МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Олексій Горський
“____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
МАГІСТР

Тема: “Інформаційна технологія моніторингу забруднення атмосферного повітря”

Виконавець: Стриков Максим Олександрович

Керівник: к.т.н. доцент Семко Олексій Вікторович

Нормоконтролер: ст.викл. Гололобов Дмитро Олександрович

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Олексій Горський
"___" _____ 2023 р

ЗАВДАННЯ
на виконання дипломної роботи студента
Стрикова Максима Олександровича

1. Тема кваліфікаційної роботи: “Інформаційна технологія моніторингу забруднення атмосферного повітря” затверджена наказом ректора від 29.09.2023 № 1994/ст.

2. Термін виконання проекту: з 02.10.2023 р. до 31.12.2023 р.

3. Вихідні дані до роботи: програмний продукт розробити мовою JAVA з використанням фреймворків Spring, Maven.

4. Зміст пояснювальної записки:
1. Аналіз існуючих підходів до реалізації збору та аналізу даних для роботи системи моніторингу забруднення атмосферного повітря
2. Вимоги до інформаційної системи моніторингу забруднення атмосферного повітря
3. Проектування системи моніторингу забруднення атмосферного повітря
4. Реалізація системи моніторингу забруднення атмосферного повітря

5. Перелік обов'язкових слайдів презентації:
1. Опис предметної області.
2. Вимоги до проекту.
3. Архітектура проекту.
4. Діаграма класів.
5. Прототип проекту.

6. Календарний план-графік:

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Розробка та затвердження графіка роботи.	02.10.23 - 08.10.23	
2.	Підготовка та написання 1 розділу. Відсилка керівнику	09.10.23 - 20.10.23	
3.	Підготовка та написання 2 розділу. Відсилка керівнику	21.10.23 - 05.11.23	
4.	Підготовка та написання 3 розділу. Відсилка керівнику	06.11.23 - 19.11.23	
5.	Підготовка та написання 4 розділу. Відсилка керівнику	20.11.23 - 30.11.23	
6.	Редагування та друк пояснювальної записки, графічного матеріалу Відсилка ПЗ для перевірки на плагіат одним файлом.	01.12.23 - 14.12.23	
7.	Проходження нормо-контролю, перепліт пояснювальної записки. Отримання відгуку керівника. Підготовка презентації та тексту доповіді.	04.12.23 - 10.12.23	
8.	Передзахист (наявність віддрукованої ПЗ, презентації, положительного відгуку керівника). При наявності цього приймається рішення про допуск к захисту дипломного проекту перед Екзаменаційною комісією. Що оформлюється протоколом засідання кафедри	11.12.23 - 17.12.23	
9.	Отримання рецензії.	18.12.23 - 24.12.23	
10.	Здати секретарю ДЕК: ПЗ, ГМ, CD-R з електронними версіями ПЗ, ГМ, презентацію, відгук керівника, рецензію, довідку про успішність, 2 папки, 2 конверта)	22.12.2023	
11.	Захист дипломної роботи перед ЕК	25.12.23 - 31.12.23	

7. Дата видачі завдання: 02.10.2023 р.

Керівник дипломної роботи: к.т.н. доцент Олексій СЕМКО

Завдання прийняв до виконання: Максим СТРИКОВ

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Інформаційна технологія моніторингу забруднення атмосферного повітря»: 93 сторінки, 50 рисунків, 2 таблиці, 15 використаних джерел.

ЗАБРУДНЕННЯ ПОВІТРЯ, ПЗ, JAVA, ANDROID, SPRING, MAVEN, МОБІЛЬНИЙ ДОДАТОК, БАГАТОРІВНЕВА АРХІТЕКТУРА, БАЗА ДАНИХ.

Об’єкт дослідження – система моніторингу забруднення атмосферного повітря.

Предмет дослідження – є методи та інструменти моніторингу забруднення атмосферного повітря.

Мета дипломної роботи – є розробка програмного забезпечення з – сервісу моніторингу забруднення атмосферного повітря, аналіз предметної області та аналіз існуючих методів моніторингу забруднення атмосферного повітря.

В процесі роботи, були проведені роботи по аналізу сучасних підходів до побудови ПЗ, дослідженні можливості та перспективи використання технології моніторингу забруднення атмосферного повітря та проаналізовані методи та інструменти моніторингу за забрудненням повітря.

В результаті представлено програмне забезпечення – система моніторингу забруднення атмосферного повітря.

Результати роботи можуть бути використані для таких цілей, як:

- Управління екологією та захист довкілля

Допомагає владі та екологічним організаціям в реагуванні на забруднення повітря та вживанні заходів для його зменшення.

- Здоров’я населення

Надає інформацію громадянам про рівень забруднення повітря, допомагаючи їм уникати небезпеки та забезпечуючи своєчасне інформування про стан довкілля.

- Дослідження та наука

Забезпечує науковців та дослідників даними для аналізу трендів забруднення повітря та розробки стратегій зменшення впливу на довкілля.

- Підприємства та промисловість

Використовується для контролю та оптимізації викидів виробництв та промислових підприємств, забезпечуючи відповідність екологічним стандартам.

- Транспортна система

Використовується для моніторингу викидів транспортних засобів та розробки заходів для зменшення впливу транспорту на довкілля.

Розробка проводилась за допомогою та під управлінням ОС Windows. Средою розробки були IntelliJ IDEA та Android Studio, на мові програмування Java.

ABSTRACT

Explanatory note to the diploma thesis "Information technology of atmospheric air pollution monitoring": 93 pages, 50 figures, 2 table, 15 used sources.

AIR POLLUTION, SOFTWARE, JAVA, ANDROID, SPRING, MAVEN, MOBILE APP, MULTILEVEL ARCHITECTURE, DATABASE.

The object of the research is the atmospheric air pollution monitoring system.

The subject of research is methods and tools for monitoring atmospheric air pollution.

The purpose of the thesis is to develop software for air pollution monitoring service, analysis of the subject area and analysis of existing methods of monitoring atmospheric air pollution.

In the course of the work, work was carried out on the analysis of modern approaches to the construction of software, the possibilities and prospects of using air pollution monitoring technology were studied, and the methods and tools of air pollution monitoring were analyzed.

As a result, software is presented - an air pollution monitoring service.

The results of the work can be used for such purposes as:

• Ecological management and environmental protection

Assists authorities and environmental organizations in responding to air pollution and taking measures to reduce it.

• Public health

Provides information to citizens about the level of air pollution, helping them avoid danger and providing timely information on the state of the environment.

• Research and science

Provides scientists and researchers with data to analyze air pollution trends and develop strategies to reduce environmental impact.

• Enterprises and industry

It is used to control and optimize emissions of production and industrial enterprises, ensuring compliance with environmental standards.

• Transport system

It is used to monitor vehicle emissions and develop measures to reduce the impact of transport on the environment.

The testing was carried out under the control of Windows OS. The development environment is IntelliJ IDEA and Android Studio, based on Java programming.

Зміст

ВСТУП 12

РОЗДІЛ 1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО РЕАЛІЗАЦІЇ ЗБОРУ ТА АНАЛІЗУ ДАНИХ ДЛЯ РОБОТИ СИСТЕМИ МОНІТОРИНГУ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ 14

1.1. Аналіз предметної області технології моніторингу забруднення атмосферного повітря 14

1.2. Загальний огляд системи моніторингу забруднення атмосферного повітря 18

1.3. Огляд існуючих систем моніторингу повітря 19

1.4. Технології та методи, що використовуються в існуючих системах моніторингу 22

Висновки до розділу 24

РОЗДІЛ 2 ВИМОГИ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ 26

2.1. Цілі створення системи 26

2.2. Вимоги користувача та функціональні вимоги 27

2.3. Бізнес вимоги 28

2.4. Нефункціональні вимоги 30

2.5. Системні вимоги 31

Висновки до розділу 32

РОЗДІЛ 3 ПРОЕКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ 33

3.1. Архітектурна концепція 33

3.2. Інтеграція та взаємодія з датчиками моніторингу забруднення атмосферного повітря 40

3.3. Проектування бази даних 41

Висновки до розділу 44

РОЗДІЛ 4 РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ 45

4.1. 4545

4.2. 5151

4.3. 8787

Висновки до розділу 89

ВИСНОВКИ 91

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 92

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IoT	–	мережа інтернет речей;
БД	–	база даних;
ГІС	–	геоінформаці́йна систе́ма
RAM	–	random access memory
GUI	–	graphic user interface;
HTTP	–	hypertext transfer protocol;
TCP	–	transmission control protocol;
IP	–	internet protocol;
UML	–	unified modeling language;
ПБД	–	проектування бази даних;
ER	–	мережа інтернет речей;
JVM	–	java virtual machine;
API	–	application programming interface;
VCS	–	version control system
СУБД	–	система управління базою даних;
ПК	–	мережа інтернет речей;
POM	–	project object model;
CI	–	continuous integration;
CD	–	continuous deployment;
XML	–	extensible markup language.

ВСТУП

Актуальність теми. Актуальність створення інформаційної системи моніторингу забруднення атмосферного повітря обумовлена зростаючим впливом людської діяльності на стан навколишнього середовища. Сучасний світ став свідком збільшення викидів різних забруднюючих речовин у атмосферу через промислові процеси, транспорт, енергетичні системи та інші джерела.

Інформаційна система моніторингу забруднення повітря може стати ефективним інструментом для відстеження та аналізу рівня забруднення в реальному часі. Вона дозволить збирати дані про викиди токсичних речовин, твердих часток, парникових газів та інших забрудників з різних джерел, таких як промислові підприємства, автотранспорт та сільське господарство.

Створення такої інформаційної системи підкреслює важливість екологічної освіченості та прагнення до сталого розвитку. Вона не лише допоможе уникнути негативних наслідків забруднення атмосфери, але й сприятиме формуванню відповідального ставлення до навколишнього середовища та забезпеченню здоров'я майбутніх поколінь.

Мета і завдання дослідження. Дослідження та розробка інформаційної системи моніторингу забруднення атмосферного повітря.

Для досягнення поставленої цілі, необхідно розв’язати наступні задачі:

- Дослідити сучасні системи моніторингу забруднення атмосферного повітря.
- Проаналізувати методи моніторингу забруднення атмосферного повітря.
- Реалізувати програмне забезпечення з метою покращення моніторингу забруднення атмосферного повітря.

Об’єкт дослідження – система моніторингу забруднення атмосферного повітря.

Предметом дослідження є методи та інструменти моніторингу забруднення атмосферного повітря.

Методи дослідження, для виконання задачі кваліфікаційної роботи використано наступні методи: емпіричний, аналізу, проектування комп’ютерних систем.

Практична цінність. Розроблений програмний додаток можна застосовувати для моніторингу за забрудненням атмосферного повітря.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО РЕАЛІЗАЦІЇ ЗБОРУ ТА АНАЛІЗУ ДАНИХ ДЛЯ РОБОТИ СИСТЕМИ МОНІТОРИНГУ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ

1.1. Аналіз предметної області технології моніторингу забруднення атмосферного повітря

Технології моніторингу забруднення атмосферного повітря є важливим інструментом для визначення та управління якістю повітря в містах та промислових регіонах. Забруднення атмосфери стало актуальною проблемою в середині XX століття внаслідок індустріального розвитку та економічного зростання.

Перші вказівки на необхідність вивчення якості повітря в атмосфері з'явилися в 1950-1960-х роках. Шервуд Роуман (Sherwood Rowland), Луїс Моліна (Mario Molina) та Роланд Карлсон (Frank Sherwood Rowland) вивчали вплив хімічних речовин на озоновий шар, і їхні дослідження дали початок розумінню проблеми забруднення атмосфери. Вони отримали Нобелівську премію з хімії в 1995 році за внесок у цю область. Відзначено було, зокрема, вплив хімічних сполук, таких як хлорфторвуглеводи (CFC), на озоновий шар та викиди газів, які впливають на якість повітря.

На сучасному етапі, світ демонструє підвищений рівень підсвідомості та занепокоєння стосовно забруднення атмосферного повітря. Перш за все, це пов'язано з великими міськими агломераціями, де індустріалізація, транспорт та інші джерела викидів спричиняють зростання рівня забруднення. Країни та міста ведуть активні дії для контролю та зменшення викидів, впроваджуючи технології моніторингу для визначення джерел забруднення та ефективного управління викидами.

Використання передових технологій моніторингу, таких як дрони, сенсори та системи Інтернету речей (IoT), дозволяє в реальному часі визначати рівні

забруднення, виявляти джерела викидів та ефективно реагувати на них. Організації та науковці активно співпрацюють для розробки нових методів та інструментів для боротьби з забрудненням атмосфери.

Загальна свідомість про екологічні проблеми, включаючи забруднення повітря, зростає. Глобальні ініціативи, такі як Паризька угода, зобов'язують країни до прийняття заходів для зменшення викидів та покращення якості повітря. Такий підхід свідчить про те, що проблема забруднення атмосфери стала не лише технічним аспектом, але й важливим завданням глобального екологічного розвитку

Забруднення повітря має серйозний вплив на здоров'я людей, призводячи до ряду захворювань та погіршення якості життя. В основному, шкідливі викиди атмосферних забруднюючих речовин, таких як оксиди азоту, сіркові сполуки, вуглеводні та тверді частки, можуть викликати захворювання верхніх та нижніх дихальних шляхів, серцево-судинні захворювання та інші системні проблеми.

Найчастіші захворювання, пов'язані з забрудненням повітря, включають астму, бронхіт, хронічну обструктивну хворобу легень (ХОЗЛ), алергічні реакції та інші респіраторні проблеми. Діти, літні люди та особи із передвісною схильністю до респіраторних захворювань є особливо вразливими перед негативними наслідками забрудненого повітря.

Країнами з найгіршим показником якості повітря являються країни востоку, а точніше Чад, Ірак, Пакистан і тд(більш детально на таблиці 1.1).

Таблиця 1.1.

Топ-10 найбільш забруднених країн світу

Рейтинг	**Країна**	**2022**	**2021**	**2020**	**2019**	**2018**	**Населення**
1	Chad	89.7	75.9	--	--	--	17,179,740
2	Iraq	80.1	49.7	--	39.6	--	43,533,592

Рейтинг	Країна	2022	2021	2020	2019	2018	Населення
3	Pakistan	70.9	66.8	59	65.8	74.3	231,402,117
4	Bahrain	66.6	49.8	39.7	46.8	59.8	1,463,265
5	Bangladesh	65.8	76.9	77.1	83.3	97.1	169,356,251
6	Burkina Faso	63	--	--	--	--	22,100,683
7	Kuwait	55.8	29.7	34	38.3	56	4,250,114
8	India	53.3	58.1	51.9	58.1	72.5	1,407,563,842
9	Egypt	46.5	29.1	--	18	--	109,262,178
10	Tajikistan	46	59.4	30.9	--	--	9,750,064

Для прикладу можно взяти тенденцію захворювань у Бангладеші. Бангладеш, будучи однією з найбільш заселеними країн світу, стикається з серйозними викликами, пов'язаними з якістю повітря та її впливом на здоров'я населення. Захворювання, спричинені забрудненням повітря, стали серйозною загрозою для громадського здоров'я та ведуть до високого рівня смертності в країні. За останні роки смертність в Бангладеш значно зросла через захворювання, які виникають в результаті забруднення повітря. Найбільш поширеними серед них є респіраторні захворювання, такі як бронхіт та астма, які атакують особливо вразливі групи

населення, зокрема дітей та літніх людей. Хронічна обструктивна хвороба легень (ХОЗЛ) також стала поширеним явищем, викликаючи довготривалі проблеми здоров'я(див.рис.1.1).

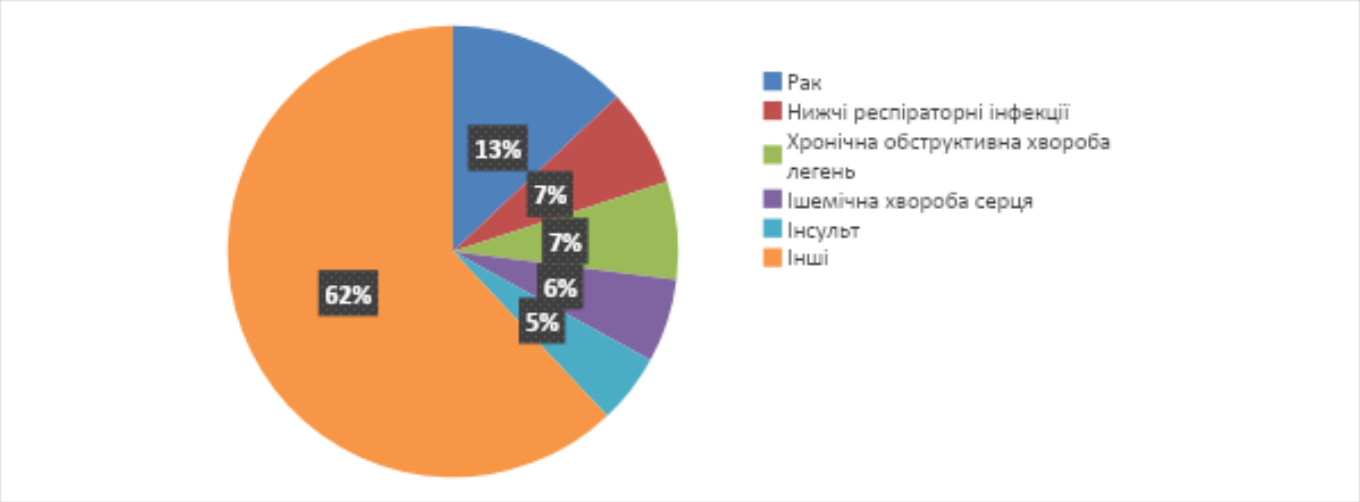


Рис. 1.1. Діаграма причин смертності в Бангладеші

Сучасна індустріалізація стала не лише джерелом економічного зростання, а й серйозним викликом для якості повітря, яке ми вдихаємо. Проблеми забруднення повітря змушують нас пристосовуватися до нової реальності, де індустріальні викиди та автомобільний рух стають причинами погіршення атмосферного середовища. Результатом є поширення різних захворювань, зокрема респіраторних та серцево-судинних, які впливають на загальний стан здоров'я населення.

На світовій мапі забруднення повітря(див рисунок 1.2), можна виявити регіони, де ситуація є особливо напруженою. Часто це мегаполіси та промислові центри, де інтенсивна діяльність призводить до небезпечного накопичення забруднюючих речовин у повітрі.

Це нове випробування для людства вимагає об'єднання всіх націй та громадян у глобальних зусиллях для подолання кризи. Забруднення повітря не знає кордонів, і його наслідки впливають на кожного. Саме зараз, коли ми конфронтовані з цією проблемою, важливо взяти на себе відповідальність та діяти наполегливо. Подолання цього виклику передбачає використання чистих технологій, зменшення викидів, та підтримку сталого способу життя.

Збереження природи та забезпечення чистоти повітря — це ключові завдання для нас усіх. Світова спільнота повинна співпрацювати, аби забезпечити майбутнє, де кожен може дихати свіжим повітрям та насолоджуватися здоровими умовами життя.

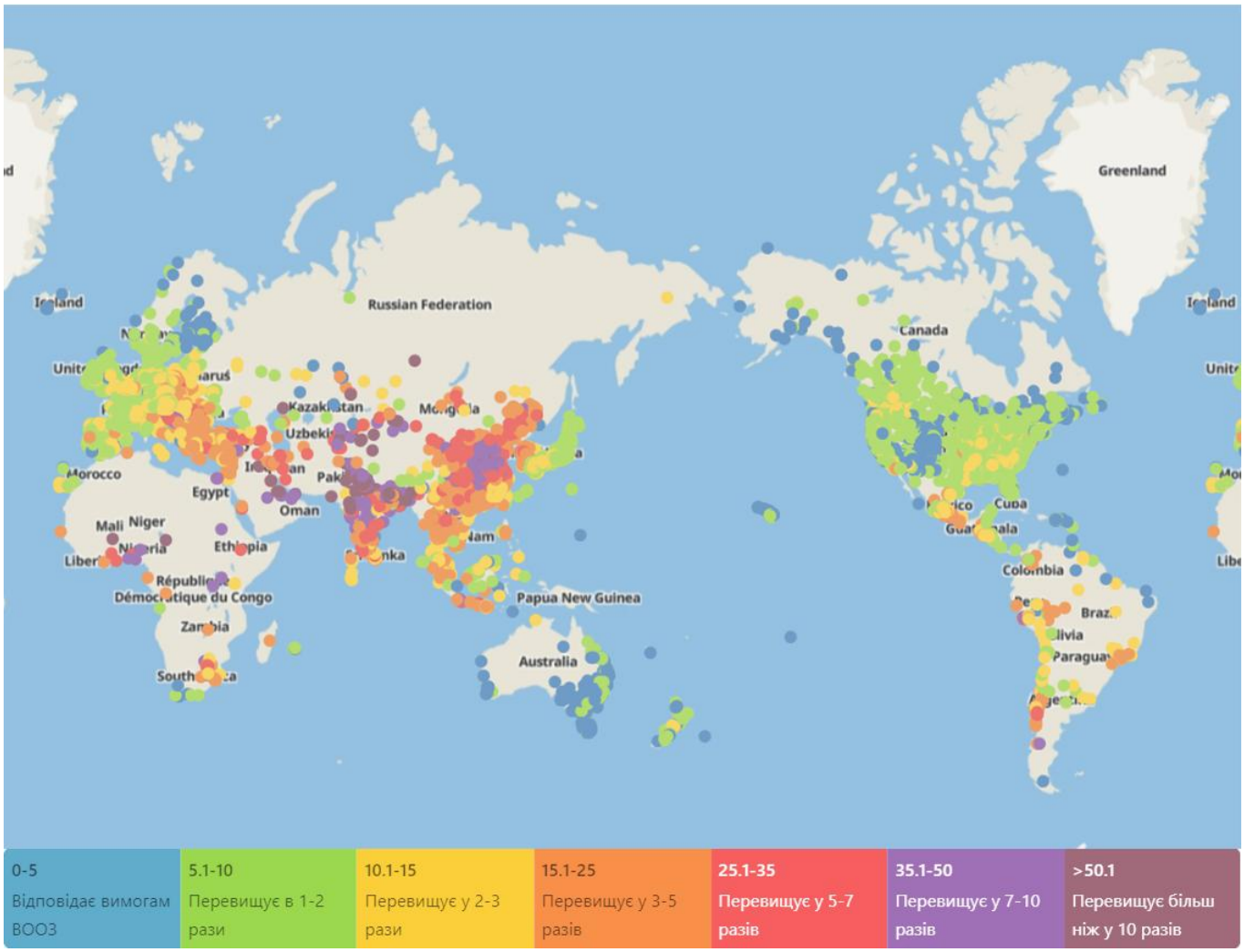


Рис.1.2. Карта концентрації PM2.5 за містами у 2022 р.

1.2. Загальний огляд системи моніторингу забруднення атмосферного повітря

Система для моніторингу атмосферного забруднення – це комплексна інфраструктура, що включає в себе датчики, засоби збору та обробки даних, програмне забезпечення та комунікаційні засоби для вимірювання, аналізу та відображення якості повітря в реальному часі. Основна мета такої системи – надавати інформацію про рівень забруднення атмосфери, щоб вживати заходи для зменшення негативного впливу на здоров'я людей та навколишнє середовище.

Основні компоненти системи моніторингу атмосферного забруднення включають:

- Датчики та вимірювальні прилади: Це пристрої, які встановлюються в різних точках для вимірювання різних параметрів якості повітря, таких як рівні токсичних речовин, твердих часток, парникових газів та інших забрудників.
- Телекомунікаційна інфраструктура: Інформація, зібрана датчиками, передається через телекомунікаційні мережі до центральної системи обробки та аналізу даних.
- Центральний обчислювальний блок: Цей блок обробляє отримані дані, виконує їх аналіз і генерує звіти щодо якості повітря. Він може використовувати різні алгоритми для ідентифікації та класифікації забруднень.
- Геопросторова інформаційна система (ГІС): Дані про забруднення можуть бути візуалізовані на картах за допомогою ГІС, що дозволяє аналізувати просторовий розподіл забруднень.
- Система управління та сповіщення: Якщо рівні забруднення перевищують допустимі стандарти, система може генерувати автоматичні повідомлення для органів влади, служб безпеки чи громадськості.
- База даних: Всі зібрані дані можуть зберігатися в базі даних для подальшого аналізу, порівняння та створення звітів.

Створення комплексної системи моніторингу забруднення атмосфери дозволяє ефективно виявляти та контролювати забруднення повітря, забезпечуючи інформацією для прийняття рішень щодо зменшення викидів та поліпшення якості навколишнього середовища.

1.3. Огляд існуючих систем моніторингу повітря

Існують системи моніторингу якості повітря в різних країнах, кожна з яких має свої особливості та принципи функціонування. Давайте розглянемо кілька прикладів існуючих систем моніторингу якості повітря та їхні основні особливості:

1. **США:**

- **Air Quality Index (AQI)**: Сполучені Штати мають національну систему моніторингу, відому як AQI, яка враховує рівень забруднення повітря та надає інформацію для громадськості та рекомендації щодо захисту здоров'я. AQI враховує параметри, такі як PM2.5, PM10, озон, оксиди азоту та сірки.
- **Система моніторингу національного рівня:** США мають обширну мережу станцій моніторингу національного рівня, розташованих у більшості великих міст та регіонів.
- **Доступ до даних**: Дані про якість повітря доступні для громадськості через веб-сайти та мобільні додатки, що дозволяє користувачам слідкувати за рівнем забруднення в реальному часі.

2. **Китай:**

- **Система моніторингу великого масштабу:** Китай має одну з найбільших систем моніторингу якості повітря в світі, з тисячами станцій по всій країні.
- **Фокус на частках PM2.5:** Забруднення частками PM2.5 є серйозною проблемою в Китаї, тому система акцентує увагу на цьому параметри.
- **Система оповіщення для громадськості:** Китай розробив систему оповіщення та попередження для населення у випадках погіршення якості повітря.

3. **Європейський союз:**

- **European Air Quality Index (EAQI):** Європейський союз використовує EAQI для визначення якості повітря в регіоні. Цей індекс враховує параметри, такі як PM10, PM2.5, озон, оксиди азоту та сірки.
- **Мережа моніторингових станцій**: ЄС має розвинену мережу моніторингових станцій, розташованих в різних країнах-членах.
- **Рекомендації та обмін інформацією:** ЄС надає рекомендації для країн-членів та сприяє обміну інформацією щодо якості повітря між країнами.

4. **Індія:**

- **Система моніторингу індексу якості повітря:** Індія має власну систему моніторингу, яка враховує рівень забруднення та надає рекомендації для громадськості.
- **Додатки та платформи для громадськості:** В Індії є додатки та онлайн-платформи, що надають інформацію про якість повітря для користувачів.
- **Аварійні сповіщення:** Індія розробляє систему аварійних сповіщень та рекомендацій для населення в разі збільшення забруднення повітря.

Ці системи моніторингу мають різні особливості, орієнтовані на потреби кожної країни та спрямовані на вирішення конкретних проблем забруднення повітря. Вони допомагають забезпечувати якість повітря та захищати здоров'я населення в умовах різних кліматичних, географічних та соціальних умов.

Таблицю порівняння систем моніторингу у таких країнах як США, Китай, ЄС та Індія представлена нижче(див.табл.1.2).

Таблиця 1.2

Порівняння систем моніторингу забруднення повітря

Ознака/Країна	США	Китай	Європейський Союз	Індія
Рівень покриття	Широке покриття	Велике регіональне охоплення	Різноманітне	Залежить від регіону, нерівномірне
Технічна інфраструктура	Розвинена мережа сенсорів і датчиків	Значна кількість мережевих станцій	Висока технічна оснащеність	Зростаюча кількість мереж станцій
Доступ до даних	Відкритий доступ до багатьох даних	Доступ до обмеженої кількості даних	Різний доступ у різних країнах	Залежить від рівня транспарентності
Моніторинг в реальному часі	Так	Так	Так	Обмежено

Види забруднюючих речовин	Різноманіття забруднюючих речовин	Різноманіття забруднюючих речовин	Різноманіття забруднюючих речовин	Проблеми з вуглеводневим забрудненням
Інтерактивні карти	Є	Є	Є	Частково

1.4. Технології та методи, що використовуються в існуючих системах моніторингу

Існують різні методи та технології моніторингу, які використовуються в системах моніторингу якості повітря(за систему моніторингу відповідає не тільки інформаційна система, але й система з застосуванням людей). Ось деякі з них:

1. Датчики та аналізатори:
    - Датчики PM2.5 і PM10: Вимірюють концентрацію часток з діаметром менше 2,5 мікрон та 10 мікрон.
    - Датчики газів: Використовуються для вимірювання рівнів газів, таких як оксиди азоту (NOx), оксиди сірки (SOx), оксид вуглецю (CO), озон (O3) тощо.
    - Датчики важких металів: Визначають концентрацію шкідливих важких металів, таких як свинець, ртуть, кадмій тощо.
    - Датчики вуглеводнів та органічних речовин: Вимірюють концентрацію речовин, які можуть виходити з промислових викидів або автомобілів.
    - Датчики аерозолів: Допомагають виміряти концентрацію та склад аерозольних часток в повітрі.
2. Метеорологічні Спостереження:

Включають в себе вимірювання температури, вологості, швидкості та напрямку вітру. Ці дані важливі для розуміння розподілу та розсіювання забруднюючих речовин.

3. Спектральний Аналіз:

Використовує аналіз світла, яке пропускається через атмосферу, для визначення концентрації різних забрудників.

4. Засоби супутникового спостереження:

Супутники Модис та Copernicus: Використовуються для моніторингу забруднення повітря з космосу(див.рис.1.3). Вони надають інформацію про глобальні тенденції та зміни в якості повітря.

5. Лазерні Технології:

Використовують лазери для вимірювання концентрації та складу певних газів в атмосфері.

6. Біосенсори:

Використовують організми або їх частини (бактерії, клітини і т. д.) для визначення різних забруднюючих речовин.

7. Моніторинг За Допомогою Дронів:

Використовують дрони для отримання даних важкодоступних або небезпечних місць.

8. Моделювання та програмне забезпечення:

- Комп'ютерні моделі: Використовуються для прогнозування рівня забруднення повітря на основі різних факторів, включаючи погоду, транспорт, промисловість тощо.
- Географічні інформаційні системи (ГІС): Дозволяють аналізувати та візуалізувати дані про забруднення повітря на мапах.

9. Додатки та платформи для громадськості:

- Мобільні додатки: Надають громадськості можливість слідкувати за рівнем забруднення повітря та надають рекомендації щодо захисту здоров'я.
- Онлайн-платформи: Надають доступ до даних про якість повітря для широкої громадськості.

Рис.1.3. Знімок супутника Копернік

Ці технології та методи працюють разом для забезпечення збору, аналізу та представлення інформації про якість повітря, що допомагає владі та громадськості приймати обгрунтовані рішення щодо збереження здоров'я та навколишнього середовища.

Висновки до розділу

Розділ, присвячений технологіям та інструментам систем моніторингу забруднення атмосферного повітря, дозволяє зрозуміти глибину та різноманітність підходів до цього важливого завдання. Використання розгалуженої мережі датчиків, аналізаторів та станцій моніторингу забезпечує комплексний підхід до вимірювання різних параметрів якості повітря. Супутникові системи, такі як Modis та Copernicus, додають глобальний розмір, дозволяючи відстежувати та реагувати на забруднення на великій території.

Комп'ютерні моделі та алгоритми прогнозування дозволяють не лише реагувати на поточні рівні забруднення, а й передбачати їхні можливі зміни. Це

робить системи моніторингу ефективним інструментом для влади та громадськості у впровадженні заходів для збереження здоров'я громадян та покращення стану навколишнього середовища.

Засоби передачі даних, такі як Інтернет речей (IoT) та системи зв'язку, роблять інформацію про якість повітря доступною та зрозумілою для широкого загалу. Додатки та платформи для громадськості сприяють активній участі громадян у моніторингу та сприйманні рішень.

Загальна концепція використання цих технологій полягає в створенні ефективних механізмів виявлення та контролю забруднення повітря, забезпечуючи тим самим безпеку та здоров'я суспільства.

РОЗДІЛ 2
ВИМОГИ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ

2.1. Цілі створення системи

Створення системи моніторингу забруднення атмосферного повітря має значущий позитивний вплив на життя людей та сприяє поліпшенню якості довкілля. Ось деякі цілі з точки зору користі для людини:

- **Охорона здоров'я:** Моніторинг забруднення повітря допомагає зменшити ризик розвитку респіраторних та інших захворювань, спричинених забрудненням повітря, і зберігати здоров'я людей.
- **Попередження кризових ситуацій:** Система дозволяє вчасно виявляти надмірні рівні забруднення і запроваджувати необхідні заходи для попередження серйозних кризових ситуацій, таких як смог, отруйні викиди та інші аварійні ситуації.
- **Збереження якості життя:** Покращення якості повітря сприяє покращенню якості життя, сприяє комфорту та збільшує продуктивність людей.
- **Сприяння екологічній свідомості:** Публічний доступ до даних про забруднення сприяє зростанню свідомості про проблеми забруднення повітря і підтримує ініціативи з охорони навколишнього середовища.
- **Підвищення ефективності заходів зниження забруднення:** Дані, зібрані системою моніторингу, допомагають владі та організаціям розробляти та впроваджувати ефективні заходи для зменшення забруднення повітря.
- **Економічні вигоди:** Зменшення забруднення повітря може призвести до зменшення витрат на лікування захворювань, пов'язаних із забрудненням, і сприяти розвитку секторів, що пов'язані із зеленими технологіями та збереженням довкілля

- **Забезпечення екологічної стабільності:** Система сприяє збереженню природних ресурсів, біорізноманіття та збалансованому використанню природи, що корисно для майбутніх поколінь.

Таким чином, створення системи моніторингу забруднення атмосферного повітря спрямоване на поліпшення якості життя та довкілля, забезпечення безпеки і здоров'я людей і сприяє сталому розвитку суспільства.

2.2. Вимоги користувача та функціональні вимоги

Користувач системи моніторингу забруднення атмосферного повітря повинен мати змогу виконувати різні дії, щоб отримувати необхідну інформацію та приймати заходи на основі цієї інформації. Ось деякі з можливих дій, які користувач повинен мати можливість виконувати:

- Перегляд інформації: Користувач повинен мати можливість переглядати поточні та минулі дані про якість атмосферного повітря, такі як рівні різних забруднювачів .
- Реєстрація у системі.
- Авторизація.
- Навігація та пошук.
- Відображення на карті: Система може надавати інформацію на мапі, показуючи рівні забруднень у різних місцях і областях.
- Отримання оповіщень: Користувач може встановлювати оповіщення про перевищення нормативів забруднень та отримувати повідомлення, коли це відбувається.
- Аналіз трендів: Можливість аналізувати дані для виявлення трендів та змін в рівнях забруднень впродовж часу.

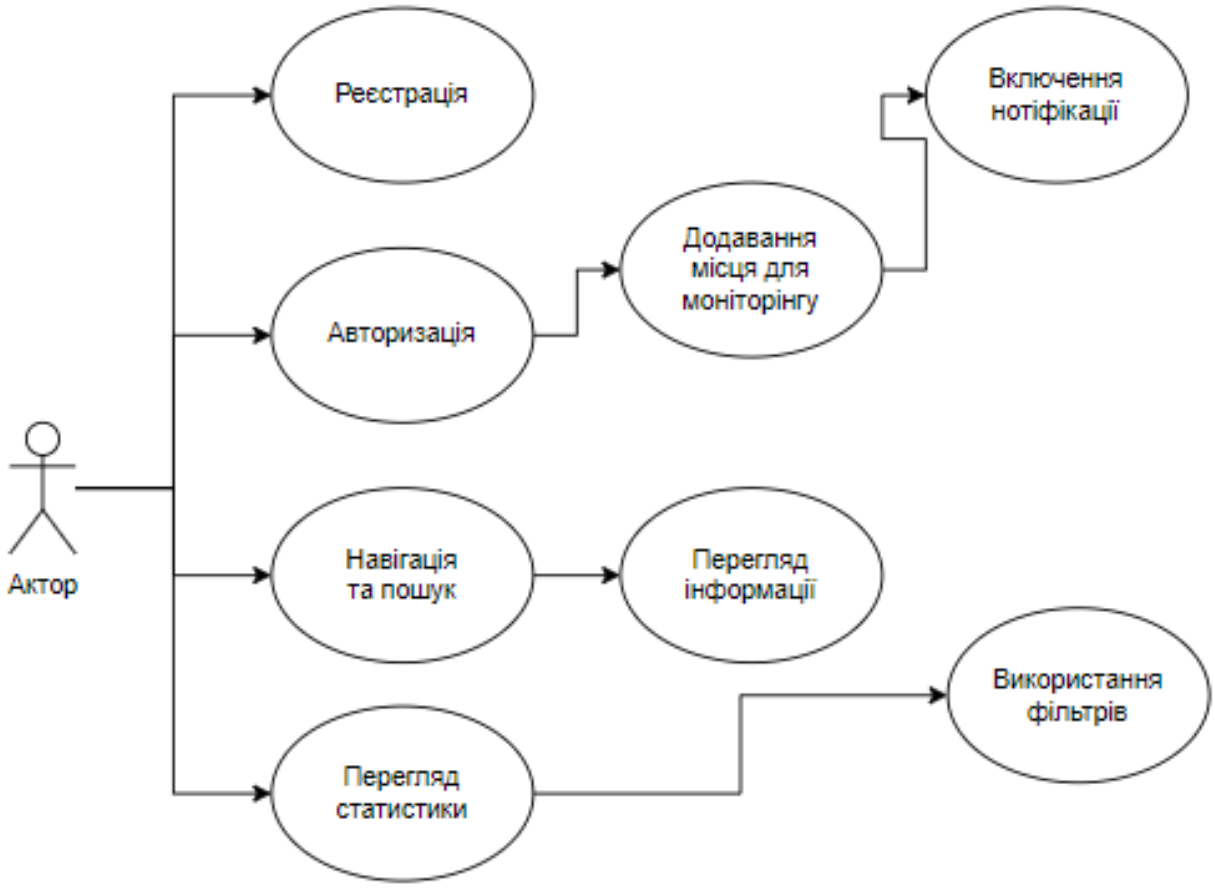


Рис. 2.1. Діаграма варіантів використання інформаційної системи моніторингу забруднення атмосферного повітря

2.3. Бізнес вимоги

Бізнес-вимоги є важливою складовою процесу визначення мети системи. Під час формулювання цих вимог, враховуються потреби, які повинні бути задоволені програмним забезпеченням. Визначення бізнес-вимог допомагає встановити критерії прийняття проекту замовником та визначити функціональну структуру проекту. Основною метою бізнес-вимог є визначення потреб, які стоять за створенням системи. Розуміючи ціль створення програмного додатку, можна ідентифікувати його потенційну цільову аудиторію та попит на продукт, що в свою чергу може призвести до збільшення прибутку від розробки продукту

Базове завдання системи моніторингу забруднення атмосферного повітря полягає в наданні інформації про рівні забруднення атмосфери та якості повітря в різних локаціях та в різний час. Головною метою є вимірювання, збір, аналіз та

візуалізація даних про концентрації різних забруднювачів, таких як токсичні гази та частки, що мають вплив на якість повітря та здоров'я людей. Система надає можливість:

1. Моніторити рівні забруднення повітря у реальному часі, а також протягом певних періодів часу.
2. Аналізувати дані для виявлення трендів, сезонних змін, аномалій та ідентифікації джерел забруднення.
3. Визначати відповідність рівнів забруднення нормативам та нормам якості повітря.
4. Забезпечувати оповіщення та інформування користувачів та владних органів про небезпеку або перевищення норм.
5. Підтримувати публічний доступ до даних, щоб інформувати громадськість та дослідників.

Потенційними користувачами системи можуть бути:

Громадськість: Люди, які мешкають у різних місцях та цікавляться якістю повітря, можуть користуватися системою для отримання інформації та оповіщень.

Владні органи та органи охорони здоров'я: Представники владних органів можуть використовувати дані для прийняття рішень щодо здоров'я громадян і прийняття заходів для зменшення забруднення.

Дослідники та науковці: Вчені можуть використовувати дані для проведення досліджень та аналізу впливу забруднення на довкілля та здоров'я.

Підприємства та індустрія: Підприємства можуть використовувати систему для моніторингу власних викидів та вдосконалення екологічних стандартів.

Екологічні організації: Організації, що відстоюють інтереси довкілля, можуть використовувати дані для стеження за дотриманням стандартів та привертання уваги до екологічних питань.

Система моніторингу забруднення атмосферного повітря призначена для широкого кола користувачів, включаючи громадськість, органи влади, науковців та організації, і має сприяти збереженню якості повітря та охороні здоров'я людей.

2.4. Нефункціональні вимоги

Нефункціональні вимоги - це вимоги до програмного продукту, які не стосуються його функціональності, але визначають якісні або нефункціональні характеристики, які повинні бути властиві системі або продукту. Ці вимоги визначають параметри якості, характеристики безпеки, продуктивності, доступності та інші аспекти, які не стосуються конкретного функціоналу програми.

- Доступність: Доступність означає, що система моніторингу повинна бути доступною для користувачів в усі часи, без важливих перерв або відключень. Це важливо, оскільки інформація про якість повітря може бути необхідною в будь-який момент, зокрема при ризикових подіях. Щоб забезпечити високу доступність, система повинна мати дублювання серверів, автоматичне відновлення та план відновлення після відмов
- Легкість в використанні: Система повинна бути інтуїтивно зрозумілою та легкою в користуванні. Це важливо для того, щоб різні категорії користувачів, включаючи громадськість та владні органи, могли легко отримувати доступ до інформації про якість повітря без необхідності спеціалізованого навчання. Інтерфейс системи повинен бути інтуїтивно зрозумілим та дружнім до користувача, а також підтримувати різні мови.
- Безпека: Безпека важлива як для даних, збережених в системі, так і для користувачів системи. Система повинна забезпечити конфіденційність, цілісність та доступність даних. Це може включати в себе шифрування даних в транзиті та спокою, механізми аутентифікації та авторизації, інструменти виявлення вторгнень та заходи безпеки на рівні інфраструктури.
- Масштабованість програмного забезпечення: Система повинна бути здатною масштабуватися для обробки зростаючого обсягу даних і користувачів. При великих подіях або в областях з високим рівнем забруднення, система повинна залишати високу продуктивність та доступність. Масштабованість може бути досягнута за допомогою хмарних рішень, розподіленої обробки даних та оптимізації коду.

- Зручність супроводу: Система повинна бути добре документованою, та мати просту процедуру супроводу та оновлення. Це важливо для забезпечення, що система залишається актуальною та безпечною з плином часу. Супровід також включає в себе технічну підтримку для користувачів і вирішення проблем, які можуть виникати.
- Надійність: Система повинна бути надійною та стійкою до відмов. Вона повинна продовжувати працювати навіть під навантаженнями та в разі тимчасових відмов в окремих компонентах. Дублювання серверів, автоматичне відновлення та тестування на стійкість до відмов допоможуть забезпечити надійність.
- Продуктивність: Система повинна бути продуктивною та забезпечувати швидкий доступ до даних для користувачів. Швидкість завантаження сторінок та відповідей на запити має бути на належному рівні. Оптимізація бази даних, кешування та використання швидких серверів можуть покращити продуктивність.

Ці нефункціональні вимоги спрямовані на забезпечення того, що система моніторингу забруднення атмосферного повітря буде надійною, легкою в використанні, безпечною та здатною працювати в різних умовах та ситуаціях.

2.5. Системні вимоги

Системні вимоги - це набір характеристик, параметрів і обмежень, які визначають технічні вимоги до обладнання та програмного забезпечення, необхідних для коректної роботи конкретного програмного продукту або системи. Ці вимоги визначають мінімальні технічні характеристики, які повинні бути відповідні для того, щоб програма або система працювала задовільно і забезпечувала певну якість та продуктивність.

Мінімальні вимоги:

• Операційна система: Windows 10 і вище.

• Процесор: тактова частота від 1,7 ГГц та підтримка багатозадачності;

• RAM: 4 ГБ.

• Постійна пам'ять: не менше 10 ГБ вільного простору на жорсткому диску для зберігання даних та логів.

Також, на системі, на якій буде розгорнуте програмне забезпечення, повинно бути встановлено наступне програмне забезпечення:

• Система керування базами даних MySQL.

• Контейнер сервлетів Tomcat.

• Java.

Висновки до розділу

При створенні цього розділу були розглянуті вимоги до системи моніторингу забруднення атмосферного повітря. Під час аналізу цих вимог була створена діаграма використання, яка ілюструє взаємодію користувачів з системою та різні можливі сценарії її використання. Головною метою аналізу вимог було визначення необхідних функціональних та нефункціональних характеристик, які повинні бути властиві розробленому програмному продукту. На основі цього аналізу були сформульовані ключові вимоги, які визначають архітектуру системи та спрямують подальший процес розробки.

Усі зібрані матеріали та вимоги, які були розглянуті в цьому розділі, будуть використані як основа для проектування та розробки системи моніторингу забруднення атмосферного повітря. Вони стануть важливим вихідним пунктом при виборі архітектурних рішень, розробці функціональності та визначенні технічних параметрів системи. Такий підхід допоможе забезпечити, що система буде відповідати вимогам та очікуванням користувачів, забезпечуючи ефективний моніторинг та контроль якості повітря для покращення життя та довкілля.

РОЗДІЛ 3
ПРОЕКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ

Даний розділ присвячений етапу проектуванню програмного забезпечення, під час якого виконуються заходи з формування моделей та шаблонів, що будуть використовуватися для створення програмного додатка. Цей етап є неодмінною частиною життєвого циклу розробки програмного забезпечення та визначає архітектуру системи, структуру її компонентів, їх функціонал і взаємодію між модулями. Основною метою проектування є створення системи, яка буде ефективною, надійною та легко масштабованою, задовольняючи потреби кінцевого користувача.

3.1. Архітектурна концепція

Даний додаток буде розроблятись за допомогою клієнт-серверної моделі з використанням багаторівневої архітектури. У цілому структура буде виглядати таким чином(див.рис.3.1.).

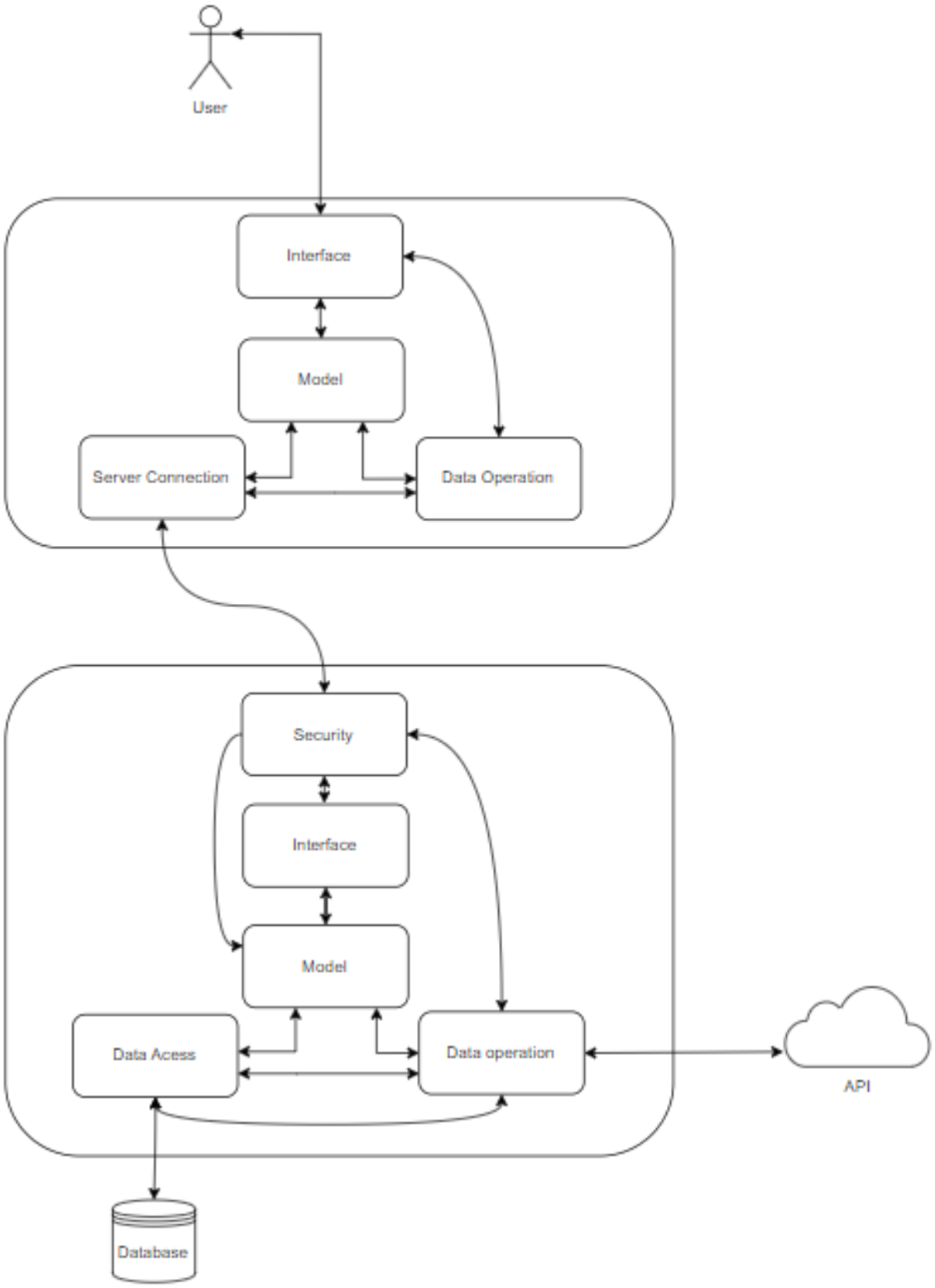


Рис.3.1. Архітектура системи моніторингу забруднення атмосферного повітря

Клієнт-серверна архітектура — це фундаментальний принцип розподіленого програмування, який уособлює взаємодію між двома ключовими складовими: клієнтом і сервером.

Клієнтська частина складається з:

- Графічний інтерфейс (GUI): частина відповідає взаємодію користувача з системою. Це включає в себе відображення елементів керування, введення даних та представлення інформації для зручного користування
- Локальна обробка даних: Клієнт може виконувати певні обчислення чи обробку даних локально, що дозволяє зменшити завантаження на сервер та забезпечити більш оперативну відповідь на дії користувача.

Серверна:

- Бізнес-логіка та обробка запитів: Сервер відповідає за виконання бізнес-логіки, що включає обчислення, обробку даних та прийняття рішень на основі отриманих від клієнта запитів. Ця частина забезпечує консистентність та точність обчислень.
- Управління базою даних: Сервер взаємодіє з базою даних для зберігання, вибору та модифікації інформації. Ефективне управління базою даних гарантує доступність актуальної інформації для виконання операцій з боку клієнта.
- Мережевий обмін та протоколи: Сервер використовує мережеві протоколи, такі як HTTP або TCP/IP, для забезпечення комунікації з клієнтською частиною. Це включає обмін даними, передачу команд та відповідей, забезпечуючи ефективну взаємодію.
- Безпека та автентифікація: Забезпечення безпеки даних та засобів сервера є важливою задачею. Серверна частина реалізує механізми аутентифікації для перевірки прав користувачів та шифрування для захисту конфіденційності даних під час передачі через мережу.

Клієнт-серверна архітектура має кілька переваг, які роблять її популярною для багатьох типів програмних додатків. Ось деякі з них:

1. **Розділення обов'язків:**

Клієнт-серверна архітектура дозволяє розділити обов'язки між клієнтом (користувачем) та сервером. Сервер відповідає за обробку запитів, зберігання даних і виконання бізнес-логіки, тоді як клієнт відповідає за відображення і взаємодію з користувачем. Це розділення спрощує розробку та підтримку системи.

2. **Легкість розширення:**

Клієнт-серверна архітектура дозволяє легко масштабувати систему. Сервер може бути оновлений або розширений без значного впливу на клієнтську частину, і навпаки. Це робить систему більш гнучкою та придатною для масштабування.

3. **Централізоване керування:**

Клієнт-серверна архітектура дозволяє централізовано управляти безпекою і аутентифікацією. Сервер може виконувати аутентифікацію користувачів та контролювати доступ до ресурсів.

4. **Ефективність та оптимізація:**

Клієнт і сервер можуть співпрацювати при виконанні завдань, використовуючи свої ресурси ефективно. Наприклад, складні обчислення можуть бути виконані на сервері, а клієнт отримує лише результат.

5. **Підтримка різних платформ:**

Клієнтські та серверні компоненти можуть бути розроблені з використанням різних технологій та платформ, що дозволяє використовувати оптимальні інструменти для кожної частини системи.

6. **Зменшення трафіку мережі:**

Часто клієнтська частина отримує лише необхідну інформацію з сервера, що допомагає зменшити обсяг передачі даних по мережі та забезпечує більш ефективну роботу системи.

Загалом, клієнт-серверна архітектура визначається своєю легкістю управління, гнучкістю та здатністю до масштабування, що робить її популярним вибором для розробки різноманітних програмних додатків.

Для кращого розуміння архітектури системи використовують діаграму класів. Діаграма класів є одним із видів структурних діаграм в області моделювання

програмного забезпечення, а саме, в рамках мови моделювання UML (Unified Modeling Language). Вона використовується для відображення класів та їх взаємозв'язків у системі. Основною її метою є візуалізація структури системи, а також виявлення інтерфейсів та взаємодій між різними класами.

Основні компоненти діаграми класів:

1. **Класи:** представляють об'єкти або сутності системи, які мають спільні атрибути, методи, інтерфейси та взаємодії.
2. **Атрибути:** описують властивості класу.
3. **Методи:** визначають функціональні можливості класу.
4. **Взаємозв'язки:** вказують на зв'язки між різними класами та їхні характеристики, такі як асоціації, агрегації, композиції тощо.

Інші елементи можуть включати інтерфейси, спадкування, абстракції тощо.

Застосування та переваги діаграм класів:

- **Моделювання системи:**

Діаграми класів використовуються для створення моделі системи, де класи представляють сутності, а атрибути та методи визначають їх характеристики та функції.

- **Документація:**

Вони служать ефективним інструментом для документування архітектури та структури коду.

- **Проектування:**

Діаграми класів допомагають при проектуванні системи, сприяючи уточненню інтерфейсів та взаємодій між класами.

- **Комунікація:**

Вони служать як інструмент комунікації між розробниками, архітекторами та іншими учасниками проекту.

- **Аналіз:**

Діаграми класів дозволяють проводити аналіз коду, виявляти залежності та покращувати структуру програмного забезпечення.

Переваги використання діаграм класів включають:

- *Візуалізація структури:* Дозволяє з легкістю розуміти взаємозв'язки та структуру класів.
- *Спрощення аналізу:* Забезпечує зручний засіб для аналізу програмного коду та взаємодій між класами.
- *Покращення комунікації:* Допомагає уникнути непорозумінь та сприяє ефективній комунікації між членами розробницького колективу.

Загалом, діаграми класів є потужним інструментом для моделювання та розуміння архітектури програмних систем.

Нижче приведено діаграму класів для серверної частини додатку (див. рис. 3.2.)

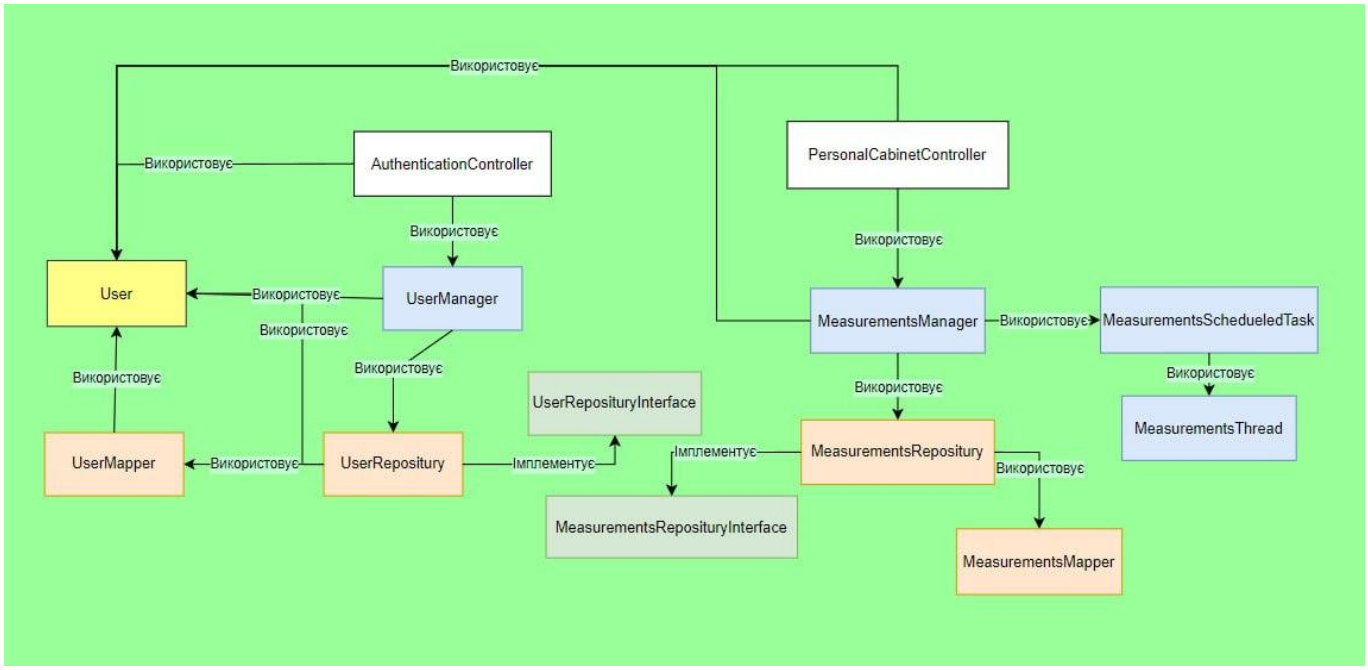


Рис. 3.2. Діаграма класів для серверної частини додатку

Ця діаграма не представляє всі класи системи, а лише надає клієнтам системи розуміння взаємодії основних класів. На діаграмі основні компоненти розділені на кольори, кожен колір відповідає за свою частину класів.

Також важливо зазначити, що на цій діаграмі показано лише частину функціональних можливостей. Серверна сторона відповідає за обробку інформації від клієнта та використання бази даних. На стороні клієнта реалізовані такі типи класів:

- Activity: отвечает за пользовательский интерфейс и взаимодействие с пользователем. Каждая активность представляет собой экран приложения и управляет жизненным циклом.
- Service: Сервисы предназначены для выполнения фоновых задач, даже если приложение не активно.
- Adapter: Адаптеры используются для связи данных с пользовательским интерфейсом, особенно с элементами, такими как списки или гриды. Они преобразуют данные из их исходного формата в формат, который может быть отображен в интерфейсе.

Переглянути, як виглядає діаграма класів клієнтської частини додатку можно на рис.3.3.

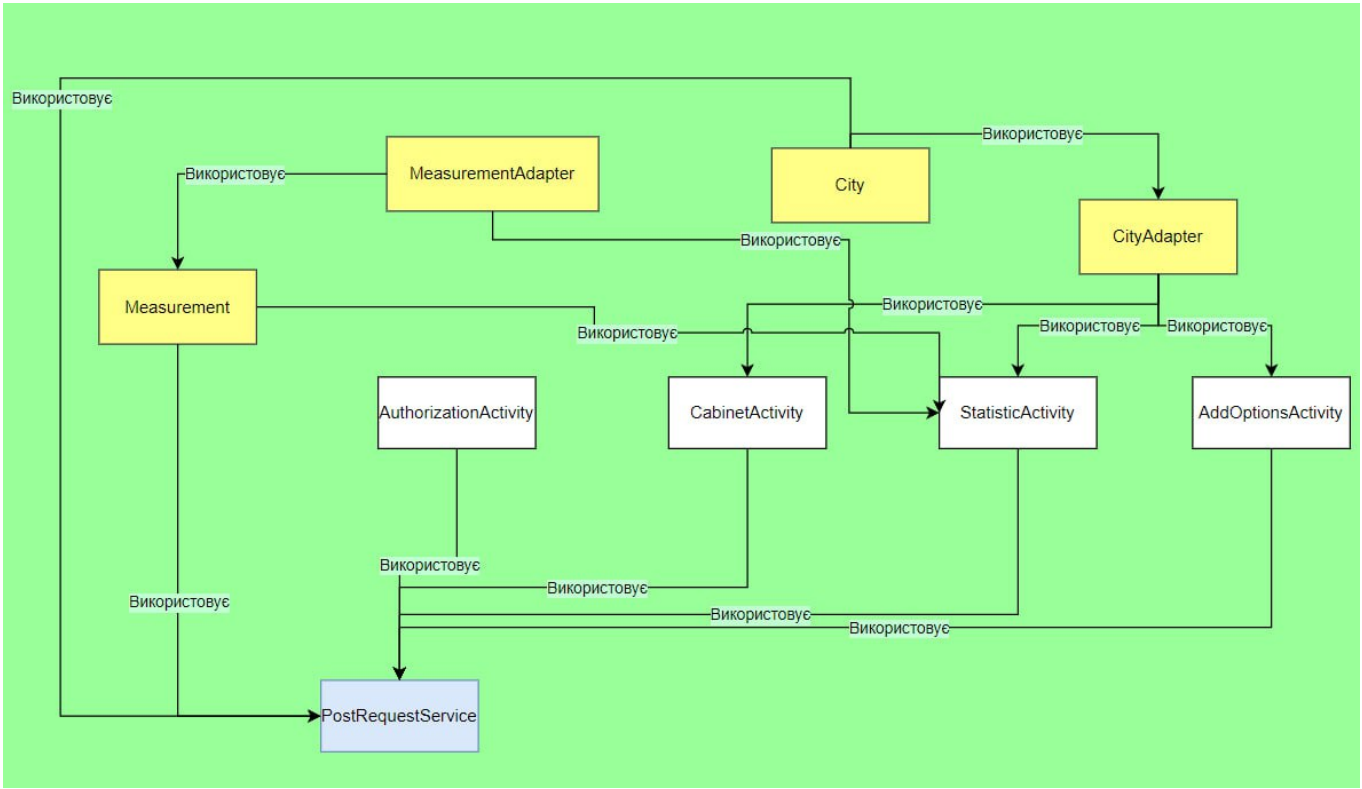


Рис. 3.3. Діаграма класів для клієнтської частини додатку

На діаграмі нижче показано ланцюжок класів, відповідальних за авторизацію та реєстрацію клієнта. Контролер отримує дані від клієнта, а потім переходить до класу менеджера для обробки, а після обробки дані надсилаються до класу сховища, який за потреби вносить зміни в базу даних.

Також важливо розуміти, що організація програмного коду може піддаватися змінам протягом всього процесу розробки. Діаграми класів, створені на етапі проектування, не завжди точно відображають поточний стан програми, яку розробляють. Вони виступають лише відомчими прикладами структури та архітектури майбутніх програм. На їхній основі буде створена система, проте за потреби може здійснюватися реорганізація структури для вдосконалення роботи програми.

3.2. Інтеграція та взаємодія з датчиками моніторингу забруднення атмосферного повітря

Взаємодія з датчиками забруднення атмосферного повітря може відбуватися через різні типи технологій та з'єднань. Вибір конкретного типу взаємодії залежить від характеристик самого датчика, його призначення, доступних технологій та вимог системи. Ось деякі типи взаємодії з датчиками забруднення повітря:

1. **Бездротова Взаємодія:**
   - *Wi-Fi:* Датчики можуть мати можливість підключатися до мережі Wi-Fi для віддаленого моніторингу та передачі даних в хмарне середовище.
   - *Bluetooth:* Використовується для локального з'єднання та обміну даними між датчиками та сумісними пристроями.
   - *LoRa (Long Range):* Дозволяє віддалену взаємодію на велику відстань, ідеальний для датчиків в розташуваннях, де немає доступу до Wi-Fi або мережі зв'язку.
   - *NB-IoT (*Використовується для надійного та ефективного з'єднання в Інтернеті речей (IoT), особливо в умовах обмеженого споживання енергії.
2. **Провідна Взаємодія:**
   - *USB або Ethernet:* Деякі датчики можуть мати можливість підключення до системи за допомогою провідних з'єднань.
3. **Інфрачервоні Технології:**

- *Використання інфрачервоних сигналів для передачі даних між датчиками та іншими пристроями.*

Отже, для розробки інформаційної системи моніторингу забруднення атмосферного повітря було вирішено використовувати існуючий API від компанії The World Air Quality Project, її обрано через те, що компанія реалізувала збір даних з датчиків з різних куточків світу.

У цій дипломній роботі використовуватиметься саме інформація, яка свідчить про рівень забруднення у конкретний момент часу. Отримані дані будуть систематично зберігатись у базі даних, щоб мати можливість подальшого аналізу та використання у межах розробленої інформаційної системи.

3.3. Проектування бази даних

Процес проектування бази даних (ПБД) є ключовим етапом в розробці інформаційних систем і грає важливу роль у забезпеченні ефективного та надійного функціонування програмного забезпечення. Важливість цього процесу визначається кількома ключовими аспектами:

- **Організація та зберігання даних:**

ПБД допомагає визначити оптимальну структуру для зберігання та організації інформації. Це включає в себе визначення таблиць, полів та взаємозв'язків між ними.

- **Забезпечення інтеграції та зменшення дублювання:**

Процес проектування бази даних дозволяє використовувати методи нормалізації для уникнення дублювання даних та забезпечення цілісності даних.

- **Обмеження та валідація:**

Встановлення правил обмежень та валідації допомагає утримувати дані відповідно до певних стандартів та забезпечує консистентність даних.

- **Індексація та оптимізація:**

Врахування потреб користувачів у запитах дозволяє оптимізувати структуру бази даних за допомогою індексації та інших методів для ефективного виконання запитів.

- **Безпека та керування доступом:**

ПБД включає в себе визначення прав доступу, що допомагає забезпечити безпеку даних, забезпечуючи лише необхідний рівень доступу користувачам.

- **Масштабованість та розширюваність:**

Планування на майбутнє, враховуючи можливість розширення та зростання обсягів даних.

- **Документація:**

Проектування бази даних включає створення документації, що полегшує обслуговування та розуміння структури для інших розробників.

- **Відповідність законодавству та правилам:**

Врахування вимог до конфіденційності, доступу та зберігання даних відповідно до законодавства та стандартів.

При проектуванні БД було створено ER-діаграму, іншими словами діаграму сутність-зв’язок. ER-діаграма (Entity-Relationship Diagram) – це графічне представлення структури бази даних, яке використовується для моделювання та опису відносин між різними сутностями в системі. Цей тип діаграми входить в стандартні інструменти моделювання баз даних та є частиною мови моделювання UML (Unified Modeling Language).

Ключові елементи ER-діаграми:

- **Сутності (Entities):**

Представляють об'єкти або поняття, для яких зберігається інформація в базі даних. Наприклад, "Клієнт", "Продукт" або "Замовлення".

- **Відносини (Relationships):**

Вказують на зв'язки між різними сутностями. Наприклад, зв'язок "Має" між "Клієнт" і "Замовленням".

- **Атрибути (Attributes):**

Представляють конкретні дані, які зберігаються для кожної сутності. Наприклад, "Ім'я", "Адреса" або "Ціна".

- **Ключі (Keys):**

Вказують на унікальний ідентифікатор сутності або відносини, який дозволяє однозначно визначити запис.

ER-діаграма допомагає візуалізувати структуру бази даних та взаємозв'язки між різними сутностями. Вона стає ефективним інструментом для аналізу та проектування баз даних перед їхнім реальним створенням. За допомогою цього типу діаграм можна легко розібрати взаємодії між об'єктами та спростити процес розробки бази даних.

ER-діаграми застосовуються не тільки для створення нових баз даних, але і для аналізу та вдосконалення вже існуючих структур. Основні концепції ER-діаграм використовуються в багатьох підходах до реляційного моделювання та дизайну баз даних.

Для інформаційної системи моніторингу забруднення атмосферного повітря було створено наступну діаграму сутностей(див. рис. 3.4.).

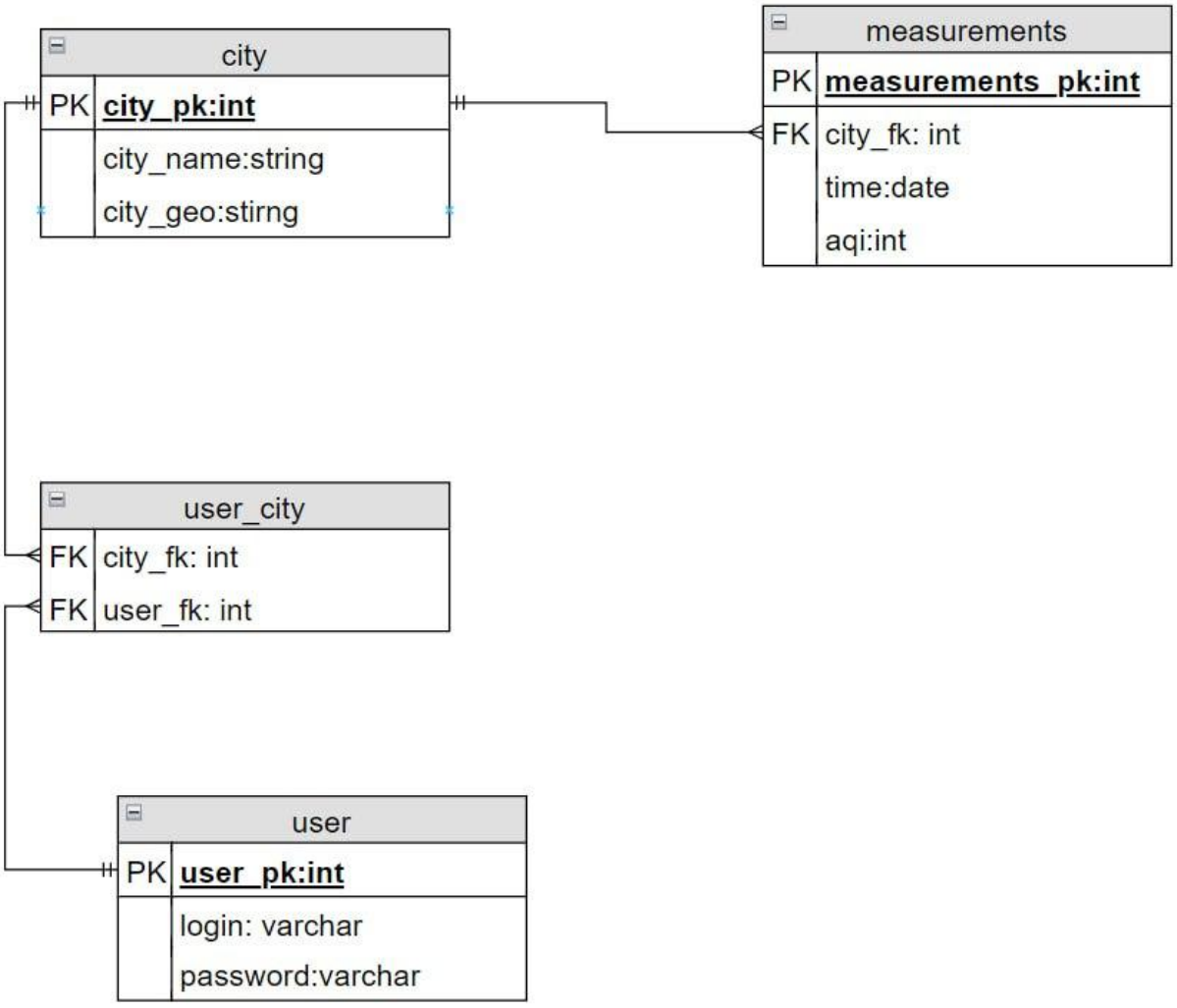


Рис. 3.4. ER-діаграма для інформаційної системи моніторингу забруднення атмосферного повітря

Висновки до розділу

У процесі створення структури інформаційної системи для моніторингу забруднення атмосферного повітря було вивчено та ретельно проаналізовано обширний обсяг матеріалу, пов'язаного з етапом проектування програмного забезпечення. За допомогою отриманого висновків були розроблені допоміжні компоненти, такі як схема архітектури системи, діаграма класів та діаграма сутностей для бази даних. Ці інструменти стануть основою для подальшого розроблення самої системи моніторингу забруднення атмосферного повітря.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ

4.1. Інструменти для розробки системи

Існує багато інструментів для розробки програмного забезпечення (ПЗ), які відрізняються за призначенням, мовою програмування, областю використання та іншими параметрами.

При виборі інструментів враховуються такі фактори, як:

- **Гнучкість**

Обраний інструмент розробки програмного забезпечення повинен бути гнучким для забезпечення правильної, точної та ефективної реалізації програмного продукту. Зазвичай більшість сучасних інструментів розробки програмного забезпечення достатньо гнучкі, щоб підтримувати різні мови програмування та надавати швидкий розвиток за допомогою можливостей кросплатформенного розробки. Вибір платформи для розробки програмного забезпечення, яка працює на різних операційних системах і забезпечує значну гнучкість та масштабованість у процесі розробки, допоможе вам значно.

- **Накладні витрати**

Деякі інструменти вимагають придбання ліцензій, що може призвести до значних витрат. Наприклад, платні інтегровані середовища розробки (IDE), бази даних чи системи управління версіями. Якщо нові інструменти мають інтегруватися з існуючими системами, це може призвести до додаткових витрат на розробку та тестування інтеграції. Якщо ви використовуєте хмарні сервіси для розробки чи тестування, це може призвести до витрат на інфраструктуру та обчислювальні ресурси.

- **Зручність для користувача**

Зручність використання інструменту розробки програмного забезпечення – один із найважливіших аспектів, який слід враховувати під час пошуку відповідного

інструмента розробки програмного забезпечення. Навіть якщо розробники програмного забезпечення володіють великим технічним досвідом і знаннями, вони повинні обрати платформу для розробки програмного забезпечення, яка допоможе їм автоматизувати процеси та зробити весь процес розробки простішим і швидш

- **Сумісність**

Сумісність інструментів програмування має значущий вплив на процедури розробки. Зазвичай, слід обирати інструменти розробки програмного забезпечення, які можуть працювати за різних вимог, щоб забезпечити ефективний та швидкий процес розробки. Важливо пам'ятати, що не всі засоби програмування є сумісними з усіма середовищами.

- **Цільова аудиторія**

Різні інструменти розробки програмного забезпечення мають свою власну цільову аудиторію та стандарти розробки. Деякі інструменти програмування призначені для малих підприємств та початківців розробників. З іншого боку, відомі інструменти розробки програмного забезпечення зазвичай призначені як для новачків, так і для досвідчених розробників. Тому перед придбанням чи реєстрацією інструменту розробки програмного забезпечення слід ознайомитися з усіма його аспектами, щоб переконатися, що він відповідає вашим вимогам до розробки.

- **Функціональні можливості**

Навіть якщо основні функціональні можливості більшості інструментів розробки програмного забезпечення залишаються незмінними, існують невеликі відмінності, про які важливо знати, щоб відрізняти інструменти розробки та вибрати найкращий серед них згідно з вашими вимогами.

Вибір мови програмування є стратегічним рішенням і залежить від конкретних вимог та контексту проекту. Java, як мова програмування, має кілька переваг, які можуть зробити її привабливою для ряду проектів:

- **Переносимість:**

Java є "write once, run anywhere" (WORA) мовою, що означає, що програмний код може бути написаний один раз і запущений на будь-якій платформі, яка

підтримує віртуальну машину Java (JVM). Це дозволяє забезпечити велику переносимість програмного забезпечення між різними операційними системами.

- **Велика спільнота та екосистема:**

Java має велику та активну спільноту розробників. Це означає доступ до безлічі бібліотек, фреймворків і інструментів, які полегшують розробку. Крім того, існують багато матеріалів для самонавчання, форумів та блогів.

- **Об'єктно-орієнтований підхід:**

Java є повністю об'єктно-орієнтованою мовою програмування, що сприяє покращенню структури та організації коду, а також полегшує повторне використання коду.

- **Безпека та надійність:**

Java володіє вбудованими механізмами безпеки, які дозволяють розробникам створювати захищені додатки. Також, завдяки системі управління пам'яттю та виняткам, Java відзначається високою надійністю та стабільністю.

- **Великі корпоративні проекти:**

Java часто використовується для великих корпоративних проектів, таких як ентерпрайз-застосунки та веб-додатки, оскільки мова пропонує ефективний та масштабований розвиток.

- **Підтримка многозадачності:**

Java має вбудовану підтримку многозадачності та паралельного програмування, що полегшує створення додатків, які можуть ефективно працювати в умовах багатозадачності.

Важливо враховувати, що вибір мови програмування залежить від конкретних потреб проекту, здатностей команди розробників та інших факторів. Java відзначається своєю універсальністю і ефективністю в ряді областей розробки програмного забезпечення.

Для розробки серверної частини проекту було обрано як середу розробки IntelliJ IDEA. IntelliJ IDEA є однією з найпопулярніших інтегрованих середовищ розробки (IDE) для мови програмування Java, і є кілька причин, чому розробники обирають цей інструмент для роботи:

- **Повна підтримка java:**

IntelliJ IDEA надає повну підтримку мови програмування Java, включаючи останні версії мови. Він автоматично розпізнає і інтегрує нові функції та API, що полегшує роботу з оновленнями Java.

- **Розширені функції рефакторингу:**

Інтелектуальні інструменти рефакторингу в IntelliJ IDEA допомагають змінювати структуру коду безпечно та ефективно. Вони включають в себе перейменування, витягування методу, оптимізацію і багато іншого.

- **Підтримка Різних Фреймворків та Технологій:**

IntelliJ IDEA вбудовано підтримує різноманітні фреймворки і технології, такі як Spring, Hibernate, JavaFX, Maven, Gradle і багато інших. Це дозволяє легко створювати різноманітні типи додатків та проектів.

- **Відмінна робота з кодом:**

Інтерфейс IntelliJ IDEA дуже зручний та ергономічний. Розуміння та редагування коду стає зручним завдяки різноманітним функціям, таким як автоматичне завершення коду, витягування коду, інспекції коду та інші.

- **Вбудовані інструменти тестування:**

IntelliJ IDEA має інтегровані інструменти для тестування, що полегшують написання, виконання та аналіз тестів коду.

- **Підтримка многозадачності:**

Вбудовані інструменти для роботи з системами керування версіями (VCS), такими як Git, дозволяють командам ефективно співпрацювати.

- **Активна та жива спільнота:**

IntelliJ IDEA має велику та активну спільноту розробників, яка підтримується виробником. Це гарантує швидке вирішення проблем, а також наявність багатьох плагінів та розширень.

Загалом, вибір IntelliJ IDEA для розробки на Java визнаний його продуктивністю, функціональністю та зручністю використання.

Для розробки БД було обрано MySQL. Вибір конкретно MySQL має свої переваги:

- **Відкритий вихідний код:**

MySQL є відкритою СУБД, що означає доступ до вихідного коду. Це дає розробникам можливість модифікувати систему під свої потреби та досліджувати її роботу.

- **Велика Спільнота та Підтримка:**

MySQL користується великою та активною спільнотою розробників. Це означає швидке виявлення та виправлення помилок, а також наявність багатьох ресурсів для самонавчання та допомоги.

- **Широка розповсюдженість:**

MySQL є однією з найпоширеніших СУБД у світі. Багато веб-застосунків, великих підприємств та стартапів використовують MySQL для зберігання та управління даними.

- **Висока швидкодія:**

MySQL відомий своєю високою швидкодією та ефективністю роботи з даними. Він оптимізований для швидкого виконання операцій читання та запису.

- **Підтримка ACID:**

MySQL гарантує дотримання принципів ACID (Atomicity, Consistency, Isolation, Durability), що робить його надійним і неперевершеним в транзакційних операціях.

- **Розширені можливості:**

MySQL має багатий набір функцій та розширень, таких як реплікація, класифікація, розподілена база даних, що дозволяє адаптувати його для різних потреб і масштабів проектів.

- **Сумісність та інтеграція:**

MySQL легко інтегрується з різними мовами програмування та платформами, що робить його зручним для використання в різних екосистемах.

- **Безкоштовний та Економічний:**

MySQL є відкритою СУБД з безкоштовною ліцензією GPLv2, що робить його доступним і економічним для використання, особливо для стартапів та невеликих підприємств.

Для розробки інтерфейсу було обрано створення мобільного додатку на базі Android. Обрання ОС Android через те, що це найпоширеніша мобільна платформа у світі. Розробка для Android дозволяє звернутися до великої аудиторії користувачів, що користуються різноманітними пристроями від різних виробників.

Android Studio є офіційним інтегрованим середовищем розробки (IDE) для створення мобільних додатків на платформі Android. Вибір Android Studio має декілька переваг, що роблять його вигідним вибором для розробки мобільних додатків на Android:

- **Офіційність та Підтримка Google:**

Android Studio розроблено безпосередньо Google як офіційне середовище для розробки Android-додатків. Це означає, що воно має повну підтримку від Google, і всі оновлення та нові функції Android найшвидше імплементуються саме в цій IDE.

- **Інтеграція З Android SDK:**

Android Studio поставляється з вбудованим Android SDK, яке включає всі необхідні інструменти та бібліотеки для розробки Android-додатків. Це спрощує процес налаштування середовища розробки.

- **Emulator та Debugging:**

Android Studio постачається з емулятором Android, який дозволяє тестувати додатки на різних пристроях та версіях Android. Також надається потужний інструмент для відлагодження та профілювання додатків.

- **Широка Підтримка Мов Програмування:**

Android Studio підтримує різні мови програмування для Android, включаючи Java, Kotlin та C++. Це надає розробникам можливість обирати мову залежно від їхніх уподобань та потреб.

- **Майстер Швидкого Старту та Шаблони Проектів**:

Android Studio має вбудований майстер швидкого старту та шаблони проектів, що полегшують початок роботи та прискорюють процес розробки.

- **Підтримка Android Jetpack та Бібліотек Google:**

Android Studio повністю підтримує Android Jetpack та інші бібліотеки Google, які допомагають прискорити розробку та робити додатки більш ефективними та масштабованими.

- **Гнучкість та Розширюваність:**

Android Studio дозволяє розробникам використовувати різні плагіни та розширення, що робить його гнучким і адаптованим до різних потреб розробників.

- **Інтеграція з Системами Керування Версіями:**

Android Studio легко інтегрується з системами керування версіями, такими як Git, що полегшує співпрацю розробничих команд.

Огляд зазначених переваг підтримує вибір Android Studio як потужного та зручного інструменту для розробки мобільних додатків на платформі Android.

4.2. Опис реалізації проекту

При розробці проекту основною ціллю є написання коду системи. В попередніх розділах вже встановлено вимоги, архітектурну концепцію та створення діаграми класів на основі яких буде реалізовуватись програмний код, отже далі описано як створювався код системи на різних рівнях.

4.2.1. Створення бази даних

Як було описано в пункті про інструменти розробки, для створення БД використовується СУБД MySQL. Для роботи з нею потрібно зайти на офіційний сайт MySQL Server, обрати відповідну версію до вашою операційної системи та завантажити на компьютер. При встановлені на компьютер потрібно погодитись з ліцензійною угодою, обрати тип встановлення та очікувати завантаження файлів(див.рис.4.1). Далі відкриється вікно налаштувань, де потрібно обрати порт за яким буде здійснюватись з’єднання з сервером(див.рис.4.2.), потім вказати користувацьке ім’я та пароль для root користувача(див.рис.4.3.).

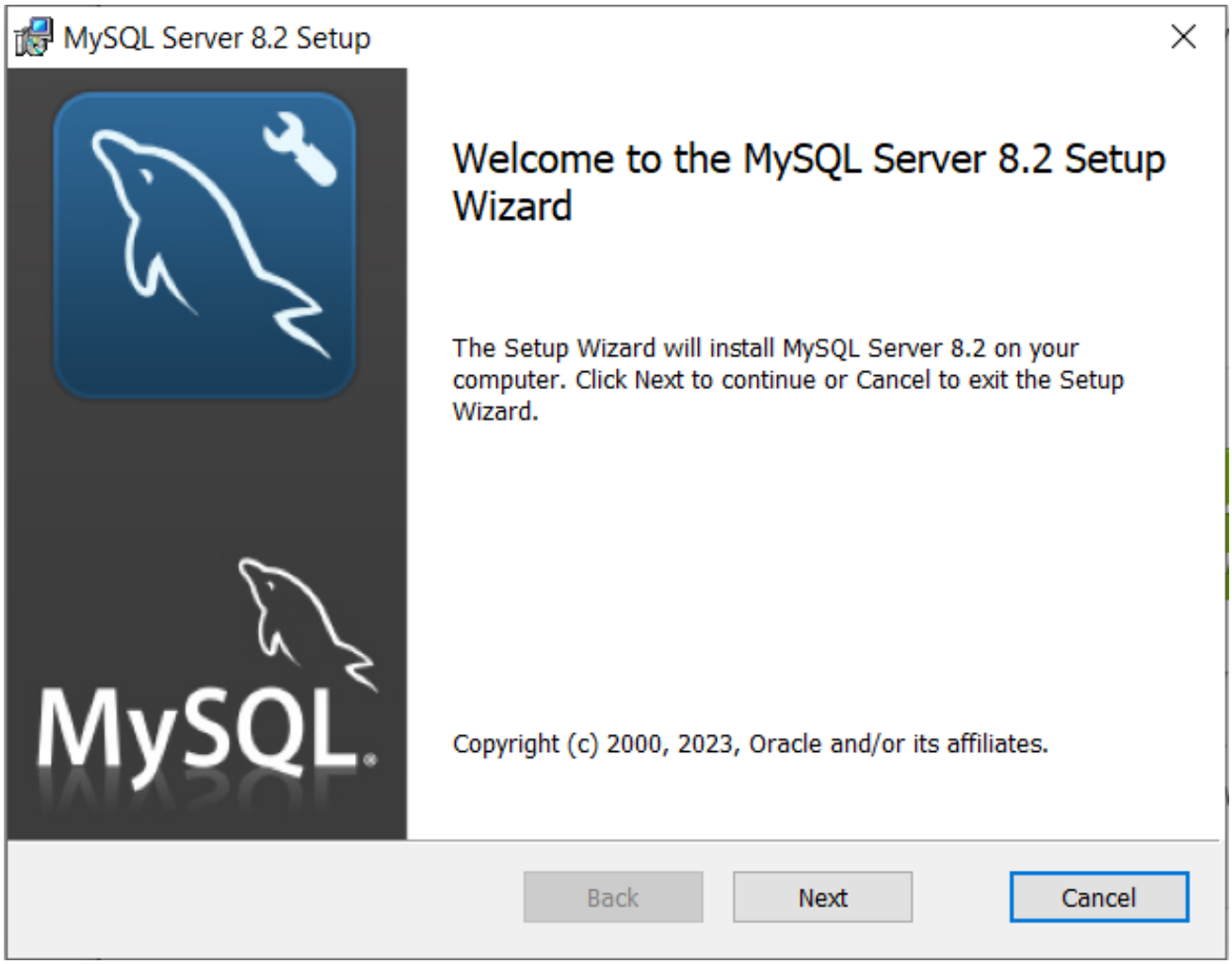


Рис. 4.1. Початок інсталяції MySQL Server

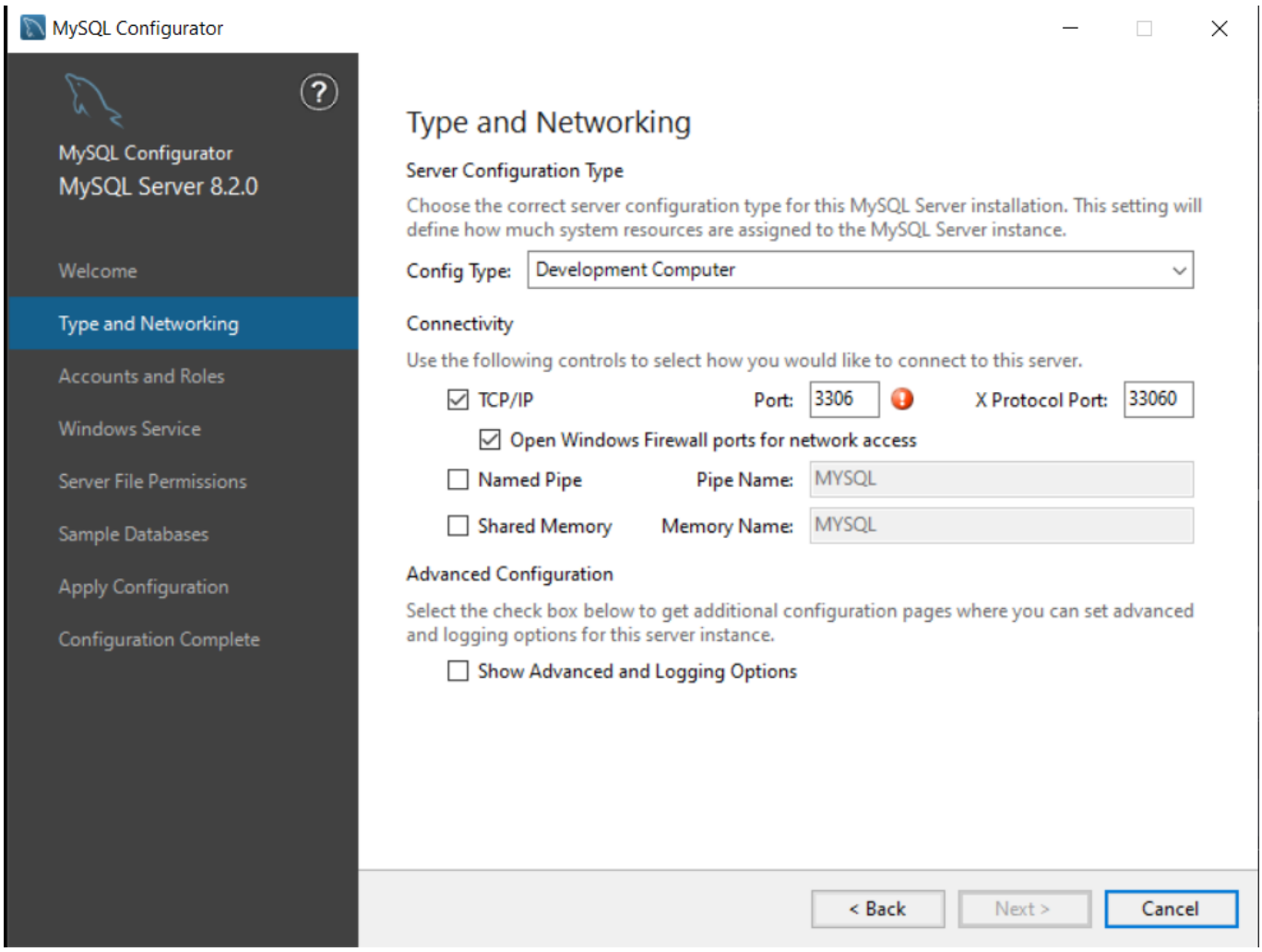


Рис. 4.2. Налаштування MySQL Server

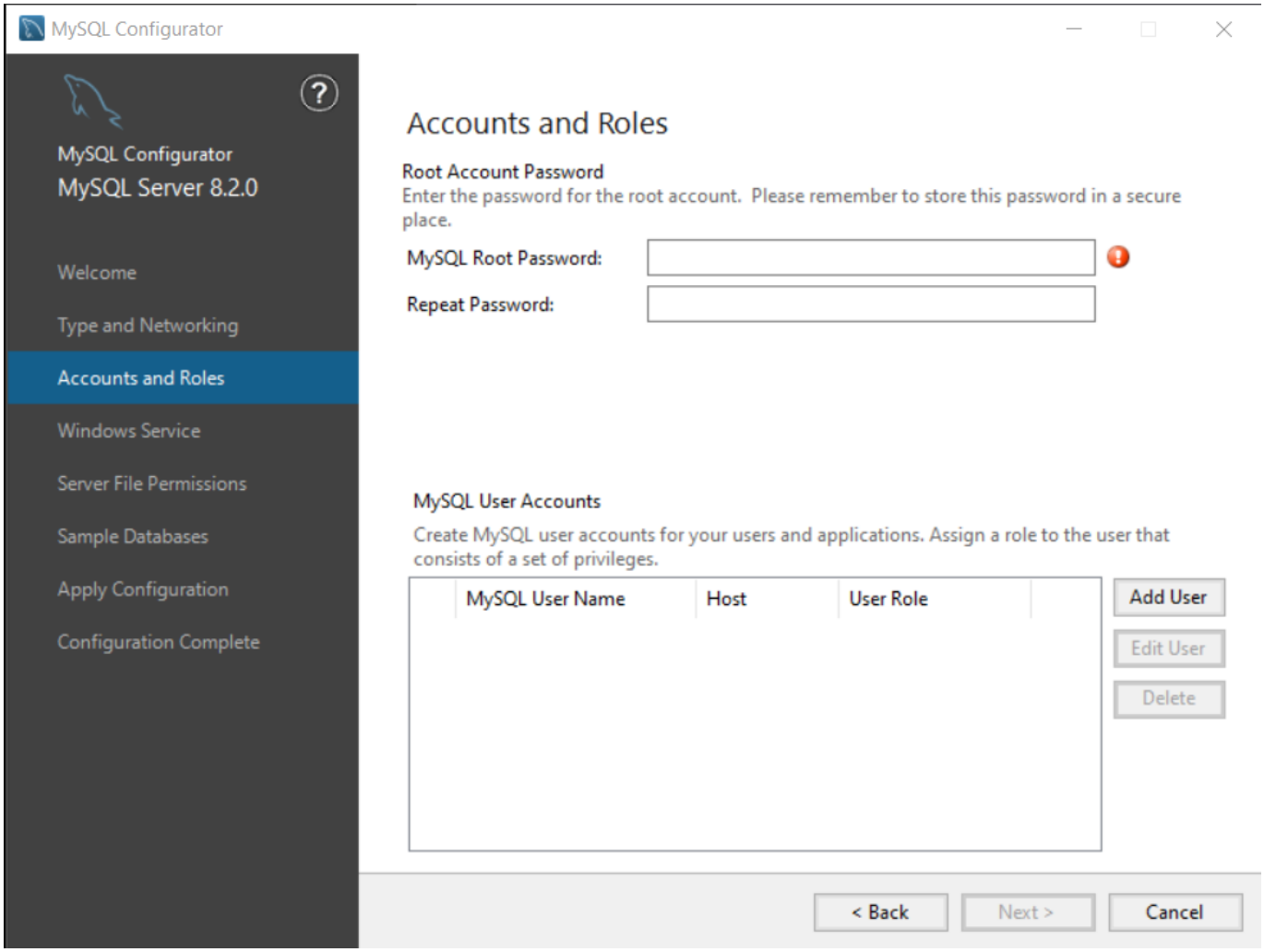


Рис. 4.3. Створення ролей в MySQL Server

При створенні користувацького аккаунту також потрібно було створити пароль та обрати роль користувача (див.рис.4.4.).

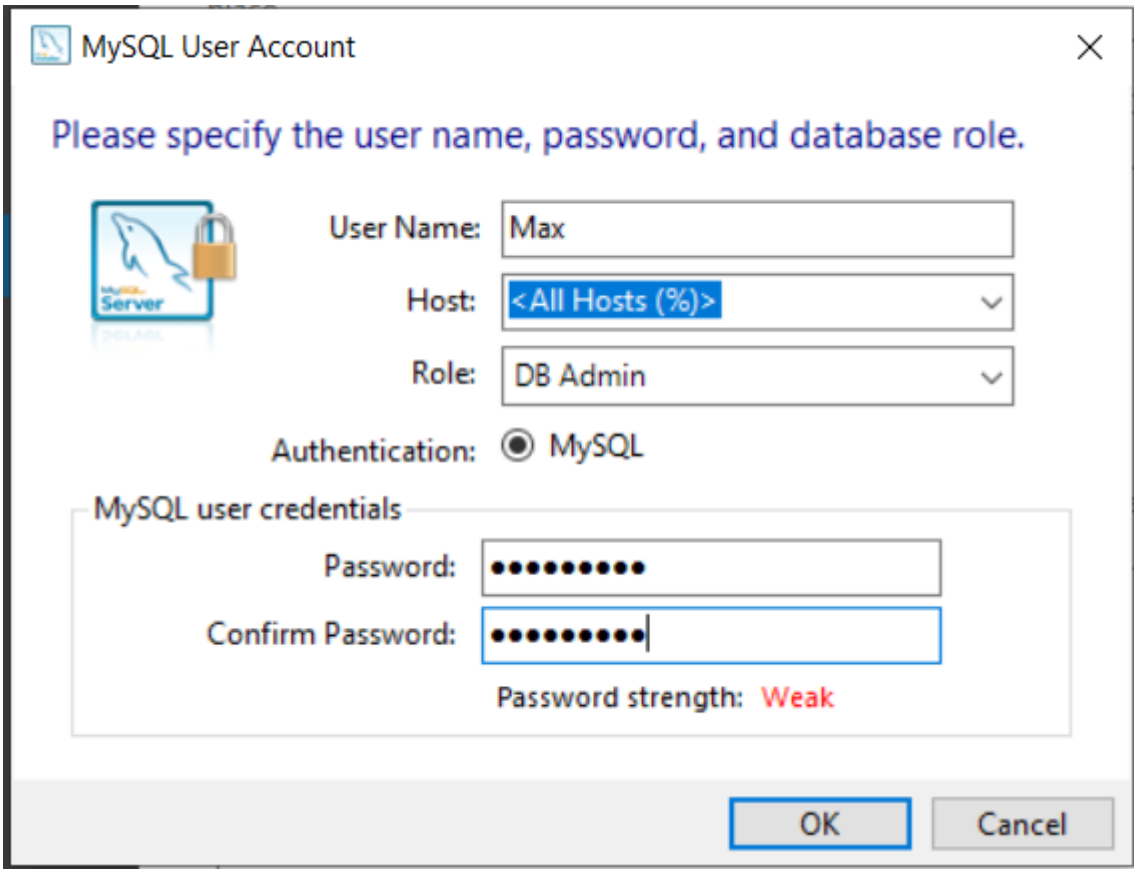


Рис. 4.4. Створення User Account в MySQL Server

Щоб працювати з GUI MySQL Server потрібно встановити MySQL Workbench(див.рис.4.5.).

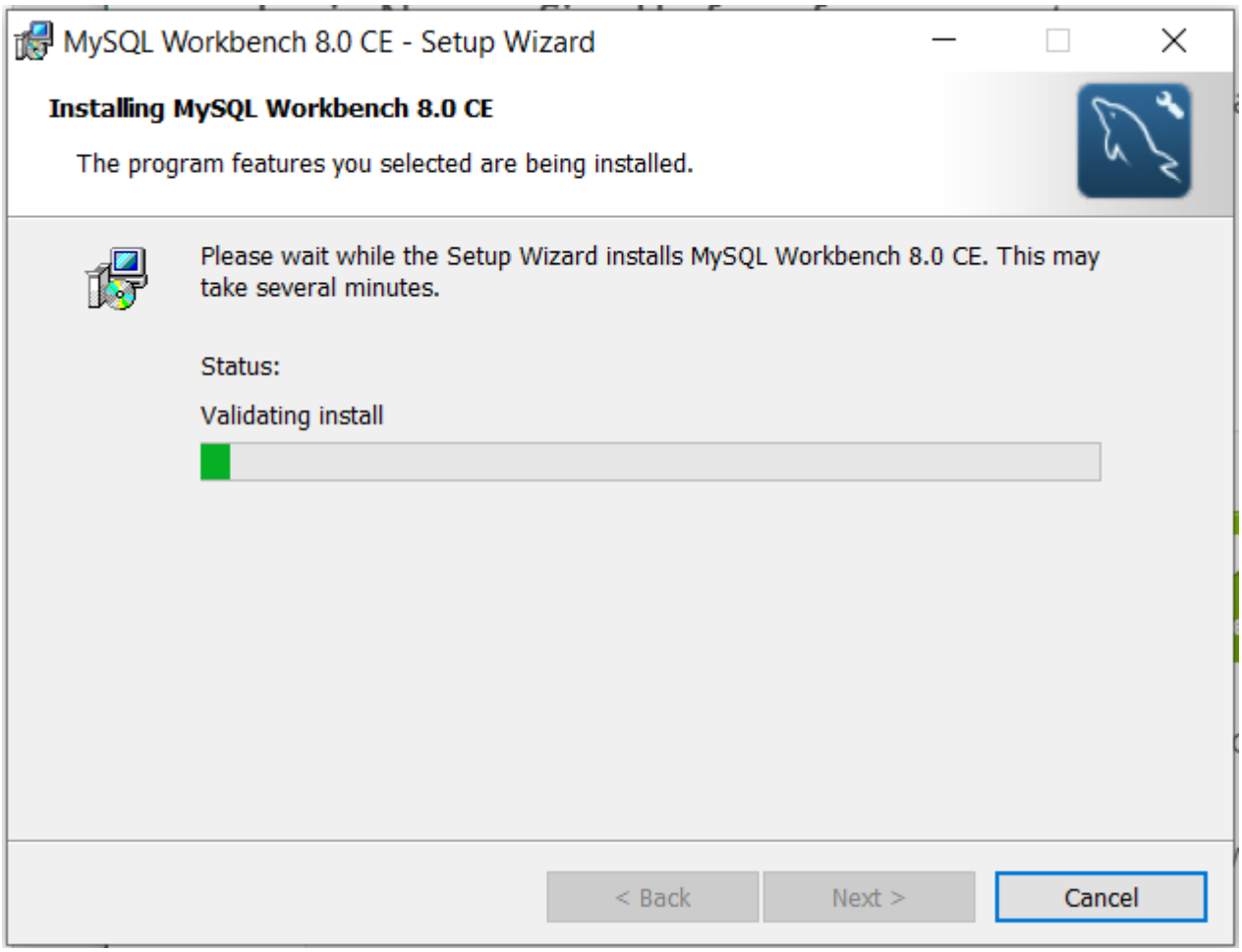


Рис. 4.5. Встановлення MySQL Workbench

Після запуску програми MySQL Workbench, на стартовому вікні натискаємо кнопку «+» і відкривається віконце встановлення з’єднання(див.рис.4.6), вводимо назву з'єднання та натискаємо “Ok”.

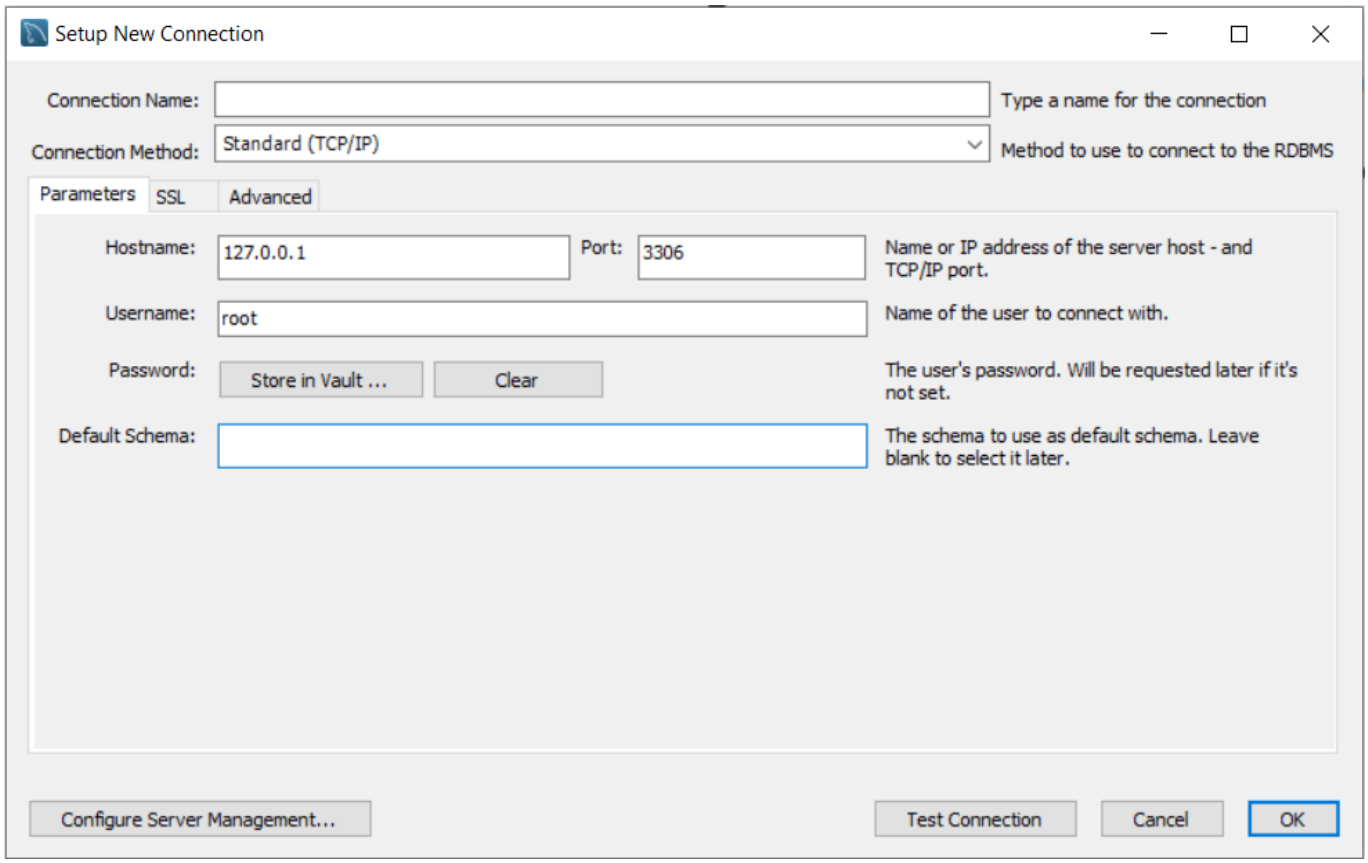


Рис. 4.6. Встановлення з’єднання в MySQL Workbench

Переглянути встановлені з’єднання можно на стартовій сторінці додатку(див.рис.4.7).

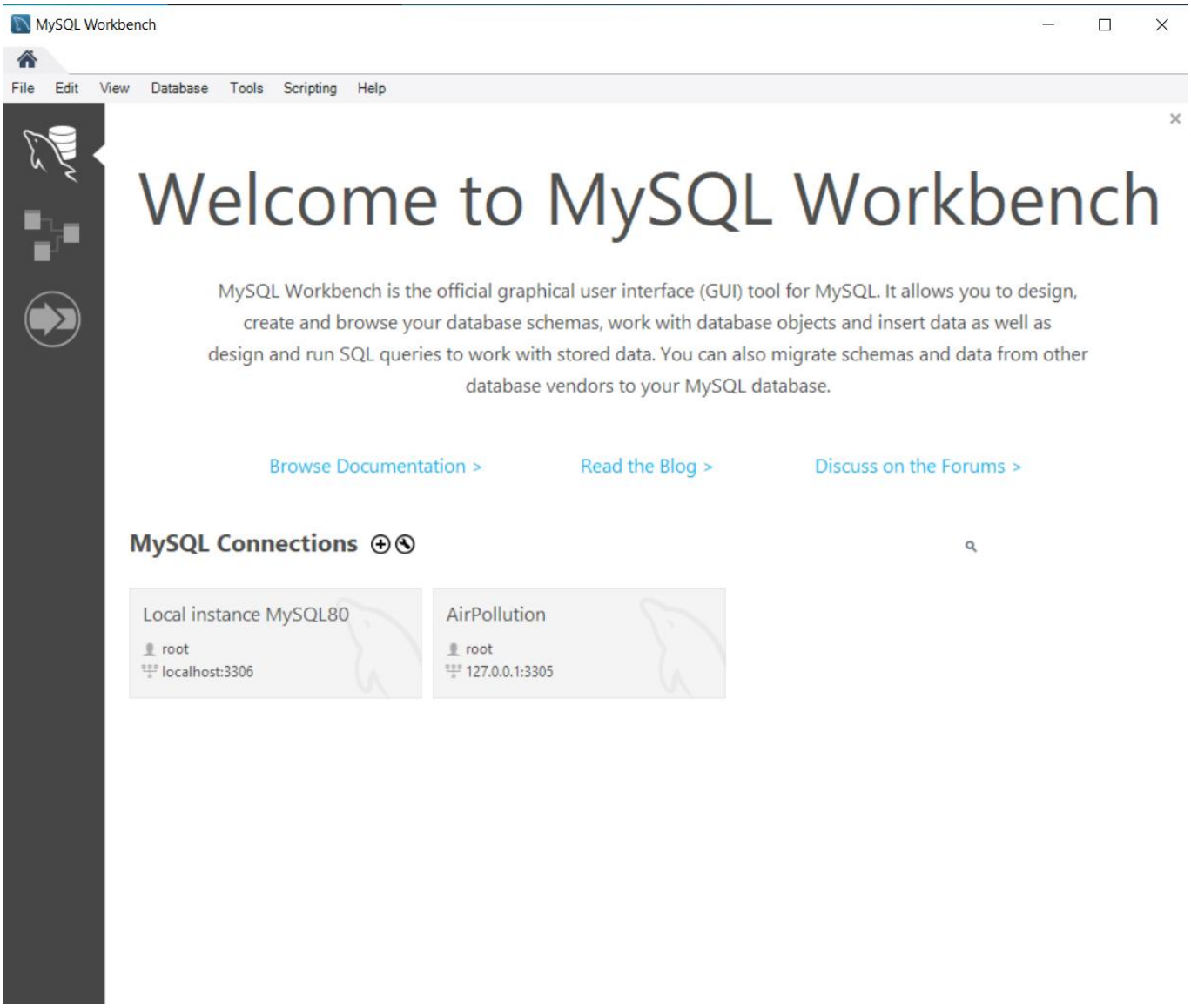


Рис. 4.7. Створенні з’єднання в MySQL Workbench

Далі переходимо на створення з’єднання AirPollution та бачимо створені БД(див.рис.4.8).

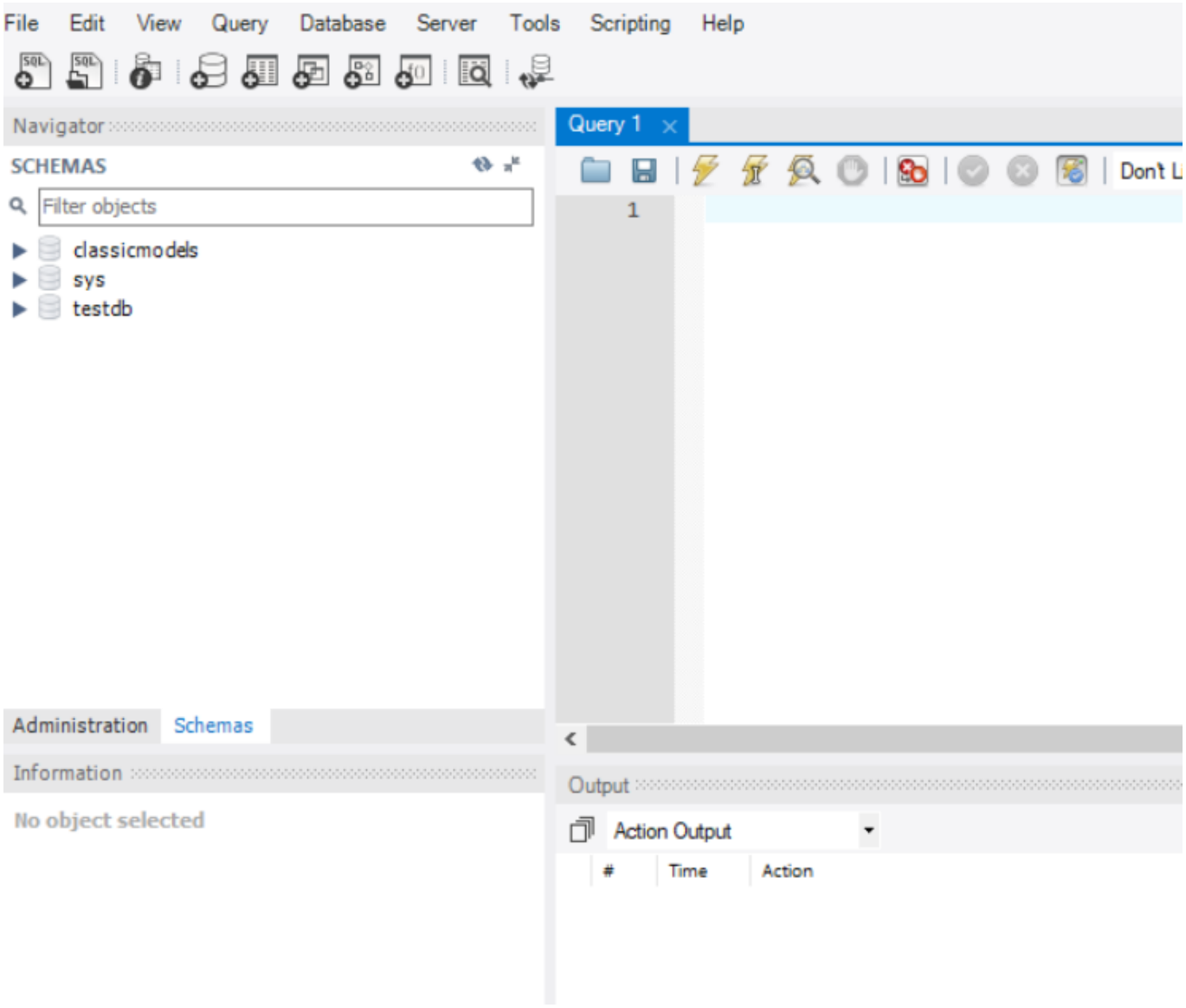


Рис. 4.8. Створені БД в MySQL Workbench

Далі, спираючись на спроектовану діаграму сутність-зв’язок (див. рис. 3.4), були описані запити для створення БД та таблиць, враховуючи усі зв’язки які були встановлені завчасно. Далі на зображені приведені скрипти створення таблиць(див.рис.4.9).

```sql
create database air_check;

    create table city(
     city_pk int(10) auto_increment primary key,
     name varchar(100) not null,
     geo varchar(200) not null,
     station varchar(200) not null,
     constraint geo_unique unique(name,geo)
    );

    create table measurements(
     measurements_pk int(10) auto_increment primary key,
     time datetime not null,
     aqi int not null,
     city_fk int(10) not null,
     FOREIGN KEY (city_fk)  REFERENCES city (city_pk)
    );
    create table user(
     user_pk int(10) auto_increment primary key,
     login varchar(100) not null unique,
     email varchar(100) not null unique,
     refresh_date datetime,
     password varchar(1000) not null
    );

    create table user_city(
     user_fk int(10) not null,
     city_fk int(10) not null,
     FOREIGN KEY (user_fk)  REFERENCES user (user_pk),
     FOREIGN KEY (city_fk)  REFERENCES city (city_pk)
    );
```

Рис. 4.9. Результат виконання скриптів створення бази даних

Початковий рядок коду на зображенні відповідає створенню бази даних і виконується лише один раз. Для виконання створення таблиць або БД потрібно виділити необхідний рядок та нажати кнопку блискавки (див.рис.4.10).

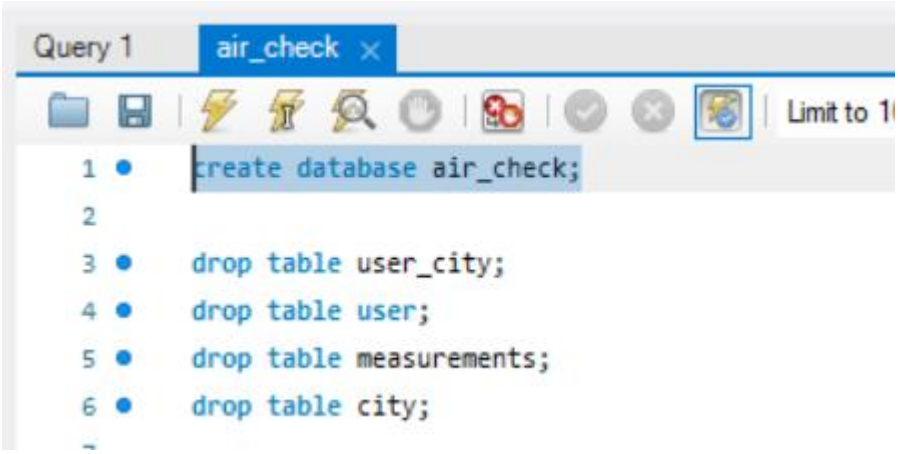
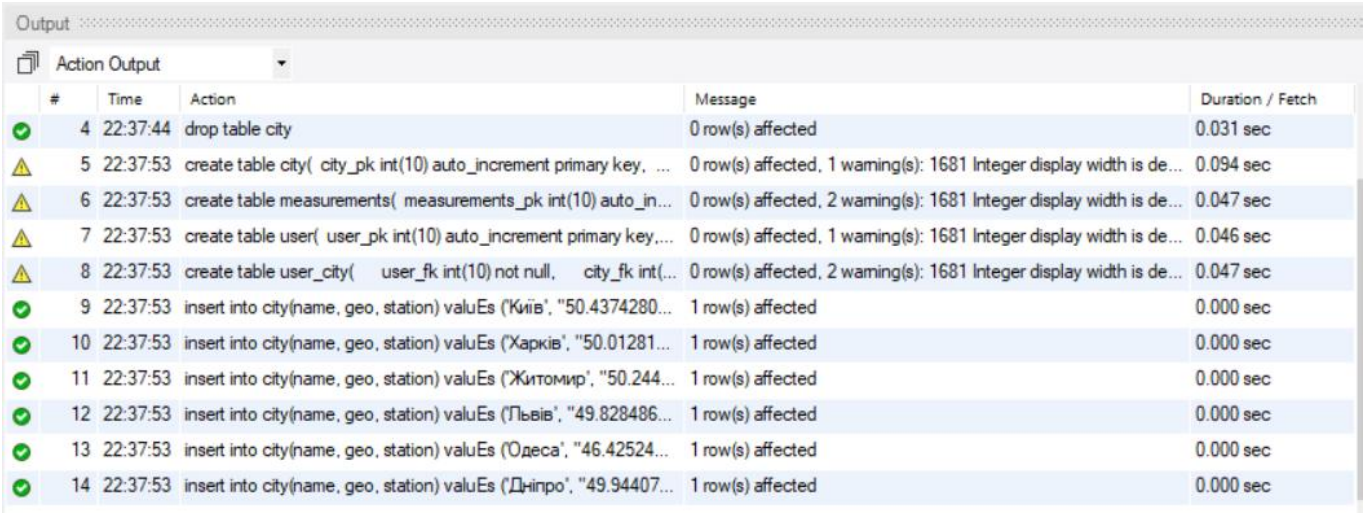


Рис. 4.10. Виконання скриптів створення бази даних

Далі наведені команди для створення таблиць; наприклад, починаючи з вісімнадцятого рядка і до двадцять четвертого включно є команда для створення таблиць користувачів. Під час створення бази даних іноді може виникнути необхідність повного очищення, що передбачає видалення всіх таблиць та їх повторне створення. Для таких випадків також написані скрипти для видалення таблиць (див. рис. 4.11.). Проте повторне виконання команди створення бази даних (перший рядок на рисунку 4.9.) не є необхідним.

```
drop table user_city;
drop table user;
drop table measurements;
drop table city;
```

Рис. 4.11. Скрипти видалення таблиць


Output

Action Output

#	Time	Action	Message	Duration / Fetch
4	22:37:44	drop table city	0 row(s) affected	0.031 sec
5	22:37:53	create table city(city_pk int(10) auto_increment primary key, ...	0 row(s) affected, 1 warning(s): 1681 Integer display width is de...	0.094 sec
6	22:37:53	create table measurements(measurements_pk int(10) auto_in...	0 row(s) affected, 2 warning(s): 1681 Integer display width is de...	0.047 sec
7	22:37:53	create table user(user_pk int(10) auto_increment primary key,...	0 row(s) affected, 1 warning(s): 1681 Integer display width is de...	0.046 sec
8	22:37:53	create table user_city(user_fk int(10) not null, city_fk int(...	0 row(s) affected, 2 warning(s): 1681 Integer display width is de...	0.047 sec
9	22:37:53	insert into city(name, geo, station) valuEs ('Київ', "50.4374280...	1 row(s) affected	0.000 sec
10	22:37:53	insert into city(name, geo, station) valuEs ('Харків', "50.01281...	1 row(s) affected	0.000 sec
11	22:37:53	insert into city(name, geo, station) valuEs ('Житомир', "50.244...	1 row(s) affected	0.000 sec
12	22:37:53	insert into city(name, geo, station) valuEs ('Львів', "49.828486...	1 row(s) affected	0.000 sec
13	22:37:53	insert into city(name, geo, station) valuEs ('Одеса', "46.42524...	1 row(s) affected	0.000 sec
14	22:37:53	insert into city(name, geo, station) valuEs ('Дніпро', "49.94407...	1 row(s) affected	0.000 sec

Рис. 4.12. Результат виконання скриптів створення бази даних

В СУБД було успішно створено таблиці, через що свідчать зелені статус коди(див.рис.4.12.)

Також були описані запити до БД, які дозволяють отримувати або видаляти виборочні дані(див.рис.4.13).

```
47  select c.city_pk, c.name, c.geo, c.station
48  from city c
49  where NOT EXISTS
50  (SELECT c.city_pk
51     from user u, user_city uc
52      where c.city_pk = uc.city_fk
53      and u.user_pk = uc.user_fk
54      and u.user_pk = "1");
55
56  select *
57  from city c
58  where c.city_pk = "1";
59
60  select m.aqi, m.time, m.measurements_pk, m.city_fk
61  from measurements m, city c
62  where m.city_fk = c.city_pk
63  and c.city_pk = 1;
64
65  select avg(m.aqi) as "aqi", m.time, m.measurements_pk, m.city_fk
66  from measurements m, city c
67  where m.city_fk = c.city_pk
68  and c.city_pk = 1;
69
70  DELETE FROM user_city
71  WHERE city_fk = 1
72  and user_fk = 1;
```

Рис. 4.13. Запити до БД

Також про успішне створення таблиць свідчить можливість утворити ER-діаграму в MySQLWorkbench(див.рис.4.14).

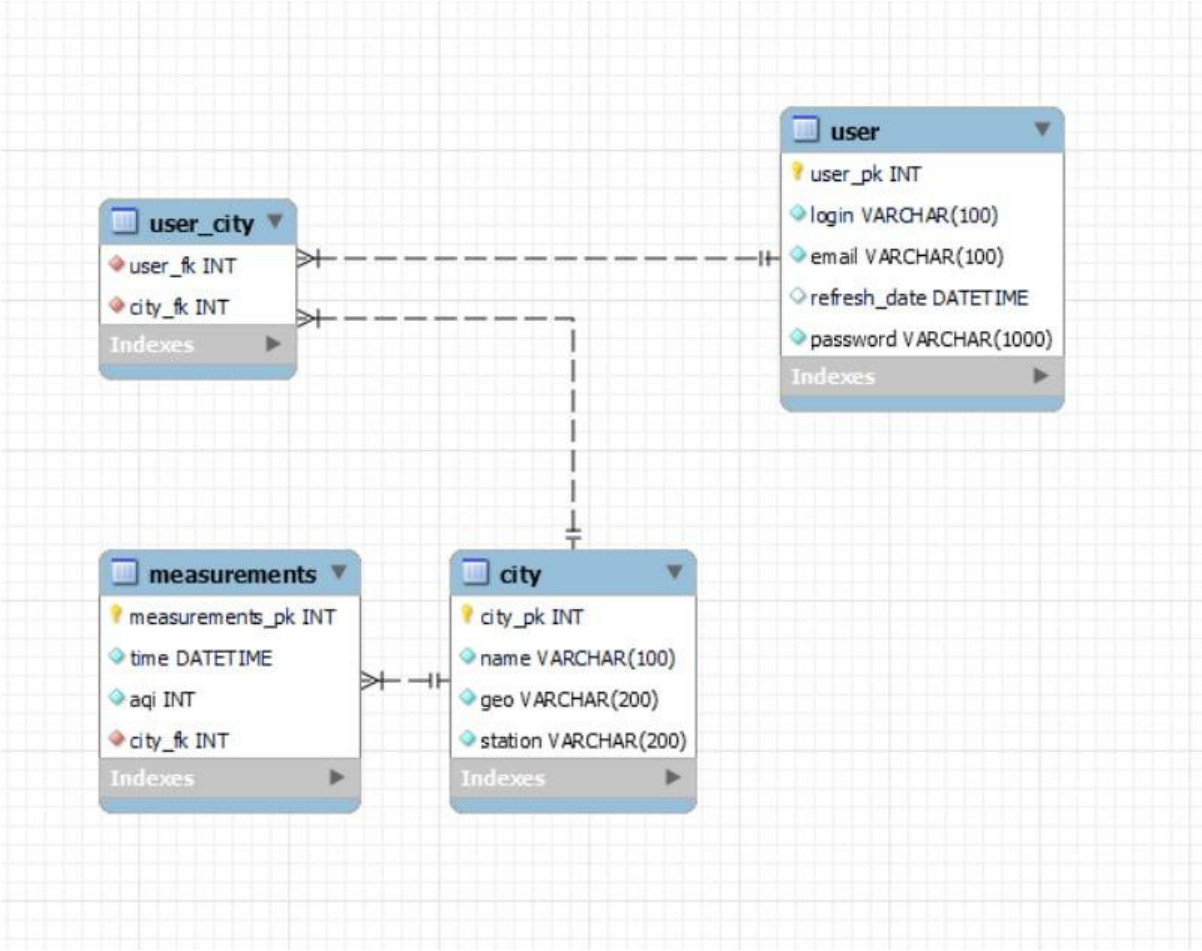


Рис. 4.14. ER-діаграма в MySQLWorkbench

4.2.2. Створення бекенду

При створенні бекенду системи було використано мову Java та ісінтрументарій Inteliji IDEA від компанії JetBrains.

Щоб створити проект, необхідно було виконати такі дії:

1. Завантажено Intellij IDEA та встановлено на ПК(див.рис.4.15).

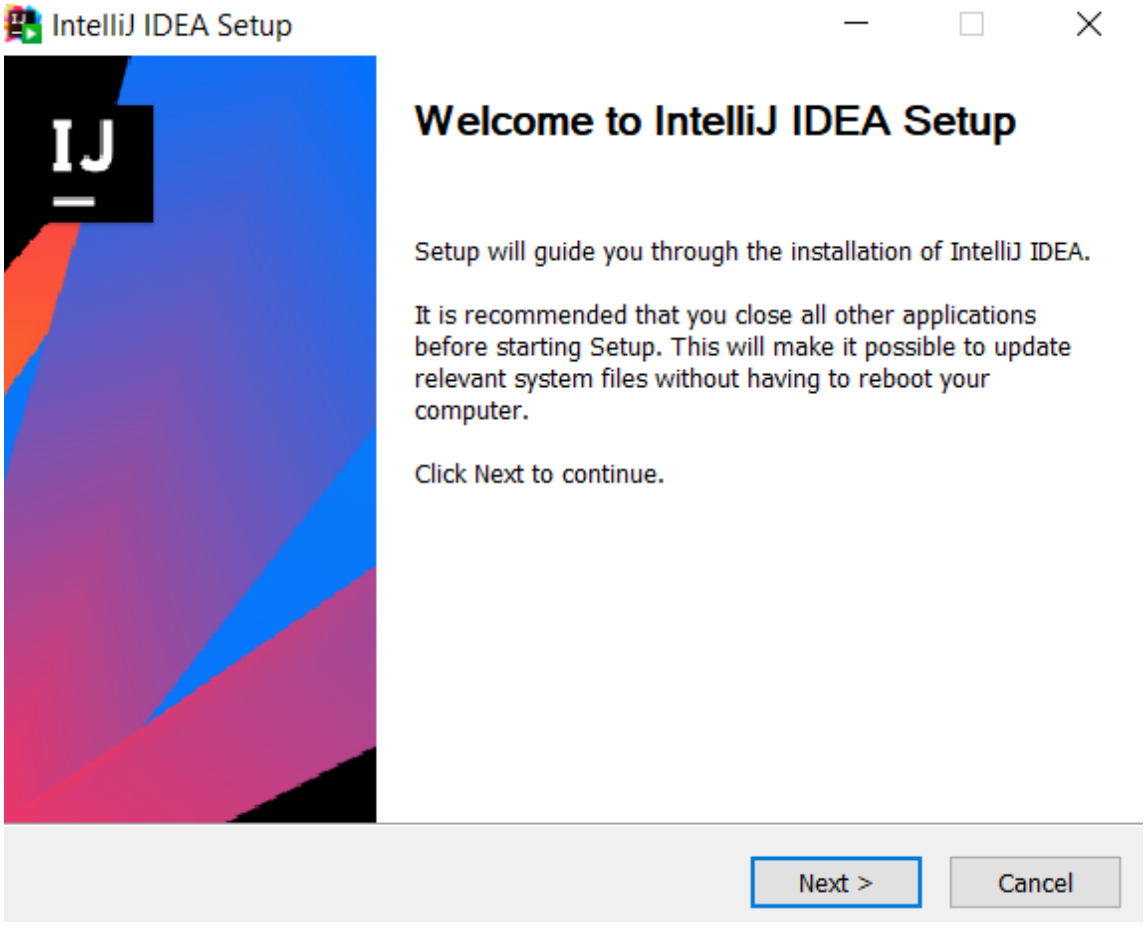


Рис. 4.15. Встановлення Intellij IDEA

2. Запущено Intellij IDEA(рис.4.16).

Рис. 4.16. Встановлення Intellij IDEA

3. Активувати ліцензію(рис.4.17).

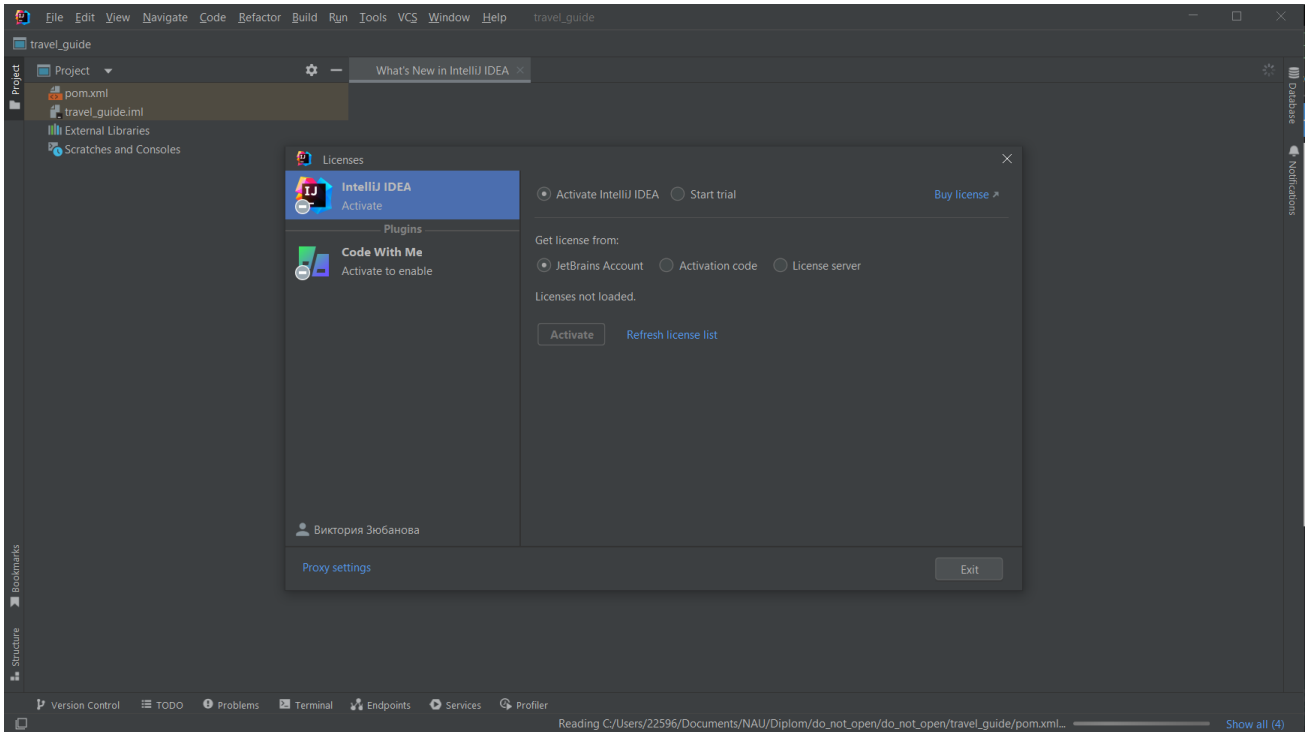


Рис. 4.17. Активація ліцензії Intellij IDEA.

4. Далі, було створено проект скориставшись шляхом “File” > “New” > “Project”(див.рис.4.18)

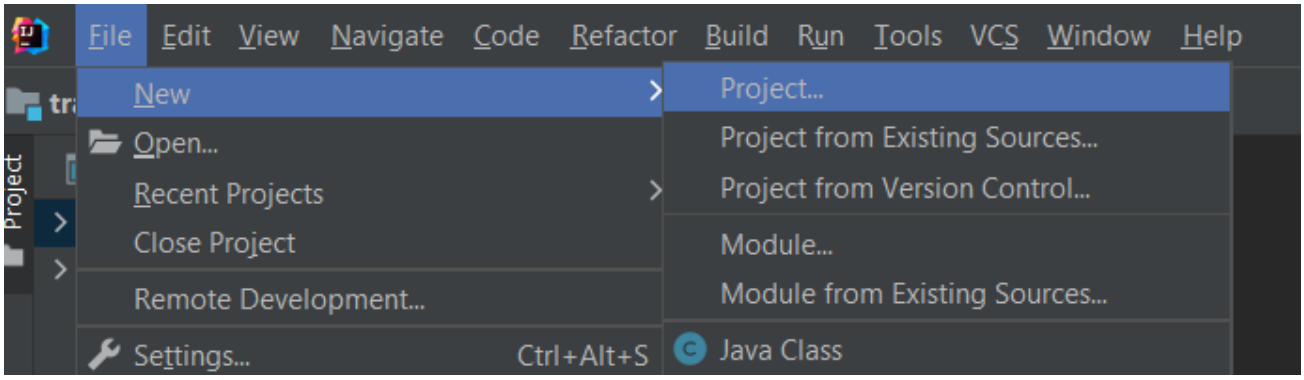


Рис. 4.18. Створення проекту

5. При створені проекту потрібно було введено назву та обрано мову програмування, білд системи та інші налаштування проекту(див.рис.4.19).

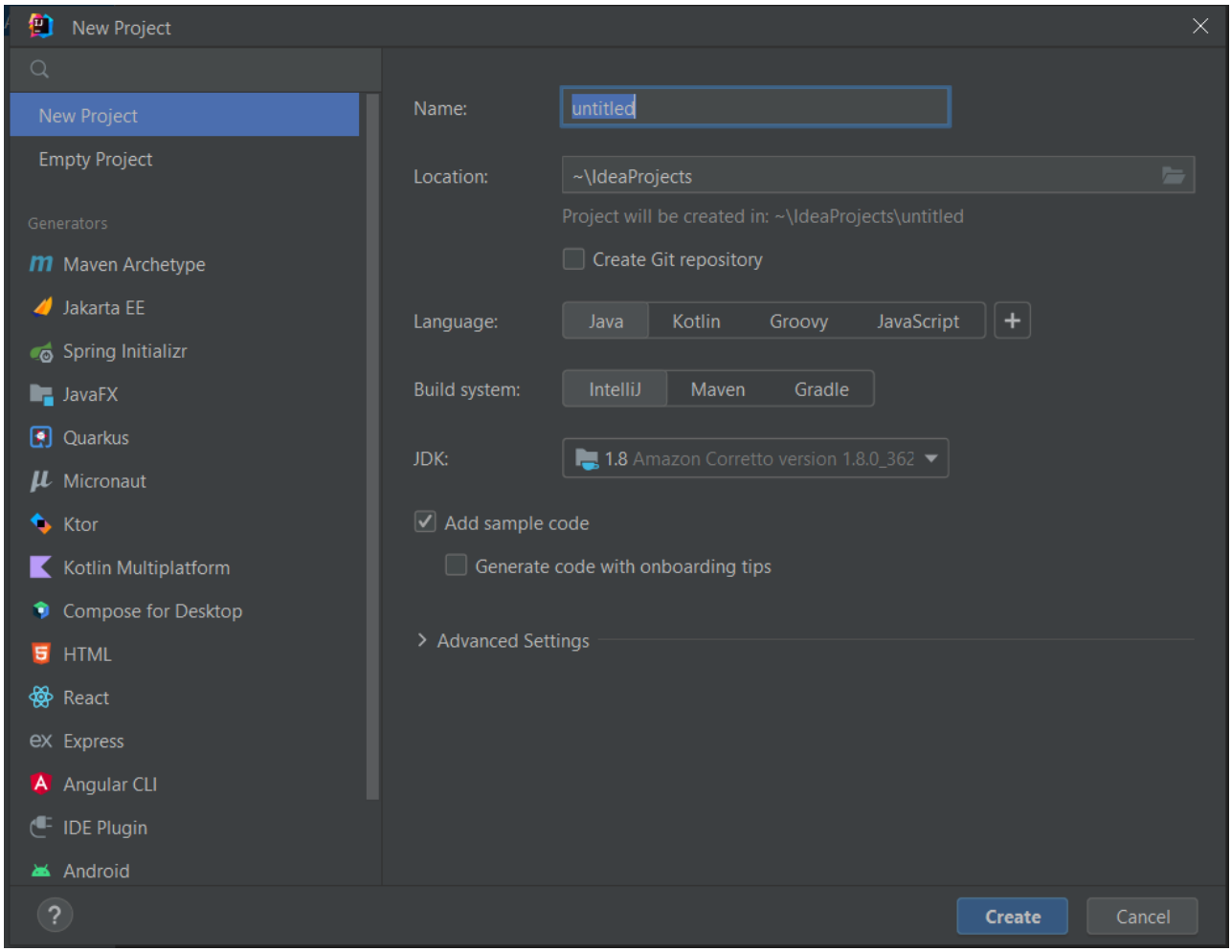


Рис. 4.19. Налаштування проекту

6. При ініціалізації проекту було створені необхідні пакети та необхідна структура додатку(див.рис.4.20)

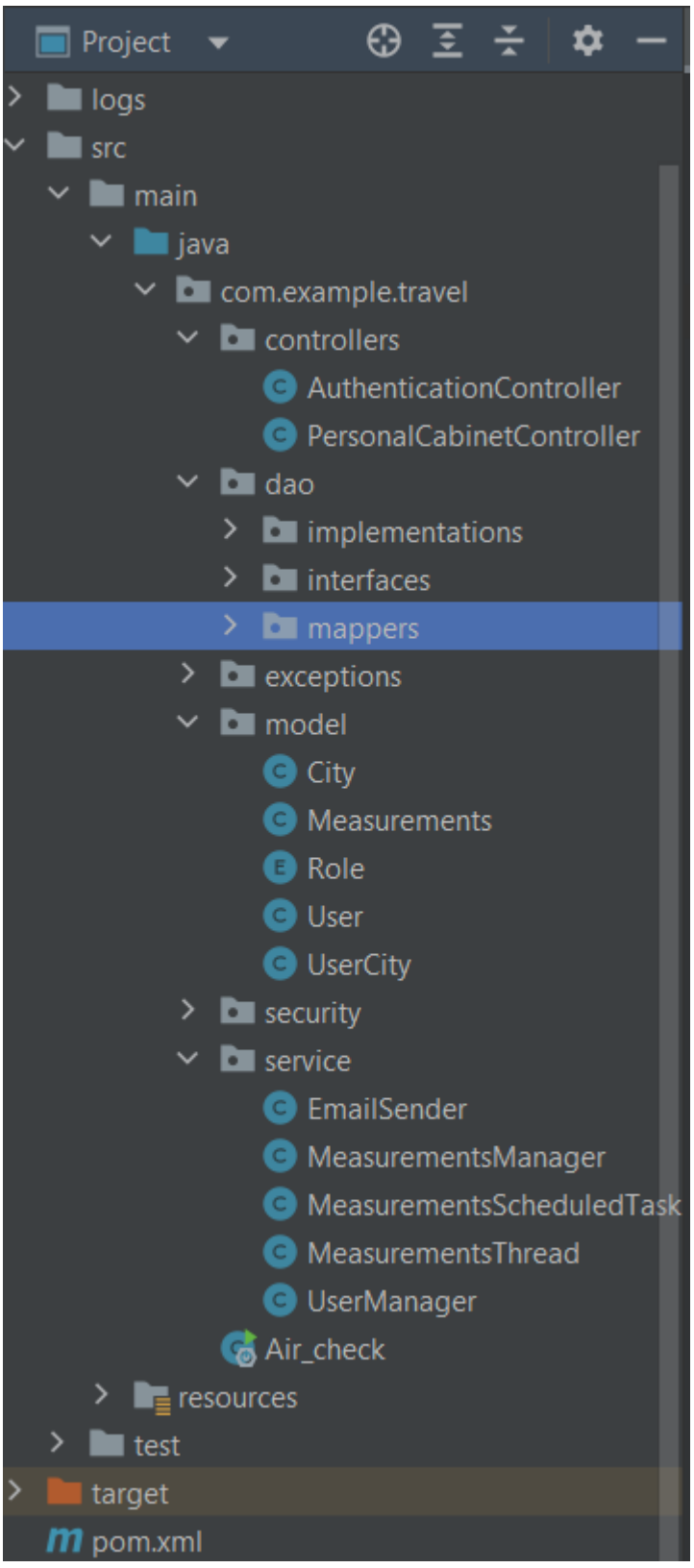


Рис. 4.20. Створення пакетної структури серверної частини додатку

Після створення пакетної структури необхідно створити файл для впровадження залежностей у проект. Завдяки залежностям до проекту можна легко додати фреймворки, бібліотеки та плагіни. Без використання цих фреймворків та

інших сторонніх матеріалів розробка програмного забезпечення стане вкрай важкою та часовитратною, адже весь функціонал доступний з використанням бібліотек та фреймворків необхідно буде писати самому. Наразі для впровадження залежностей у проект використовують різні збирачі, яскравими прикладами подібних збирачів служать Maven та Gradle. У якості збирача проекту виступає Maven.

Apache Maven – це інструмент для автоматизації процесів управління проектами в розробці програмного забезпечення. Він надає стандартні процедури для збирання, тестування та розгортання програм, а також для керування залежностями проекту.

Основні завдання, для яких використовується Maven, включають:

- **Управління залежностями:**

Maven забезпечує централізоване управління залежностями проекту, що включає в себе бібліотеки та інші залежності. Він автоматично завантажує необхідні компоненти з Інтернету та забезпечує їх актуальність.

- **Збірка проекту:**

Maven автоматизує процеси збірки, тестування та пакування програмного забезпечення. Це спрощує інтеграцію та розгортання.

- **Структуризація проекту:**

Maven накладає стандартизовану структуру на проекти, що полегшує розуміння та співпрацю розробників. Він використовує POM (Project Object Model) для централізованого управління конфігурацією проекту.

- **Централізована конфігурація:**

Збереження конфігурацій у файлах POM дозволяє централізовано встановлювати параметри для всього проекту та його модулів.

- **Генерація звітів:**

Maven може генерувати різноманітні звіти, такі як звіти про покриття коду тестами, звіти з аналізу якості коду, а також звіти для сторінок проекту.

- **Інтеграція з іншими інструментами:**

Maven легко інтегрується з різними інструментами розробки та системами для забезпечення безперервної інтеграції (CI) та постачання (CD).

- **Підтримка різних проектів:**

Maven може бути використаний для різних типів проектів та мов програмування, що робить його універсальним і підходящим для різних відомостей.

В цілому, Maven допомагає розробникам стандартизувати та автоматизувати багато процесів у розробці програмного забезпечення, забезпечуючи при цьому ефективність та стабільність у роботі з проектами.

Для керування залежностіями проекту був створений файл pom.xml, в якому було прописано усі потрібні залежності(див.рис.4.21.) .

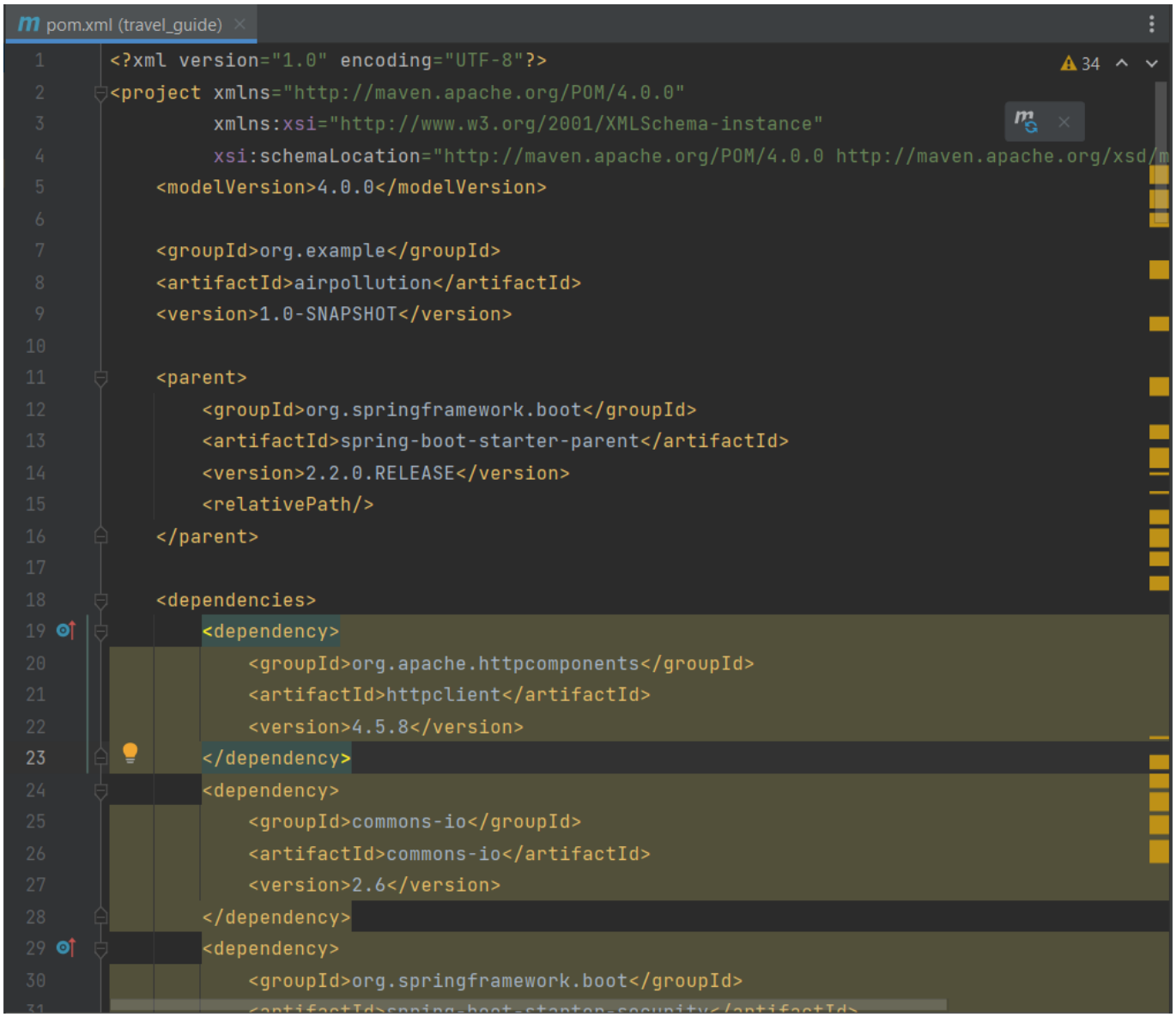


Рис. 4.21. Файл pom.xml

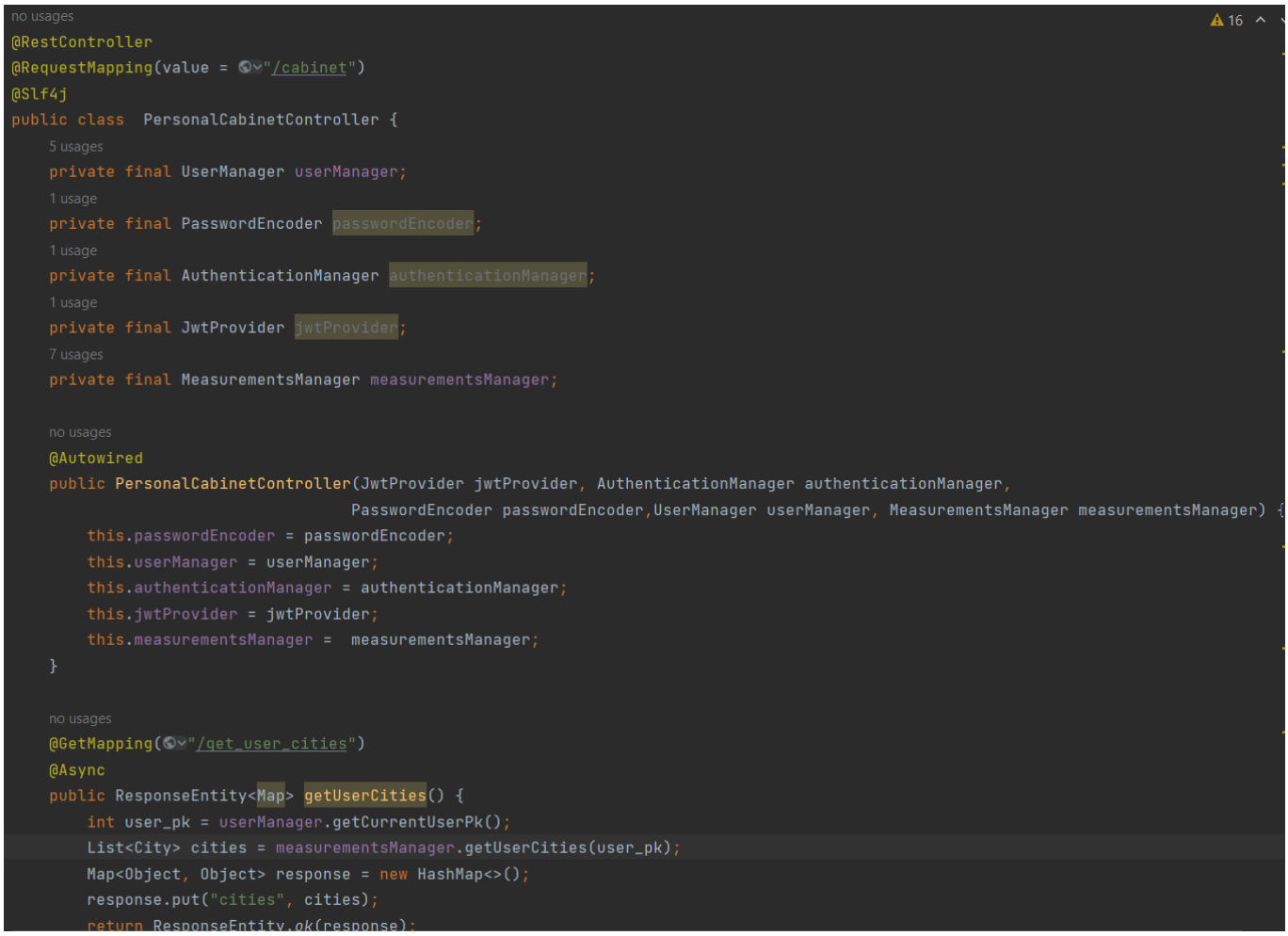
Наступними кроками було створено файли класів, та самі файли було наповненні відповідним кодом, який створює необхідний функціонал до системи моніторингу забруднення атмосферного повітря. Для прикладу було приведено класи моделей(рис.4.24) та класи контролери (рис.4.22-4.23).

```
@RestController
@RequestMapping(value = "/user_service")
@Slf4j
public class AuthenticationController {
    2 usages
    private final UserManager userManager;
    2 usages
    private final PasswordEncoder passwordEncoder;
    2 usages
    private final AuthenticationManager authenticationManager;
    2 usages
    private final JwtProvider jwtProvider;
    1 usage
    private final MeasurementsManager measurementsManager;

    no usages
    @Autowired
    public AuthenticationController(JwtProvider jwtProvider, AuthenticationManager authenticationManager,
                                    PasswordEncoder passwordEncoder,UserManager userManager,MeasurementsManager measurementsManager) {
        this.passwordEncoder = passwordEncoder;
        this.userManager = userManager;
        this.authenticationManager = authenticationManager;
        this.jwtProvider = jwtProvider;
        this.measurementsManager =measurementsManager;
    }

    no usages
    @PostMapping("/signup")
    @Async
    public ResponseEntity<Map> register(User user){
        try {
            log.info("POST /register/user [{}, {}]", user.getLogin(), user.getEmail());
            user.setPassword(passwordEncoder.encode(user.getPassword()));
            userManager.register(user);
        }catch (LoginExistException e){
            Map<Object, Object> response = new HashMap<>();
            response.put("message",e.getMessage());
            return ResponseEntity.status(HttpStatus.CONFLICT).body(response);
        }catch (EmailExistException e){
```

Рис.4.22. Клас контролер який приймає запити на авторизацію та реєстрацію



```
no usages                                                                                                    ⚠ 16
@RestController
@RequestMapping(value = "/cabinet")
@Slf4j
public class  PersonalCabinetController {
    5 usages
    private final UserManager userManager;
    1 usage
    private final PasswordEncoder passwordEncoder;
    1 usage
    private final AuthenticationManager authenticationManager;
    1 usage
    private final JwtProvider jwtProvider;
    7 usages
    private final MeasurementsManager measurementsManager;

    no usages
    @Autowired
    public PersonalCabinetController(JwtProvider jwtProvider, AuthenticationManager authenticationManager,
                                     PasswordEncoder passwordEncoder,UserManager userManager, MeasurementsManager measurementsManager) {
        this.passwordEncoder = passwordEncoder;
        this.userManager = userManager;
        this.authenticationManager = authenticationManager;
        this.jwtProvider = jwtProvider;
        this.measurementsManager =  measurementsManager;
    }

    no usages
    @GetMapping("/get_user_cities")
    @Async
    public ResponseEntity<Map> getUserCities() {
        int user_pk = userManager.getCurrentUserPk();
        List<City> cities = measurementsManager.getUserCities(user_pk);
        Map<Object, Object> response = new HashMap<>();
        response.put("cities", cities);
        return ResponseEntity.ok(response);
```

Рис.4.23. Клас контролер кабінету користувача

Рис. 4.24. Код класу моделі яка відповідає за користувача

Також було налаштоване підключення до БД, де прописувалась строка підключення та користувач з паролем(див.рис.4.25).

```
spring.mail.default-encoding=UTF-8
spring.mail.host=smtp.gmail.com
spring.mail.username=MonitoringServiceNoReply@gmail.com
spring.mail.password=MonSer123
spring.mail.port=2525
spring.mail.protocol=smtp
spring.mail.test-connection=false
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

spring.servlet.multipart.max-file-size = 1MB
spring.servlet.multipart.max-request-size = 1MB

jwt.token.secret=8f58811c1b7015d09535008e5aa6d2747fba52b71a81500e03b418
jwt.token.expired=3600000
jwt.token.secondPause=1000

spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/air_check?serverTimezone=UTC
spring.datasource.username=user
spring.datasource.password=user
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

logging.file.name=./logs/app-logs.log
```

Рис. 4.25. Підключення до БД

Для работи с API датчиків, було обрано систему https://aqicn.org/, яка дозволяє отримати доступ до великої кількості даних, які надходять від датчиків забруднення повітря, API надає доступ до географічного розташування(на основі широти/довготи або IP-адреси), дозволяє отримувати точні погодні умови та шукати станції за назвами. Для роботи з нею був отриманий унікальний токен, який був прописаний в MeasureManager (див.рис.4.26).

```java
public class MeasurementsManager {
    10 usages
    private MeasurementsRepositoryInterface measurementsRepository;
    1 usage
    private final CloseableHttpClient httpClient = HttpClients.createDefault();
    1 usage
    Timer time = new Timer();

    no usages
    @Autowired
    public MeasurementsManager(MeasurementsRepositoryInterface measurementsRepository) {
        this.measurementsRepository = measurementsRepository;
    }

    3 usages
    public Measurements getMeasurements(int city_pk) throws IOException, JSONException {
        Measurements measurements = new Measurements();
        City city = measurementsRepository.getCity(city_pk);
        HttpGet requestTo = new HttpGet( uri: "https://api.waqi.info/feed/"+city.getStation()+"/?token=7a312d89e27090a27344164d2e9bb
        requestTo.addHeader( name: "custom-key",  value: "mkyong");
        requestTo.addHeader(HttpHeaders.USER_AGENT,  value: "Googlebot");
        CloseableHttpResponse responseFromUrl = httpClient.execute(requestTo);

        if(responseFromUrl.getStatusLine().getStatusCode() == 200) {
            BufferedReader in = new BufferedReader(new
                    InputStreamReader(responseFromUrl.getEntity().getContent()));

            StringBuffer sb = new StringBuffer("");
            String line = "";

            while ((line = in.readLine()) != null) {
                sb.append(line);
                break;
```

Рис. 4.26. Measure Manager

Після створення всіх необхідних класів та налаштувань бази даних, був проведений тестовий запуск серверної частини системи. Це було зроблено для перевірки відсутності помилок та підтвердження успішного початку роботи програмного забезпечення(рис.4.27).

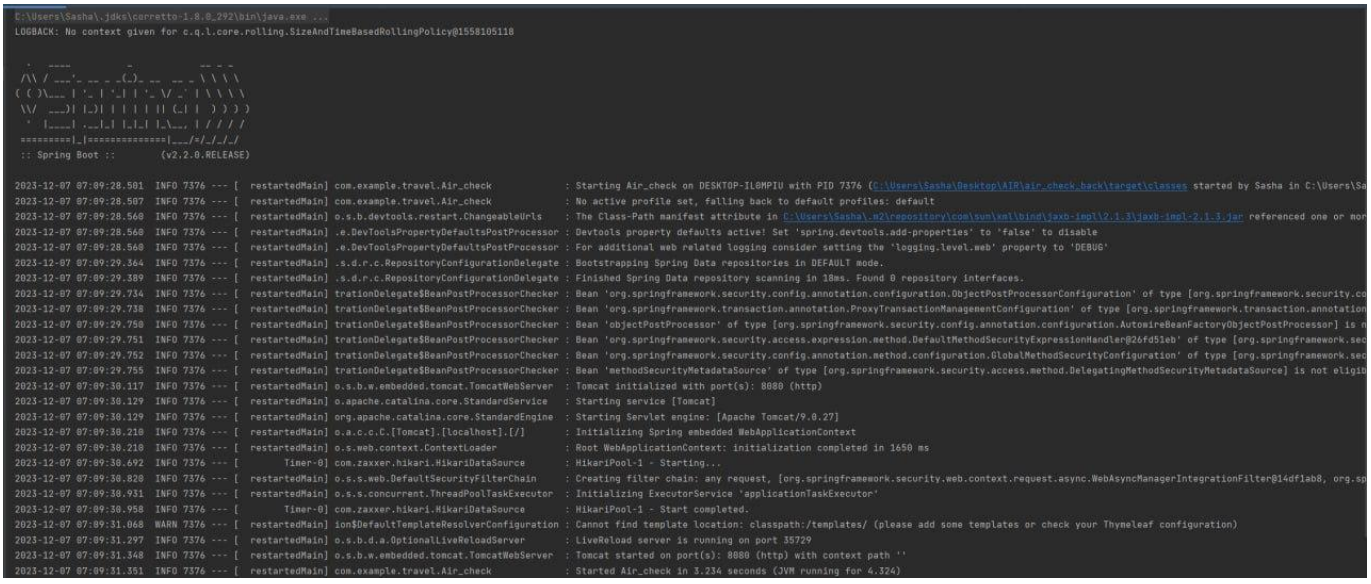


Рис. 4.27. Результат успішного запуску серверу системи

4.2.3. Створення мобільного додатку для інформаційної системи моніторингу забруднення атмосферного повітря

Після створення бекенду системи, було розпочато роботу над мобільним додатком, який відповідає за фронтенд частину системи. Розробка відбувається в середовищі Android Studio, чому обрана вона і які переваги несе, описано в пункті 4.1.

Отже на початку роботи над мобільним додатком була створена файлова структура проекту, створення її на початку несе в собі такі переваги як:

- Структуровані проекти забезпечують зрозуміліше розміщення коду
- Якщо проект зростає в розмірах, добре спроектована файлова структура полегшує масштабування.
- Модульна структура дозволяє визначити окремі компоненти та функціонально розділити їх.
- Коректно структурований проект полегшує додавання та управління залежностями.

Отже, переглянути, як виглядає файлова структура можно на рис.4.28.

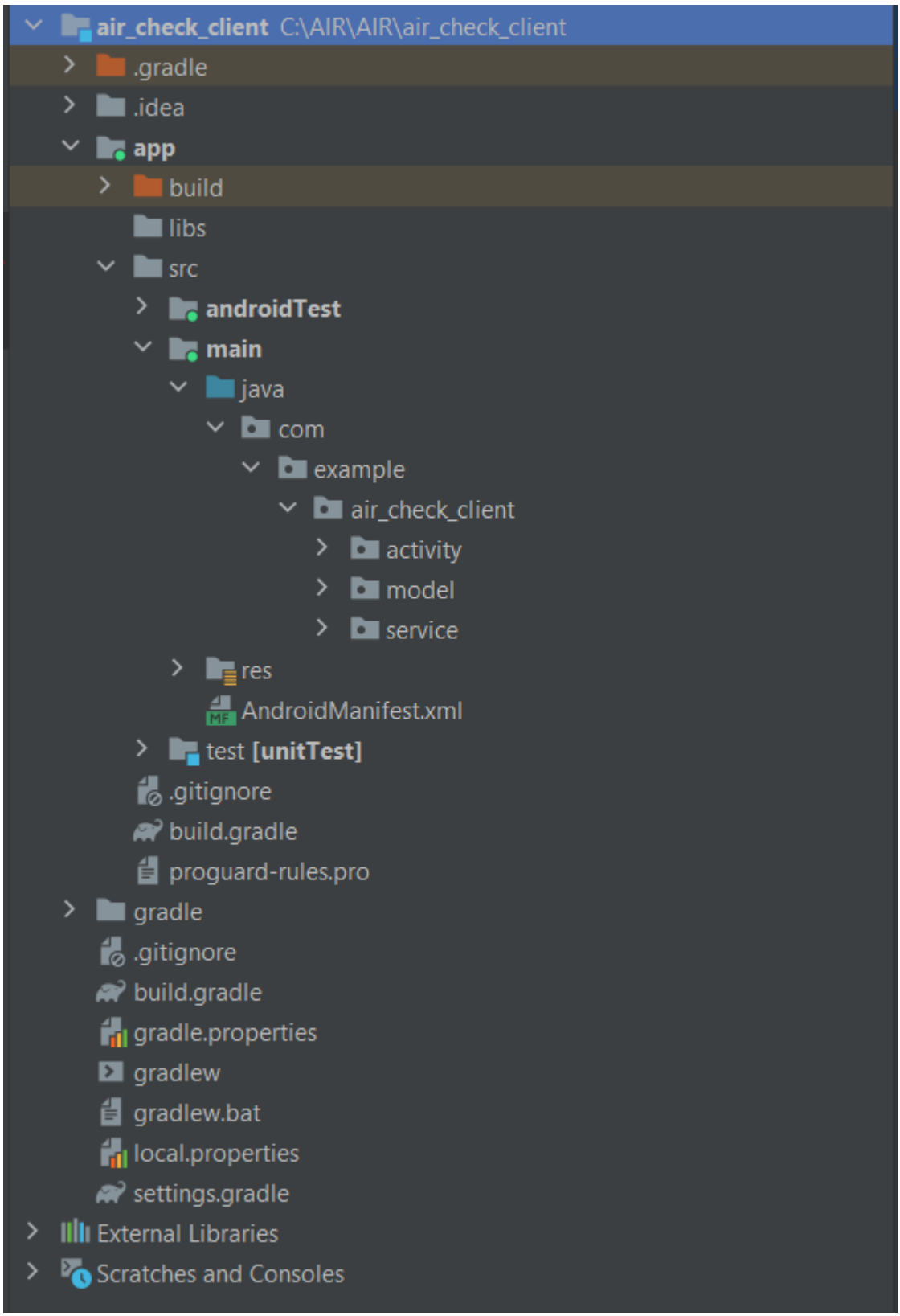


Рис. 4.28. Пакетна структура мобільного додатку

Після створення файлової структури, було створено файл залежностей(див.рис.4.29) та написані необхідні класи системи(рис.4.30).

```
plugins {
    id 'com.android.application'
}

android {
    namespace 'com.example.air_check_client'
    compileSdk 32

    defaultConfig {
        applicationId "com.example.air_check_client"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
```

Рис. 4.29. Файл з залежностями для мобільного додатку

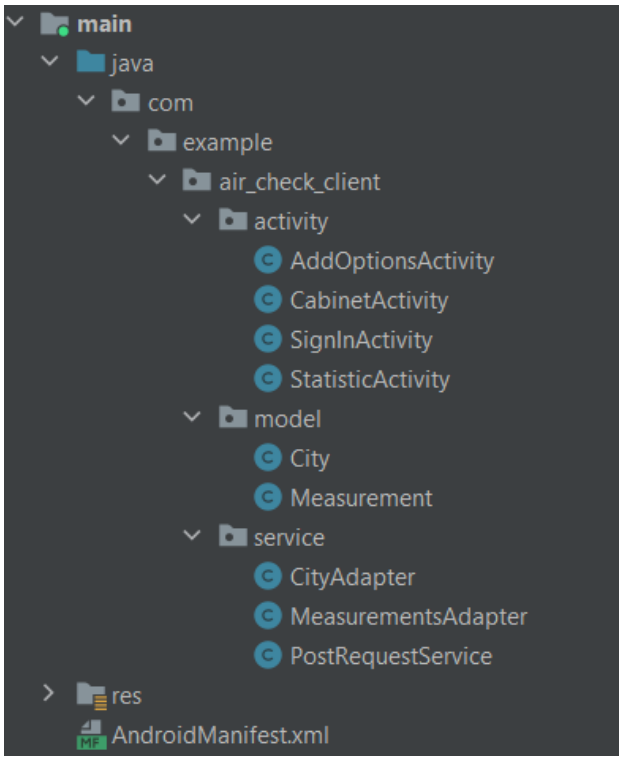


Рис. 4.30. Список класів системи

Далі був розроблений інтерфейс системи за допомогою інтегрованого візуального редактору інтерфейсу ”Layout editor”, завдяки йому можна створити інтерфейс, перетягуючи та розміщуючи елементи за допомогою миші або клавіатури(див.рис.4.31).

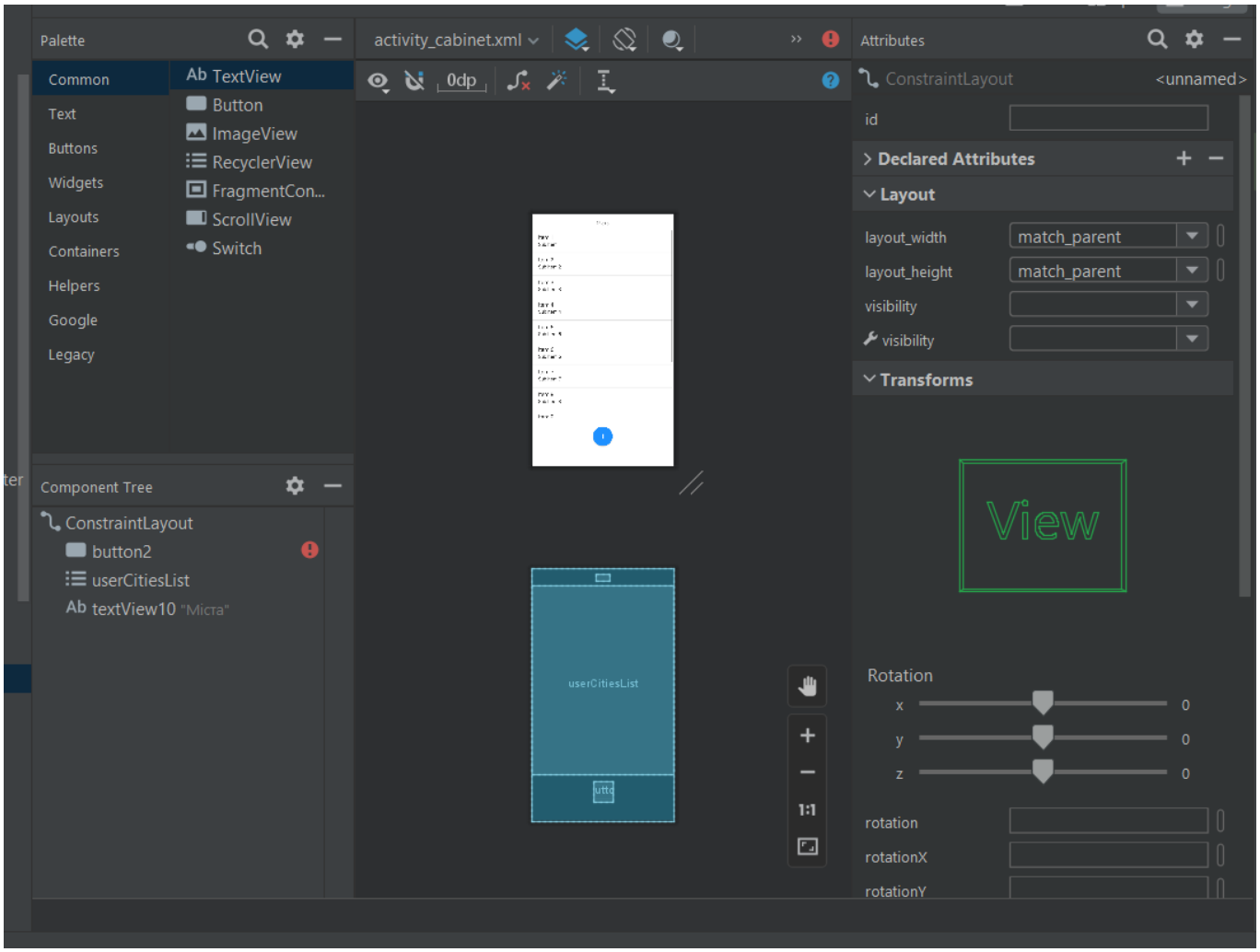


Рис. 4.31. Зовнішній вигляд Layout Editor

Layout Editor включає в себе такі компоненти як:

- Palette - панель інструментів, яка містить різноманітні елементи для вставки в макет, такі як кнопки, текстові поля, зображення б світчери, списки, карти і багато інших елементів, які можна вставити в макет. Підтримує функцію Drag-and-Dropб тобто можно перетягнути елементи з панелі.
- Attributes (властивості) – це панель інструментів, яка дозволяє розробникам налаштовувати різні властивості обраних елементів інтерфейсу

користувача в режимі дизайну. Властивості розділені на категорії, такі як "Layout", "Text", "Appearance", "ID" тощо, щоб легше знаходити і редагувати конкретні параметри. Можно редагувати значення властивостей, є автоматичні підказки і довідка про те, які значення можливі для конкретної властивості. Також можливий перехід по коду, тобто ви можете перейти до значення в XML-коді і відредагувати його там(див.рис.4.32).

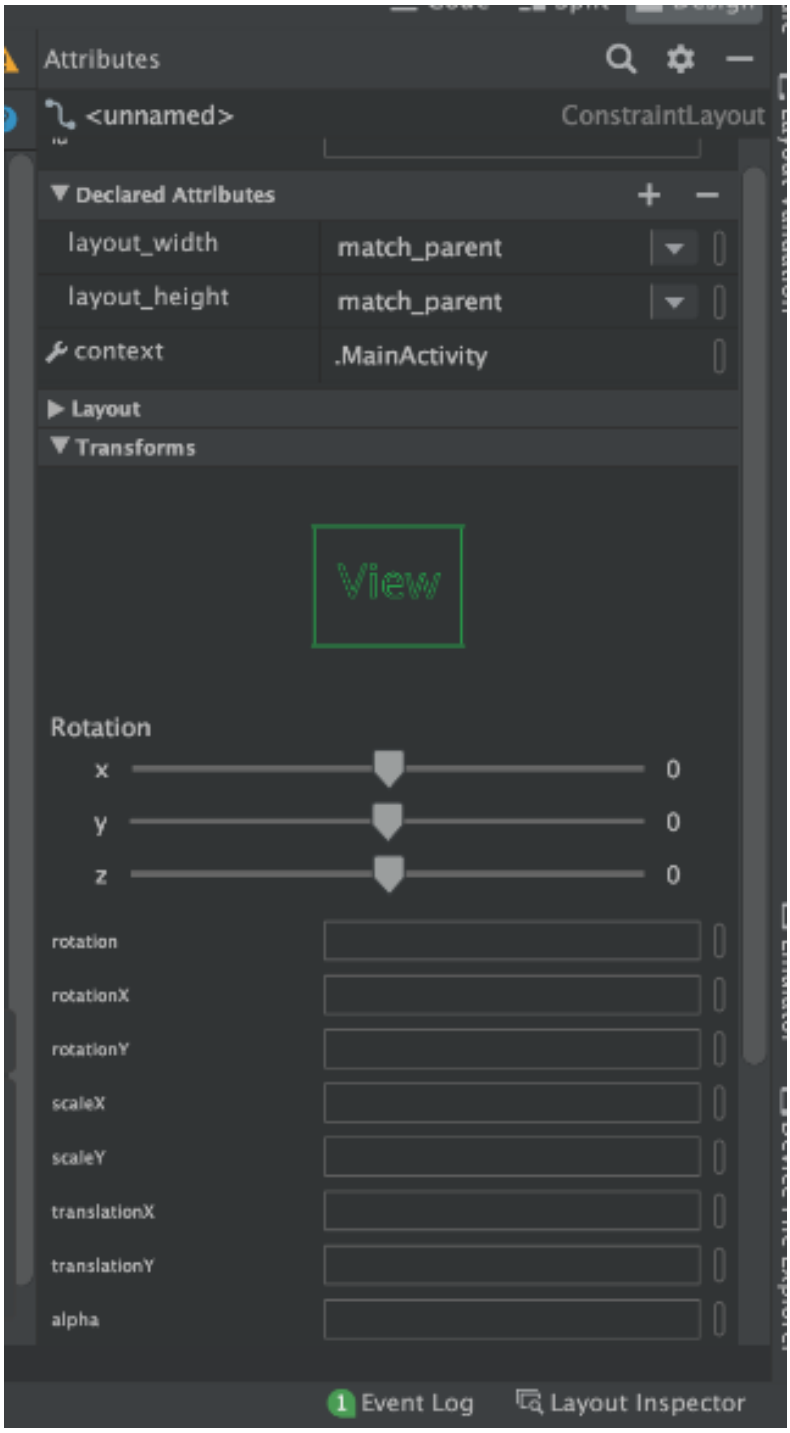


Рис. 4.32. Зовнішній вигляд Layout Editor

- ConstraintLayout Editor – це інструмент, який дозволяє визначати відносини і обмеження між елементами інтерфейсу, можно створювати взаємозв’язки між елементами в контейнері, контролювати виривнювання, розміри і поведінку елементів(див.рис.4.33).

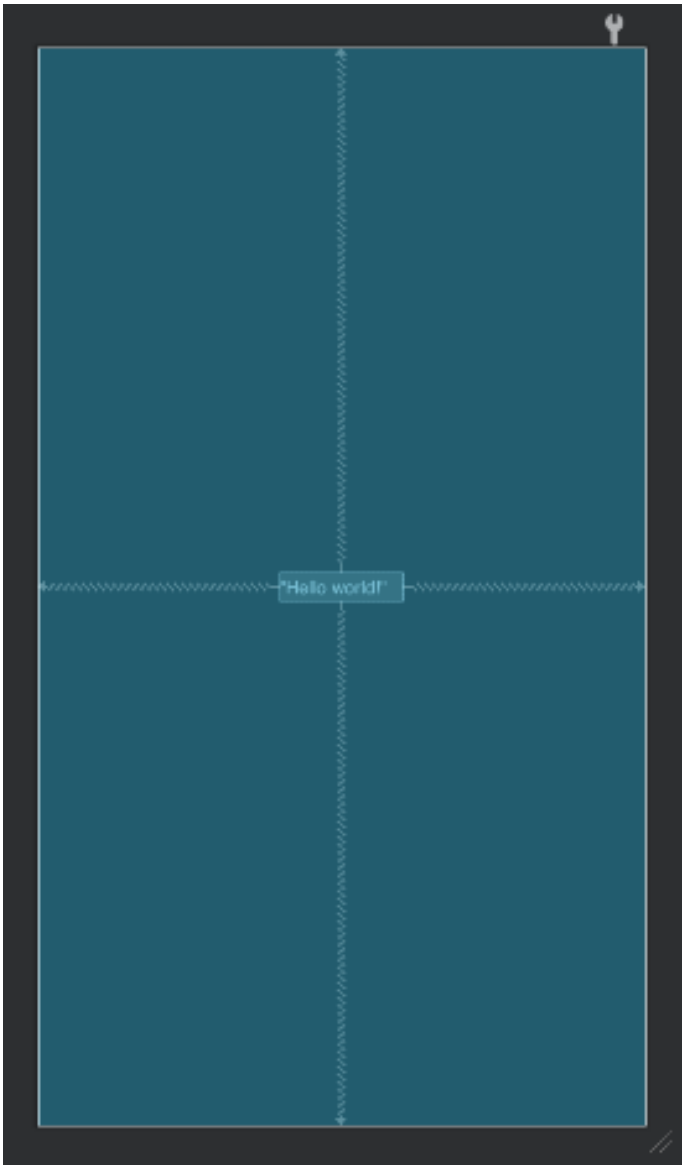


Рис. 4.33. Зовнішній вигляд ConstraintLayout Editor

- Vector Asset Studio - це інструмент призначений для роботи з векторною графікою. Він дозволяє виконувати імпорт векторних зображень як з комп'ютеру так і з Google Material Design і Adaptive Icons. Також можна налаштовувати різні параметри векторних зображень, такі як колір, розмір і інші властивості .

- Device Emulator - це вбудований емулятор пристроїв в Android Studio, який дозволяє розробникам тестувати свої додатки на різних віртуальних пристроях Android. Також можна вибрати різні версії андроїду, конфігурації екрану та характеристики пристрою.

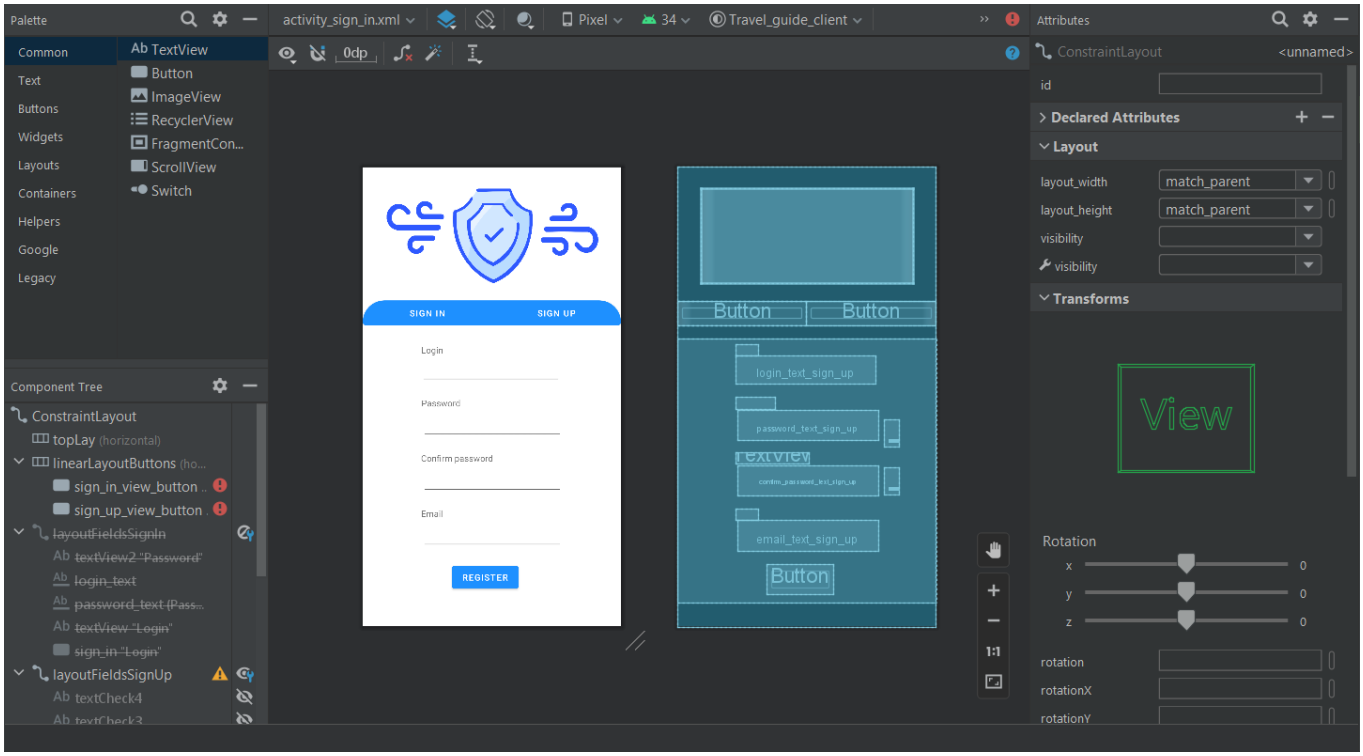


Рис. 4.34. Процес створення сторінки авторизації

Процес створення сторінок можно розглянути на рис.4.34-4.36. Були прописана основна бізнес логіка процесу реєстарації та авторизації(див.рис.4.37).

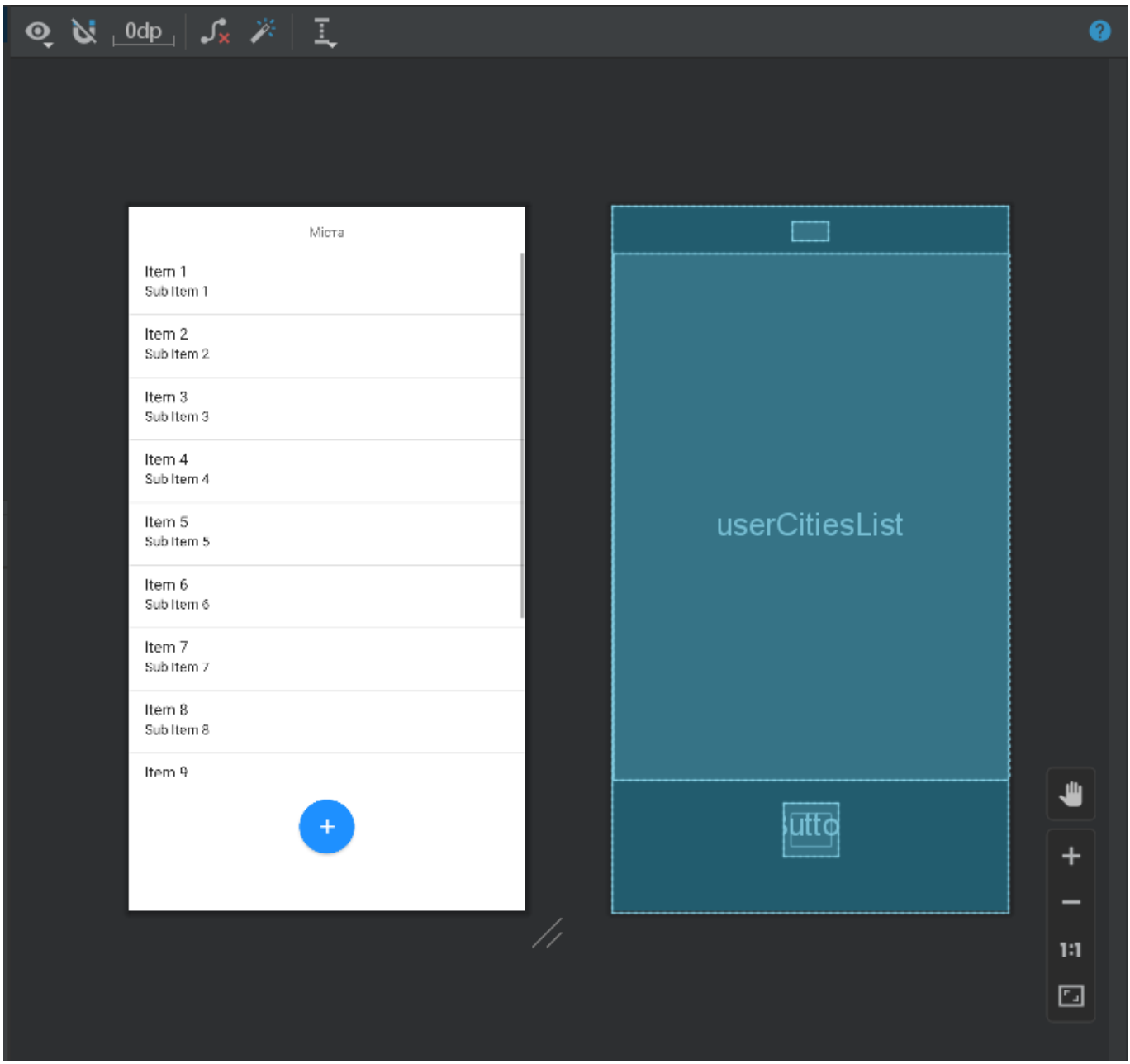


Рис. 4.35. Процес створення сторінки кабінету

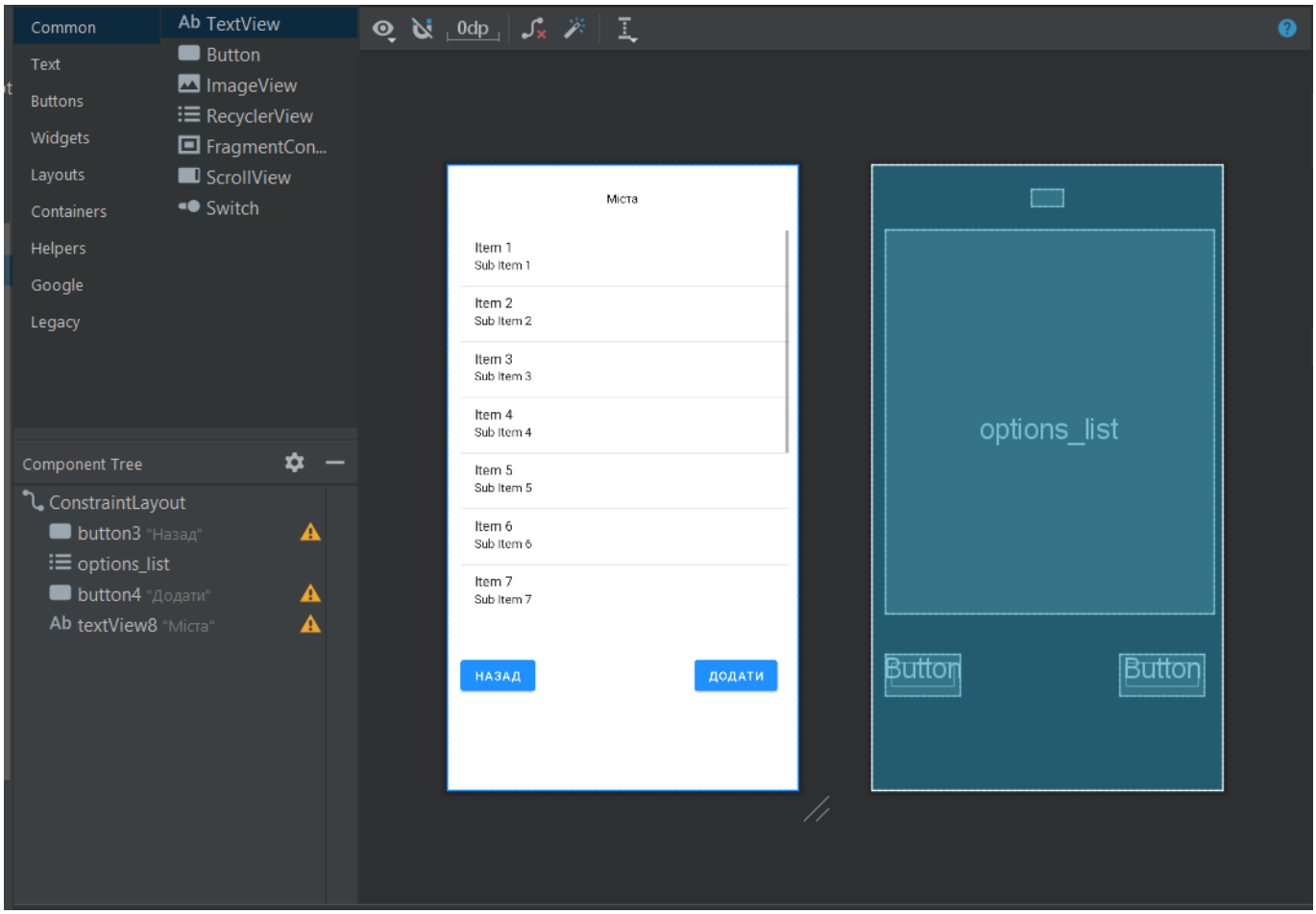
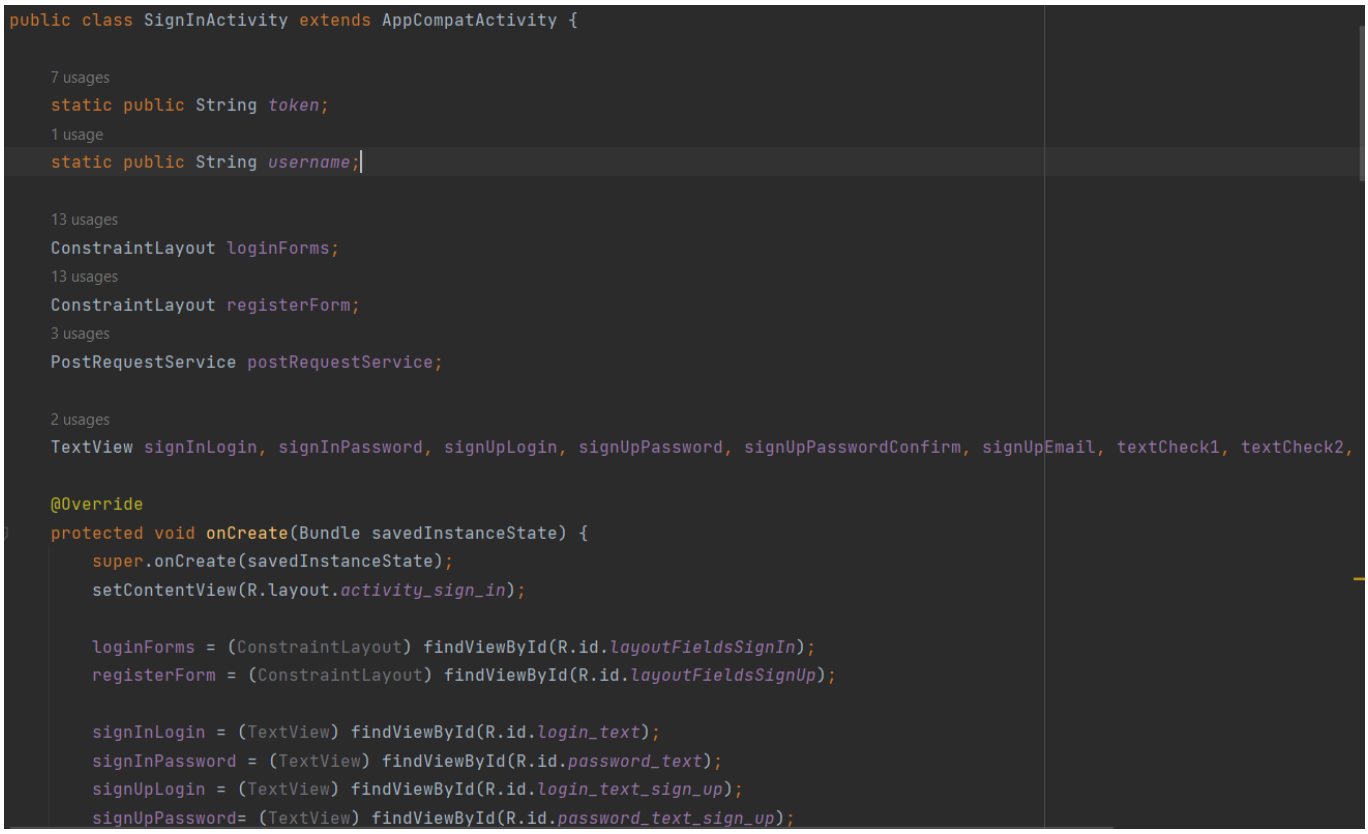


Рис. 4.36. Процес створення сторінки додавання міст

При додаванні елементів на сторінку, можна переглянути як пошарово вони додані, дивитись у лівий нижній куток на рисунку 4.36.



```
public class SignInActivity extends AppCompatActivity {

    7 usages
    static public String token;
    1 usage
    static public String username;

    13 usages
    ConstraintLayout loginForms;
    13 usages
    ConstraintLayout registerForm;
    3 usages
    PostRequestService postRequestService;

    2 usages
    TextView signInLogin, signInPassword, signUpLogin, signUpPassword, signUpPasswordConfirm, signUpEmail, textCheck1, textCheck2,

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

        loginForms = (ConstraintLayout) findViewById(R.id.layoutFieldsSignIn);
        registerForm = (ConstraintLayout) findViewById(R.id.layoutFieldsSignUp);

        signInLogin = (TextView) findViewById(R.id.login_text);
        signInPassword = (TextView) findViewById(R.id.password_text);
        signUpLogin = (TextView) findViewById(R.id.login_text_sign_up);
        signUpPassword= (TextView) findViewById(R.id.password_text_sign_up);
```

Рис. 4.37. SignInActivity

Далі було виконано тестовий запуск сторінки авторизації та перевірено її роботу(рис.4.38). Створена сторінка виявилась робочою та успішною.

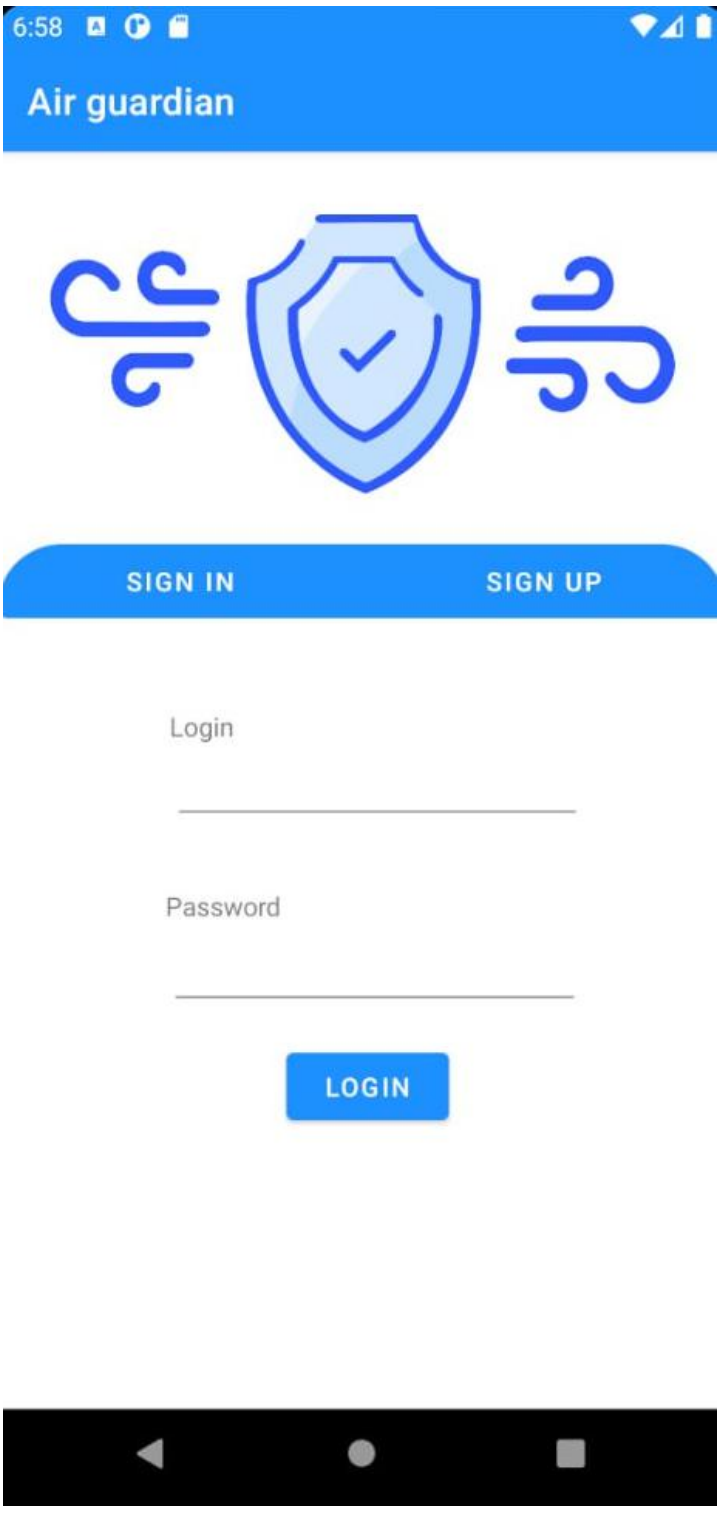


Рис. 4.38. Сторінка авторизації мобільно додатку

Далі відбулися схожі дії з кожною сторінкою додатку.

4.3. Інтерфейс системи

Отже, як було зазначено вище, для реалізації інтерфейсу використовувалась інтегрований візуальний редактор Android Studio, було створено кожну сторінку додатку завдяки йому та прописано логіку процесів, далі продемонстровано сторінки додатку(рис.4.13-4.19).

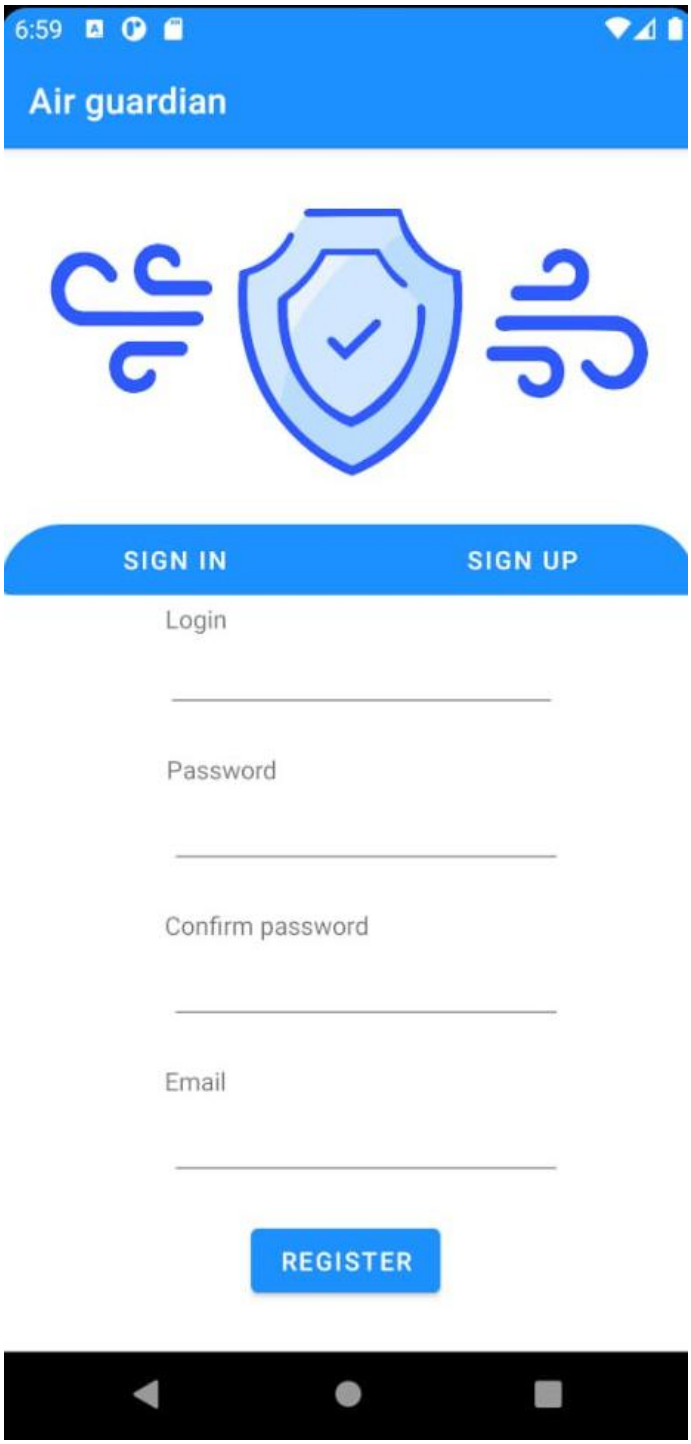


Рис. 4.39. Сторінка реєстрації

ис. 4.40. Сторінка «Кабінет»

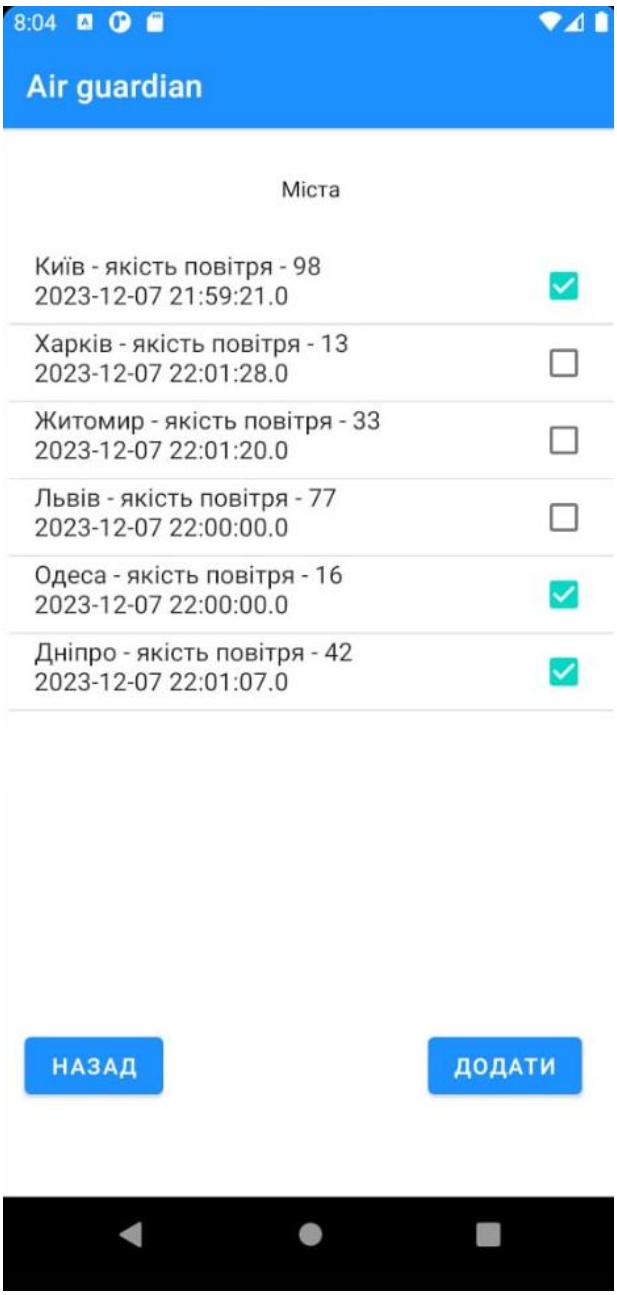


Рис. 4.41. Сторінка «Обрання міст»

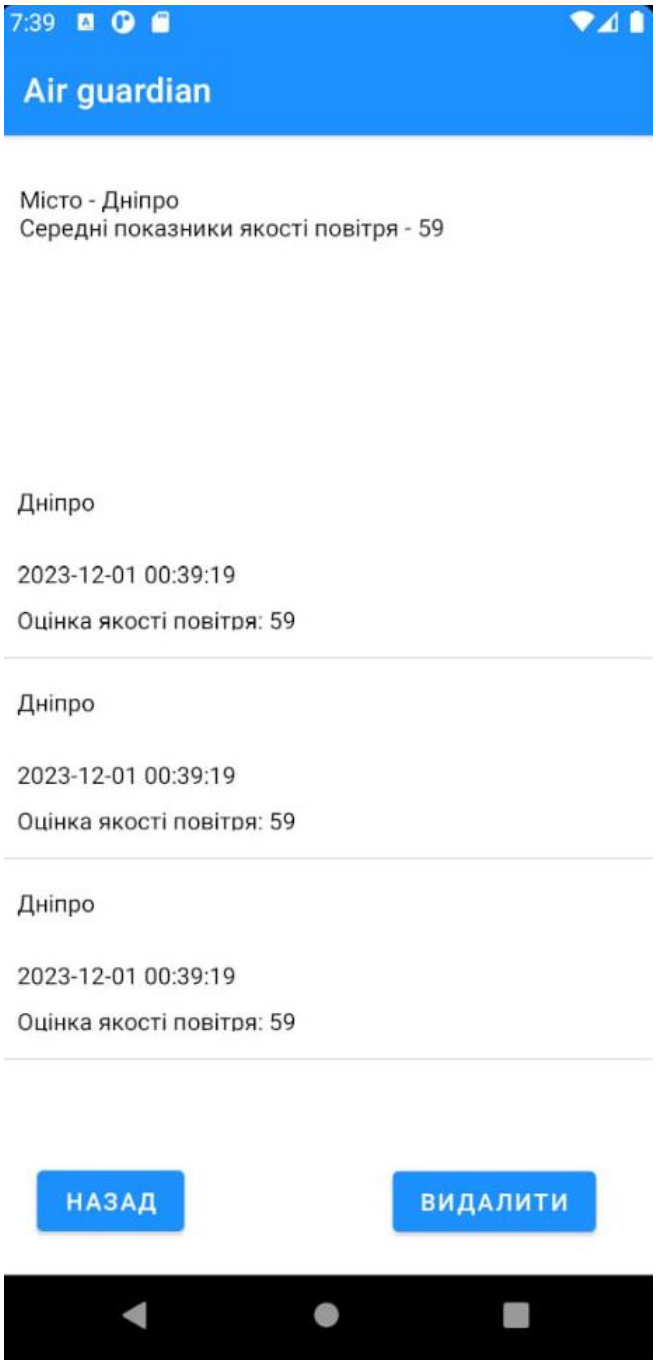


Рис. 4.42. Статистика по місту

Висновки до розділу

Для виконання роботи були використані сучасні передові технології, такі як мова програмування Java, середа розробки Intellij IDEA, СУБД MySQL та Android Studio, що надають широкий функціонал при розробці додатків.

Створений інтерфейс, розроблений за допомогою інтегрованого візуального редактора в Android Studio, дозволяє ефективно взаємодіяти з користувачем та надає зручний та привабливий дизайн.

Запуск серверної та мобільної сторін додатку підтверджує успішну інтеграцію та взаємодію між сервером та клієнтською частиною. Результати тестового запуску підтверджують відсутність критичних помилок та вказують на коректну роботу програмного продукту на обох платформах.

Отже, даний розділ відображає високий рівень досягнутого технічного виконання та відповідність розробленого додатку вихідним вимогам та цілям проекту.

ВИСНОВКИ

У ході виконання дипломної роботи було проведено аналіз існуючих підходів до реалізації збору та аналізу даних для систем моніторингу забруднення атмосферного повітря. Оцінено загальний стан існуючих систем моніторингу, їх технології та використовувані інструменти.

На основі отриманих результатів розглянуто вимоги до інформаційної системи моніторингу забруднення атмосферного повітря. Сформульовані цілі створення системи, визначені вимоги користувача, функціональні та бізнес вимоги, а також нефункціональні та системні вимоги.

У наступному розділі проведено проектування системи моніторингу, включаючи архітектурну концепцію, інтеграцію та взаємодію з датчиками моніторингу забруднення атмосферного повітря, а також розроблено структуру бази даних.

У четвертому розділі виконана реалізація системи моніторингу забруднення атмосферного повітря. Застосовані інструменти для розробки системи, надано опис створення бази даних, бекенду та мобільного додатку.

Отже, інформаційна технологія моніторингу забруднення атмосферного повітря, розроблена в рамках даної дипломної роботи, є ефективним інструментом для відстеження та аналізу якості повітря. Її впровадження сприятиме покращенню екологічної ситуації та забезпеченню безпеки громадського здоров'я. Такий інноваційний підхід до моніторингу дозволить ефективно реагувати на зміни у складі атмосферного повітря, покращуючи загальний екологічний стан нашого середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. European Environment Agency [Electronic resource] – Mode of access: https://www.eea.europa.eu/en. – Назва з екрана.
2. Principles of Air Quality Management 2nd Edition / Roger D. Griffin – CRC Press, 2020. – 358 с.
3. Air Quality 6th Edition / Wayne T. Davis , Joshua S. Fu, Thad Godish– CRC Press, 2023. – 401 с.
4. Ait pollution Health and Invironmental Impacts / Bhola R. Gurjar, Luisa T. Molina, C.S. P. Ojha – CRC Press, 2010. – 556 с.
5. Earth data [Electronic resource] – Mode of access: https://www.earthdata.nasa.gov/. – Назва з екрана.
6. World Health Organization [Electronic resource] – Mode of access: https://www.who.int/ – Назва з екрана.
7. National Institute of Environmental Health Sciences [Electronic resource] – Mode of access: https://www.niehs.nih.gov/– Назва з екрана.
8. Fundamentals of Air Pollution/ Richard w. Boubel, Donald L. Fox, D. Bruce Turner, Arthur C. Stern –, Elsevier Inc, 2008. – 942 с.
9. Введення залежностей (Dependency Injection) [Electronic resource] – Mode of access: https://en.wikipedia.org/wiki/Dependency_injection – Назва з екрана.
10. Аналіз програмного забезпечення / Ю.І.Грицюк – Львівська політехніка, 2018. – 456 с.
11. Maven : Definitive Guide/ Timothy M.O’Brien, Eric Redmond – O'Reilly Media, 2008. – 468 с.
12. Spring Start Here: Learn what you need and learn it well / Laurentiu Spilca - Manning, 2021 – 416 с.
13. High Performance MySQL: Proven Strategies for Operating at Scale, 4th edition /Jeremy Tinley, Silvea Botros - O'Reilly Media, 2021 – 386 с.

14. Android (Operation Systems) [Electronic resource] – Mode of access: https://en.wikipedia.org/wiki/Android_(operating_system) – Назва з екрана.
15. Centers for Desease Control and Preventation [Electronic resource] – Mode of access: https://www.cdc.gov/globalhealth/countries/bangladesh/default.htm – Назва з екрана.