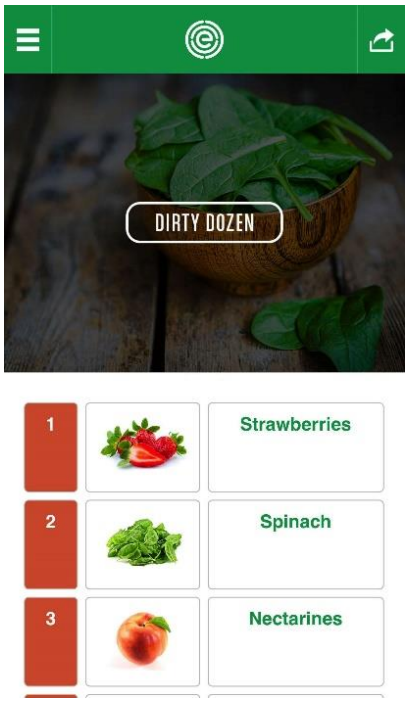

Рисунок 1.2 – Обраний список

На третій вкладці (див. рис. 1.3) обраний елемент списку, на якому зображено інформацію про обраний продукт.

Рисунок 1.3 – інформація про продукт

Недоліком засобу є те, що він орієнтований тільки на сільськогосподарський ринок США.

1.2.2 Розгляд застосування Earth-Now

Earth-Now – це засіб, створений NASA, який зображує інформацію глобального клімату землі отриману із супутників Earth Science.

На рис. 1.4 наведено модель землі та кнопки вибору необхідної інформації.

Рисунок 1.4 – Головний екран

На другій вкладці (див. рис. 1.5) зображено список категорій доступної інформації.

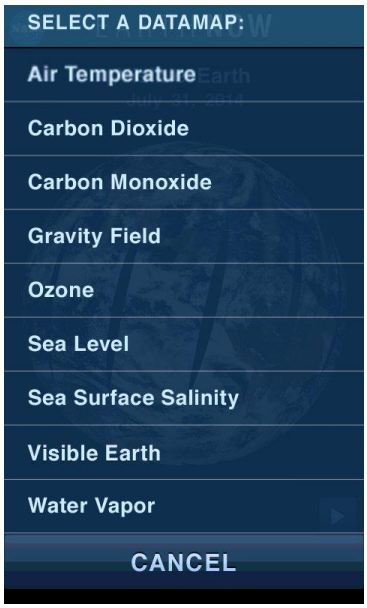

Рисунок 1.5 – Меню вибору інформації для відображення

Для того щоб побачити модель землі з обраною інформацією, наприклад температура повітря на поверхні землі удень (див. рис. 1.6), потрібно обрати категорію даних з меню.

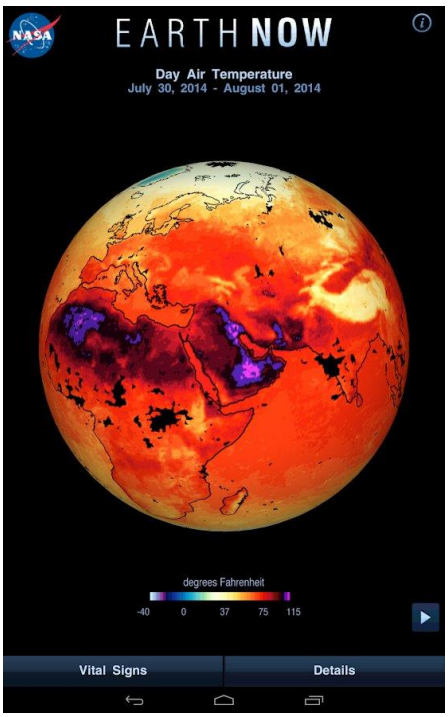

Рисунок 1.6 – Модель землі з показниками температури на поверхні

1.2.3 Розгляд застосування Loss of the night

Loss of the night – це застосування для виявлення наскільки забруднене нічне небо та наскільки погано проходить світло від зірок.

На першій вкладці (див. рис. 1.7) зображено сузір’я і кнопка для початку тесту забруднення зіркового світла.

Рисунок 1.7 – Стартовий екран застосування

Після початку тесту (див. рис. 1.8) застосування наводить користувача до необхідної зірки.

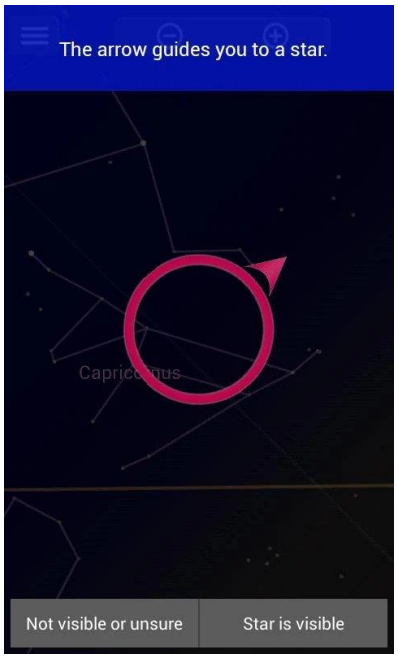

Рисунок 1.8 – екран з інструкціями для користувача

На наступній вкладці (див. рис. 1.9) вже знайдена зірка, і застосування опитує користувача наскільки її видно.

Рисунок 1.9 – екран опитування щодо видимості зірки для користувача

На останній вкладці (див. рис. 1.10) зображено меню завершення роботи застосування та питання чи хоче користувач відправити дані до вчених або продовжити тест.

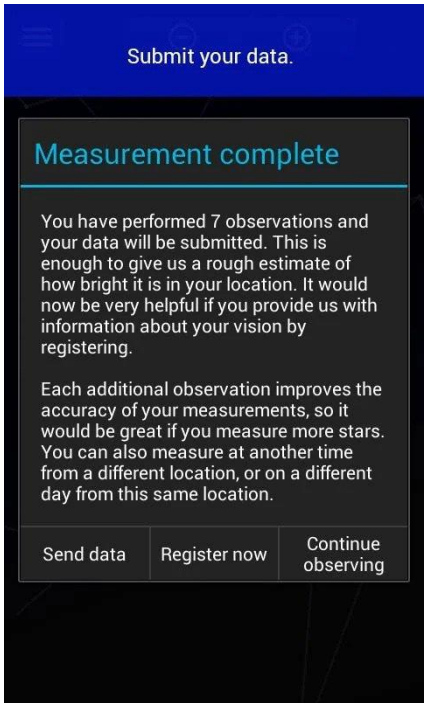

Рисунок 1.10 – Екран завершення тесту

Всі ці застосування є яскравими прикладами необхідності аналогічного ПЗ для різних галузей екології.

Нажаль подібної програми для розрахунку ПХДД і ПХДФ немає.

1.3 Аналіз інструментів розробки

Як розглядалося у вступі (стор.), основними завданнями при розробці ПП є:

- вибір СКБД для створення БД ПП;
- вибір системи розробки програми.

1.4 Вибір платформи розробки

При виборі платформи були розглянуті та порівняні програмні платформи Java та .NET Framework.

1.4.1 Розгляд Java

Платформа Java – ряд програмних продуктів і специфікацій розроблених компанією Sun Microsystems, нині дочірньої компанії корпорації Oracle, які спільно надають систему для розробки прикладного програмного забезпечення та вбудовування її в будь-яке крос-платформне програмне забезпечення. Java використовується в самих різних комп'ютерних платформах від вбудованих пристроїв і мобільних телефонів в нижньому ціновому сегменті, до корпоративних серверів і суперкомп'ютерів у вищому ціновому сегменті.

Програмний код, написаний на Java, віртуальна машина Java виконує байт-код Java. Однак є компілятори байт-коду для інших мов програмування, таких як Ada, JavaScript, Python, і Ruby. Також є кілька нових мов програмування, розроблених для роботи з віртуальною машиною Java. Це такі мови як Scala, Clojure and Groovy. Синтаксис Java (англ.) В основному запозичений з C і C ++, але об'єктно-орієнтовані можливості засновані на моделі, використовуваної в Smalltalk і Objective-C. В Java відсутні певні низького рівня конструкції, такі як покажчики, також Java має дуже просту модель пам'яті, де кожен об'єкт розташований в купі і всі змінні об'єктного типу є посиланнями. Управління пам'яттю здійснюється за допомогою інтегрованої автоматичної збірки сміття, яку виконує JVM.

Програмна платформа Java – це ім'я пакета програм компанії Sun Microsystems, які дозволяють розробляти і запускати програми, написані на мові програмування Java. Ця програмна платформа не є специфічною для якого-небудь одного процесора або операційної системи, але механізм виконання

(званий віртуальною машиною) і компілятор з набором бібліотек, які реалізовані для різного апаратного забезпечення і різних операційних систем, щоб Java-програми могли працювати скрізь однаково.

Також Java платформа має чотири спеціалізованих технології.

Першою є технологія Java Card яка дозволяє невеликим Java-застосуванням (аплетам) надійно працювати на смарт-картах та інших подібних пристроях з малим об'ємом пам'яті.

Друга технологія – Java ME, включає в себе декілька різних наборів бібліотек (відомих як профілі) для пристроїв з обмеженим обсягом місця для зберігання, невеликим розміром дисплея і батареї. Часто використовується для розробки застосувань для мобільних пристроїв, КПК, ресиверів цифрового телебачення та принтерів.

Наступна – третя технологія яка має назву Java SE використовується для використання на настільних ПК, серверах і іншому подібному обладнанні.

І остання – четверта технологія під назвою Java EE: Java SE плюс API, корисне для багаторівневих клієнт-серверних бізнес-застосувань [7][1)].

1.4.2 Розгляд .NET Framework

Платформа .NET Framework – це технологія, яка підтримує створення і виконання нового покоління застосувань і веб-служб XML. При розробці платформи .NET Framework враховувалися наступні цілі.

По-перше, забезпечення узгодженого об'єктно-орієнтованого середовища програмування для локального збереження і виконання об'єктного коду, для локального виконання коду, розподіленого в Інтернеті, або для віддаленого виконання.

По-друге, забезпечення середовища виконання коду, що мінімізує конфлікти при розгортанні програмного забезпечення та управлінні версіями.

[1)] [7] Java (программная платформа) – Википедия. URL: https://ru.wikipedia.org/wiki/Java_(программная_платформа)(дата звернення 06.06.19).

По-третє, забезпечення середовища виконання коду, що гарантує безпечне виконання коду, враховуючи код, створений невідомим або не повністю довіреним стороннім постачальником;

По-четверте, забезпечення середовища виконання коду, що виключає проблеми з продуктивністю середовищ виконання сценаріїв або інтерпретованого коду.

По-п’яте, забезпечення єдиних принципів розробки для різних типів застосувань, таких як застосування Windows і веб-застосування.

По-шосте, взаємодія на основі промислових стандартів, яке гарантує інтеграцію коду платформи .NET Framework з будь-яким іншим кодом.

Платформа .NET Framework складається з загальномовного середовища виконання (середовища CLR) і бібліотеки класів .NET Framework. Основою платформи .NET Framework є середовище CLR. Середовище виконання можна вважати агентом, який керує кодом під час виконання і надає основні служби, такі як керування пам'яттю, керування потоками і віддалене взаємодія. При цьому середовищем накладаються умови суворої типізації та інші види перевірки точності коду, що забезпечують безпеку і надійність. Фактично основним завданням середовища виконання є керування кодом. Код, який звертається до середовища виконання, називають керованим кодом, а код, який не обертається до середовища виконання, називають некерованим кодом. Бібліотека класів є комплексною об'єктно-орієнтованою колекцією повторно використовуваних типів, які застосовуються для розробки застосувань – починаючи з звичайних застосувань, що запускаються з командного рядка, і застосувань з графічним інтерфейсом (GUI) і закінчуючи застосуваннями, що використовують останні технологічні можливості ASP.NET, такі як веб-форми і веб-служби XML.

Платформа .NET Framework може розміщуватися некерованими компонентами, які завантажують середовище CLR у власні процеси і запускають виконання керованого коду, створюючи таким чином програмне середовище, що

дозволяє використовувати засоби як керованого, так і некерованого виконання. Платформа .NET Framework не тільки надає кілька базових середовищ виконання, але також підтримує розробку базових середовищ виконання незалежними виробниками [8][1)].

За результатами аналізу було прийнято рішення, що розробка ПП буде проводитись на платформі .NET Framework на мові C#. Вибір обґрунтовано тим, що кінцевому користувачу немає потреби встановлювати додаткове ПЗ, наприклад JVM.

1.5 Вибір СКБД

1.5.1 Опис Microsoft SQL Server

Microsoft SQL Server – комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів – Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI/ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

Microsoft SQL Server як мову запитів використовує версію SQL, що отримала назву Transact-SQL (скорочено T-SQL), яка є реалізацією SQL-92 (стандарт ISO для SQL) з багатьма розширеннями. T-SQL дозволяє використовувати додатковий синтаксис процедур, що зберігаються і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим застосунком). Microsoft SQL Server та Sybase ASE для взаємодії з мережею використовують протокол рівня застосунка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних).

[1)] [8] Общие сведения о платформе .NET Framework. URL: https://docs.microsoft.com /ru-ru/dotnet/framework/get-started/overview(дата звернення 06.06.19).

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC) – інтерфейс взаємодії застосунків з СКБД. Версія SQL Server 2005 надає можливість підключення користувачів через веб-сервер-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, кроссплатформенно з'єднуватися з SQL Server. Microsoft також випустила сертифікований драйвер JDBC, що дозволяє застосункам під керування Java (таким як BEA і IBM Websphere) з'єднуватися з Microsoft SQL Server 2000 і 2005.

SQL Server підтримує дзеркалювання та кластеризацію баз даних. Кластер серверу SQL – це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між декількома серверами. Усі сервери мають одне віртуальне ім'я, а дані розподіляються за IP-адресами машин кластеру протягом робочого циклу. Також у разі відмови або збою на одному з серверів кластеру доступне автоматичне перенесення навантаження на інший сервер.

SQL Server підтримує дублювання даних за трьома сценаріями.

Перший сценарій – Знімок: Виконується «знімок» бази даних, який сервер відправляє одержувачам.

Другий сценарій – Історія змін: Всі зміни бази даних безперервно передаються користувачам.

Третій сценарій – Синхронізація з іншими серверами: Бази даних декількох серверів синхронізуються між собою. Зміни усіх баз даних відбуваються незалежно на кожному сервері, а під час синхронізації відбувається звірка даних. Дублювання такого типу передбачає можливість вирішення протиріч між базами даних.

SQL Server 2005 має вбудовану підтримку .NET Framework. Завдяки цьому, процедури бази даних, що зберігаються, можуть бути написані на будь-якій мові платформи .NET з використанням повного набору бібліотек, доступних для .NET Framework. На відміну від інших процесів, .NET Framework ви-

діляє додаткову пам'ять і будує засоби керування SQL Server, не використовуючи вбудовані засоби Windows. Це підвищує продуктивність порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налагоджені для використання у структурах SQL Server[9][1)].

1.5.2 Опис PostgreSQL

PostgreSQL – об'єктно-реляційна система керування базами даних (СКБД). Є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite).

Порівняно з іншими проектами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення.

Сервер PostgreSQL написаний на мові C. Зазвичай розповсюджується у вигляді набору текстових файлів із сирцевим кодом. Для інсталяції необхідно відкомпілювати файли на своєму комп'ютері і скопіювати в деякий каталог. Весь процес детально описаний в документації.

Функції дозволяють виконувати деякий код безпосередньо сервером бази даних. Ці функції можуть бути написані на SQL, який має деякі примітивні програмні оператори, такі як галуження та цикли. Але гнучкішою буде функція написана на одній із мов програмування, з якими PostgreSQL може працювати. До таких мов належать:

- Вбудована мова, яка зветься PL/pgSQL, подібна до процедурної мови PL/SQL компанії Oracle;

[1)] [9] Microsoft SQL Server – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft _SQL_Server(дата звернення 06.06.19).

- Мови розробки сценаріїв: PL/Perl, PL/Python, PL/Tcl, PL/Ruby, PL/sh;
- Класичні мови програмування C, C++, Java (за допомогою PL/Java).

Функції можуть виконуватись із привілеями користувача, який її викликав, або із привілеями користувача, який її написав.

PostgreSQL підтримує одночасну модифікацію БД декількома користувачами за допомогою механізму Multiversion Concurrency Control (MVCC). Завдяки цьому виконуються вимоги ACID, і практично відпадає потреба в блокуванні зчитування.

Таблиці можуть успадковувати характеристики та набори полів від інших таблиць (батьківських). При цьому дані, які додаються до породженої таблиці, автоматично будуть брати участь (якщо це не вказано окремо) в запитах до батьківської таблиці. Цей функціонал в поточний час не є повністю завершеним. Однак він достатній для практичного використання[10][1)].

1.5.3 Опис SQLite

SQLite – полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання, тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою

[1)] [10] PostgreSQL – Вікіпедія. URL: https://uk.wikipedia.org/wiki/PostgreSQL(дата звернення 06.06.19).

частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті постачання йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС.

Сама бібліотека SQLite написана мовою С. Проте є реалізація бібліотеки на JavaScript sql.js, яка дозволяє обробляти файли БД безпосередньо в браузері[11][1)].

В результаті аналізу було визначено, що для ПП потрібна SQLite, бо БД яка перебуває на локальному сервері може видати помилку при спробі підключення до сервера. У випадку SQLite у нас буде портативна БД для якої сервер не потрібний.

[1)] [11]SQLite – Вікіпедія. URL: https://uk.wikipedia.org/wiki/SQLite(дата звернення 06.06.19).

1.6 Вибір IDE

1.6.1 Розгляд Eclipse

Eclipse – вільне модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для програмістів на мові Java, системи контролю версій, конструктори GUI тощо. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи Ada, C, C++, COBOL, Fortran, Perl, PHP, Python, R, Ruby (включно з каркасом Ruby on Rails), Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP.

Початок коду йде від IBM VisualAge, він був розрахований на розробників Java, складаючи Java Development Tools (JDT). Але користувачі могли розширяти можливості, встановлюючи написані для програмного каркасу Eclipse плагіни, такі як інструменти розробки під інші мови програмування, і могли писати і вносити свої власні плагіни і модулі.

Випущена на умовах Eclipse Public License, Eclipse є вільним програмним забезпеченням. Він став одним з перших IDE під GNU Classpath і без проблем працює під IcedTea.

Eclipse це фреймворк для розробки модульних платформонезалежних застосунків із низкою особливостей:

- можливість розробки ПЗ на багатьох мовах програмування (рідною є Java);
- платформонезалежна;
- модульна, призначена для подальшого розширення незалежним розробниками;
- з відкритим сирцевим кодом;

- розробляється і підтримується фондом Eclipse, куди входять такі постачальники ПЗ, як IBM, Oracle, Borland.

Спочатку проект розроблявся в IBM як корпоративний стандарт IDE для розробки на багатьох мовах під платформи IBM. Потім проект було перейменовано на Eclipse і надано для подальшого розвитку спільноті.

Eclipse насамперед повноцінна Java IDE, націлена на групову розробку, має засоби роботи з системами контролю версій (підтримка CVS входить у поставку Eclipse, активно розвиваються кілька варіантів SVN модулів, існує підтримка VSS та інших). З огляду на безкоштовність, у багатьох організаціях Eclipse – корпоративний стандарт для розробки ПЗ на Java[12][1)].

1.6.2 Опис MonoDevelop

MonoDevelop – це комплексне середовище розробки з відкритим кодом для Linux, MacOS і Windows. Основна увага приділяється розробці проектів, що використовують рамки Mono та .NET. MonoDevelop об'єднує функції, подібні до функцій NetBeans і Microsoft Visual Studio, такі як автоматичне завершення коду, керування джерелами, графічний інтерфейс користувача та вебдизайнер. MonoDevelop об'єднує дизайнера графічного інтерфейсу Gtk # під назвою Stetic. Він підтримує Boo, C, C++, C#, CIL, D, F#, Java, Oxygene, Vala, JavaScript, TypeScript і Visual Basic.NET.

MonoDevelop можна використовувати на Windows, MacOS і Linux. Офіційно підтримувані дистрибутиви Linux включають CentOS, Debian, Fedora, openSUSE, SUSE Linux Enterprise, Red Hat Enterprise Linux і Ubuntu, причому багато інших дистрибутивів забезпечують власні неофіційні збірки MonoDevelop у своїх сховищах. MacOS і Windows офіційно підтримуються з версії 2.2.

[1)] [12] Eclipse – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Eclipse(дата звернення 06.06.19).

MonoDevelop включає можливості подібні до NetBeans та Microsoft Visual Studio, такі як автоматичне доповнення, інтеграція контролю коду, графічний користувацький інтерфейс і веб-дизайнер. В MonoDevelop інтегрований Gtk# GUI дизайнер під назвою Stetic[13][1)].

1.6.3 Опис Microsoft Visual Studio

Microsoft Visual Studio є інтегрованим середовищем розробки (IDE) від Microsoft. Він використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-застосувань, веб-служб і мобільних застосувань. Visual Studio використовує платформи розробки програмного забезпечення Microsoft, такі як Windows API, Windows Forms, Windows Presentation Foundation, Магазин Windows і Microsoft Silverlight. Вона може виробляти як власний код, так і керований код.

Visual Studio включає в себе редактор коду, що підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. Інтегрований відладчик працює як відладчик вихідного рівня, так і відладчик на рівні машини. Інші вбудовані інструменти включають в себе профайлер коду, дизайнер форм для створення GUI-застосувань, веб-дизайнер, дизайнер класів і дизайнер схеми бази даних. Він приймає плагіни, що підвищують функціональність практично на кожному рівні, включаючи додавання підтримки систем керування джерелами (наприклад, Subversion і Git) і додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для доменних мов або наборів програм для інших аспектів життєвого циклу розробки (наприклад, клієнт Team Foundation Server: Team Explorer).

Visual Studio підтримує 36 різних мов програмування і дозволяє редактору коду і відладчику підтримувати (в різному ступені) практично будь-яку

[1)] [13] MonoDevelop – Wikipedia. URL: https://en.wikipedia.org/wiki/MonoDevelop (дата звернення 06.06.19).

мову програмування, за умови наявності спеціальної мовної служби. Вбудовані мови включають C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML і CSS. Підтримка інших мов, таких як Python, Ruby, Node.js і M серед інших, доступна через плагіни. Java (і J#) підтримувалися в минулому[14][1)].

По завершенні розгляду, для розробки ПП було обрано середовище Visual Studio. Так як обрана мова C# є крос-платформною то в майбутньому буде можливість перенести застосування на інші платформи, такі як UNIX, MAC OS або Android.

Для розробки ПП буде використовуватись Visual Studio, тому що в ній є Team Foundation Server, що дозволяє багатьом розробникам працювати над одним проектом. Також Visual Studio має сервера на яких зареєстрований користувач може зберегти код, що є практичним у разі знищення фізичної копії через різні можливі обставини.

[1)] [14] Microsoft Visual Studio – Wikipedia. URL: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio(дата звернення 06.06.19).

2 ПРОЕКТНА ЧАСТИНА

2.1 Постановка завдань

За результатами огляду і аналізу предметної області, ринку подібних проектів і вимог до ПП, були виділені такі завдання:

- спроектувати та розробити БД для ПП;
- спроектувати та розробити UML діаграму діяльності;
- спроектувати та розробити зовнішній вигляд ПП;
- спроектувати та розробити систему розрахунку та побудови графіку.

2.2 Проектування структури БД ПП

Кожна БД складається з різної кількості об’єктів, такі об’єкти називають сутностями. Сутністю БД може бути будь-який об’єкт, який можна виділити із предметної області, для якої розробляється база даних. Кожна сутність має атрибути. Атрибути потрібні для опису сутності. Модель сутностей бази даних можна побачити на рис. 2.1.

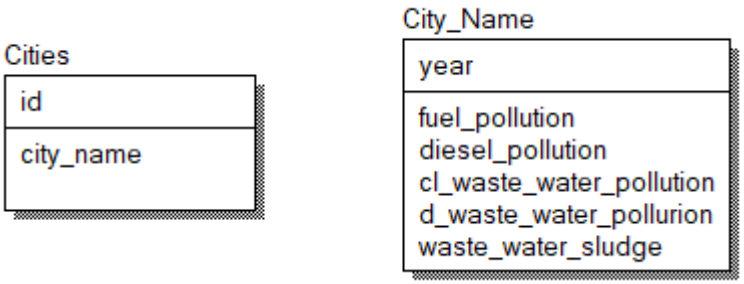

Рисунок 2.1 – Схема БД застосування

На схемі зображено дві сутності.

Перша сутність – міста (Cities), має атрибути (властивості): ідентифікатор (id), назва міста (city_name).

Друга сутність – назва міста (City_Name), має наступні атрибути (властивості): забруднення бензином (fuel_pollution), забруднення дизелем (diesel_pollution), забруднення очищеними стічними водами (cl_waste_water_pollution), забруднення неочищеними стічними водами (d_waste_water_pollution), забруднення стічними водами з активним мулом (waste_water_sludge).

БД ПП буде зберігати лише дані екземплярів сутностей, а процедури і таблиці будуть реалізовані за допомогою функцій застосування.

2.3 Побудова UML діаграми діяльності

Як вказано в джерелі [15][1)] при моделюванні поведінки проектованої або аналізованої системи виникає необхідність не тільки представити процес зміни її станів, але і деталізувати особливості алгоритмічної і логічної реалізації виконуваних системою операцій. Традиційно для цієї мети використовувалися блок-схеми або структурні схеми алгоритмів. Кожна така схема акцентує увагу на послідовності виконання певних дій або елементарних операцій, які в сукупності призводять до отримання бажаного результату.

Алгоритмічні і логічні операції, що вимагають виконання в певній послідовності, оточують нас постійно. Звичайно, ми не завжди замислюємося про те, що подібні операції відносяться до настільки наукових категорій. Наприклад, щоб зателефонувати, нам спочатку потрібно зняти трубку або включити його. Для приготування кави або заварювання чаю необхідно спочатку закип'ятити воду. Щоб виконати ремонт двигуна автомобіля, потрібно здійснити цілий ряд нетривіальних операцій, таких як розбирання силового агрегату, зняття генератора і деяких інших.

Важливо підкреслити ту обставину, що зі збільшенням складності системи суворе дотримання послідовності виконуваних операцій набуває все більш важливе значення. Якщо спробувати заварити каву холодною водою, то

[1)] [15] Леоненков, самоучитель UML. URL: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl7/gl7.html (дата звернення 07.06.19).

ми можемо тільки зіпсувати одну порцію напою. Порушення послідовності операцій при ремонті двигуна може привести до його виходу з ладу. Ще більш катастрофічні наслідки можуть статися в разі відхилення від встановленої послідовності дій при зльоті або посадці авіалайнера, запуска ракети, регламентних роботах на АЕС.

Діаграми діяльності - це графічні представлення робочих процесів поетапної діяльності та дій з підтримкою вибору, ітерації та паралельності. У уніфікованій мові моделювання діаграми діяльності призначені для моделювання як обчислювальних, так і організаційних процесів (тобто робочих процесів), а також потоків даних, що перетинаються з відповідними діями. Хоча діаграми активності в першу чергу показують загальний потік управління, вони також можуть включати елементи, що показують потік даних між діями через один або більше сховищ даних.

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

- еліпси являють собою дії;
- ромб являє собою прийняття рішення;
- стовпці представляють початок (розбиття) або кінець (приєднання) одночасних дій;
- чорне коло означає початок (початковий вузол) робочого процесу;
- обведене чорне коло представляє кінець (кінцевий вузол).

Стрілки йдуть від початку до кінця і являють собою порядок дій.

Діаграми діяльності можна розглядати як форму структурованої блок-схеми у поєднанні з традиційною схемою потоку даних. Типовим методам блок-схеми не вистачає конструкцій для вираження паралельності. Тим не менш, символи об'єднання та розбиття у діаграмах діяльності дозволяють це лише для простих випадків; сенс моделі незрозумілий, коли вони довільно поєднуються з рішеннями або циклами[16][1)].

[1)] [16] Activity diagram – Wikipedia. URL: https://en.wikipedia.org/wiki/Activity_diagram (дата звернення 07.06.19).

На рис. 2.2 зображено діаграма діяльності для ПП.

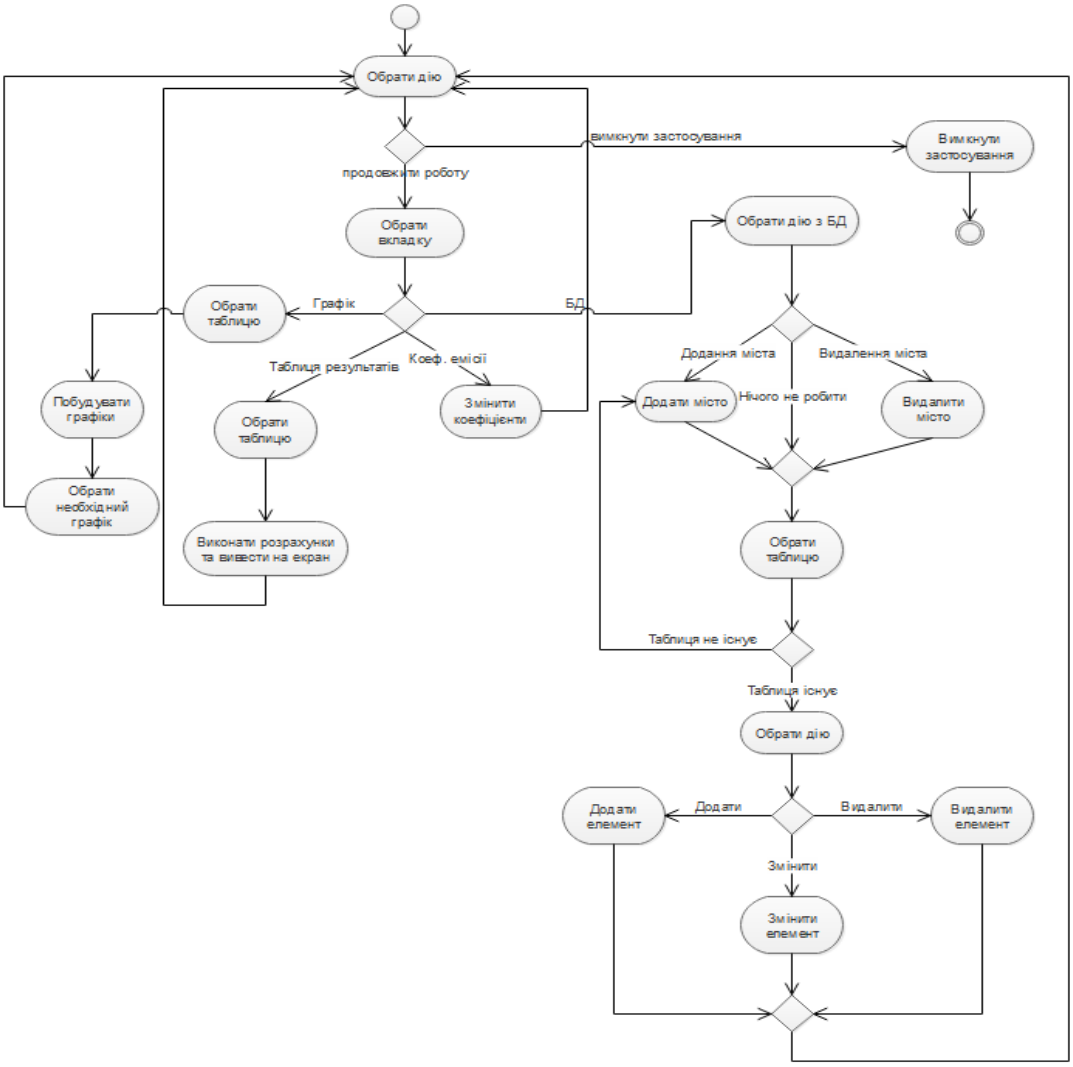

Рисунок 2.2 – UML діаграма діяльності

На діаграмі зображено послідовність дій, які застосування виконую по проханню користувача, та рішень які користувач може приймати.

Наприклад після початку роботи застосування користувач може почати якусь послідовність дій, або просто припинити роботу застосування.

Після прийняття рішення на продовження роботи застосування перед користувачем стоїть задача обрати вкладку:

– БД;

- графік;
- таблиця результатів;
- коефіцієнти емісії.

Якщо користувач вибере вкладку БД, то наступною дією буде вибрати дію з БД.

Користувач може додати нове місто, що створить нову таблицю, видалити існуюче місто із БД, або працювати з існуючими таблицями.

Далі користувач повинен обрати таблицю з якою буде працювати, якщо обраної таблиці не існує то користувач повертається на дію створення таблиці, інакше користувачу потрібно обрати дію з таблицею. Серед цих дій:

- додати новий елемент у таблицю;
- змінити існуючий елемент;
- видалити існуючий елемент.

Вкладка графік дає користувачу обрати таблицю вхідних даних, після чого будує графіки і дає змогу користувачу обрати вкладку необхідного йому графіку.

Після вибору вкладки таблиця результатів користувач обирає таблицю вхідних даних, після чого застосування виконує розрахунки.

При виборі вкладки коефіцієнти емісії – користувачу надається змога змінити необхідні йому коефіцієнти емісії.

Після завершення кожної дії, окрім припинення роботи, застосування повертаються до першого прийняття рішення користувачем.

2.4 Проектування зовнішнього вигляду ПП

У ПП буде декілька вкладок для роботи з БД, обробки даних та відображення графіків.

При запуску застосування буде відображатися вкладка для роботи з БД (див. рис. 2.3). На цій вкладці користувач зможе створювати або видаляти таблиці з інформацією про різні міста. Або працювати з цими таблицями, а саме:

додавати, видаляти та змінювати окремі елементи у таблиці. Також на вкладці буде відображена обрана на даний час таблиця.

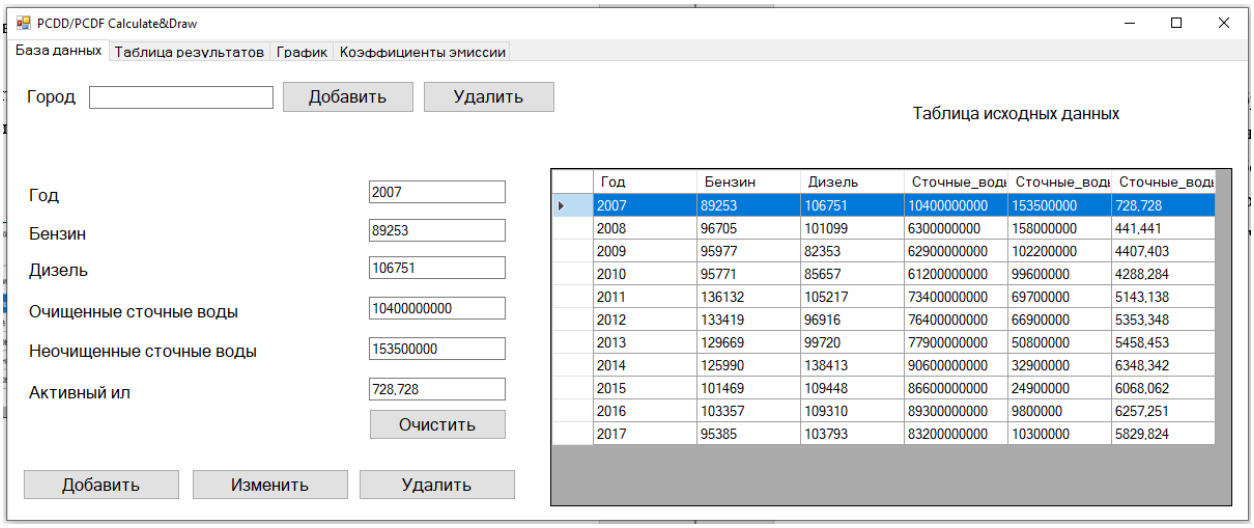

Рисунок 2.3 – Вкладка для роботи з БД

На другій вкладці буде відображена таблиця з результатами розрахунків (див. рис. 2.4).

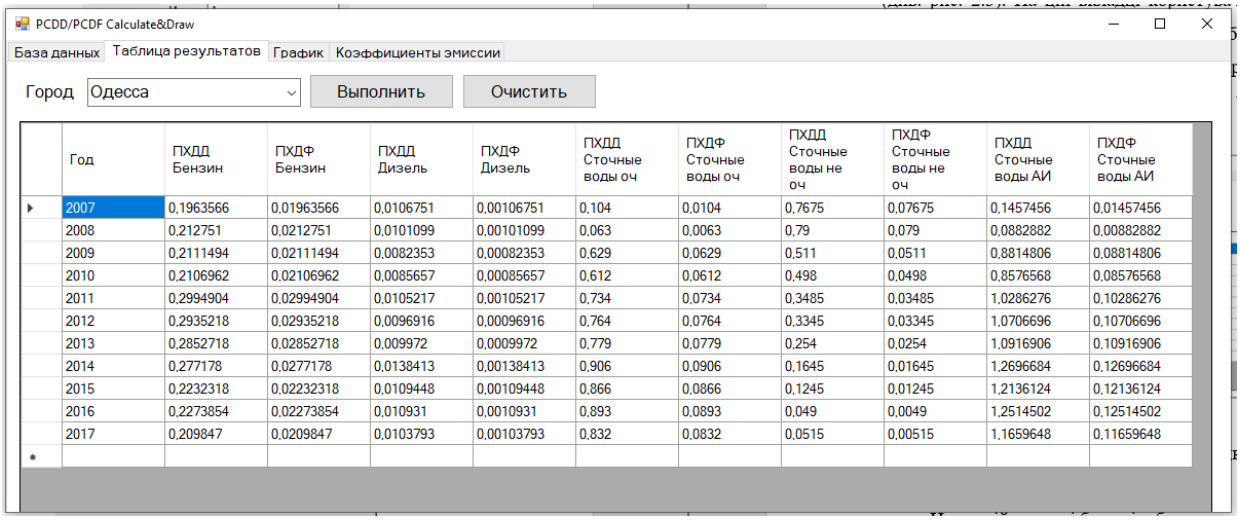

Рисунок 2.4 – Вкладка з таблицею результатів

Дані які будуть відображатися на вкладці таблиця результатів, є результатами розрахунку за допомогою спеціальних коефіцієнтів. Коефіцієнти встановлені за замовчуванням, але за необхідності їх можна змінити на вкладці під назвою коефіцієнти емісії (див. рис. 2.5)

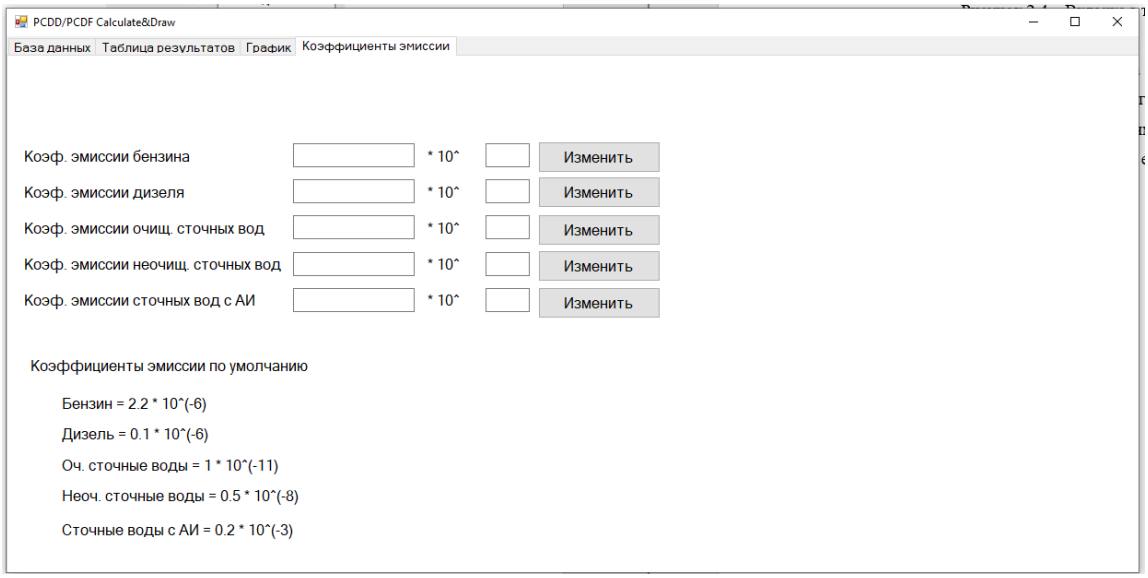

Рисунок 2.5 – Зовнішній вигляд вкладки «Коефіцієнти емісії»

Також для відображення графіка буде вкладка графік (див. рис. 2.6).

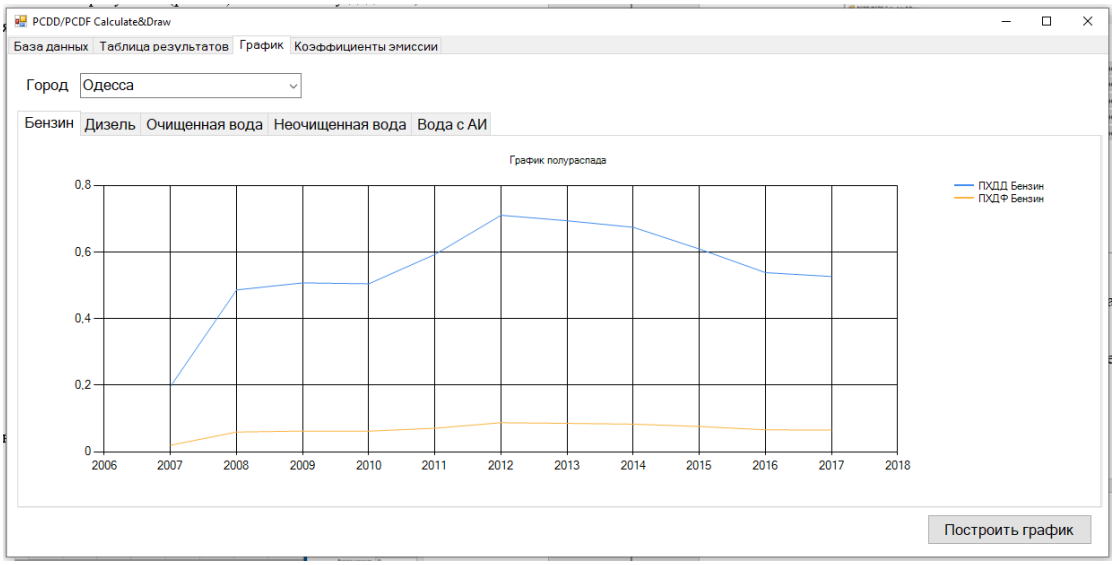

Рисунок 2.6 – вигляд графіку у вкладці «Графік»

На цій вкладці будуть відображені графіки напіврозпаду ПХДД та ПХДФ протягом років, по обраному місту. Також для графіків існують декілька вкладок, для різних джерел забруднення.

3 ОПИС ОКРЕМИХ ПРОГРАМНИХ ЕЛЕМЕНТІВ РОЗРОБКИ

3.1 Розробка інструментів «спілкування» з БД ПП

Для розробки ПП ми будемо використовувати бібліотеки System.Data.SQLite, вона має усі необхідні методи для вирішення питань, яка можливо виникнуть під час розробки методів «спілкування» з БД.

Для отримання даних з БД потрібно вирішити декілька питань, такі як:

- отримання уявлення таблиці БД;
- отримання значень з певного осередку таблиці;
- отримання значень з певного стовпця;
- виконання процедур.

Усі методи та змінні мають модифікатори private і static тому як усе знаходиться у класі Form1 тому, як у нас немає інших класів яким потрібен доступ до методів та змінних.

Для вирішення цих питань оголошено декілька змінних.

```
private SQLiteDataReader db_reader;
private SQLiteConnection db_con;
private SQLiteCommand db_cmd;
private SQLiteDataAdapter DB;
private DataSet DS = new DataSet();
private DataTable DT = new DataTable();
```

Далі зображено метод який організовує з’єднання з БД

```
public void SetConnection()
        {
            db_con = new SQLiteConnection("Data
Source=database.db;Version=3;New=False;Compress=True;");
        }
```

Після виконання методу для підключення та відключення до БД виконуються команди.

```
db_con.Open();
db_con.Close();
```

Наведені вище команди виконують підключення до БД для отримання необхідних даних, і відключення від БД по завершенню.

Також перед реалізацією будь-яких дій з БД реалізовано метод для виконання SQL-запитів.

```
private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }
```

Метод ExecuteQuery() приймає на вхід текст SQL-запиту, потім встановлює з’єднання з БД за допосогою методу SetConnection(), який було зображено раніше та виконує команди для створення SQL-запиту. По завершені відключається від БД.

Наступним реалізовано метод який буде отримувати таблицю з вхідними даними та вносить їх у таблицю. Назва таблиці буде обиратися користувачем за допомогою випадаючого списку.

```
private void LoadData(string city)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            string CommandText = "Select * from "+city+"";
            DB = new SQLiteDataAdapter(CommandText, db_con);
            DS.Reset();
            DB.Fill(DS);
            DT = DS.Tables[0];
            dtGridInfo.DataSource = DT;
            db_con.Close();
        }
```

Даний метод приймає на вхід назву міста та створює SQL-запит на отримання даних з БД. За допомогою SQLiteDataAdapter та об’єкту DataSet з типом DataTable ми отримуємо дані та заносимо їх в таблицю dtGridInfo.

Наступний метод заповнює випадаючий список назвами міст, назви яких є назвами таблиць які будуть виводитися у dataGridView. Інакше кажучи отримуються вхідні параметри для попереднього методу.

```
private void combofill()
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            string command_text = "SELECT * from Города ;";
            db_cmd.CommandText = command_text;
            db_reader = db_cmd.ExecuteReader();
            while (db_reader.Read())
            {
                string sName = db_reader.GetString(1);
                comBoxCityShow.Items.Add(sName);
                comboBoxCityCalc.Items.Add(sName);
                comBoxGraph.Items.Add(sName);
            }
            db_con.Close();
        }
```

Метод, який наведено вище, після встановлення з’єднання з БД використовує клас SQLiteDataReader для того, щоб отримати стовпець із таблиці з назвами міст, і заносить їх у випадаючі списки у всіх вкладках застосування.

Нижче наведені методи роботи з таблицями у БД.

Наступні методи – по натисненню на кнопку виконують створення таблиці міста та додає новий запис у таблицю з назвами міст.

```
private void btnAddCity_Click(object sender, EventArgs e)
        {
             if (cityAdd.Text != "")
             {
                string addcity = "insert into
Города(Название)values('" + cityAdd.Text + "')";
                string addtable = "CREATE TABLE " + cityAdd.Text
+ " (Год INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, Бензин DOU-
BLE, Дизель DOUBLE, Сточные_воды_оч DOUBLE, Сточные_воды_н_оч
DOUBLE, Сточные_воды_АИ DOUBLE); ";
                ExecuteQuery(addcity);
                ExecuteQuery(addtable);
                comBoxCityShow.Items.Add(cityAdd.Text);
                comboBoxCityCalc.Items.Add(cityAdd.Text);
                comBoxGraph.Items.Add(cityAdd.Text);
             }
        }
```

Показаний вище метод по натисненні на кнопку перевіряє чи задана назва нового міста користувачем. Якщо назва задана, то метод починає роботу, а саме:

- додає до таблиці з назвами міст нове місто;
- створює таблицю нового міста;
- додає нове місто до всіх випадаючих списків у застосуванні.

Далі реалізовано метод для видалення таблиці за обраною назвою з випадаючого списку та видалення запису з назвою міста із таблиці назв.

```
private void btn_delete_city_Click(object sender, EventArgs e)
        {
            string cityName = comBoxCityShow.Text;
            string rmvcity = "DELETE FROM 'Города' WHERE Назва-
ние = '"+ cityName + "';";
            string rmvtable = "DROP TABLE IF EXISTS " + cityName
+ ";";
            ExecuteQuery(rmvcity);
            ExecuteQuery(rmvtable);
            comBoxCityShow.Items.Remove(cityName);
            comboBoxCityCalc.Items.Remove(cityName);
            comBoxGraph.Items.Remove(cityName);
        }
```

Виконання методу схоже на метод додавання, але замість запису назви міста користувачем, необхідна назва обирається з випадаючого списку, що нейтралізує помилку користувачем при введені назви самостійно. Після цього, коли користувач натисне на кнопку метод видалить запис з назвою міста з таблиці міст, і видалить таблицю самого міста, якщо вона існує, а потім видалить це місто з всіх випадаючих списків у застосуванні.

Далі зображено реалізацію дій з даними з таблиці міста, до яких користувач буде мати доступ:

- додавати дані до таблиці;
- видаляти дані з таблиці;
- змінювати існуючі дані у таблиці.

Для цього використані елементи типу Button, у коді яких реалізовані наступні три методи.

Метод додання:

```
private void btnAdd_Click(object sender, EventArgs e)
        {
            string add = "insert into "+ comBoxCityShow.Text +
"(Год,Бензин,Дизель,Сточные_воды_оч,Сточные_воды_н_оч,Сточ-
ные_воды_АИ) values ('"+txtYear.Text+ "','"+txtFuel.Text+"', '"+
txtDiesel.Text + "', '"+ txtClWater.Text + "', '"+
txtDWater.Text + "', '"+ txtActive.Text + "');";
            ExecuteQuery(add);
            LoadData(comBoxCityShow.Text);
        }
```

Метод видалення:

```
private void btnDelete_Click(object sender, EventArgs e)
        {
            string txtQuery = "DELETE FROM " +
comBoxCityShow.Text + " WHERE Год = '" + txtYear.Text+"';";
            ExecuteQuery(txtQuery);
            LoadData(comBoxCityShow.Text);
        }
```

Метод для зміни:

```
private void btnEdit_Click(object sender, EventArgs e)
        {
            string txtQuery = "UPDATE "+comBoxCityShow.Text+ "
SET Бензин = '" + txtFuel.Text + "',Дизель = '" + txtDiesel.Text
+ "',Сточные_воды_оч = '" + txtClWater.Text + "',Сточ-
ные_воды_н_оч = '" + txtDWater.Text + "',Сточные_воды_АИ = '" +
txtActive.Text + "' WHERE Год = '"+txtYear.Text+"';";
            ExecuteQuery(txtQuery);
            LoadData(comBoxCityShow.Text);
        }
```

Також для більшої зручності користувача, йому надано змогу по натисненню на комірку в dataGridView – відповідний рядок для введення буде автоматично заповнений даними з натиснутої комірки. Реалізується це за допомогою методу dtGridInfo_CellClick().

Приклад команд у методі:

```
txtYear.Text =
dtGridInfo.SelectedRows[0].Cells[0].Value.ToString();
```

```
txtFuel.Text =
dtGridInfo.SelectedRows[0].Cells[1].Value.ToString();
```

Тобто для значення тексту в елементі textbox для рядку введення присвоюється значення вибраної комірки.

3.2 Розробка інструментів розрахунку по вхідним даним

Для розробки методів розрахунку нам потрібні методи класу Math. Ще для перед початком розрахунку потрібно оголосити глобальні змінні, які будуть використані для обчислення коефіцієнту емісії. А також потрібно оголосити коефіцієнт перетворення ПХДД в ПХДФ, який завжди дорівнює 0,1.

```
private double fuel_pow = Math.Pow(10, (-6));
private double fuel_emis = 2.2;

private double diesel_pow = Math.Pow(10, (-6));
private double diesel_emis = 0.1;

private double clwater_pow = Math.Pow(10, (-11));
private double clwater_emis = 1;

private double dwater_pow = Math.Pow(10, (-8));
private double dwater_emis = 0.5;

private double actwater_pow = Math.Pow(10, (-3));
private double actwater_emis = 0.2;

private double diox_to_furan = 0.1;
```

Коефіцієнти показані вище були надані кафедрою екології та охорони довкілля Одеського державного екологічного університету. За необхідності ці коефіцієнти можна змінити, для цього розроблено п’ять методів.

Приклад методу:

```
private void btn_fuelcoef_Click(object sender, EventArgs e)
        {
            fuel_pow = Math.Pow(10,
(Convert.ToInt32(txtpow_fuel.Text)));
            fuel_emis = Convert.ToDouble(txt_fuel_koef.Text);
        }
```

Метод показаний вище буде, по натисненню на кнопку, привласнювати значенням коефіцієнта значення вказані у рядку введення.

Для розрахунку емісії будуть використовуватися методи реалізовані нижче.

Як і в інших методах встановлюється з'єднання з БД, та отримання вхідних даних для проведення розрахунку. Для цього за допомогою класу SQLiteDataReader отримати дані з кожного стовпця таблиці, та занести їх у список. Зробити це можна за допомогою цикла while:

Приклад:

```
while (db_reader.Read())
            {
                int yearnum = db_reader.GetInt32(0);
                year_list.Add(yearnum);

                double fuel = db_reader.GetDouble(1);
                fuel_quant.Add(fuel);


            }
```

Як показано на прикладі while буде працювати до тих пір, поки в таблиці є дані.

Потім для більш зручної реалізації розрахунку отримані списки перетворюються у масиви. Це реалізовано за допомогою методу класу System.Collections та методу ToArray() який перетворює список з будь-яких елементів на відповідний йому масив.

Для проведення розрахунку, який виконуватиметься у циклі for. А саме множення кожного елементу масиву на коефіцієнт емісії, та занесення отриманого числа до заздалегідь оголошеного масиву для ПХДД. Та множення масиву ПХДД на коефіцієнт перетворення для отримання масиву ПХДФ, і внесення цих масивів у таблицю DataGridView.

Приклад:

```
for (int i = 0; i < fuel_array.Length; i++)
```

```
        {
                fueldiox_arr[i] = fuel_array[i];
                fueldiox_arr[i] *= fuel_k;
                fuelfuran_arr[i] = fueldiox_arr[i] *
diox_to_furan;

                dieseldiox_arr[i] = diesel_array[i];
                dieseldiox_arr[i] *= diesel_k;
                dieselfuran_arr[i] = dieseldiox_arr[i] *
diox_to_furan;

                dtGridCalc.Rows.Add(year_array[i].ToString(),
fueldiox_arr[i].ToString(), fuelfuran_arr[i].ToString(),
dieseldiox_arr[i].ToString(), dieselfuran_arr[i].ToString(),
waterdiox_arr[i].ToString(), waterfuran_arr[i].ToString(),
dwaterdiox_arr[i].ToString(), dwaterfuran_arr[i].ToString(),
actwaterdiox_arr[i].ToString(),
actwaterfuran_arr[i].ToString());
            }
```

3.3 Технічні рішення при розробці елементів відображання

Головний елемент відображення складається з:

- таблиця відображення(DataGridView);
- кнопки додавання(Button);
- кнопки видалення(Button);
- кнопки зміни(Button);
- вкладки(tabPage);
- випадаючого списку(combobox);
- графіку(Chart).

Для побудови графіку буде використовуватися метод аналогічний методу розрахунку.

Це означає що при побудові графіка всі результати будуть розраховуватися знову, це не дуже оптимальне рішення але швидкодія C# дозволяє таке виконання.

Окрім основних розрахунків перед побудовою графіку розраховується напіврозпад речовин протягом років. Напіврозпад розраховується по закону, однаковому для всіх речовин.

Розрахунок коефіцієнтів:

```
double tau_fuel_diox = air / (Math.Log((fueldiox_arr[0]) / 2));
double tau_fuel_furan = air / (Math.Log((fuelfuran_arr[0]) /
2));
double tau_diesel_diox = air / (Math.Log((dieseldiox_arr[0]) /
2));

double fuel_coef_diox = (-1) / tau_fuel_diox;
double fuel_coef_furan = (-1) / tau_fuel_furan;
```

Розрахунок точок графіка:

```
double[] graph_fuel_diox = new double[fueldiox_arr.Length];
double[] graph_fuel_furan = new double[fuelfuran_arr.Length];
graph_fuel_diox[0] = fueldiox_arr[0];
graph_fuel_furan[0] = fuelfuran_arr[0];
```

Далі зображена реалізація побудови самого графіка.

Приклад:

```
for (int i = 1; i < years_count; i++)
            {
                graph_fuel_diox[i] = (fueldiox_arr[i - 1] *
Math.Exp(fuel_coef_diox) + fueldiox_arr[i]);
                graph_fuel_furan[i] = (fuelfuran_arr[i - 1] *
Math.Exp(fuel_coef_furan) + fuelfuran_arr[i]);
            }


            for (int i = 0; i < years_count; i++)
            {
                chart_fuel.Series[0].Points.AddXY(year_array[i],
graph_fuel_diox[i]);
                chart_fuel.Series[1].Points.AddXY(year_array[i],
graph_fuel_furan[i]);
            }
```

Так як зовнішній вигляд однаковий для усіх таблиць БД, то при створенні елементу вказується, які дані у таблицю завантажувати і які дії будуть виконуватися при натисканні кнопок.

Тому за допомогою методу:

```
private void LoadData(string city)
        {
```

```
        SetConnection();
        db_con.Open();
        db_cmd = db_con.CreateCommand();
        string CommandText = "Select * from "+city+"";
        DB = new SQLiteDataAdapter(CommandText, db_con);
        DS.Reset();
        DB.Fill(DS);
        DT = DS.Tables[0];
        dtGridInfo.DataSource = DT;
        db_con.Close();
    }
```

При виконанні кожної дії таблиця DataGridView оновлюється.

4 РОЗРОБКА ІНСТРУКЦІЇ КОРИСТУВАЧА

4.1 Комплектація застосування

Фінальне застосування є дуже вузькоспеціалізованим, то виникає необхідність навчити користувача його використовувати.

Спочатку сама комплектація застосування. Так як для роботи програми необхідні бібліотеки для роботи з БД. Ще на комп'ютері користувача повинен знаходитися .NET Framework не раніше версії 3.0.

Отже для розповсюдження застосування необхідно використовувати архів. Файли які знаходяться всередині зображені на рис. 4.1-4.2.

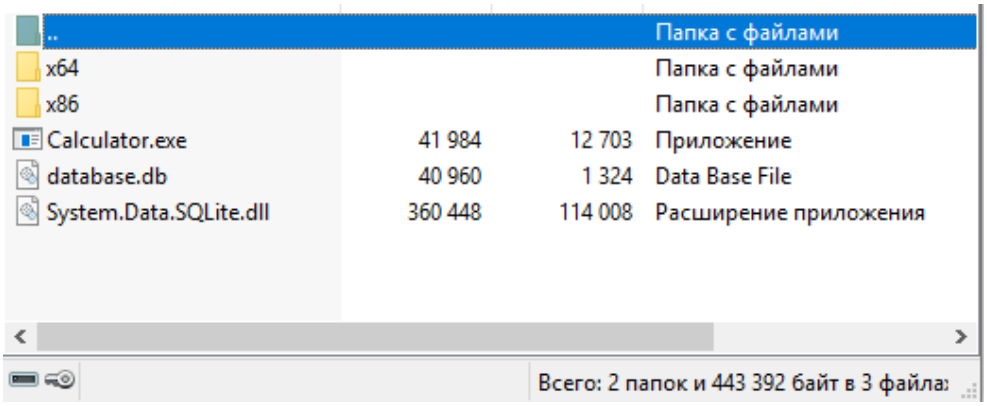

Рисунок 4.1 – Зміст архіву застосування

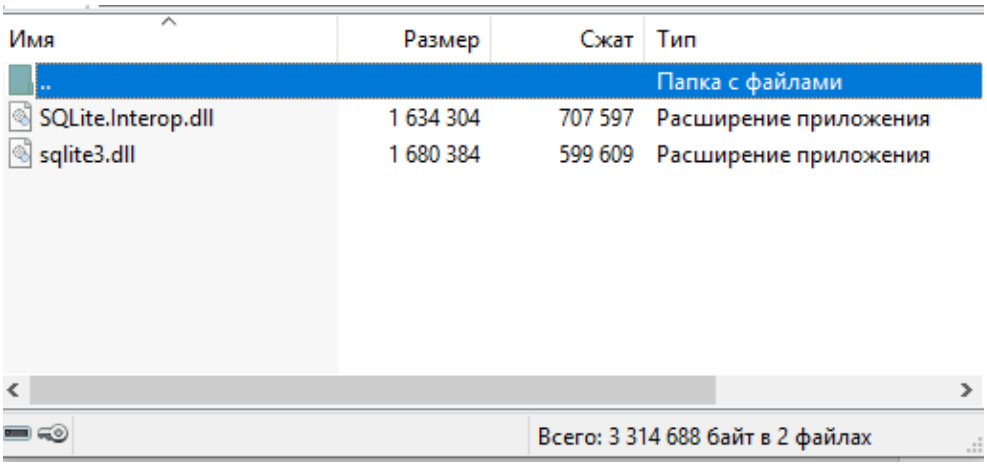

Рисунок 4.2 – Зміст папок x64 та x86

4.2 Інструкція по використанню застосування

Коли користувач запускає виконавчий файл програми, то першою перед ним завантажується вкладка «База данных», де користувач виконує роботу з БД. Користувач може створити таблицю з назвою нового міста, ввівши назву в поле «Город» та натиснувши на кнопку «Добавить» (див. рис. 4.3).

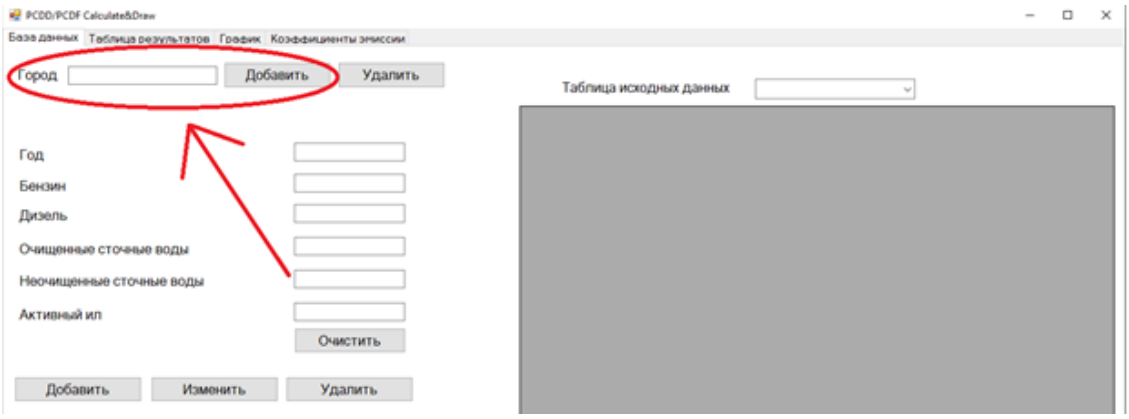

Рисунок 4.3 – Додати нове місто до БД

Або користувач може обрати вже існуюче місто з БД або видалити його (див. рис. 4.4). Для цього користувачу потрібно обрати місто з випадаючого списку, та натиснути кнопку «Удалить», при бажані видалити.

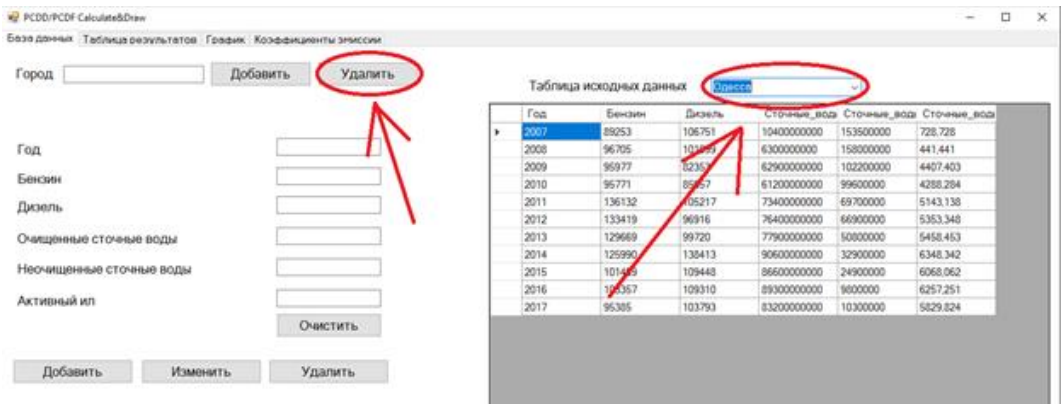

Рисунок 4.4 – Можливість обрати таблицю та видалити

Після того як таблиця була обрана користувачем він може проводити з таблицею операції додання, видалення та зміни елементу в таблиці (див. рис 4.4). Також користувачу надається змога по натисненню на комірки таблиці вносити дані з комірок у відповідні поля.

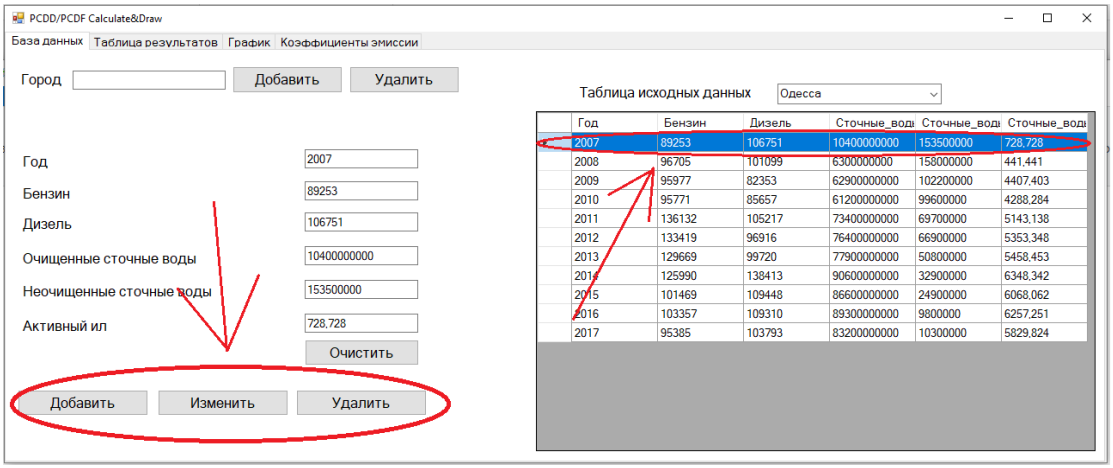

Рисунок 4.4 – Операції з таблицею

Після введення всіх необхідних даних до таблиці користувач повинен перейти до вкладки «Таблица результатов» (див. рис. 4.5), де після вибору таблиці вхідних даних, та натиснення на кнопку «Выполнить» виконується розрахунок ПХДД та ПХДФ.

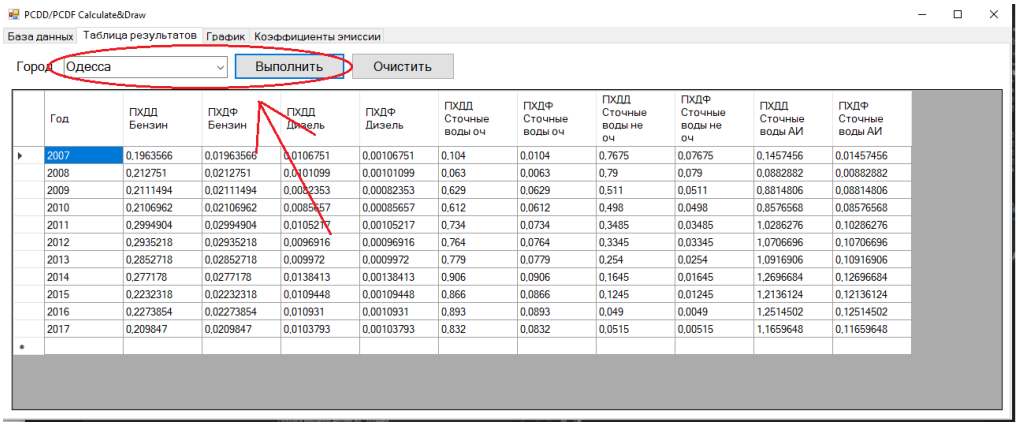

Рисунок 4.5 – Розрахунок ПХДД та ПХДФ

Коефіцієнти, по яким виконуються розрахунки на вкладці «Таблица результатов», задаються користувачем на вкладці «Коэффициенты эмиссии» (див. рис. 4.6). Коефіцієнти по перезапуску застосування скидаються до значень за замовченням. Значення за замовченням були надані кафедрою екології та охорони довкілля Одеського державного екологічного університету.

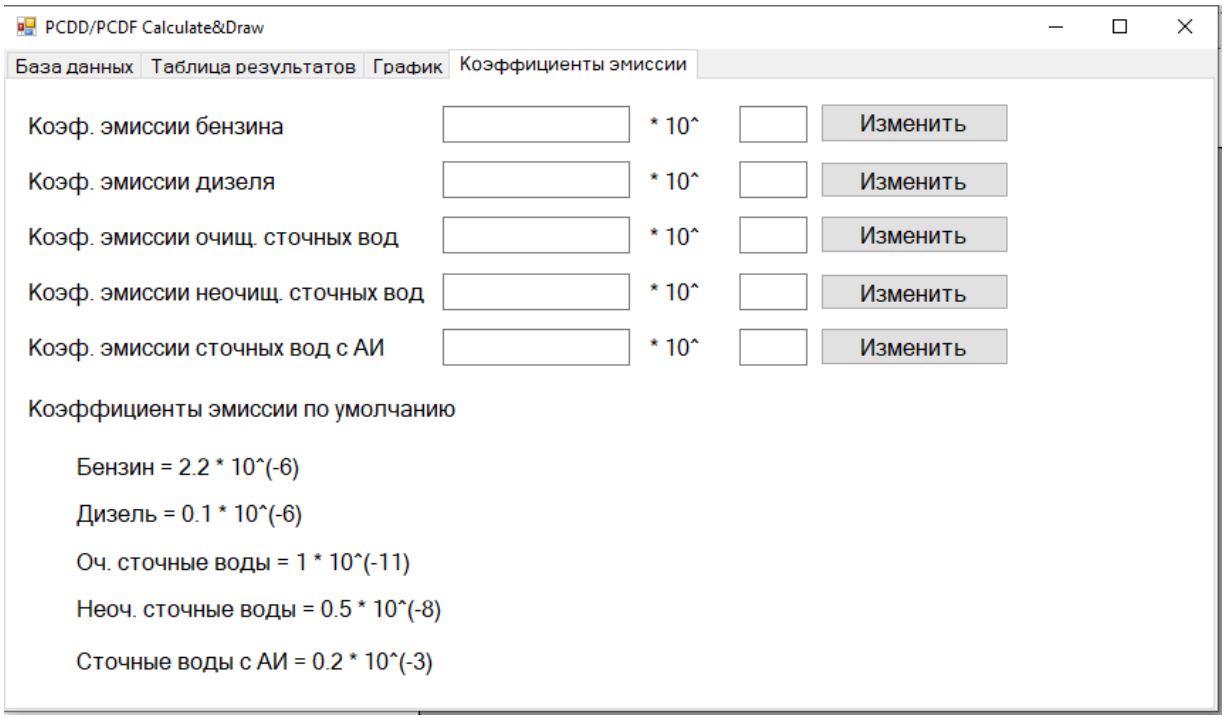

Рисунок 4.6 – Зміна коефіцієнтів

Також для користувача є можливість побачити графік напіврозпаду, при натисненні на кнопку «Построить график», ПХДД та ПХДФ (див. рис. 4.7), для різних типів забруднень упродовж років вказаних у таблиці. Після побудови графіків користувач може перемикатися між вкладками графіків, для спостереження за різними типами забруднень по обраному місту.

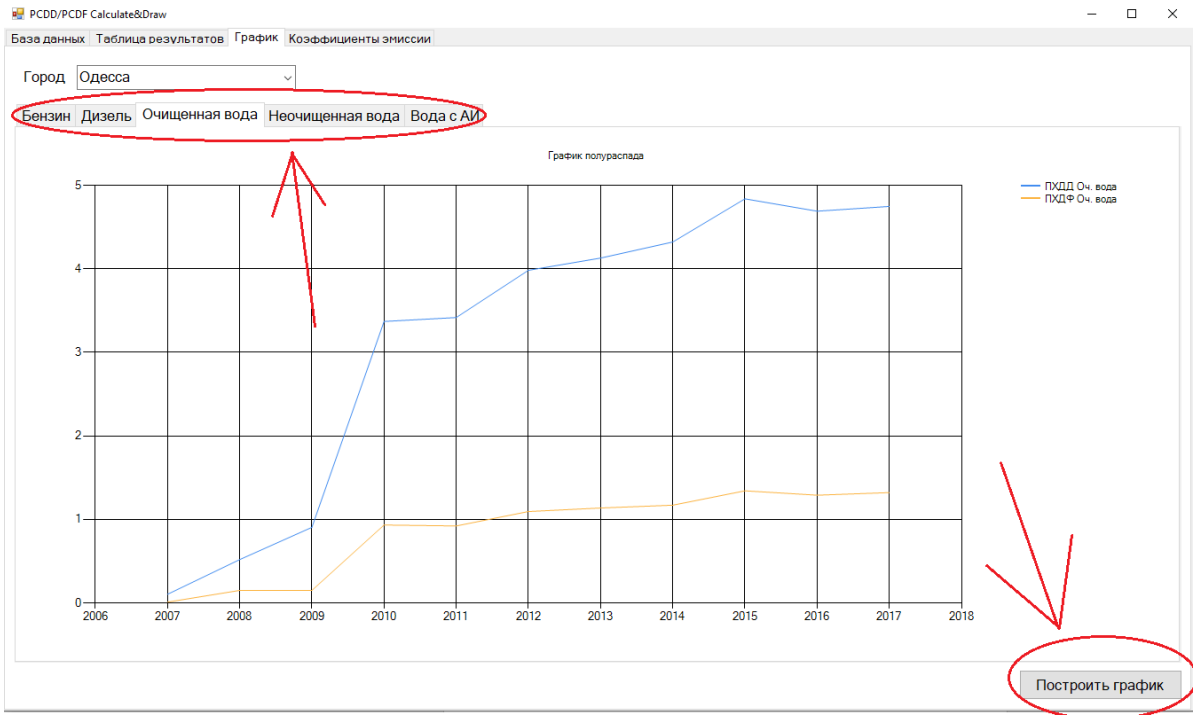

Рисунок 4.7 – Побудова графіків

ВИСНОВКИ

У результаті роботи, був виконаний аналіз предметної галузі. Були виявлені аналогічні ПП, проаналізовані їх недоліки і переваги.

Було проведено моделювання і обрані мова програмування та середовище розробки. З розглянутих середовищ розробки, було обрано Visual Studio і .Net.

Було виконане моделювання БД, логічна та фізична схема БД.

В результаті розробки, одержаний ПП, який зможе проводити облік ПХДД і ПХДФ, та обробляти дані по ним. А також допоможе вченим визначати рівень забруднення, та запобігти подальшому забрудненню навколишнього середовища.

ПЕРЕЛІК ДЖЕРЕЛ І ПОСИЛАНЬ

1. Солдатов А. И. Источники загрязнения среды обитания. Часть 1: Стойкие органические загрязнители: конспект лекций – Челябинск: Издательский центр ЮУрГУ, 2015. 157с.
2. Стойкие органические загрязнители: глобальная проблема, глобальное решение. URL: http://www.unido-russia.ru/archive/num_14/art14_5 (дата звернення 10.06.2019).
3. Ревич Б. А. Основные сведения о стойких органических загрязнителях (СОЗ).: [б. м.]
4. Dirty Dozen for Android – APK Download. URL: https://apkpure.com/dirty-dozen/com.ewg.dirtydozen(дата звернення 10.06.19).
5. Приложения в Google Play – Earth-Now. URL: https://play.google.com/store/apps/details?id=gov.nasa.jpl.earthnow.activity&hl=ru(дата звернення 10.06.19).
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 10.06.19).
7. Java (программная платформа) – Википедия. URL: https://ru.wikipedia.org/wiki/ Java_(программная_платформа)(дата звернення 06.06.19).
8. Общие сведения о платформе .NET Framework. URL: https://docs.microsoft.com /ru-ru/dotnet/framework/get-started/overview(дата звернення 06.06.19).
9. Microsoft SQL Server – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft _SQL_Server(дата звернення 06.06.19).
10. PostgreSQL – Вікіпедія. URL: https://uk.wikipedia.org/wiki/PostgreSQL(дата звернення 06.06.19).

11. SQLite – Вікіпедія. URL: https://uk.wikipedia.org/wiki/SQLite(дата звернення 06.06.19).
12. Eclipse – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Eclipse(дата звернення 06.06.19).
13. MonoDevelop – Wikipedia. URL: https://en.wikipedia.org/wiki/MonoDevelop (дата звернення 06.06.19).
14. Microsoft Visual Studio – Wikipedia. URL: https://en.wikipedia.org/wiki/ Microsoft_Visual_Studio(дата звернення 06.06.19).
15. Леоненков, самоучитель UML. URL: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl7/gl7.html (дата звернення 07.06.19)
16. Activity diagram – Wikipedia. URL: https://en.wikipedia.org/wiki/ Activity_diagram (дата звернення 07.06.19).