МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

КУРСОВИЙ ПРОЕКТ

з дисципліни “Бази даних-3”

спеціальність 121 – Інженерія програмного забезпечення
на тему «СИСТЕМА МОНІТОРИНГУ ЯКОСТІ АТМОСФЕРНОГО ПОВІТРЯ МІСТ ТА МІСТЕЧОК УКРАЇНИ»

Студент
групи КП-81 Ничепорук Захар Анатолійович

(підпис)

Викладач
к.т.н, доцент кафедри
СПіСКС Петрашенко Андрій Васильович

(підпис)

Київ – 2021

АНОТАЦІЯ

Метою розробки даного курсового проекту є набуття практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з пост реляційними базами даних, а також здобуття навичок оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті виконання курсового проекту було опановано навик розробляти програмне забезпечення для пост реляційних баз даних, володіння основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних.

Темою даного курсового проекту є створення системи моніторингу якості атмосферного повітря. У документі викладена актуальність та проблематика аналізу великого обсягу даних, аналіз використаного інструментарію (опис мови програмування, використаних бібліотек та СУБД), описана структура бази даних, опис розробленого програмного забезпечення (загальний, опис модулів та основних алгоритмів роботи), аналіз функціонування засобів масштабування, та опис результатів проведеного аналізу.

Результатами даного проекту стали діаграми та графіки, що зображають результати моніторингу якості атмосферного повітря. З ними можна ознайомитися в додатку А.

ЗМІСТ

ВСТУП

У ході роботи була створена система спостереження за якістю атмосферного повітря. За dataset були взяті останні дані з кожної станції моніторингу якості повітря, які було зібрано проектом SaveDnipro.

Актуальність. На сьогоднішній день проблема чистоти атмосферного повітря стоїть дуже гостро. Різноманітні організації повинні слідкувати за якістю повітря, перевіряти рівень небезпеки для людей, робити все можливе, щоб зменшити кількість небезпечних речовин у повітрі. Виробничі заводи витрачають величезні кошти на встановлення фільтрів, розробку більш екологічно чистого виробничого процесу. Очевидно, що система спостереження за якістю повітря є, без перебільшення, дуже впливовою на дану галузь. Можливість отримати дані, на основі яких можна зробити оцінку якості повітря, та визначити її залежність від різноманітних факторів надзвичайно корисна для даної галузі.

Проект збирає дані зі станцій моніторингу, оброблює їх та видає середні показники стану атмосферного повітря в містах України. Також система аналізує тренди та моду різних атмосферних параметрів, вираховує залежність оцінки якості повітря від кількості тонкодисперсних частинок.

Мета розробки: створення програмного забезпечення, що забезпечує роботу наведених далі пунктів:

1. Попередня обробка даних

• Засоби генерації даних. Розроблення утиліти для збору інформації про параметри атмосферного повітря.

• Засоби фільтрації і валідації даних. Розроблення додаткового функціоналу у вищезазначеній утиліті задля корегування отриманих даних та переходу до їх подальшої обробки та структуризації.

2. Основний модуль: процесу. Встановлення залежності між оцінкою якості повітря та кількістю тонкодисперсних частинок в повітрі. Складання таблиці атмосферних показників в містах України.

3. Забезпечення масштабування та реплікації даних: шардинг та створення репліка сетів у Mongo DB

Дані для аналізу були запозичені із відкритого API за посиланням https://api.saveecobot.com/output.json.

АНАЛІЗ ІНСТРУМЕНТАРІЮ ДЛЯ ВИКОНАННЯ КУРСОВОГО ПРОЕКТ

Аналіз СКБД

В процесі виконання цього курсового проекту перед нами встала потреба кешувати та зберігати дані про вакансії між запусками аналізатора. Кожного разу читати дані із CSV - файлів є дуже дорогою операцією, тож було прийняте рішення використати СКБД. В якості СКБД були розглянуті варіанти: PostgreSQL, MongoDB. З порівняльною характеристикою цих СУБД можна ознайомитися в таблиці 1.

таблиця 1. Порівняльна характеристика СКБД

Критерій порівняння	Назва СКБД		
	MongoDB	PostgreSQL	CassandraDB
Має відкритий вихідний код	так	так	так
Схема даних	динамічна	статична і динамічна	статична і динамічна
Підтримка ієрархічних даних	так	так (з 2012)	ні
Реляційні дані	ні	так	так
Транзакції	ні	так	так
Атомарність операцій	всередині документа	по всій БД	всередині партиції
Мова запитів	JSON/JavaScript	SQL	CQL
Найлегший спосіб масштабування	горизонтальний	вертикальний	горизонтальний
Підтримка шардингів	так	так (важка конфігурація)	так (може зберігати партиції на різних машинах)

Приклад використання	Великі дані (мільярди записів) з великою кількістю паралельних оновлень, де цілісність і узгодженість даних не потрібно.	Транзакційні і операційні програми, вигода яких в нормалізованому формі, об'єднаннях, обмеження даних і підтримки транзакцій.	Багато запитів на запис/читання у одиницю часу, до даних можна задіяти партиціювання за ключем, дані мають лише первинні індекси
Наявність бібліотек для мови програмування JavaScript	так	так	так
Підтримка реплікації	так, автоматичне переобрання головного процесу	За принципом master-slave	так, через партиціювання
Засіб збереження та відновлення даних	mongodump	pg_dump	не має окремого доданка, виконується засобами CQL
Форма збереження даних	документи JSON	таблиця	таблиця

За результатами порівнянні цих СУБД було прийнято рішення зупинитися на NoSQL рішеннях. Оскільки вона чудово поєднує в собі переваги неструктурованих баз даних та простоту використання горизонтального масштабування. Крім цього класичним прикладом використання NoSQL СУБД є системи збору та аналізу даних, до яких можна застосувати індексування за первинними та вторинними ключами.

Ця база даних є об’єктно орієнтованою та дозволяє зберігати великі масиви неструктурованих даних. На відміну від SQL баз даних ми можемо зберігати дані у “сирому” об’єктному вигляді, який використовується програмою та є більш близьким за структурою до моделі даних, яку буде використовувати ПЗ написане з використанням мови програмування JavaScript. Це пришвидшить збір, збереження та отримання даних програмним забезпеченням. Оскільки MongoDB є представником NoSQL баз даних, вона не потребує жорсткої схеми даних, що дозволяє пришвидшити процес розробки та зробити його більш гнучким. Окрім

цього дана СУБД підтримує горизонтальне масштабування за допомогою шардингу з метою зменшення навантаження на кожен окремий вузол шляхом розподілення навантаження між ними всіма.

Нижче наведено перелік основних переваг:

- Підтримка ієрархічних даних
- Динамічна схема
- Швидкість запису у колекцію
- Швидкість читання із колекції
- Простота масштабування та відновлення даних

Обґрунтування вибору мови програмування

Мовою програмування для ПЗ було обрано JavaScript. Оскільки це сильна динамічно типізована мова програмування. Це дозволяє самостійно обирати типізацію і поєднувати швидкість розробки ПЗ із можливістю в майбутньому додати типи до модулів програми.

Обґрунтування вибору бібліотек і фреймворків

Використані бібліотеки:

chartjs — це безкоштовна бібліотека JavaScript з відкритим кодом для візуалізації даних, яка підтримує 8 типів діаграм: смуга, лінія, область, пиріг (пончик), міхур, радіолокатор, полярний та розкиданий.

Створена лондонським веб-розробником Ніком Дауні в 2013 році, тепер вона підтримується ком'юніті і є другою за популярністю бібліотекою для графіків JS на GitHub за кількістю зірок після D3.js, що вважається значно простішою у використанні, хоча і менш налаштованою, ніж остання. Chart.js відображається в canvas HTML5 і широко висвітлюється як одна з найкращих бібліотек візуалізації даних. Вона доступна за ліцензією MIT.

mongoose — це бібліотека JavaScript, яка створена для взаємодії з базами даних включаючи MongoDB

simple-statistics — це бібліотека JavaScript, яка реалізує статистичні методи. Вона допомагає програмістам використовувати силу статистики, а

статистики розуміють код. Бібліотека вичерпно задокументована, написана у простому та доброзичливому стилі та ретельно перевірена

express.js, або просто Express — фреймворк для розробки серверної частини веб-застосунків для Node.js, реалізований як вільне і відкрите програмне забезпечення під ліцензією MIT. Він спроектований для створення веб-застосунків і API. Де-факто є стандартним фреймворком для Node.js. Автор фреймворка, TJ Holowaychuk, описує його як створений на основі написаного на мові Ruby фреймворка Sinatra, маючи на увазі, що він мінімалістичний, але має велику кількість плагінів, що підключаються.

СТРУКТУРА БАЗИ ДАНИХ

База даних складається з однієї колекції, в якій зберігаються відомості про параметри атмосферного повітря в місті. Загальна структура документа в базі даних приведена у таблиці 2.

таблиця 2. Опис властивостей документа у базі даних

Назва властивості	**Тип**	**Опис**
_id	ObjectId	Ідентифікатор запису
name	String	Назва міста чи станції
date	String	Дата найновішого виміру
AQI	String	Загальна оцінка якості повітря
PM2_5	String	Вміст тонкодисперсних матеріалів PM2.5 в повітрі
PM10	String	Вміст тонкодисперсних матеріалів PM10 в повітрі
temperature	String	Температура
pressure	String	Тиск
humidity	String	Вологість
stations	Array	Масив станцій, станція містить такі само поля як і описані вище, а також поле cityName, в якому зберігається назва міста в якому знаходиться станція
cityName	String	Назва міста в якому знаходиться станція

АНАЛІЗ ФУНКЦІОНУВАННЯ ЗАСОБІВ МАСШТАБУВАННЯ

У якості засобів масштабування було обрано реплікацію та шардинг.

Реплікація - це процес синхронізації даних на декількох серверах. Даний механізм зменшує витрати ресурсів і збільшує доступність даних, копії яких зберігаються на різних серверах. Реплікація захищає базу даних (далі - БД) від втрати єдиної сервера і дозволяє зберегти дані в разі технічної несправності на одному з серверів.

У MongoDB реплікація досягається шляхом використання набору копій (replica set). Це група примірників mongod, який зберігають однакові набори даних. У копії один вузол - це ключовий вузол, який отримує всі операції запису. Всі інші вузли - вторинні, приймають операції з першого, таким чином, зберігаючи такі ж записи, як і первинний вузол. Набір копій може мати тільки один первинний вузол.

Шардінг - це процес зберігання документів на декількох серверах і це спосіб, яким MongoDB справляється з великими даними. З ростом кількості даних, один сервер не може зберігати всі даних, ні записувати їх, ні давати до них доступ. Шардінг вирішує проблему шляхом горизонтального масштабування. Завдяки даному механізму ми можемо підключати додаткові сервери для зберігання, записи і читання даних.

Для емуляції існування багатьох серверів із СКБД було використано Docker машину. Конфігурація docker-compose наведено у Додатку А.

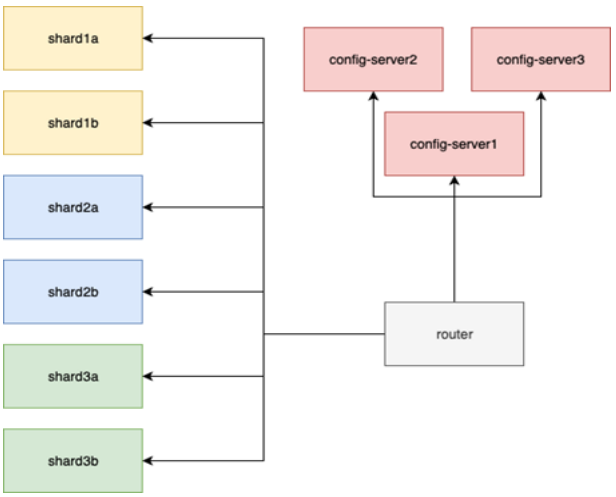


рис 1.Схема використаної моделі шардингу MongoDB у даному ПЗ

Загальний алгоритм додавання нового кластеру шардингу, який складається з кількох серверів реплікації

1. Запустити на виконання один або декілька процесів mongod, які стануть вузлами шардингу за допомоги команди у таблиці 3.

таблиця 3. Скрипт запуску нових вузлів шардингу

```
mongod --port <PORT> --shardsvr --replSet <SHARD_1_ID> --noprealloc --smallfiles
--oplogSize 16

mongod --port <PORT> --shardsvr --replSet <SHARD_2_ID> --noprealloc --smallfiles
--oplogSize 16
```

2. Дочекатися їх ініціалізації, під’єднатися до процесу mongod одного з них та виконати команду наведену у таблиці 4.

таблиця 4. Скрипт створення нових вузлів шардингу

```
rs.initiate(
  {
     _id: "<SHARD_1_ID>",
     version: 1,
     members: [
        { _id: 0, host : "<SHARD_HOST_1>:<PORT>" },
        { _id: 1, host : "<SHARD_HOST_1>:<PORT>" },
     ]
  }
)
```

3. Під’єднатися до консолі процесу mongoose сервера роутера та виконати команду, наведену у таблиці 5. Це зареєструє новий сервер конфігурації шардингу у сервері маршрутизації.

таблиця 5. Скрипт додавання нового кластеру до існуючого шардингу

```
sh.addShard("<SHARD_HOST_1>/<SHARD_1_ID>:<PORT>")

sh.addShard("<SHARD_HOST_1>/<SHARD_2_ID>:<PORT>")
```

Загальний алгоритм додавання нового вузла реплікації серверів конфігурації

Запустити на виконання один або декілька процесів mongod, які стануть вузлами реплікації.

1. Дочекатися їх ініціалізації, під’єднатися до процесу mongod одного з них та виконати команду наведену у таблиці 6.

таблиця 6. Скрипт додавання нового серверу конфігурації до реплікації

```
rs.add( { host: "<hostnameNew>:<portNew>", priority: 0, votes: 0 } )
```

ОПИС РЕЗУЛЬТАТІВ АНАЛІЗУ ПРЕДМЕТНОЇ ГАЛУЗІ

В результаті виконання курсового проекту було проаналізовано дані про атмосферне повітря в містах України. Було отримано наступні дані:

- Проаналізована залежність загальної оцінки якості повітря від вмісту тонкодисперсних матеріалів у повітрі;
- Складена таблиця міст України з показниками атмосферних параметрів;
- Знайдені мода та медіана атмосферних параметрів;
- Знайдені кореляції між загальною оцінкою якості повітря та двома показниками вмісту тонкодисперсних матеріалів: PM2.5 та PM10;
- Розроблений графічний інтерфейс для візуалізації отриманих даних;
- Додана функція експорту та імпорту бази даних.

ВИСНОВКИ

В процесі виконання даного курсового проекту було отримано практичні навички обробки великих масивів даних за допомогою мови програмування JavaScript та СУБД MongoDB.

Було проаналізовано сучасні методи та інструменти для роботи із великими даними, знайдено гарно працюючу комбінацію із мови програмування JavaScript та бібліотек до нього. Було складено порівняльну таблицю із трьох баз даних, які найчастіше використовують для роботи із часовими рядами. На основі цих даних було обрано в якості СУБД NoSQL рішення MongoDB.

Для забезпечення горизонтального масштабування було використано засоби MongoDB, такі як: репліка сет та шардинг. Практичним шляхом було знайдено спосіб партиціювання даних часових рядів на основі хеш-функції від мітки часу. Це дозволило розподілити дані, які зберігаються в рамках однієї колекції між багатьма серверами кластерів шардингу. За допомогою реплікації було досягнуто відмовостійкість системи, в результаті чого, під час тестування ми впевнилися, що система продовжує функціонування після виходу із ладу кількох реплік.

На основі зібраних даних було проаналізовано атмосферне повітря. Знайдено кореляції та залежності між різними показниками. Дані аналізу приведені у Додатку Б. Текстовий опис надано в відповідному розділу цього курсового проекту.

В ході виконання даного курсового проекту було досягнуто поставленої мети: було набуто практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з NoSQL базами даних, а також були здобуті навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті виконання курсового проекту я навчився писати програмне забезпечення для NoSQL баз даних, володіти основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних, а саме відкритими бібліотеками мови JavaScript.

Код курсової роботи розміщено за наступним посиланням: https://github.com/znycheporuk/db

ЛІТЕРАТУРА

1. Hashed sharding [Електронний ресурс]. Режим доступу до ресурсу: https://docs.mongodb.com/manual/core/hashed-sharding/

2. Aggregation [Електронний ресурс]. Режим доступу до ресурсу: https://docs.mongodb.com/manual/aggregation/

3. 5 причин, почему машинное обучение станет технологией 2018 [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://blogs.oracle.com/russia/machine-learning-2018;

4. Сложности накопления данных для интеллектуального анализа [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: https://habr.com/post/154787/;

5. PostgreSQL лучше чем MongoDB[Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://habr.com/post/272735/;

6. 27 шпаргалок по машинному обучению [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://proglib.io/p/ds-cheatsheets/;

7. Горизонтальное масштабирование: когда и как? [Електронний ресурс]. – Режим доступу до ресурсу: https://habr.com/company/oleg-bunin/blog/319526

ДОДАТКИ

Додаток А

таблиця А1. Вміст файлу docker-compose.yml

docker-compose.yml

```
version: '3.7'
services:
  config01:
    image: mongo:4.0
    command: mongod --port 27017 --configsvr --replSet configserver --noprealloc
--smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/config01:/data/db
  config02:
    image: mongo:4.0
    command: mongod --port 27017 --configsvr --replSet configserver --noprealloc
--smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/config02:/data/db
  config03:
    image: mongo:4.0
    command: mongod --port 27017 --configsvr --replSet configserver --noprealloc
--smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/config03:/data/db
  shard01a:
    image: mongo:4.0
    command: mongod --port 27018 --shardsvr --replSet shard01 --noprealloc
--smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard01a:/data/db
  shard01b:
    image: mongo:4.0
    command: mongod --port 27018 --shardsvr --replSet shard01 --noprealloc
--smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard01b:/data/db
  shard02a:
    image: mongo:4.0
    command: mongod --port 27019 --shardsvr --replSet shard02 --noprealloc
--smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard02a:/data/db
  shard02b:
    image: mongo:4.0
    command: mongod --port 27019 --shardsvr --replSet shard02 --noprealloc
--smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard02b:/data/db
  shard03a:
    image: mongo:4.0
```

```
    command: mongod --port 27020 --shardsvr --replSet shard03 --noprealloc
--smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard03a:/data/db
  shard03b:
    image: mongo:4.0
    command: mongod --port 27020 --shardsvr --replSet shard03 --noprealloc
--smallfiles --oplogSize 16
    volumes:
      - ./scripts:/scripts
      - ./data/shard03b:/data/db
  router:
    image: mongo:4.0
    command: mongos --port 27017 --configdb
configserver/config01:27017,config02:27017,config03:27017 --bind_ip_all
    environment:
      USER: user
      PSW: psw
      DB: mydb
    ports:
      - "27017:27017"
    volumes:
      - ./scripts:/scripts
      - ./data/router:/data/db
    depends_on:
      - config01
      - config02
      - config03
      - shard01a
      - shard01b
      - shard02a
      - shard02b
      - shard03a
      - shard03b
```

Додаток Б

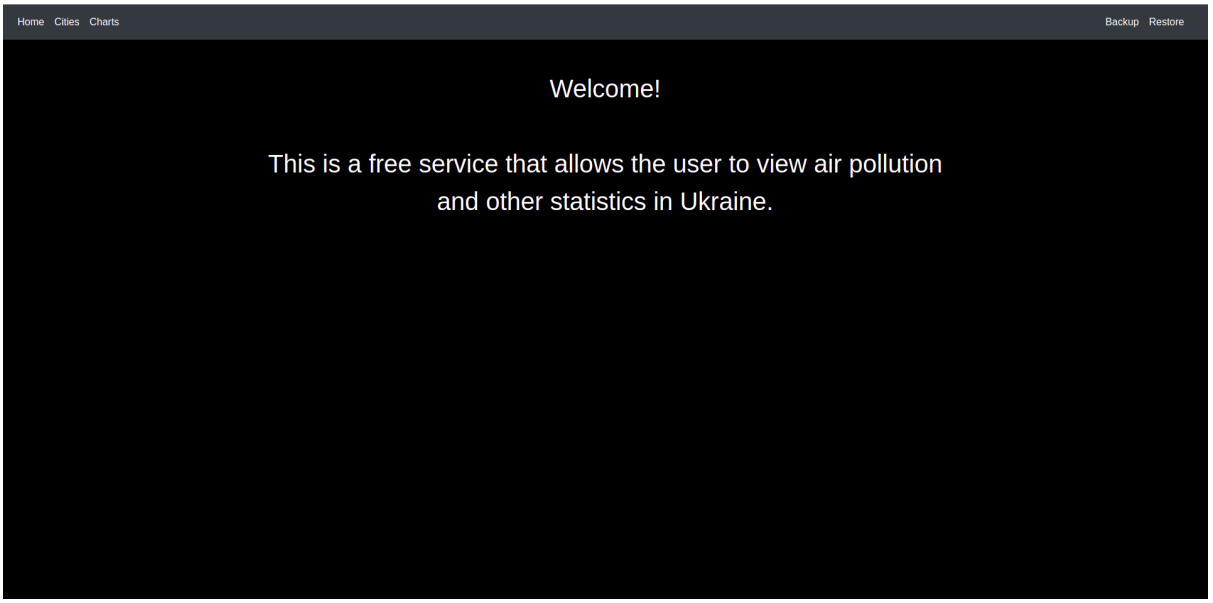
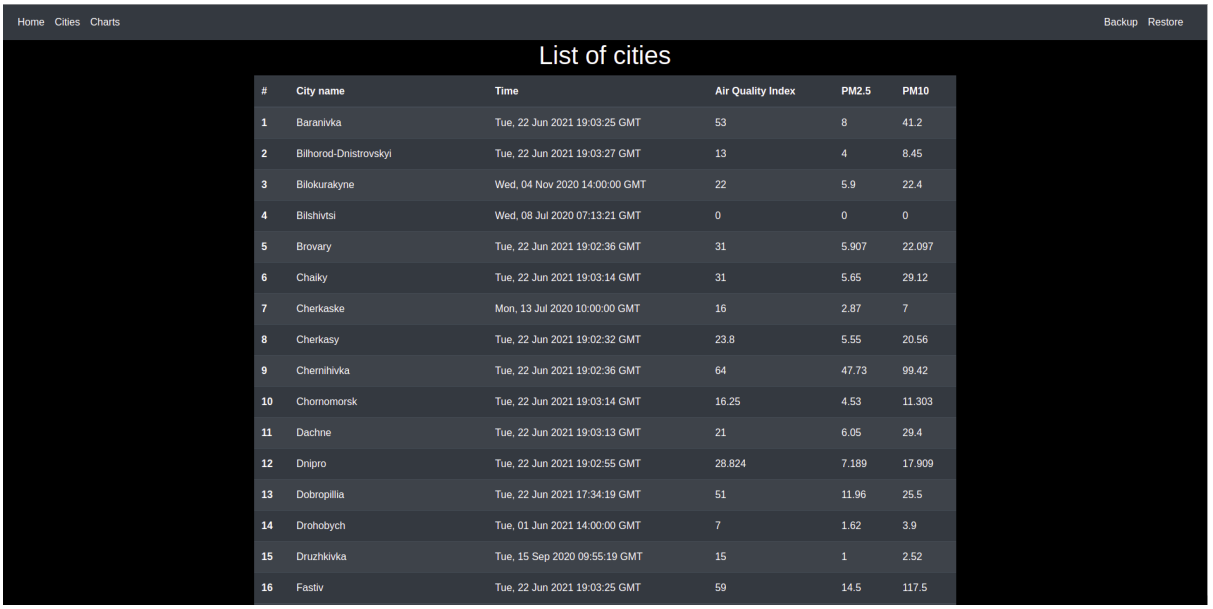


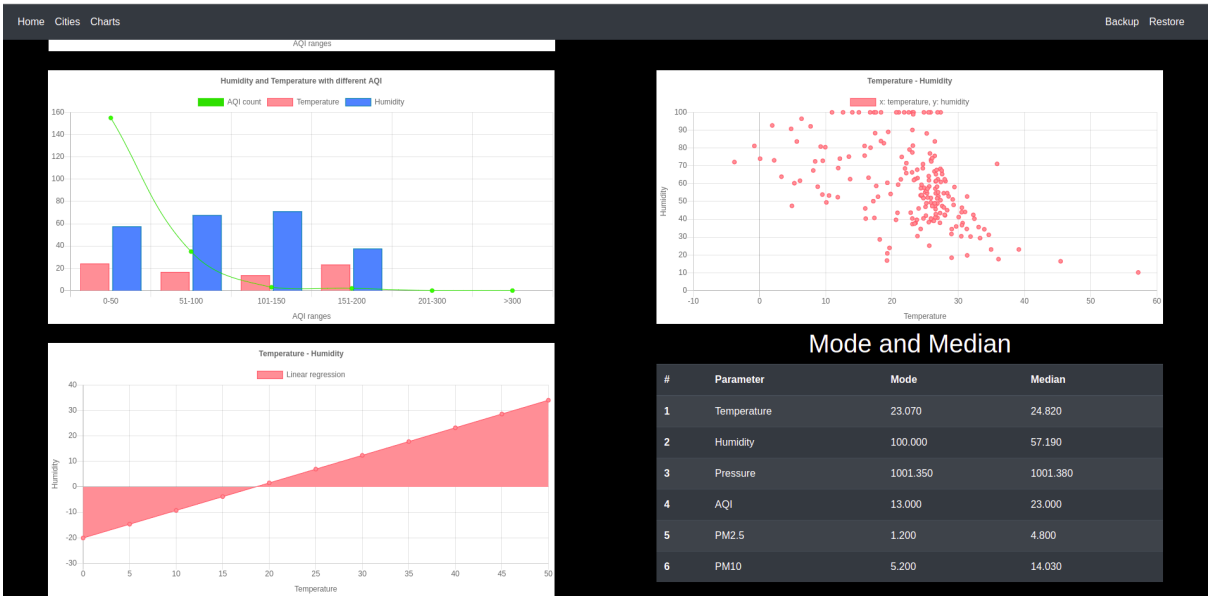
рис Б1.



#	City name	Time	Air Quality Index	PM2.5	PM10
1	Baranivka	Tue, 22 Jun 2021 19:03:25 GMT	53	8	41.2
2	Bilhorod-Dnistrovskyi	Tue, 22 Jun 2021 19:03:27 GMT	13	4	8.45
3	Bilokurakyne	Wed, 04 Nov 2020 14:00:00 GMT	22	5.9	22.4
4	Bilshivtsi	Wed, 08 Jul 2020 07:13:21 GMT	0	0	0
5	Brovary	Tue, 22 Jun 2021 19:02:36 GMT	31	5.907	22.097
6	Chaiky	Tue, 22 Jun 2021 19:03:14 GMT	31	5.65	29.12
7	Cherkaske	Mon, 13 Jul 2020 10:00:00 GMT	16	2.87	7
8	Cherkasy	Tue, 22 Jun 2021 19:02:32 GMT	23.8	5.55	20.56
9	Chernihivka	Tue, 22 Jun 2021 19:02:36 GMT	64	47.73	99.42
10	Chornomorsk	Tue, 22 Jun 2021 19:03:14 GMT	16.25	4.53	11.303
11	Dachne	Tue, 22 Jun 2021 19:03:13 GMT	21	6.05	29.4
12	Dnipro	Tue, 22 Jun 2021 19:02:55 GMT	28.824	7.189	17.909
13	Dobropillia	Tue, 22 Jun 2021 17:34:19 GMT	51	11.96	25.5
14	Drohobych	Tue, 01 Jun 2021 14:00:00 GMT	7	1.62	3.9
15	Druzhkivka	Tue, 15 Sep 2020 09:55:19 GMT	15	1	2.52
16	Fastiv	Tue, 22 Jun 2021 19:03:25 GMT	59	14.5	117.5

рис Б2.

рис Б3.



#	Parameter	Mode	Median
1	Temperature	23.070	24.820
2	Humidity	100.000	57.190
3	Pressure	1001.350	1001.380
4	AQI	13.000	23.000
5	PM2.5	1.200	4.800
6	PM10	5.200	14.030

рис Б4.