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”

спеціальність 121 – Програмна інженерія

на тему: **Моніторингова система кліматичних показників**

Студент
групи КП-83 Василець Данило Андрійович _____
(ПІБ) (підпис)

Аспірант кафедри
СПіСКС **Радченко К.О.** _____
(підпис)

Захищено з оцінкою _____

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 моніторингової системи кліматичних показників. 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).

Результатами даного проекту стали діаграми та графіки, що зображають результати дослідження кліматичних показників.

Зміст

Вступ

Під час виконання проекту було створено систему моніторингу погоди. Всі данні було отримано з сайту Open Weather, що пропонує можливість безкоштовного доступу до API з погодою.

Актуальність

На сьогоднішній день кожен користується сайтами з погодою, для того щоб знати що сьогодні вдягти, яка погода буде завтра або на вихідних. Сайти з погодою є досить корисними на даний час, хоча погода часто змінюється і прогнози погоди часто є хибними, нам все одно хочеться знати хочаб приблизну погоду.

Програма збирає дані з 15 різних міст України і зберігає данні про температуру, вологість, тиск та швидкість вітру. Також вираховує моду і медіану показників, та будує графік залежності температури від вологості, зображує графік температур у містах України.

Мета розробки:

Створення програмного забезпечення, що забезпечує роботу наведених далі пунктів:

1. Попередня обробка даних

• Засоби генерації даних. Підключення до API сайту з погодою для отримання даних для різних міст.

• Засоби фільтрації і валідації даних. Розроблено ПЗ, що фільтрує данні, отримані з API та залишає лише потрібні, також переводить різні системи вимірювань даних.

2. Основний модуль процесу. Ззібрання даних клімату різних міст України, побудова деяких графіків.

Дані для аналізу були запозичені із відкритого API за посиланням https://openweathermap.org/current.

Аналіз інструментарію для виконання курсового проект

Аналіз СКБД

Кожного разу читати дані із CSV було б досить витратно, тож було прийняте рішення використати СКБД. В якості СКБД були розглянуті варіанти: PostgreSQL, MongoDB, CassandraDB. З порівняльною характеристикою цих СУБД можна ознайомитися в таблиці 1.

таблиця 1. Порівняльна характеристика СКБД

Критерій порівняння	Назва СКБД		
	MongoDB	PostgreSQL	CassandraDB
Реляційні дані	ні	так	так
Схема даних	динамічна	статична і динамічна	статична і динамічна
Підтримка ієрархічних даних	так	так	ні
Має відкритий вихідний код	так	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 Node 12	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	документы 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. 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:

- Динамічна схема
- Швидкість запису у колекцію
- Швидкість читання із колекції

Обґрунтування вибору мови програмування

Мовою програмування для ПЗ було обрано JavaScript, з серверним фреймворком Node js. Оскільки це динамічна, об'єктно-орієнтована прототипна мова програмування. Вона дозволяє створити гнучку систему, що спрощує розробку та подальше розширення програми.

Обґрунтування вибору бібліотек і фреймворків

Використані бібліотеки:

Google Charts - це багатофункціональний набір інструментів для візуалізації даних. За допомогою нього можна відносно легко будувати графіки та діаграми на сайті.

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

simple-statistics — це бібліотека JavaScript, яка реалізує статистичні методи.

Express — програмний каркас 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 каркасом для Node.js.

Структура бази даних

База даних складається з однієї колекції, в якій зберігаються відомості міста з кліматичними показниками. 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
date	String	Дата виміру
temperature	String	Температура
pressure	String	Тиск
humidity	String	Вологість
windSpeed	String	Швидкість вітру

Опис результатів аналізу предметної галузі

В результаті виконання курсового проекту було проаналізовано кліматичні дані в різних містах України. Було виконано наступне:

- Проаналізована залежність температури повітря від вологості;
- Складена таблиця міст України з показниками атмосферних параметрів;
- Знайдені мода та медіана атмосферних параметрів;
- Розроблений графічний інтерфейс для візуалізації отриманих даних;
- Додана функція експорту та імпорту бази даних.

Висновки

В Ході роботи було отримано практичні навички обробки даних та роботи з данними за допомогою мови програмування JavaScript та MongoDB.

Також було отримано нові навички роботи з певними бібліотеками, які спрощують та покращують роботу з JavaScript. Було складено таблиці порівняння баз данних. На основі цих даних було обрано в якості СУБД NoSQL рішення MongoDB.

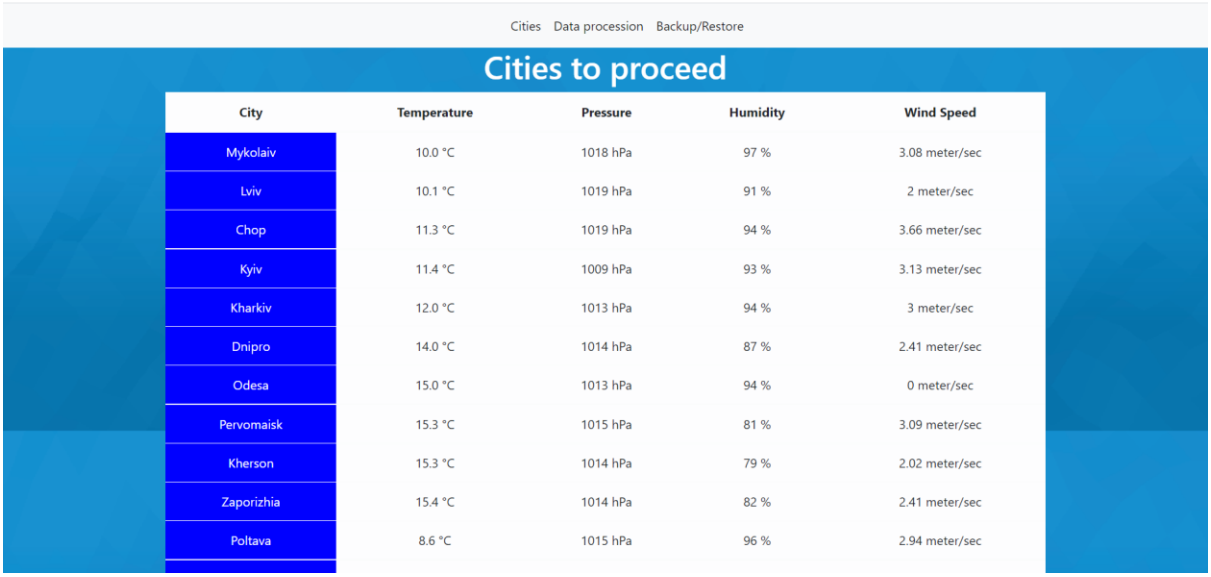
На основі зібраних даних було побудовано таблиці та графіки. Дані аналізу приведені у Додатку А.

В ході виконання даного курсового проекту було набуто практичних навичок розробки програмного забезпечення, що взаємодіє з NoSQL базою даних, а також були покращені навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. Також було набуто навиків в роботі з бібліотеками для побудови графічної частини роботи.

Література

1. PostgreSQL vs MongoDB https://habr.com/post/272735/;
2. Визуализация с Google Chart Tools API https://habr.com/ru/post/304296/
3. Mongodb https://docs.mongodb.com/
4. JavaScript https://www.jetbrains.com/help/phpstorm/viewing-javascript-reference.html?gclid=Cj0KCQjwirz3BRD_ARIsAImf7LOHP8rqF-M7XZyl8hxRoDFd0Ta74D6cXVM3tT1jlHxOe3NAIa39JRwaAvfoEALw_wcB

Додаток А

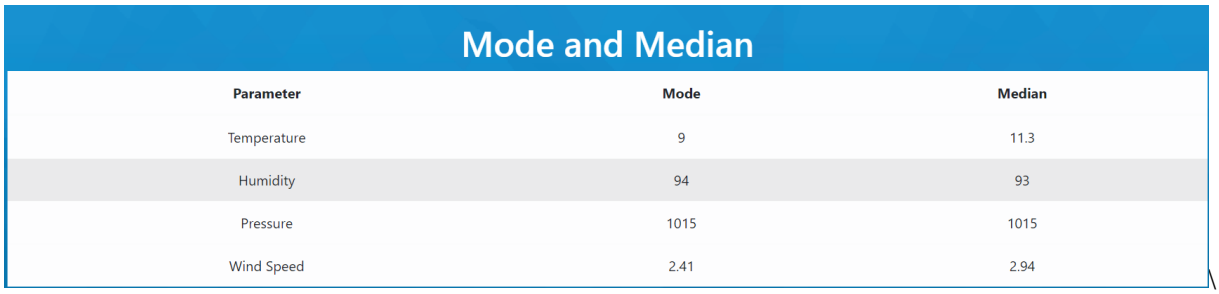


Cities Data procession Backup/Restore

Cities to proceed

City	Temperature	Pressure	Humidity	Wind Speed
Mykolaiv	10.0 °C	1018 hPa	97 %	3.08 meter/sec
Lviv	10.1 °C	1019 hPa	91 %	2 meter/sec
Chop	11.3 °C	1019 hPa	94 %	3.66 meter/sec
Kyiv	11.4 °C	1009 hPa	93 %	3.13 meter/sec
Kharkiv	12.0 °C	1013 hPa	94 %	3 meter/sec
Dnipro	14.0 °C	1014 hPa	87 %	2.41 meter/sec
Odesa	15.0 °C	1013 hPa	94 %	0 meter/sec
Pervomaisk	15.3 °C	1015 hPa	81 %	3.09 meter/sec
Kherson	15.3 °C	1014 hPa	79 %	2.02 meter/sec
Zaporizhia	15.4 °C	1014 hPa	82 %	2.41 meter/sec
Poltava	8.6 °C	1015 hPa	96 %	2.94 meter/sec

Рис.1 – таблиця з кліматичними даними



Mode and Median

Parameter	Mode	Median
Temperature	9	11.3
Humidity	94	93
Pressure	1015	1015
Wind Speed	2.41	2.94

Рис.2 – мода та медіана показників

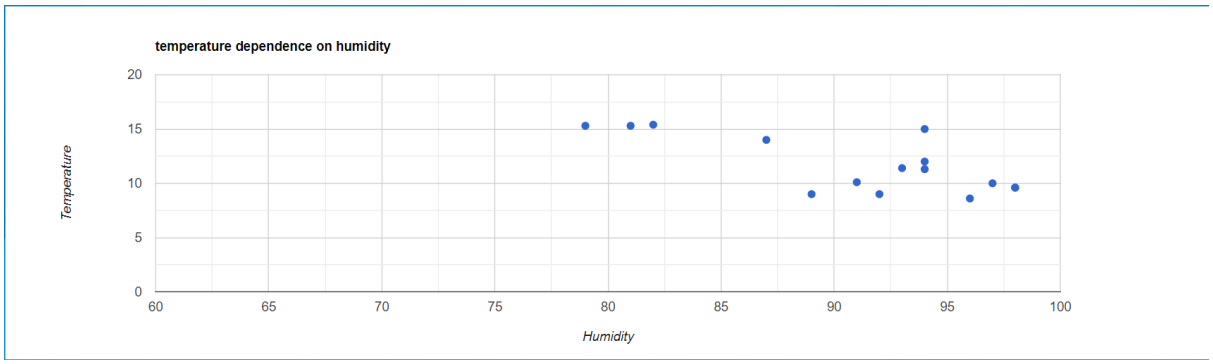


Рис.3 – залежність вологості від температури

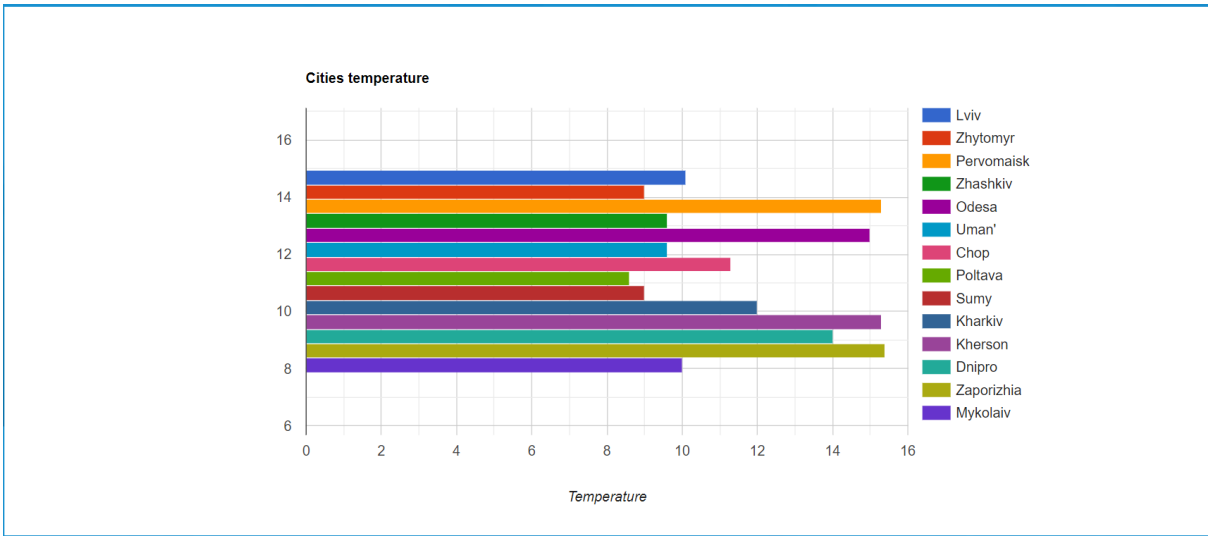


Рис.4 – температура у різних містах України

Додаток В

Код серверної частини

app.js

```
const express = require('express');
const mustache = require('mustache-express');
const path = require('path');
const child_process = require('child_process');
const bodyParser = require('body-parser');
const busboyBodyParser = require('busboy-body-parser');
const mongoose = require('mongoose');
const sstatistics = require("simple-statistics")

const app = express();

const config = require('./config');

const cookieParser = require('cookie-parser');
const session = require('express-session');

const City = require("./models/city.js");

const viewsDir = path.join(__dirname, 'views');
app.engine("mst", mustache(path.join(viewsDir, "partials")));
app.set('views', viewsDir);
app.set('view engine', 'mst');

app.use(bodyParser.json()); // support json encoded bodies
app.use(bodyParser.urlencoded({ extended: true })); // support encoded bodies
app.use(busboyBodyParser({ limit: '5mb' }));

app.use(express.static('public'));

// new middleware
app.use(cookieParser());
app.use(session({
    secret: config.SecretString,
    resave: false,
    saveUninitialized: true
```

```
}));

const PORT = config.ServerPort;
const databaseUrl = config.DatabaseUrl;
const connectOptions = { useNewUrlParser: true};

mongoose.connect(databaseUrl, connectOptions)
    .then(() => console.log(`Database connected: ${databaseUrl}`))
    .then(() => app.listen(PORT, function() { console.log('Server is ready'); }))
    .catch(err => console.log(`Start error ${err}`));

const dbUpdate = require('./update_db');
dbUpdate.getCitiesInfo();
setInterval(dbUpdate.getCitiesInfo, 5000);

app.get('/backup', function(req, res)
{
    const pathToSave = path.join(__dirname, `./backup/backup.bson`);
    const command = `cd C:\\Program Files\\MongoDB\\Tools\\100\\bin & mongodump --host
localhost:27017 -u "user" -pwd "user"  --db mydb --out ${pathToSave} `;
    child_process.exec(command);
    console.log(pathToSave)

    res.redirect('/peps?backup=successful');
});

app.get('/restore', function(req, res)
{
    City.clearCities()
    .then(() =>
    {
        const pathToRestore = path.join(__dirname, `./backup/backup.bson`);
        const command = `cd C:\\Program Files\\MongoDB\\Server\\4.4\\bin & mongorestore --
host localhost:27017 -u "user" -pwd "user"  --db mydb ${pathToRestore} `;
        console.log(child_process.exec(command));
        res.redirect('/peps?restore=successful');
    });
});
app.get('/', function(req, res)
{
    res.redirect('/cities');
});

app.get('/peps', function(req, res)
{
    res.render('index', {});
});

app.get('/cities', function(req, res)
{
    City.getAllCities()
    .then(cities =>
    {
        cities.sort(function(a, b) {
            return (a.temperature).localeCompare(b.temperature);
        });

        let idx = 1;
        res.render('cities', { cities, "index": function() {return idx++;} });
    })
    .catch(err => res.status(500).send(err.toString()));
});

app.get('/data', function(req, res)
{
    City.getAllCities()
```

```
    .then(cities =>
    {
        let citiesArray = [];
        cities.map(value => citiesArray.push(value));
        citiesArray = [].concat.apply([], citiesArray);

        let weatherData = {temperature: [], humidity: [], pressure: [], windSpeed: []};
        for(let i = 0; i < citiesArray.length; i++)
        {
            let city = citiesArray[i];

            weatherData.temperature.push(parseFloat(city.temperature));
            weatherData.humidity.push(parseFloat(city.humidity));
            weatherData.pressure.push(parseFloat(city.pressure));
            weatherData.windSpeed.push(parseFloat(city.windSpeed));
        }

        let mode = { temperature: sstatistics.mode(weatherData.temperature), humidity:
sstatistics.mode(weatherData.humidity),
            pressure: sstatistics.mode(weatherData.pressure), windSpeed:
sstatistics.mode(weatherData.windSpeed)};
        let median = { temperature: sstatistics.median(weatherData.temperature), humidity:
sstatistics.median(weatherData.humidity),
            pressure: sstatistics.median(weatherData.pressure), windSpeed:
sstatistics.median(weatherData.windSpeed)};
        let idx = 1;
        res.render('charts', {mode, median, "index": function() {return idx++;}});
    })
    .catch(err => res.status(500).send(err.toString()));});

app.get('/allcities', function(req, res)
{
    City.getAllCities()
    .then(cities =>
    {
        res.send(cities);
    });
});
```