ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного засобу для моніторингу метеорологічних умов мовою JavaScript»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Дмитро РУЖАНСЬКИЙ
(підпис)

Виконав: здобувач вищої освіти групи ПД-44

Дмитро РУЖАНСЬКИЙ

Керівник: Вікторія ЖЕБКА
д.т.н., професор

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ружанському Дмитру Олексійовичу

1. Тема кваліфікаційної роботи: «Розробка програмного засобу для моніторингу метеорологічних умов мовою JavaScript»
керівник кваліфікаційної роботи д.т.н., професор, завідувач кафедри ТЦР Вікторія ЖЕБКА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
Схема функціонування додатків на JavaScript
Інформація про відкриті погодні сервіси в мережі Інтернет;
Програмна документація на мову JavaScript.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Основні підходи до створення додатків на мові JavaScript.
2. Системи інформування про метеорологічні умови.
3. Проектні рішення щодо розробки додатку для відстежування метеорологічних умов.

4. Опис реалізації та тестування додатку для відстежування метеорологічних умов.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Алгоритм роботи застосунку.
6. Діаграма класів.
7. Екранні форми.
8. Апробація результатів дослідження

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Дослідження архітектури веб-додатків	14.03-18.03.2024	
4	Розробка алгоритмів роботи веб-додатку	19.03-25.03.2024	
5	Програмна реалізація засобу	26.03-18.04.2024	
6	Тестування програмного засобу	19.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Дмитро РУЖАНСЬКИЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Вікторія ЖЕБКА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 60 стор., 17 рис., 14 джерел.

Мета роботи – зменшення затрат часу, необхідних середньостатистичному користувачеві мережі Інтернет, для отримання інформації про поточні метеорологічні умови у заданій місцевості, а також підвищення зручності даного процесу.

Об'єкт дослідження – процес отримання інформації про погоду для типового користувача мережі Інтернет.

Предмет дослідження – програмне забезпечення типу «Web», що може застосовуватися для відстежування погодних умов.

Короткий зміст роботи: В роботі проаналізовано алгоритми та методи для обробки природньої мови в контексті задачі формування програмного засобу для моніторингу метеорологічних умов мовою JavaScript. Проаналізовано інструментальні засоби для моніторингу метеорологічних даних: OpenWeatherMap API, WeatherStack API, AccuWeather API. Розроблено алгоритм роботи засобу та програмно реалізовані ключові функціональні можливості, зокрема: завантаження метеорологічних даних, їх обробка, вибір мови, візуалізація даних у реальному часі, налаштування повідомлень про погіршення погодних умов. Проведено функціональне та модульне тестування додатку. В роботі використано бібліотеку Axios для обробки запитів, Chart.js для візуалізації даних, та React для створення користувацького інтерфейсу. Сферою використання засобу є надання актуальної інформації про метеорологічні умови користувачам у реальному часі.

КЛЮЧОВІ СЛОВА: ПОГОДА, МЕТЕОРОЛОГІЧНІ УМОВИ, МОНІТОРИНГ, NODE.JS, JAVASCRIPT, ПРОГРАМНА РЕАЛІЗАЦІЯ.

ЗМІСТ

ВСТУП

Незважаючи на бурхливий розвиток мультимедійних технологій, широке поширення аудіо та навіть відеозв'язку, передача чисто текстової інформації була і залишається одним із найбільш ефективних способів вербальної комунікації. Текстове спілкування дозволяє мати у прямому доступі одразу великі обсяги даних, ґрунтовно обдумувати питання та відповіді, швидко пересуватися по записам, що обмежується лише технікою читання людини і т.д. Відповідно, не дивним є і той факт, що при текстовому спілкуванні люди можуть обирати собі у співбесідники автоматичні сутності, програми, які називають веб-додатками. «Спілкування» із такими роботами відбувається зазвичай у добровільному режимі, а людина прекрасно розуміє, з ким ведеться спілкування. Звичайно, вести світську бесіду з роботом не має сенсу, але вирішити якісь конкретні практичні питання з його допомогою цілком можливо. Однією із простих задач, яка періодично постає перед будь-якою сучасною людиною, є визначення поточних, а також і майбутніх (у найближчій чи довшій перспективі) погодних метеорологічних умов. В наш час швидких погодних змін обізнаність у метеорологічних умовах стає уже не тільки запорукою гарного зовнішнього вигляду, а й забезпечення здоров'я (а в окремих, екстремальних, умовах – і життя). Таким чином, розробка веб-додатків, зокрема і для відстежування метеорологічних умов є **актуальною задачею** для фахівців у галузі сучасних інформаційних технологій.

Метою роботи зменшення затрат часу, необхідних середньостатистичному користувачеві мережі Інтернет, для отримання інформації про поточні метеорологічні умови у заданій місцевості, а також підвищення зручності даного процесу.

Задачами роботи, відповідно, є:

а) аналіз предметної галузі: особливостей програмних продуктів типу «веб-додаток», існуючих засобів по визначенню поточної погоди, невирішених частин даної проблеми;

б) проробка проектних рішень по створенню програмного продукту типу «WEB» для відстежування поточних погодних умов;

в) проведення реалізації веб-додатку для відстежування поточних погодних умов, його тестування та оцінка ефективності.

Об’єкт дослідження – процес отримання інформації про погоду для типового користувача мережі Інтернет.

Предмет дослідження – програмне забезпечення типу «веб-додаток», що може застосовуватися для відстежування погодних умов.

Практичне значення роботи полягає у створенні робочого програмного продукту, за допомогою якого будь-який користувач може швидко дізнаватися про метеорологічні умови на найближчі дні у цікавому для нього регіоні, причому з використанням потрібної йому мови.

Методи, застосовані в роботі в основному відносяться до сфери інформаційних технологій, зокрема галузі програмування.

Перспектива даної роботи полягає у розширенні системи команд, доступних для користувачів веб-додатку (зокрема, виборі кількості днів, на які потрібно показати прогноз), а також у проведенні кращої його локалізації.

1 АНАЛІЗ МОЖЛИВОСТЕЙ СУЧАСНИХ ПРОГРАМНИХ ПРОДУКТІВ ТИПУ «WEB»

1.1 Історія розвитку та основні властивості програм веб-додатків

Еволюція цифрових комунікацій і мережевих технологій, як локальних, так і глобальних (особливо Інтернету), надала людям можливість ефективного віддаленого спілкування задовго до появи технологій передачі великих мультимедійних даних. До 90-х років традиційними засобами для відправлення коротких текстових повідомлень були телеграми (через телетайп «Телекс») або звичайні поштові листи, що займало від кількох годин до днів.

З появою перших комп'ютерних мереж, обмін текстовими повідомленнями став однією з основних функцій. Наприклад, глобальна мережа Fidonet дозволяла обмінюватися повідомленнями через систему Netmail. Вже на ранньому етапі деякі користувачі прагнули автоматизувати відправлення певних типів повідомлень за допомогою програм-роботів, або ботів, які могли автоматично взаємодіяти з іншими користувачами. Такі боти виконували функції:

- надання допомоги у системах відповідей на часто задавані питання (FAQ);
- надання статистичної інформації на вимогу користувача;
- надання загальної корисної інформації (наприклад, правил поведінки, допоміжної інформації для нових користувачів);
- проведення навчальних розсилок тощо.

Однак спілкування у текстовому режимі офлайн не сприяло активному впровадженню продуктів, які ми сьогодні називаємо чат-ботами. Для їхньої повноцінної роботи необхідний був режим онлайн-спілкування. Першою системою онлайн-спілкування стала розроблена у 1988 році Internet Relay Chat (IRC) — система, що дозволяла великій кількості користувачів спілкуватися одночасно в

рамках однієї віртуальної кімнати, яку називали IRC-каналом. Цей протокол став дуже популярним, і мільйони користувачів по всьому світу почали спілкуватися онлайн. Саме в IRC вперше з'явилися чат-боти, які виглядали як звичайні користувачі. Іноді їх було важко відрізнити від реальних людей, що приводило до ситуацій, коли звичайних користувачів могли вважати ботами, а ботів — справжніми людьми, до моменту розкриття їхньої справжньої природи.

Тут слід звернути увагу на інший аспект використання веб-додатків, а саме – ступінь їх схожості з людською взаємодією. Усі веб-додатки можна поділити на відкриті, про які з самого початку відомо, що це автоматизований додаток, і таємні, про які така інформація приховується.

У випадку відкритого веб-додатка особливих вимог щодо якості взаємодії зазвичай не виникає. Навіть вибагливі користувачі, знаючи, що взаємодіють з автоматизованим додатком, не звертають уваги на якість його відповідей, а просто намагаються отримати потрібний результат. Якщо додаток прихований, його розробник має закласти в нього складне інтелектуальне підґрунтя, щоб відповіді додатка були максимально наближені до людських. У науковій літературі з штучного інтелекту існує поняття тесту Т’юринга, яке означає процедуру сліпого текстового спілкування людини з комп'ютерною програмою. В результаті цього тесту людина не повинна змогти визначити, що спілкується не з живим співрозмовником.

Для коротких неспецифічних взаємодій зі складними програмними продуктами важко встановити, що взаємодія відбувається з автоматизованим додатком. Для цього потрібно використовувати питання, які потребують точної конкретної відповіді, отриманої в результаті побудови аналогій, узагальнення, аналізу слабко формалізованих ситуацій, образного мислення тощо. Таким чином, для побудови найякісніших додатків розробникам слід імітувати ці властивості людського мозку. Однак, це є зайвим при реалізації простих відкритих веб-додатків, єдиною функцією яких є надання користувачеві різноманітної корисної інформації.

Можливості веб-додатків також могли використовувати зловмисники. Використовуючи автоматизовані додатки для масового написання в різні форуми або платформи, що неможливо було зробити вручну, зловмисник значно збільшував шанси на успіх своєї атаки, реалізованої за правилами соціальної інженерії. Соціальна інженерія – це окрема галузь на стику психології та інформаційних технологій, яка допомагає здійснювати злом комп'ютерних систем через спілкування з користувачем. Також можливості веб-додатків могли використовувати недобросовісні рекламодавці, надаючи великі обсяги рекламної інформації ширшій аудиторії.

Веб-додатки виконують корисні функції, такі як видача допомоги у системах FAQ, надання статистичної інформації на вимогу, надання загально необхідної інформації та проведення навчальних розсилок. Наступним кроком після IRC став тотальний розвиток персональних програм-месенджерів, які домінують на ринку спілкування і по даний час, лише додаючи нову функціональність, як, наприклад, голосові дзвінки через Інтернет та відео спілкування. Такими месенджерами спочатку були ICQ, QIP та інші, а пізніше – WhatsApp, Viber, Telegram. У багатьох з цих продуктів можуть використовуватися веб-додатки, тому розглянемо їх докладніше.

а) Інтернет-пейджер ICQ був дуже поширеним для настільних ПК, але не отримав такого розвитку для мобільних систем. На сьогоднішній день він може вважатися застарілим (принаймні в умовах української Інтернет-спільноти) та таким, що поступився місцем більш сучасним продуктам. Слід відмітити, що у часи власної популярності ICQ активно використовував програм-ботів, але на сьогоднішній день розробляти їх немає сенсу через занепад самого месенджера в цілому.

б) ICQ була дуже популярною серед користувачів ПК, але не змогла досягти такого успіху на мобільних пристроях і зараз вважається застарілою. У часи своєї популярності ICQ активно використовувала ботів, але сьогодні це не має сенсу через занепад месенджера.

в) Програма для відеозв’язку Skype від самого свого виникнення має

функцію передачі виключно текстових повідомлень, однак через маркетингову політику виробника не асоціюється у широких мас користувачів із текстовим обміном, і цю можливість реально використовує порівняно мало людей. Через такі обставини веб-додатки для Skype не використовуються.

г) Програма-месенджер Viber була початково створена для смартфонів, але через свою популярність отримала і версію для ПК. Цей продукт ізраїльського походження є лідером по організації текстового обміну в Україні. Він використовується не тільки для спілкування друзів та знайомих, але й для офіційного спілкування керівників з підлеглими. Веб-додатки у Viber часто виконують рекламні, допоміжні, сервісні та інформаційні функції. Проте, більшість користувачів сприймають їх як рекламні інструменти і звертаються до них тільки у разі необхідності.

д) Програмний продукт WhatsApp від компанії Facebook є майже повним аналогом Viber. Він раніше був дуже популярним, особливо у західних країнах, але після його купівлі компанією Facebook розвивається повільніше, можливо, через внутрішню конкуренцію з Facebook Messenger. Хоча WhatsApp є найпопулярнішим месенджером у світі, можливість створення ботів у ньому відсутня, тому він не підходить для цілей цієї роботи (версія WhatsApp Business, яка допускає створення ботів, не враховується через її відносно малу поширеність).

е) Facebook Messenger має значний плюс завдяки тотальній інтеграції з обліковим записом Facebook, що дуже зручно для інтенсивних користувачів соціальної мережі. Боти для Facebook Messenger є популярними засобами комунікації. Однак, через широку популярність та невисоку кваліфікацію середнього користувача, для створення ботів під цю платформу часто використовують прості конструктори чатів, такі як ChattyPeople, Botsify, Chatfuel, FlowXO, BeepBoop. Ці розробки мають типові недоліки систем конструкторів: обмеження у налаштуваннях і функціональності.

є) Додаток надає набагато ширші можливості, ніж простий текстовий обмін, зокрема для організації і забезпечення стійкої криптографії. Зручний інтерфейс та інші особливості месенджера сприяли широкому поширенню веб-додатків типу

«чат-бот». На відміну від Viber, не всі ці боти призначені для реклами; навпаки, часто вони мають допоміжний, сервісний або освітній характер, що позитивно сприймається користувачами. Відповідно, саме цей месенджер буде взятий за основу у подальшій розробці.

ж) Існують інші засоби текстового обміну, які, однак, не отримали великої популярності серед широкого загалу, тому їх навряд чи доцільно використовувати для організації веб-додатків для відстежування метеорологічних умов.

1.2 Дослідження існуючих програм-аналогів для відстежування метеорологічних умов

Перед початком розробки власного веб-додатку для відстеження погоди, важливо вивчити наявні рішення на ринку, аби розуміти їхні можливості та недоліки. Це дозволить створити конкурентоспроможний продукт, який буде максимально зручним для користувачів.

У першу чергу, більшість користувачів звертаються до сайтів для перевірки погоди. Серед найбільш популярних в Україні можна виділити наступні (рис. 1.1):

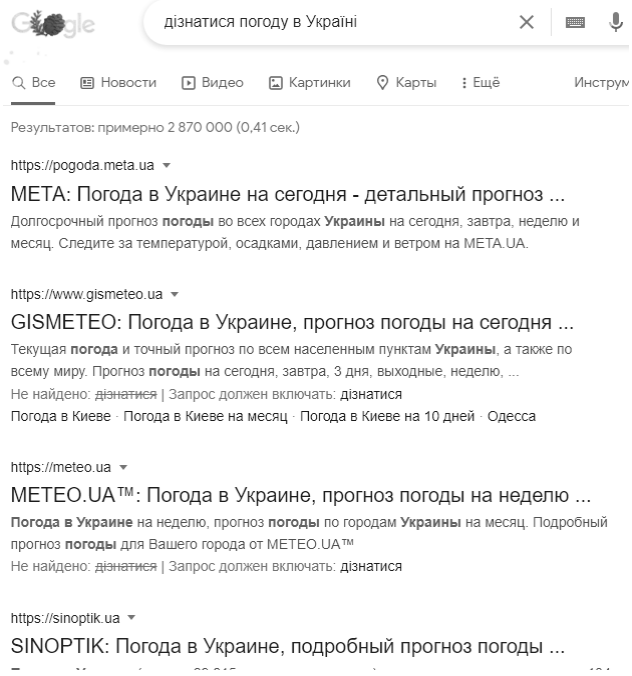


Рис. 1.1 Найбільш популярні сайти для відстежування погоди в Україні

Очевидно, що для користування будь-якими сайтами потрібний доступ до Інтернету, але для користування месенджерами мобільні оператори у багатьох тарифних планах надають повністю вільний необмежений доступ. Крім того, інтерфейс сайтів може здаватися перевантаженим, причому не тільки зайвими опціями, а й, головне, рекламою, чого немає у веб-додатку. Також використання боту саме у часто вживаному месенджері дозволяє спростити процес отримання потрібної інформації про погоду для пересічного користувача (який не має високої кваліфікації у сфері IT). Отже, розглянемо варіанти існуючих програмних продуктів – ботів для визначення погоди.

В першу чергу, можна розглянути чат-бот «Прогноз одежды», який на основі визначення метеорологічних умов одразу рекомендує, який одяг доцільно одягти у заданому місці – рис. 1.2.

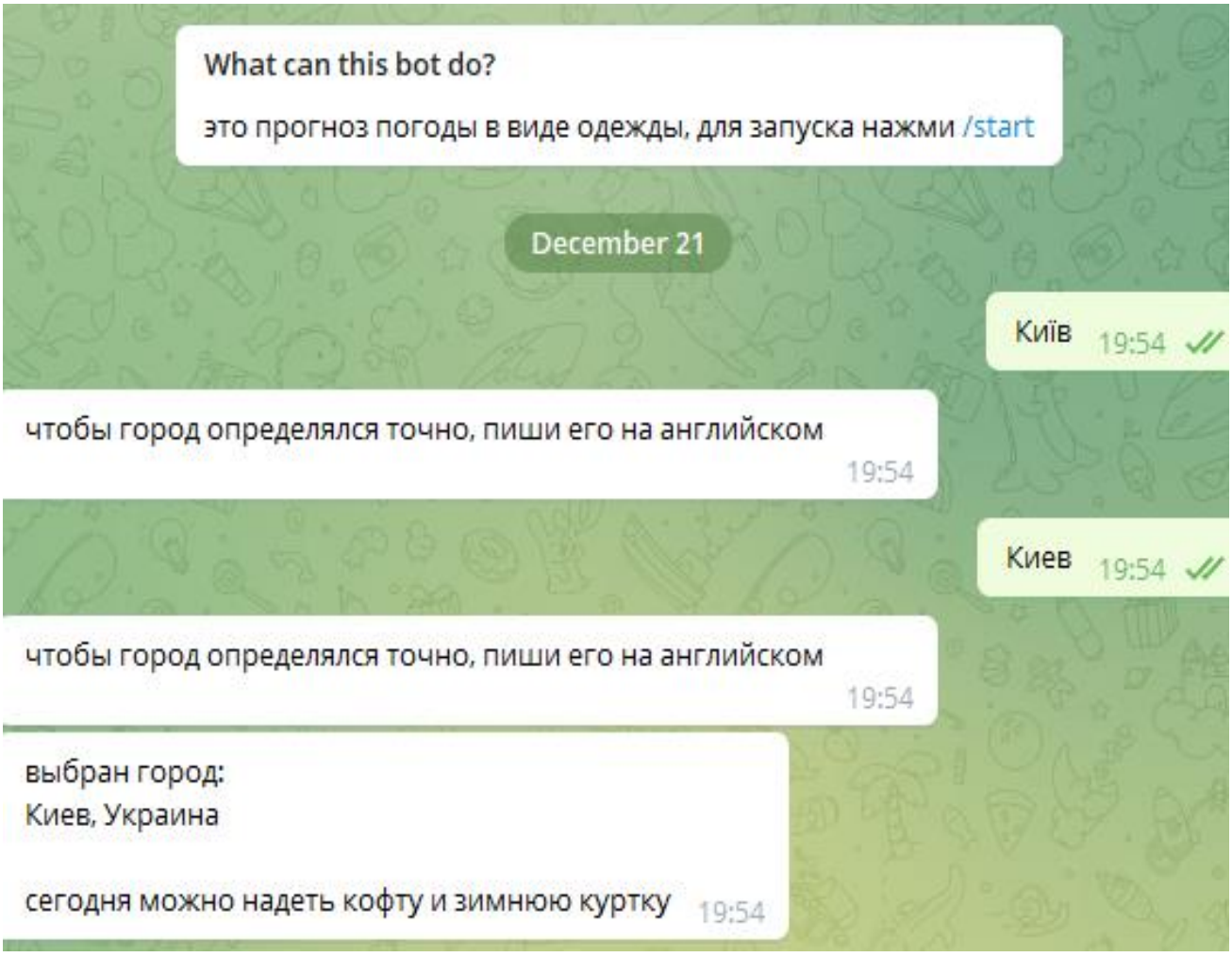


Рис. 1.2 Інтерфейс чат-боту «Прогноз одежды» для Telegram

Це рішення має кілька недоліків:

– неможливість визначити самі метеорологічні умови, а лише відповідний до них одяг.
– відсутність вибору мови.

Наступним, більш досконалим рішенням є чат-бот «Погодник». Серед явних переваг цього сервісу варто відзначити опцію вибору мови інтерфейсу, зокрема, можливість використання української. Окрім цього, користувачі мають змогу отримати вичерпну інформацію стосовно поточних погодних умов, що наочно продемонстровано на рис. 1.3.

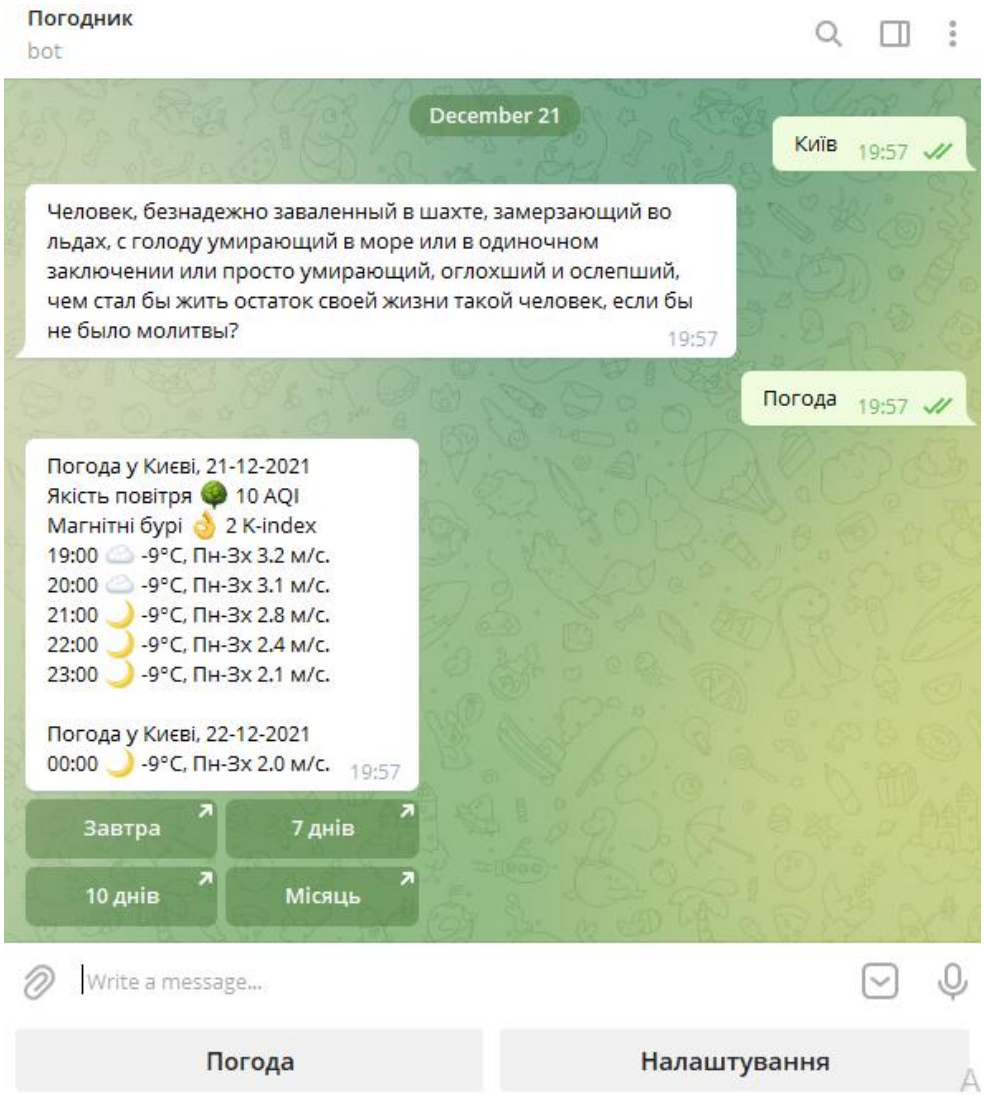


Рис. 1.3 Інтерфейс чат-боту «Погодник» для Telegram

Втім, логіка роботи боту є дещо надмірною, адже у відповідь на перший запит із назвою міста «Київ» бот видає досить велике філософське повідомлення, яке збиває з пантелику користувача і вимагає значних зусиль для того, щоби зрозуміти, що він все ж правильно потрапив саме у бот для відстежування погодних умов і, крім того, що робити далі. Насправді, цілком очевидно, що при вході у погодний бот і надсиланні назви відомого міста користувач хоче дізнатися погоду у даному місті, а не отримувати філософські тексти. Для більшості користувачів такий підхід не буде зручним, тому доцільно провести власну реалізацію програмного продукту-боту.

Ще одним схожим рішенням є бот MeteoBot – рис. 1.4.

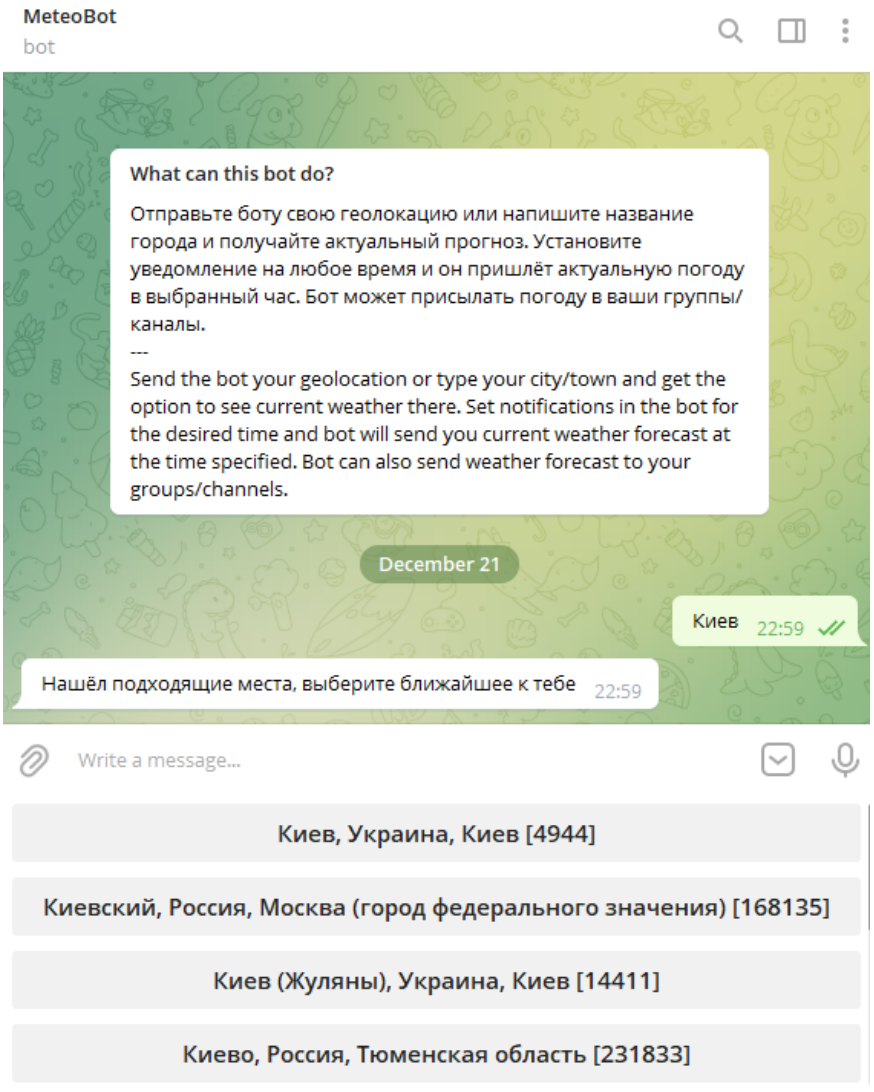


Рис. 1.4 Інтерфейс чат-боту «Meteobot» для Telegram

Meteobot має наступні недоліки:

- не має можливості вільного вибору мов, зокрема, української;

- має недоробки у тексті власних повідомлень, наприклад, на першому ж екрані боту, що показаний на рис. 1.4, написано «выберите ближайшее к тебе».

Беручи до уваги недоліки розглянутих програмних продуктів типу «WEB» для відстежування метеорологічних умов, можна зробити однозначний висновок про те, що доцільною власна розробка такого веб-додатку.

1.3 Виділення невирішених частин проблеми дослідження та вимоги до проектованого програмного продукту

Розглянуті особливості існуючих програмних продуктів спричинюють необхідність розробки власного рішення – веб-додатку мовою JavaScript, який би дозволяв проводити моніторинг погоди, або іншими словами, відстежування метеорологічних умов.

Таким чином, розробці підлягає програмний продукт для отримання актуальної метеорологічної інформації.

Вид продукту – веб-додаток мовою JavaScript.

Тип архітектури продукту – клієнт-серверна архітектура.

Цільова платформа сервера – рішення на базі програмної платформи Node.JS.

Цільова платформа клієнта – будь-яка актуальна версія інтернет-браузеру (настільна або мобільна).

Мови програмування та засоби розробки – сучасні, достатньо популярні, рішення на базі яких можуть в подальшому підтримуватися широким колом програмістів (без необхідності залучення «вузьких» спеціалістів). Передбачається використання мови JavaScript.

Протокол взаємодії – HTTP.

Керуючись даними вимогами, можна переходити до наступного етапу процесу створення програмного продукту.

2 РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ПРИ СТВОРЕННІ ДОДАТКУ ДЛЯ МОНІТОРИНГУ МЕТЕОРОЛОГІЧНИХ УМОВ

2.1 Обґрунтування вибору архітектури та типу додатку

Метою даної роботи є створення умов для ефективного інформування користувача програмного продукту про погодні умови у місцевості, що його цікавить. Звичайно, можуть існувати різні варіанти типів додатків, що виконуватимуть подібні функції. У найпростішому (елементарному) варіанті можливим є створення статичного веб-сайту із прогнозом погоди, який оновлюватиметься вручну (наприклад, кожного ранку) уповноваженою людиною оператором. Очевидно, такий підхід є вкрай неефективним, оскільки потребує величезної кількості ручної праці і майже зовсім не використовує переваг сучасних розвинутих інформаційних технологій. Більш ефективним тут є створення веб-сайту, який би мав власну базу даних та комунікацію із одним або кількома професійними серверами організацій, що публікують прогнози погоди (метеорологічних служб). Такий програмний продукт підпадає під категорію веб-додатків і, взагалі кажучи, є досить ефективним, але потребує від користувача активних дій – в першу чергу, відкриття даного сайту у браузері. Було б значно зручніше, якби необхідна інформація уже знаходилася би у пристрої користувача автоматично, а користувач міг би миттєво отримати її без вчинення додаткових дій при виникненні такої необхідності.

Вказану можливість забезпечує, наприклад, мобільний додаток, однак суттєвим недоліком такого рішення є те, що його слід спеціальним чином встановлювати у смартфон, він споживатиме ресурси смартфону, постійно знаходячись у пам’яті (а якщо він не буде працювати резидентно, то, відповідно, матиме ті ж недоліки, що і веб-додаток).

Ще однією можливістю є запровадження веб-додатку, який має наступні переваги:

- інформація надходить до пристрою користувача в автоматичному режимі, без прикладання додаткових зусиль з боку користувача, і миттєво надається йому при виникненні такої потреби;
- не потребує встановлення додаткового програмного забезпечення, що споживає апаратні ресурси клієнтського пристрою;
- використання функціональності веб-браузера, що є надзвичайно поширеним програмним продуктом.

Суттєві недоліки у такого рішення практично відсутні, через що його і було прийнято у якості основного у даній роботі.

Отже, згідно обґрунтованого вище вибору веб-платформи (а також згідно завдання на розробку), у даній роботі створюється веб-додаток. Архітектура таких програмних продуктів є фіксованою і на логічному рівні відповідає концепції «клієнт-сервер», оскільки команди, які вводить користувач у веб-інтерфейсі, передаються на серверну частину (бек-енд), яка має знаходитися на комп'ютері-сервері, що має постійне підключення до мережі Інтернет. Там ці команди обробляються і на клієнта пересилається відповідь, що відображується користувачеві. Число клієнтів, з якими спілкується сервер (тобто, власне веб-додаток), обмежується лише апаратною продуктивністю комп'ютера-сервера і може складати тисячі користувачів одночасно і більше. Таким чином, на логічному рівні маємо повну відповідність архітектурі «клієнт-сервер».

Слід сказати, що фізично усі запити і відповіді, що передаються у веб-додатку, спочатку надсилаються на сервер, який забезпечує процеси шифрування даних і взагалі керує усім процесом спілкування клієнтів з сервером. Проходження повідомлень від сервера до клієнта і назад відбувається у повністю прозорому режимі, тобто відображувати цей процес якимось чином у програмному коді клієнтської частини не потрібно.

Слід відмітити, що повнофункціональний веб-додаток запам'ятовує користувачів, які хоча б раз до нього звернулися, для чого слід записувати

ідентифікатор відповідного користувача до бази даних. Відповідно серверний код, який обробляє запити клієнтської частини, часто працює у зв’язці з сервером баз даних.

Для цього веб-додаток використовує систему аутентифікації, яка зберігає ідентифікатори користувачів та інші необхідні дані в базі даних. Це дозволяє зберігати контекст спілкування з кожним користувачем та надавати персоналізовані відповіді на їхні запити.

Серверна частина відповідає за:

- прийом та обробку запитів від клієнтської частини;
- взаємодію з базою даних для збереження і отримання необхідної інформації;
- забезпечення шифрування та безпеки переданих даних;
- відправку відповідей клієнтській частині.

Клієнтська частина, у свою чергу, відповідає за:

- відправку запитів до серверу;
- відображення отриманих відповідей;
- забезпечення зручного інтерфейсу для користувача.

Таким чином, логіка роботи веб-додатку відповідає концепції «клієнт-сервер», де сервер виконує роль основного обробника даних і керує усіма процесами взаємодії, а клієнтська частина забезпечує інтерфейс для кінцевого користувача.

Отже, архітектура проектованого веб-додатку буде такою, як показано на рис. 2.1.

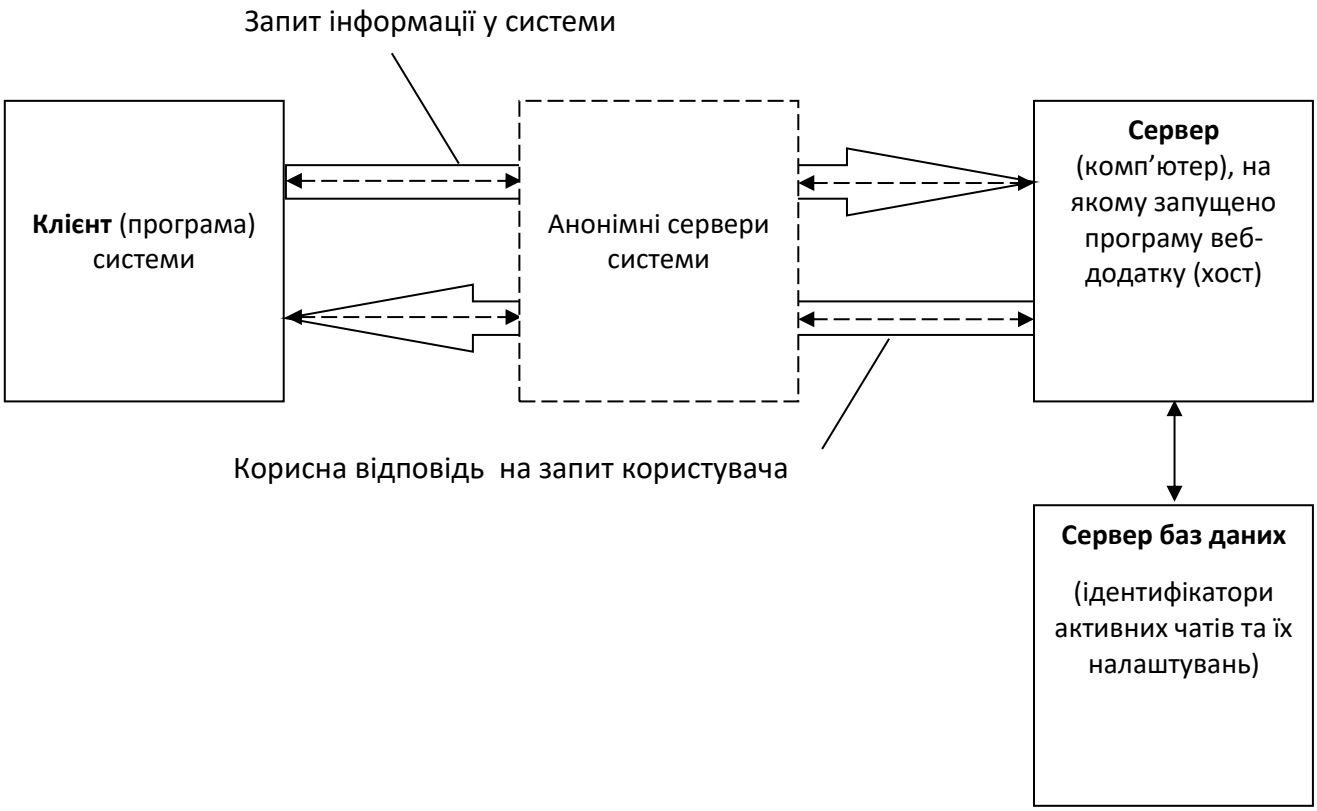


Рис. 2.1 Архітектура проектованого програмного продукту типу веб-додаток мовою JavaScript

Таким чином, архітектура проектованого програмного продукту є визначеною, тому можна переходити до вибору засобів розробки та подальшої програмної реалізації.

2.2 Вибір основних технологій для створення веб-додатку

У процесі розробки програмних продуктів одним з ключових викликів, що постають перед розробниками, є селекція оптимальної моделі, технології чи парадигми створення софту [12]. На сучасному етапі у виробничій сфері активно використовуються підходи структурного (процедурного) та об'єктно-

орієнтованого програмування. Кожен з них має свої характерні риси, сильні та слабкі сторони, які варто розглянути більш детально.

Структурне, або, як його часто називають програмісти-практики, процедурне програмування, ґрунтується на застосуванні окремих структурних блоків, передусім - підпрограм (процедур та функцій). Цей підхід передбачає конструювання програми, спираючись на три фундаментальні принципи: послідовність, розгалуження та повторення.

Послідовність полягає в тому, що оператори та блоки програмного коду виконуються у певному порядку, один за одним, без порушення черговості.

Слідування означає, що оператори і блоки програмного коду слідують і виконуються один за іншим. Розгалуження реалізується різними умовними операторами типу if (а також, оператор «?», switch/case і т.п.), і дозволяє вибирати один з декількох подальших варіантів виконання програми. Повторення зазвичай відносять на рахунок циклів (що йдуть підряд багаторазових повторів одного і того ж ділянки коду), хоча цей же принцип можна віднести і до підпрограм.

Взагалі ж, структурне програмування є більш старшою методикою програмування, на зміну якій поступово прийшло об’єктно-орієнтоване програмування (деякі сучасні мови програмування загального призначення навіть не дозволяють створити структурну програму, тільки об’єктно-орієнтовану – як, наприклад, Visual C# чи Java). Проте, при створенні невеликих програм (наприклад, до 10000 рядків коду і без передбачуваного розширення) застосування цієї методики програмування більш виправдано і код краще сприймається, ніж його об’єктно-орієнтований варіант.

Суть же методології об’єктно-орієнтованого програмування [15] полягає в тому, що система розглядається, як сукупність окремих сутностей – об’єктів, які мають набір якихось своїх внутрішніх параметрів – властивостей, а також можуть взаємодіяти між собою за допомогою деяких дій – викликів методів (або більш непрямим чином – шляхом надсилання повідомлень, оброблюваних методами об’єктів; для цього необхідна присутність активної сутності, яка роздає

повідомлення адресатам, як, наприклад, менеджер вікон в ОС Windows, який надсилає вікнам програм адресовані до них повідомлення).

Якщо говорити про програмний код, то для того, щоб оперувати об’єктом, його спочатку потрібно створити. Об’єкти створюються як змінні, у яких типом виступає клас об’єкта. Клас являє собою опис, які властивості можуть мати об’єкти такого типу (тобто яку інформацію вони можуть зберігати), і які у них є методи (тобто які дії вони можуть виконувати). Об’єктом же є набір значень, чому саме рівні властивості даного об’єкта (а свої методи кожен об’єкт отримує від свого класу, тобто методи однакові у всіх об’єктів, що належать даному класу).

Реалізовуючи всі сутності, необхідні, згідно з алгоритмом, для роботи програми, у вигляді класів і об’єктів, розробник спрощує її розуміння для самого себе. Саме тому ОО-підхід рекомендується до застосування для великих проектів (більше десятків тисяч рядків коду), коли утримувати «в голові» всю систему цілком стає важко. Можна сказати, що розбиття програми на об’єкти і проектування їх класів наближає розуміння предметної області до звичного людського образу мислення (в разі «великих» проектів). Людина мислить класами, об’єктами і зв’язками між ними.

Відзначимо, що часто, крім розглянутих міркувань, також на вибір методики програмування впливають інші чинники, наприклад, можливість майбутнього розширення функціональності, створення якомога більш зрозумілого коду (для роботи над проектом цілої команди, а не одного програміста), або просто побажання замовника застосувати найбільш сучасний підхід до програмування та ін.

Крім розбиття (декомпозиції) всієї предметної області на об’єкти (класи) і співвідношення між ними, також ОО-підхід має на увазі дотримання трьох основних його принципів: інкапсуляція, наслідування, поліморфізм.

Під інкапсуляцією мається на увазі об’єднання даних (значення властивостей класу у деякого конкретного об’єкта) та засобів їх обробки (методи класу). Це знову ж таки зручно психологічно, так як дозволяє реалізовувати окремі завершені

сутності - класи, які самі обробляють свої дані. Звернення до об’єктів цих класів відбувається за допомогою методів, що утворюють інтерфейс класу.

Наслідування є дуже корисною можливістю, тому що дозволяє значно скорочувати обсяги повторюваного коду (до чого потрібно завжди прагнути при розробці будь-якого програмного забезпечення). Згідно з цим принципом виділяється клас, який має загальний набір властивостей і методів для декількох більш розширених класів. Цей клас оголошується батьком, базовим класом для декількох похідних від нього (нащадків, спадкоємців). Всі класи-нащадки успадковують від базового всі його властивості та методи, але до цього ще мають свої власні оригінальні властивості і / або методи.

При наслідуванні іноді методи батьківського і похідного класу мають однакове призначення, але реалізовуються по-різному. Такі методи називаються перевантаженими. При цьому ще раз підкреслимо, що призначення методу в обох випадках одне і те саме.

Поліморфізм є можливістю деякої функції (яка належить якомусь класу, тобто функції-методу, або окремій, розміщеній поза усіма класами) приймати об’єкти як батьківського, так і похідних класів, і вміти викликати перевантажені методи саме того класу, об’єкт якого був переданий в функцію. Слід сказати, що це досить специфічна можливість і в загальному багато програмістів використовують ОО-підхід і без звернення до поліморфізму.

Важливими поняттями в ООП також є: статичні члени класу, абстрактні методи і класи, дружба функцій і класів, і т.д.

Ґрунтуючись на перерахованих особливостях двох існуючих методів програмування, вибираємо об’єктно-орієнтований підхід як більш сучасний, гнучкий та прогресивний, а також такий, що відповідає середнім масштабам проектованого ПЗ.

2.3 Обґрунтування вибору мови програмування та відповідної програмної платформи для розробки

Для розробки веб-додатків можна використовувати будь-яку серверну мову програмування. Аналіз навчальних матеріалів показує, що часто для цих цілей обирають мови Python та JavaScript (зокрема, на базі серверної платформи Node.js). Якщо розглянути ширший контекст створення програмного забезпечення, то Python здебільшого застосовується для розробки інтелектуальних додатків із великою кількістю складної логіки. Для додатків, що працюють за принципом "запит-аналіз-відповідь", використання всіх можливостей Python не завжди виправдано. У цій роботі ми будемо використовувати JavaScript і платформу Node.js, розглянувши їхні можливості детальніше.

JavaScript - це мова програмування, яка має ряд особливостей, що відрізняють її від інших мов. Однією з ключових характеристик JavaScript є інтерпретованість його вихідного коду, що надає мові гнучкості, але може дещо уповільнювати швидкість виконання програм. Синтаксис JavaScript подібний до мови C, а оператори в ньому розділяються крапкою з комою. На відміну від HTML, де регістр символів не має значення, в JavaScript враховується регістр літер, що може створювати певні труднощі для початківців.

Коментарі в JavaScript оформлюються так само, як і в мові C: однорядкові коментарі починаються з //, а багаторядкові обмежуються символами /* і */. Мова використовує динамічну типізацію даних, автоматично конвертуючи їх при операціях з різними типами. Дані в JavaScript поділяються на примітивні, що містять однорідні значення і передаються у функції за значенням, та складені, які включають різнорідну інформацію і передаються за посиланням.

JavaScript базується на концепції прототипів, а не класів, що відрізняє її від багатьох інших об'єктно-орієнтованих мов. Об'єкти в JavaScript поділяються на чотири категорії: вбудовані, об'єкти браузера, документа та користувацькі об'єкти. Можливості введення-виведення в JavaScript здебільшого обмежуються взаємодією з документами та користувачами, а доступ до локальної файлової

системи за замовчуванням заборонений з міркувань безпеки, хоча браузери можуть надавати спеціальні об'єкти для роботи з файлами користувача.

Скрипти JavaScript тісно взаємодіють з об'єктами веб-сторінки, для чого їх код інтегрується в HTML-документ. Існує декілька способів включення JavaScript в HTML, найчастіше для цього використовується тег <script>. Код скрипта може розміщуватися безпосередньо в HTML-документі або в окремих файлах, які потім підключаються до основного документу. Тег <script> може містити атрибут src для вказівки URL-адреси файлу зі скриптом. У разі розміщення коду в окремому файлі, теги <script> не використовуються. Контейнерний тег <script> може містити атрибут SRC, який вказує ім’я або URL-адресу текстового файлу, що містить код сценарію. Цей атрибут необхідний в тому випадку, якщо сценарій розташований не безпосередньо в HTML-документі, а в окремому файлі. Розширення файлу зі сценарієм може бути яким завгодно, але зазвичай використовують js:

<SCRIPT SRC = «myscripts.js»> </ SCRIPT>.

Якщо сценарій розташовується в окремому файлі, то в ньому, зрозуміло, теги <SCRIPT> і </ SCRIPT> не пишуть. Сценарій, завантажений з зовнішнього файлу, можна уявити собі просто як його вставку в HTML-документ.

У браузерах, що потенційно підтримують сценарії, цю функцію користувач може відключити (зокрема, із міркувань підвищеної безпеки). Крім того, існують браузери, які принципово не підтримують сценарії. У даній ситуації бажано хоча б вивести повідомлення про те, що на сторінці був сценарій, але в даному конкретному випадку він не виконується. З цією метою в HTML-документі використовують контейнерний тег <noscript>, в якому розміщують текст, який з’явиться користувачеві за умови, що сценарій з тієї, чи іншої причини не виконуватиметься. Всі браузери, які підтримують сценарії, проігнорують вміст тегів <noscript> крім тих випадків, коли підтримка сценаріїв відключена. Браузери, що принципово не підтримують сценарії, навпаки, вміст тега <script> опустять (особливо, якщо він вкладений у теги <!-- -->, які є HTML-коментарем), а тега <noscript> - відобразять. Приклад наведено нижче:

<HTML> <HEAD>

<TITLE> Приклад застосування тега NOSCRIPT </ TITLE>

</ HEAD>

<SCRIPT TYPE = "text / JavaScript»>

<! -

alert («Підтримка JavaScript включена»); // ->

</ SCRIPT>

<NOSCRIPT>

<H2> Ваш браузер не підтримує JavaScript або його підтримка відключена </ H2>

</ NOSCRIPT>

</ HTML>

Тут була використана функція alert (повідомлення) для виведення повідомлень в маленькому діалоговому вікні.

Як уже зазначалося, в окремих файлах зазвичай розміщують бібліотеки функцій (визначення функцій), а також сценарії, які використовуються в декількох HTML-документах одного або декількох сайтів. Сценарій можна також писати у вигляді рядка операторів, між якими ставиться крапка з комою. Такий рядок, взятий в лапки, служить у якості значення атрибута-події, наприклад:

<IMG SRC = "mypicture.gif" ONCLICK = "alert ('Привіт!')".

Виклики функцій і їх визначення можуть слідувати в довільному порядку, але тільки якщо вони розташовані в одному і тому ж контейнері <script>. Якщо вони розміщуються у різних контейнерах <script>, то необхідно, щоб визначення функції передувало її виклику. Аналогічно, якщо визначення функцій знаходяться в окремому файлі, то його необхідно завантажити в HTML-документ раніше викликів цих функцій.

Браузер інтерпретує теги HTML послідовно. Так, зустрівши вираз виклику функції myfunc() в одному контейнері <script>, браузер ще не має в пам'яті визначення цього об'єкта (функції), розташованого в іншому контейнері <script>. Якщо ж визначення функції і її виклик знаходяться в одному і тому ж контейнері <script>, то його вміст спочатку завантажується в пам'ять, а потім аналізується.

Коли інтерпретатор зустрічає виклик функції, то він шукає її визначення в пам'яті і в разі успіху виконує код цієї функції.

Втім, настільні додатки на Node.JS майже відсутні, а широко даний інструмент використовується для організації бек-енд складових сучасних веб-додатків. Таким чином, завдяки даній платформі, мова JavaScript стала досить поширеною мовою серверного програмування.

2.4 Проектування способу взаємодії користувача із проектованим веб-додатком

Взаємодія користувача з веб-додатком розпочинається із відкриття браузера і введення URL-адреси веб-додатку. Після завантаження сторінки користувачеві буде запропоновано почати роботу шляхом натискання кнопки "Почати" на головній сторінці.

На головній сторінці веб-додатку користувач побачить навігаційне меню з доступними функціями. Основні функції веб-додатку включають:

а) Поточна погода: Користувач може ввести назву міста в текстове поле і отримати інформацію про погодні умови в даний момент у зазначеному місті.

б) Погода на завтра: Користувач може ввести назву міста і отримати інформацію про погодні умови на наступний день у заданому місті.

в) Прогноз на тиждень: Користувач може ввести назву міста і отримати прогноз погоди на найближчі 7 днів у цьому місті.

г) Прогноз на 3 дні: Користувач може обрати раніше встановлене основне місто і отримати інформацію про погодні умови на найближчі 3 дні.

ґ) Зміна мови: Користувач може обрати мову інтерфейсу додатку, що дозволить отримувати інформацію на обраній мові.

д) Встановлення основного міста: Користувач може ввести назву міста і встановити його як основне місто, для якого будуть видаватися погодні умови за замовчуванням.

е) Перегляд основного міста: Користувач може переглянути поточно встановлене основне місто.

є) Довідка: Користувач може переглянути інструкції та список доступних функцій веб-додатку.

Даних функцій буде достатньо для того, щоб реалізувати основну функціональність, визначену завданням на розробку та постановкою задачі даної роботи.

2.5 Формалізація алгоритмів роботи проектованого додатку

Як уже зазначалося вище, алгоритми роботи з проектованим програмним забезпеченням не є досить складними, а навпаки досить просто виражаються схемою виду рис. 2.2.

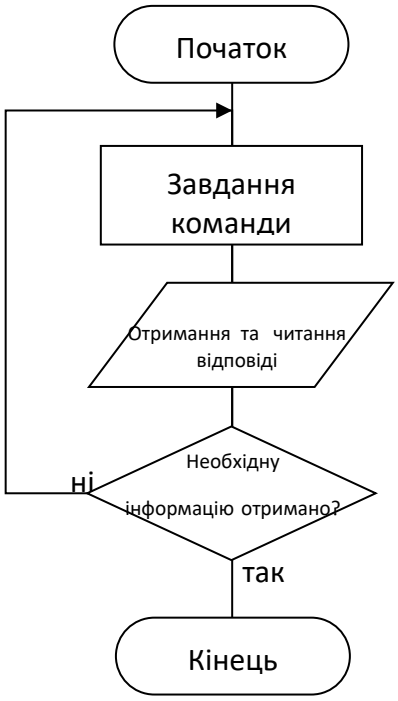


Рис. 2.2 Блок-схема алгоритму роботи веб-додатку, що надає інформацію про погодні умови

Керуючись інформацією, наведеною у даному підрозділі, а також у вищенаведеному викладі, можна переходити до процесу проведення програмної реалізації веб-додатку мовою JavaScript, що і виконується у наступному розділі.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКУ ДЛЯ ВІДСТЕЖУВАННЯ МЕТЕОРОЛОГІЧНИХ УМОВ

3.1 Налаштування програмного середовища для функціонування додатку

Для створення програмного продукту типу веб-додаток мовою JavaScript, основна функціональність якого пов’язана із наданням користувачеві інформації про погоду, необхідно виконати кілька організаційних кроків перед початком програмування. В першу чергу, для створення довільного веб-додатку потрібно отримати його ідентифікатор, або за термінологією – токен (англ. Token). За допомогою даного ідентифікатору здійснюється уся взаємодія в системі «сервер – користувач» і без нього створити програмне рішення будь-яким способом не можливо:

По-перше, необхідно встановити та налаштувати серверне середовище для розгортання веб-додатку. Це може бути локальний сервер для розробки та тестування, або віддалений хостинг для розміщення додатку в Інтернеті. Популярними варіантами є Apache, Nginx, Microsoft IIS тощо. Необхідно забезпечити підтримку відповідних технологій, таких як PHP, Node.js, .NET, залежно від вибраного стеку технологій для розробки.

По-друге, слід налаштувати середовище для роботи з базами даних, де зберігатиметься інформація про погоду та дані користувачів. Поширеними варіантами є MySQL, PostgreSQL, MongoDB та інші.

По-третє, потрібно інтегрувати веб-додаток з API сервісу погоди, такого як OpenWeatherMap, AccuWeather або інших провайдерів погодних даних. Це зазвичай передбачає реєстрацію облікового запису та отримання ключа API для автентифікації запитів.

```
xhr.open('POST', loginUrl, true);
xhr.setRequestHeader('Content-Type', 'application/json; charset=UTF-8');
xhr.addEventListener('load', function() {
  var responseObject = JSON.parse(this.response);
  console.log(responseObject);
  if (responseObject.token) {
    tokenElement.innerHTML = responseObject.token;
  } else {
    tokenElement.innerHTML = "No token received";
  }
});

var sendObject = JSON.stringify({name: user, password: password});

console.log('going to send', sendObject);

xhr.send(sendObject);
```

Рис. 3.1 Форма отримання токена, яка дозволяє створювати новий веб-додаток

Нарешті, для розробки інтерфейсу користувача потрібно налаштувати середовище для роботи з HTML, CSS та JavaScript, а також, можливо, використати фреймворки для фронтенду, такі як React, Angular або Vue.js.

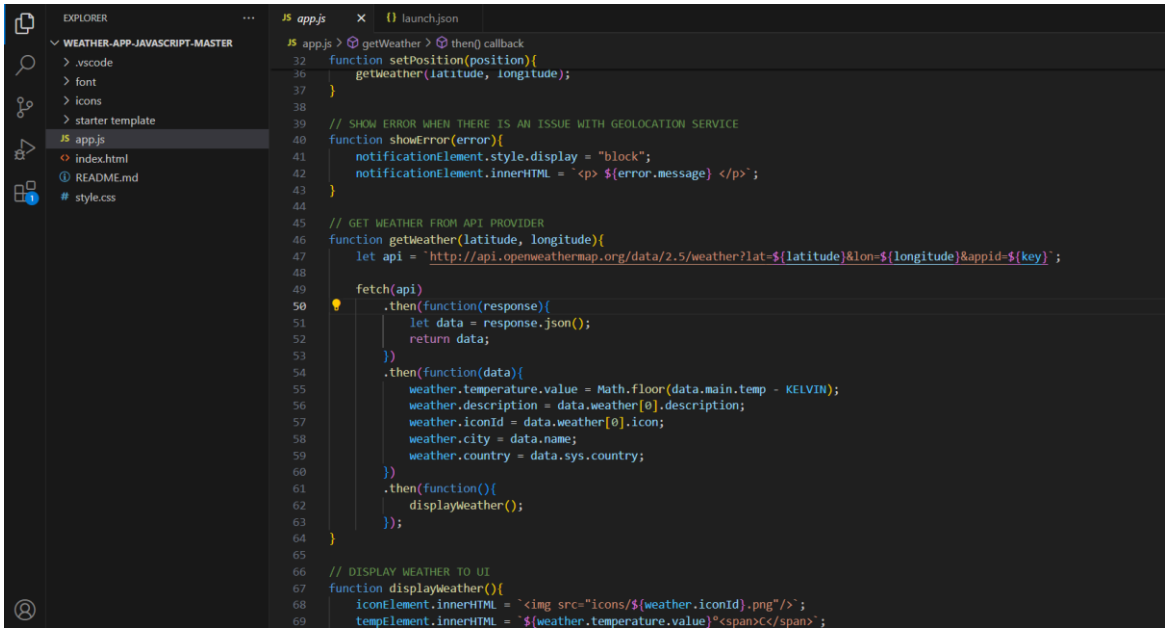


Рис. 3.2 Отримання токену безпосередньо від ресурсу моніторингу метеорологічних умов

Для створення такої програми, як веб-додаток обов’язкових налаштувань більше не потрібно (хоча з додатковими налаштуваннями можна ознайомитися

безпосередньо на рис. 3.1). Однак, продукт, що створюється саме у даній роботі призначений для відображення погодних умов. Очевидно, що їх потрібно зчитувати з одного із якихось зовнішніх джерел, оскільки створювати свій метеорологічний центр в даному випадку ані доцільно, ані можливо взагалі. Дані про погоду можна зчитувати безкоштовно за допомогою сервісу OpenWeatherMap від компанії OpenWeather (м. Лондон). Даний проект використовує у якості основи відомості з сайтів метеорологічних служб розвинутих країн (які публікують ці дані в Інтернет), тобто має функцію веб-агрегування. Однак, даний ресурс не є агрегатором у чистому вигляді, оскільки у великій мірі спирається на дані приватних метеостанцій, кількість яких зростає у всьому світі досить високими темпами. Таким чином, OpenWeatherMap має свої внутрішні алгоритми, які об'єднують інформацію з різних джерел і видають єдиний консолідований прогноз для вказаної місцевості.

При використанні сервісу OpenWeatherMap виникає два технічних моменти, що вимагають вирішення. По-перше, на ньому слід зареєструватися (сервіс допускає безкоштовний тарифний план із дещо зменшеною функціональністю, якої вистачає для цілей даної роботи), в результаті чого сервіс надає ключ, що буде потрібний для використання у боті, що проектується – рис. 3.3.

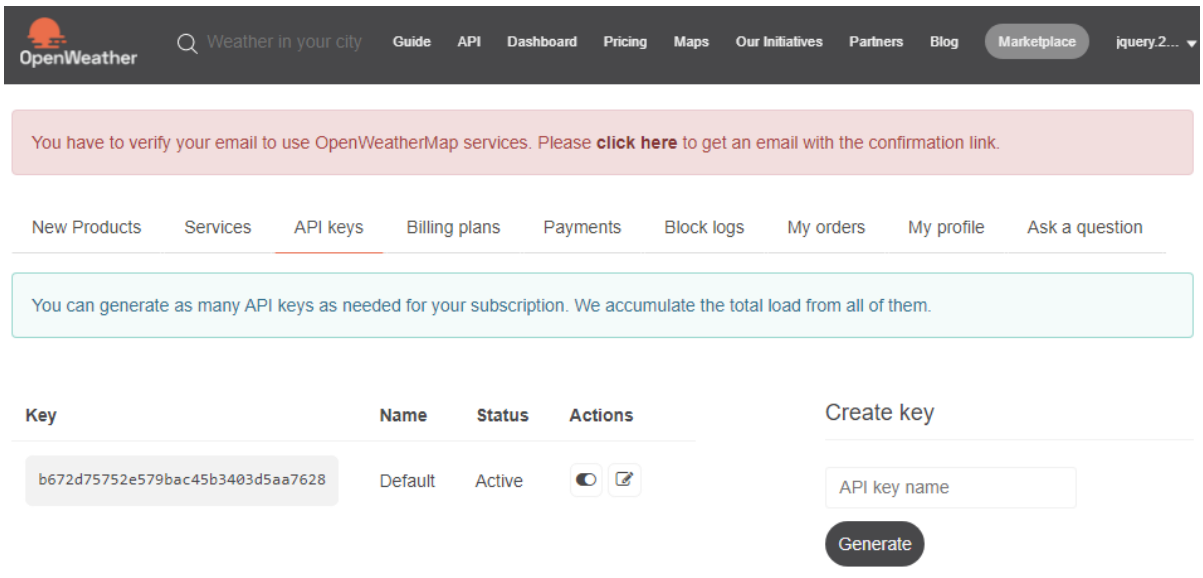


Рис. 3.3 Персональна сторінка безкоштовного тарифного плану на сервісі OpenWeatherMap

По-друге, для використання сервісу OpenWeatherMap, йому у HTTP запиті слід передавати широту і довготу місцевості, у якій треба визначити погоду. Очевидно, що користувач додатку, який хоче дізнатися погоду навіть у своєму рідному місті навряд чи знає його географічні координати. Відповідно, виникає проблема переведення імені місцевості (зазвичай, це просто ім'я населеного пункту) у точні координати – широту та довготу. Для цього можна використати ще один сторонній ресурс, що допускає вільне використання – проект OpenCage.

OpenCage є геокодувальною системою, однією із функцій якої якраз і є переведення текстової назви місцевості у географічні координати. Звичайно, для використання даного сервісу також слід зареєструватися, в результаті чого також надається відповідний ключ (Geocoding API Key) – рис. 3.4.

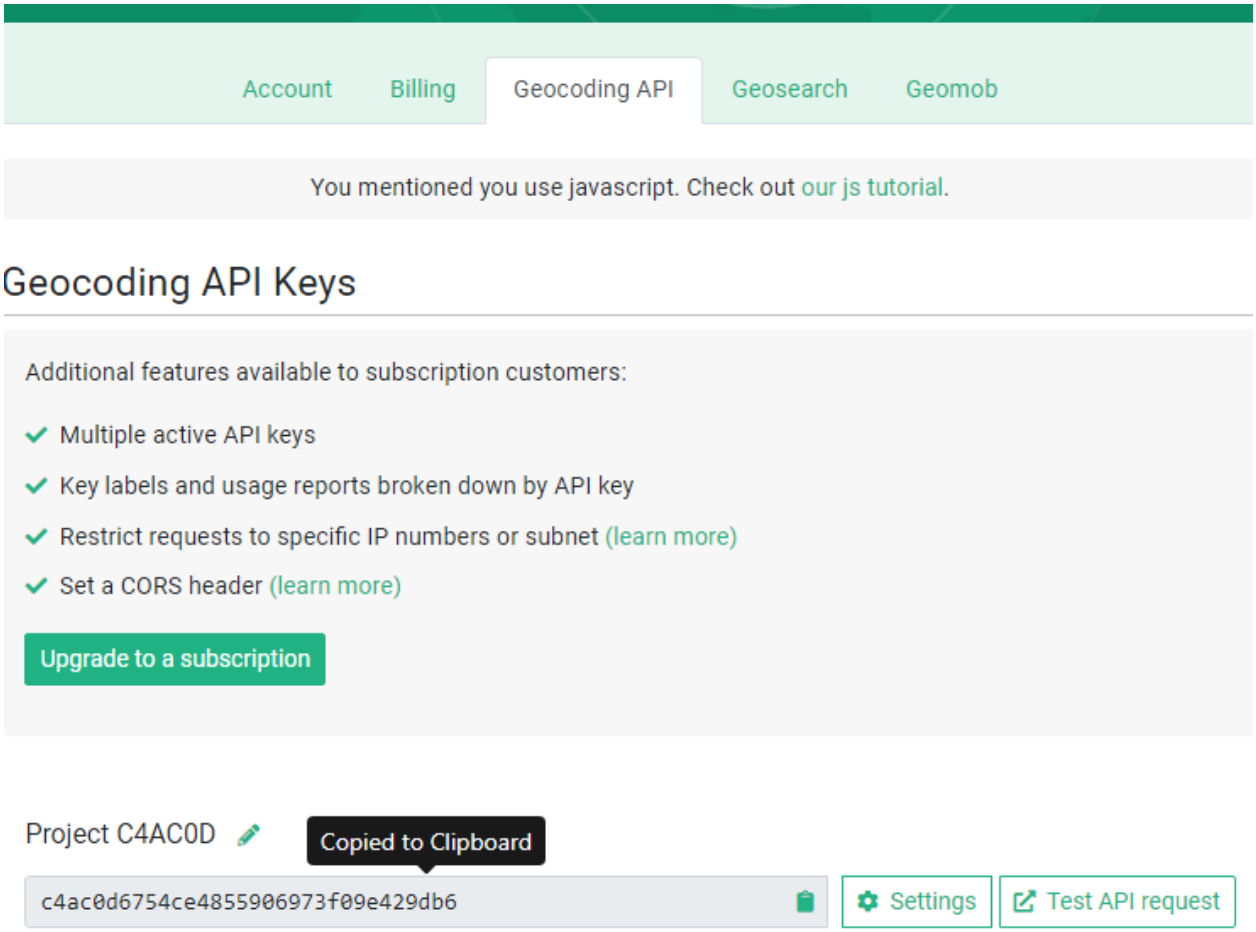


Рис. 3.4 Персональний кабінет пробного тарифного плану на платформі OpenCage

Ще одним окремим програмним продуктом, що є потрібним для зведення повноцінного програмного рішення є система управління базами даних (далі – СУБД). На сьогоднішній день такими, що реально використовуються є два класи СУБД: реляційні (або по-іншому такі, що використовують мову структурованих запитів SQL) та нереляційні (NoSQL системи). Перші є значно більш поширеними, але в одній зв’язці з Node.JS програмісти на мові JavaScript найчастіше використовують якраз NoSQL СУБД MongoDB.

Одним з простих варіантів використання MongoDB є зберігання налаштувань програми. Це зручно, оскільки такі налаштування зазвичай унікальні та неефективно розміщувати їх у таблицях. У більш загальному випадку MongoDB використовується для зберігання великих обсягів неструктурованих даних, таких як окремі документи, що робить цю СУБД документоорієнтованою.

Для початку роботи з MongoDB необхідно завантажити її з офіційного сайту mongodb.com, вибравши безкоштовну версію MongoDB Community Server.

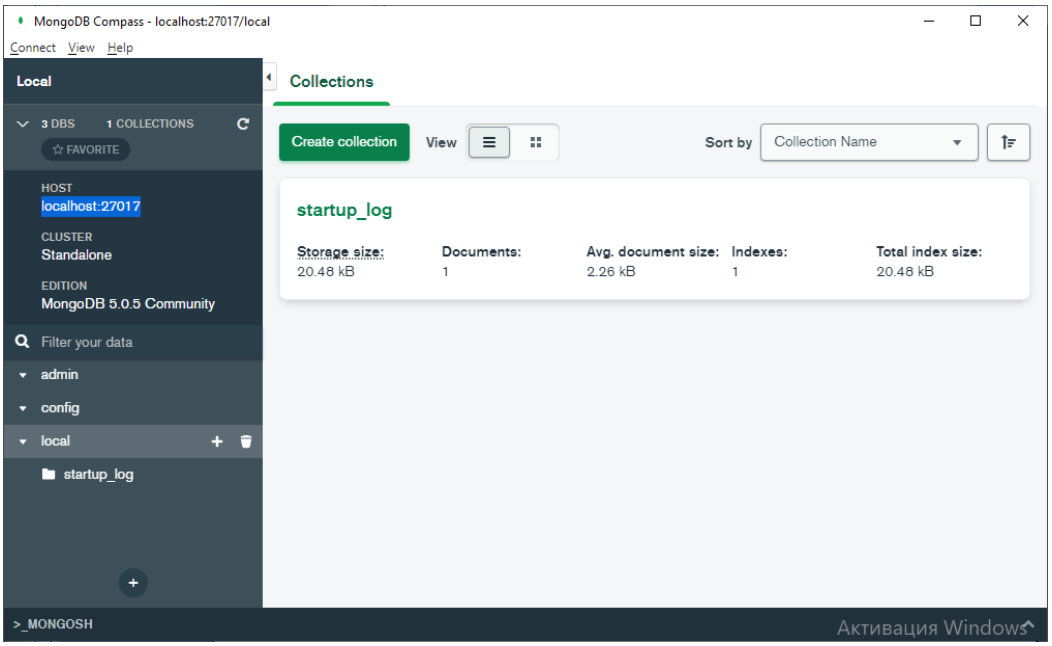


Рис. 3.5 Вікно засобу Compass для управління СУБД MongoDB

Як уже згадувалося, для реалізації проекту буде задіяна платформа Node.js, яку необхідно скачати з її офіційного веб-сайту https://nodejs.org та інсталювати на комп'ютер, де відбуватиметься розробка.

Ефективна робота над Node.js проектами вимагає наявності менеджера пакетів, доступного через командний рядок. Один з таких менеджерів, npm (Node

Package Manager), вже інтегрований в екосистему Node.js. Проте, за потреби, можна додатково встановити альтернативний менеджер пакетів, такий як yarn, що може спростити процес інсталяції та оновлення додаткових модулів Node.js, необхідних для проекту.

Таким чином, наявність Node.js та зручного менеджера пакетів на кшталт npm або yarn є необхідною передумовою для початку розробки веб-застосунку з використанням цієї платформи. Ці інструменти забезпечать розробників потужним та гнучким середовищем для створення та управління проектом. Далі слід виконати команду створення нового порожнього проекту Node.JS:

```
npm init
```

Щоб уникнути необхідності надавати відповіді на кожне запитання окремо та дозволити системі використовувати стандартні значення, необхідно додати параметр "yes" до команди:

```
npm init --yes
```

Після ініціалізації нового проекту можна встановлювати необхідні компоненти, специфічні для проекту, вручну. Для роботи з базою даних MongoDB зручно використовувати бібліотеку mongoose. Для цього потрібно виконати команду у командному рядку в каталозі проекту.

Щоб спростити взаємодію з базою даних MongoDB на мові JavaScript, рекомендується скористатися спеціальною бібліотекою mongoose. Ця ODM-бібліотека (Object Data Modelling) дозволяє зручно зіставляти об'єкти класів з документами колекцій у базі даних. Для встановлення mongoose достатньо виконати наступну команду в командному рядку з каталогу проекту:

```
npm install mongoose
```

Оскільки додаток буде обробляти HTTP-запити, доцільно також додати бібліотеку Axios за допомогою команди:

```
npm install axios
```

Для розробки веб-додатку може стати в нагоді пакет node-web-api. Його можна завантажити, скориставшись командою:

```
npm install node-web-api
```

Виконавши вищезазначені дії, можна констатувати, що підготовчий етап конфігурації програмного оточення успішно завершено і система готова до подальшої розробки. Тепер можна безпосередньо приступати до реалізації веб-додатку для моніторингу метеорологічних умов з використанням мови програмування Node.js.

3.2 Особливості реалізації алгоритмів роботи додатку

Ключі, що були отримані під час виконання дій, описаних у попередньому підрозділі вбудовуються у файл config.js:

```
const = "5292362960:AAEcNwiUlwvxxsi0Ct89IoBBIargZs6IuSU"
const OpenWeatherKey = "b672d75752e579bac45b3403d5aa7628"
const MongodbURI = "mongodb://localhost:27017"
const OpenCageKey = "c4ac0d6754ce4855906973f09e429db6"
module.exports = { TelegramToken, OpenWeatherKey, MongodbURI, OpenCageKey }
```

Далі, інформація, що є налаштуваннями додатку, як уже було сказано вище, має зберігатися у базі даних MongoDB. У такому випадку створюється схема даних сховища, яка вміщується у каталог models (як відомо, відповідно до шаблону Model-View-Controller дані, що обробляються програмою, відносяться до сутності «модель») та являє собою файл User.js:

```
const userSchema = mongoose.Schema({
  user_id: {
    type: Number,
    required: true,
    unique: true
  },
  city: {
    type: String,
    required: true
  },
  lang: {
    type: String,
```

```
    default: 'en'
  }
});
```

База даних містить інформацію про зареєстрованих користувачів додатку, включаючи їх унікальні ідентифікатори, вибране місто для відображення погодних даних та переважну мову інтерфейсу (англійська встановлена за замовчуванням). Коли до системи додається новий користувач, у базі даних автоматично генерується новий документ, структура якого виглядає наступним чином – рис. 3.6.

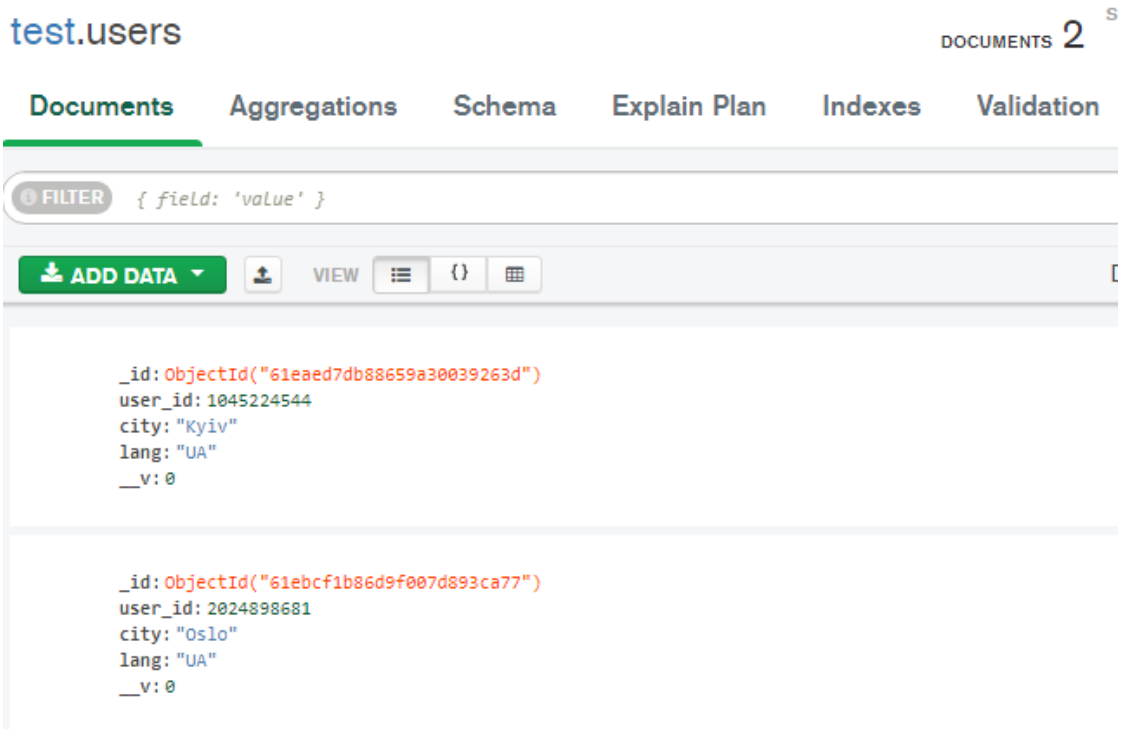


Рис. 3.6 Дані, що зберігаються у базі MongoDB

У коді програми, що є основою веб-додатку і міститься у файлі index.js, використовується 7 функцій та 12 методів для об’єкта MyApp, який ініціалізується таким чином:

```
const MyApp = new AppToken(AppToken, { polling: true });
```

Функції, розроблені у програмі, є наступними (слід відмітити, що у коді програми об’ява функцій виконана у стрілковому стилі, який є не дуже зручним для опису в тексті документа, тому тут вони наводяться у традиційній формі «назва-список аргументів»):

а) conv_time(timestamp) – функція, що перетворює час, заданий у формі TIMESTAMP UNIX (тобто це кількість секунд, що пройшли з 01.01.1970) у гарний рядок;

б) conv_date(timestamp) – функція, аналогічна до попередньої, але формується не час, а дата у вигляді рядка;

в) wEndpoint(latitude, longitude, language) – функція, що отримує погоду від сервісу OpenWeatherMap за допомогою координат;

г) wIcon(icon) – функція, що повертає картинку, яка відповідає погоді;

д) wHTMLTemplate(data, date, city) – функція, яка формує великий рядок (у форматі HTML), що містить усю необхідну інформацію про погоду, дату, місцевість;

е) getWeather(chat_id, latitude, longitude, language = 'en', index, city) – сервісна функція, яка є обгорткою для функції wEndpoint і виконує додаткові дії;

є) getInfo(chat_id, user_id, city, index) – функція ще більш високого рівня абстракції, аніж getWeather, і включає попередні запити до бази даних, з метою визначення базового міста користувача та встановленої ним мови, а також визначення координат у сервісі OpenCage за назвою міста.

Серед функцій веб-додатку з прогнозом погоди можна виділити наступні:

- Пошук міста: користувач може ввести назву міста або його поточні координати для отримання прогнозу погоди.
- Відображення поточної погоди: додаток показує температуру, опис погодних умов, швидкість вітру, вологість та інші актуальні дані для вибраного місця.
- Прогноз погоди на кілька днів: користувач може переглянути прогноз погоди на кілька наступних днів, зазвичай до 5-7 днів.
- Вибір одиниць вимірювання: можливість вибрати метричну або імперську систему одиниць температури та швидкості вітру.
- Збереження улюблених місць: користувачі можуть додавати міста до списку улюблених для швидкого доступу.
- Попередження про небезпечну погоду: додаток може надсилати сповіщення про екстремальні погодні явища, такі як урагани, зливи або сильний вітер.

– Інтеграція із соціальними мережами: можливість поділитися прогнозом погоди у своїх акаунтах в соціальних мережах.
– Деталізована інформація: детальні дані про хмарність, тиск, вологість, схід і захід сонця та інші метеорологічні показники.

Інші деталі та можливості програмного коду можна знайти у початковому тексті, наведеному у Додатку А, які можуть бути додані згідно з первинною специфікацією проекту.

3.3 Тестування створеного програмного продукту

Після завершення етапу реалізації будь-якого програмного продукту, необхідно провести його ретельне тестування. Процедура тестування для даного проекту складається з наступних кроків. Спочатку у командному рядку, перебуваючи в каталозі проекту, потрібно виконати команду для запуску проекту:

```
npm start
```

Після введення цієї команди розпочнеться процес завантаження та виконання проекту. При цьому командний рядок буде заблоковано, а сам проект перейде у режим безперервного виконання – рис. 3.7.

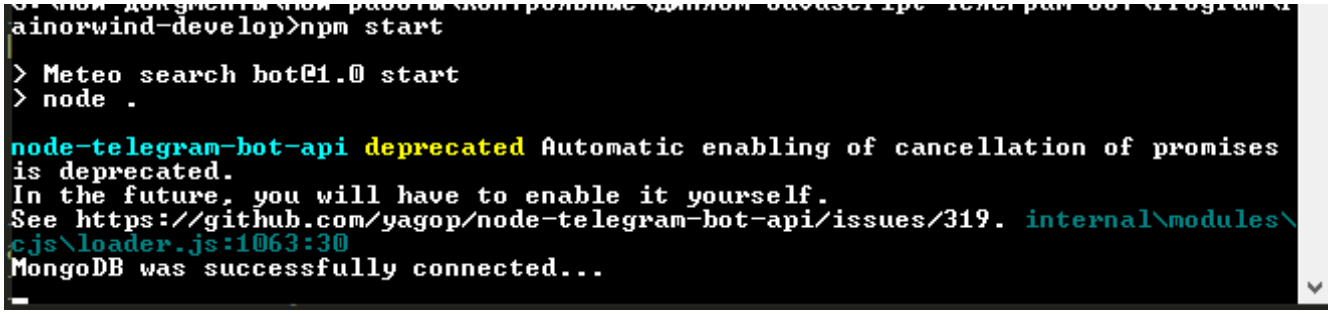


Рис. 3.7 Блокування консолі завантаженим проектом протягом всього часу його роботи

Потрібно відзначити, що робота з веб-додатком можлива лише коли на якомусь комп’ютері, який підключений до Інтернету, запущена консоль з відповідним додатком на платформі Node.JS. Це передбачає, що програмне

забезпечення веб-додатку не може бути розміщене на домашньому комп’ютері, який періодично вимикається, оскільки під час вимкнення додаток не буде працювати. Тому для розгортання програми у професійних умовах рекомендується використовувати хостинг Інтернету, який підтримує проекти на базі Node.JS та MongoDB. На жаль, зазвичай отримати безкоштовний доступ до сервера з Node.JS не є легкою задачею, оскільки політика доступу може змінюватися, а доступні сервери обмежені в кількості. Тому більш надійним варіантом є використання платних хостингів або безкоштовних альтернатив з постійним моніторингом для вчасного переходу на інші ресурси, якщо потрібно. Таким чином, процес тестування проводився виключно в локальному середовищі, де сервери Node.JS та MongoDB були розгорнуті на персональному комп'ютері розробника, підключеному до глобальної мережі Інтернет. Це дозволило перевірити функціонування застосунку в умовах, максимально наближених до реальної експлуатації, без необхідності розгортання на віддалених серверах на цьому етапі.

На початковому етапі тестування варто звернути увагу на вигляд стартового екрану додатку – рис. 3.8.

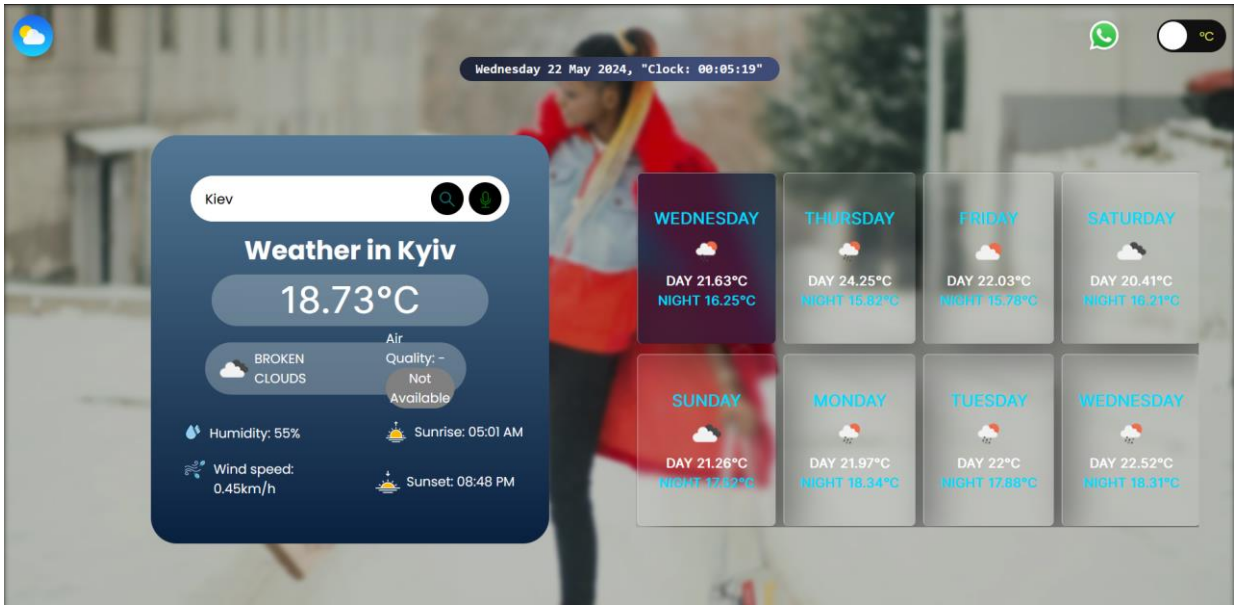


Рис. 3.8 Зовнішній вигляд стартового екрану створеного веб-додатку

При виборі опції мануалу з'являється екран, аналогічний до зображеного на рисунку 3.8.

Після введення команди з назвою певного міста веб-додаток надає правильну поточну погоду, яка відображається у вигляді картинки, що відповідає погодним умовам. Наприклад, на рисунку 3.9 картинка із невеликим дощем відображається у відповідь на текст "light rain", що свідчить про правильну роботу функції wIcon. Під цією картинкою виводиться гарно відформатоване повідомлення, яке точно відповідає HTML-шаблону, сформованому функцією wHTMLTemplate.

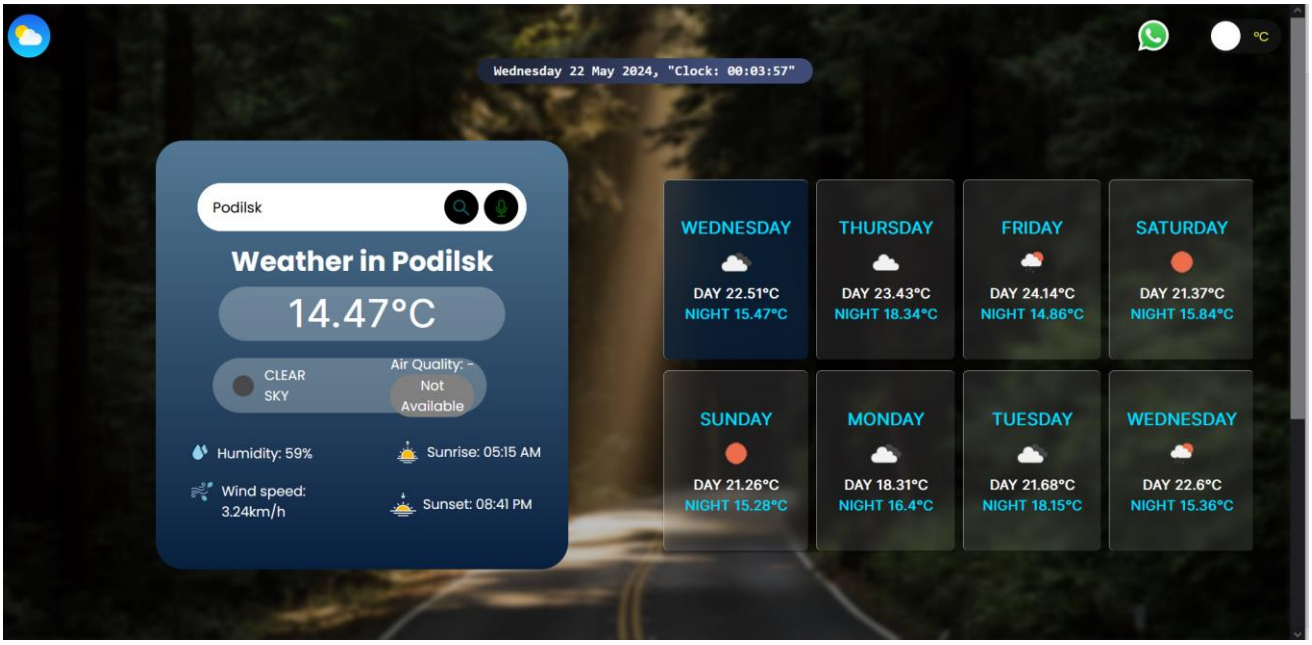


Рис. 3.9 Коректне відображення поточної погоди для заданого міста

Команди /tomorrow та /week працюють аналогічно до команди /now. Користувач завжди повинен вказувати назву міста, для якого він бажає отримати прогноз погоди. На відміну від цих команд, команда /w не потребує вказання міста, оскільки вона використовує місто, яке користувач раніше встановив за допомогою команди /set. Наприклад, як показано на рис. 3.10, спочатку користувач задав місто London, а потім використав команду /w без вказівки міста, що призвело до відображення інформації про погоду саме у Лондоні.

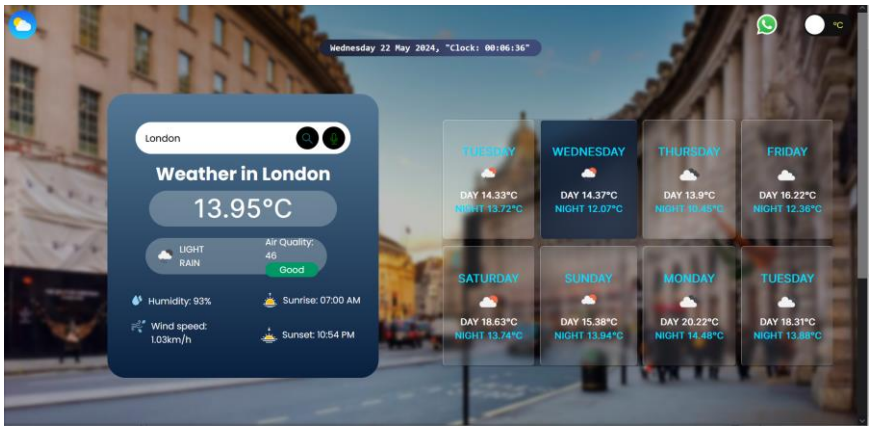


Рис. 3.10 Коректна робота команди /set для завдання міста у базі даних та /w для видачі погоди на три дні у цьому (базовому) місті

Тестування команди /lang підтвердило її коректну роботу, що відображається у зміні мови інтерфейсу веб-додатку. Однак, зазначено, що лише невелика частина інформації підлягає перекладу, тому використання цієї команди може бути не настільки доцільним, як у випадку, коли весь контент додатку має бути доступний у різних мовах.

З урахуванням цього можна зробити висновок, що хоча команда /lang є функціональною і працює коректно, її застосування може бути менш доцільним у випадку, коли основний контент веб-додатку не потребує перекладу. Однак вона може бути корисною в тих випадках, коли додаток має значну частину інтерфейсу або додаткового контенту, який може бути переведений для зручності користувачів, що володіють іншими мовами.

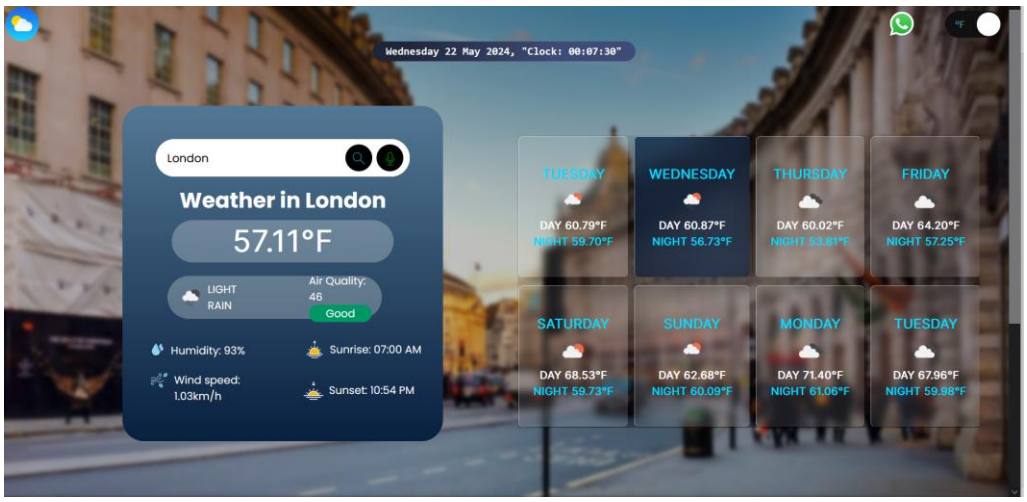


Рис. 3.11 Коректна робота функції завдання мови для вибору типу температури

Висновок з тестування вказує на те, що веб-додаток успішно виконує свою основну функціональність, яка полягає у наданні користувачеві інформації про погоду в поточний момент і прогнозування погоди на майбутні дні. Важливо зауважити, що ця функціональність працює як на різних мовах, так і в різних географічних регіонах.

Це свідчить про високу якість розробленого веб-додатку і його здатність задовольняти потреби користувачів у надійній та актуальній інформації про погоду. Такий успіх виконання функціональності сприяє позитивному враженню від використання додатку та може збільшити його популярність серед користувачів.

3.4 Аналіз ефективності розробленого рішення

Будь-яке рішення, що реалізується технічно, повинно мати економічне обґрунтування, або іншими словами економічний ефект. В окремих випадках розробка може бути обумовлена іншими видами ефекту: екологічним, політичним, соціальним і т.д. В даному випадку існує економія часу особи, що дізнається за погоду, яка виражається у фінансовому еквіваленті (за умови, що цей процес відбувається у робочий, тобто оплачуваний час).

Для того, щоби дізнатися погоду за допомогою сайту потрібно завантажити браузер, ввести адресу сайту погоди, або скористатися завчасно збереженими посиланнями, за наявності відповідної функції браузеру, та у досить великому, насиченому інформацією вікні, знайти відомості про метеорологічні умови та прочитати їх: Умовно можемо вважати, що на ці дії користувач витрачає $\Delta t = 3$ хв.

Після запровадження веб-додатку дізнатися погоду можна буде, встановивши його на персональний комп’ютер або через веб-браузер, який постійно знаходиться у пам’яті пристрою а також надає інформацію у значно більш стисненому вигляді: тільки корисні текстові дані і жодних відволікаючих факторів, накшталт рекламних оголошень для монетизації ресурсів котрі використовує

користувач веб-сайтів призначених для перегляду погодних умов. Через усі ці фактори можемо вважати, час проведення операції тепер складатиме $\Delta t' = 1$ хв.

Економія часу на однократному визначенні погоди складає:

$$\Delta t_e = \Delta t - \Delta t' = 3 - 1 = 2 \text{ хв.} \quad (3.1)$$

Можна вважати, що одна особа цікавиться погодою один раз на день (розглядаємо робочі дні), число яких у році приймаємо:

$$N = 250 \text{ днів/рік.} \quad (3.2)$$

Загальна економія часу на рік складатиме:

$$\Delta T_e = N \cdot \Delta t_e = 250 \cdot 2 = 500 \text{ хв.} = 8{,}3 \text{ год.} \quad (3.3)$$

Конкретна фінансова економія залежить від погодинної оплати праці особи, яка переглядає погоду. Якщо прийняти, що $c = 250$ грн/год (середня зарплата для кваліфікованого фахівця ІТ-сфери), то річна економія у грошовому вираженні складе:

$$C = \Delta T_e \cdot c = 8{,}3 \cdot 250 = 2075 \text{ грн.} \quad (3.4)$$

Зважаючи на те, що розробка веб-додатку відбувається в рамках написання випускної кваліфікаційної роботи, тобто виконується не за гроші, а з метою підтвердження кваліфікації, можемо вважати, що затрати на розробку складають 0 грн.

При прийнятих допущеннях розробка є економічно ефективною і у вираженні на одну особу складає 2075 грн. на рік.

Таким чином, розроблений продукт є економічно ефективним і може бути рекомендований до реального практичного використання.

ВИСНОВКИ

Таким чином, у даній роботі проведено розробку веб-додатку для моніторингу метеорологічних умов за допомогою мови програмування JavaScript на платформі Node.js. В першу чергу, визначено які функції притаманні продуктам типу веб-додаток та проаналізовано, як найбільш ефективно можна отримувати відомості про погоду. Встановлено, що використання веб-додатку є одним із найбільш ефективних рішень і має чимало плюсів, порівняно з іншими способами. Після фіксації технічних вимог до програмного продукту, було обрано його архітектуру (трирівнева, традиційна для такого типу розробки), а також інструментальні засоби для розробки. Обраною мовою програмування є найбільш популярна на сьогоднішній день мова JavaScript, а для того, щоби завантажувати серверні додатки, використано перспективну платформу Node.JS, поширення якої у веб весь час зростає.

Було розроблено необхідне алгоритмічне підґрунтя, спроектовано спосіб взаємодії додатку з користувачами, описано дані, що зберігатимуться у БД продукту. Усі ці аспекти проекту враховано при проведенні його програмної реалізації, що описано у третьому розділі. Тестування розробленого веб-додатку показало, що він у повній мірі виконує поставлену перед ним задачу, працює стабільно та без виникнення помилок системного чи логічного характеру. Відповідно, розроблений програмний продукт може бути рекомендований до впровадження та постійного використання мовою програмування JavaScript (для чого потрібно лише розмістити проект на хостингу у мережі Інтернет, що підтримує платформу Node.JS).

Робота прийшла апробацію на Всеукраїнській науково-технічній конференції «Застосування програмного забезпечення в ІКТ» за темою «Розгляд аналогів програмного забеспечення для моніторингу метеорологічних умов», м.Київ, ДУІКТ, 14 квітня 2024 року, та на Всеукраїнській науково-практичній конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті» за темою «Розробка програмного засобу для моніторингу метеорологічних умов мовою JavaScript», м.Київ, ДУІКТ, 1 травня 2024 року [14, 15].

ПЕРЕЛІК ПОСИЛАНЬ

1. Marijn H. Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming UK, 2018. – 472 с.

2. Cody Lindley JavaScript (ES2015+) Enlightenment. Available at: https://frontendmasters.com/guides/javascript-enlightenment/

3. Hawkins C. Apache Web Server Administration and E-Commerce Guide:, 2001. – 336 с.

4. Ross V. Creation of sites: HTML, CSS, PHP, MySQL. Textbook, MGDD, 2010. – 107 с.

5. Addy O. Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide US 2013. – 347 с.

6. Expressive Javascript Marijn Haverbeke - No Starch Press, 2015. – 91 с.

7. The Modern JavaScript Tutorial Available at: https://javascript.info/

8. Henrik Joreteg Human Javascript US, 2018. – 128 с.

9. David Harron Node.js. Development of server-side web applications with JavaScript / Harron David. - M.: DMK Press, 2016. – 405 с.

10. Flavio Copes The JavaScript Beginner’s Handbook 2013. - 400 с.

11. Cantelon, M. Node.js in action / M. Cantelon. 2015. – 950 с.

12. Cantelon, Mike Node.js in action / Mike Cantelon. 2014. – 564 с.

13. Ружанський Д. О. Розгляд аналогів програмного забеспечення для моніторингу метеорологічних умов. Всеукраїнсьа науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24.04.2024, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2024. С.38 - 39

14. Ружанський Д. О. Розробка програмного засобу для моніторингу метеорологічних умов мовою JavaScript. IV Всеукраїнська науково-практичноа конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2024. С.116 – 118.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного засобу для моніторингу метеорологічних умов мовою JavaScript

Виконав студент 4 курсу
групи ПД-44
Ружанський Дмитро Олексійович
Керівник роботи
д.т.н., професор, зав. кафедри ТЦР, Жебка Вікторія Вікторівна

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – спрощення моніторингу метеорологічних умов за рахунок використання веб-застосунку, розробленого мовою JavaScript.
- **Об'єкт дослідження –** моніторинг метеорологічних умов.
- **Предмет дослідження –** веб-застосунок для моніторингу метеорологічних умов.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Визначити вимоги розробки застосунку для моніторингу метеорологічних умов: розробка архітектури, визначення frontend/backend-частини, проектування класів.
2. Проаналізувати алгоритми та методи спискової структури для задачі формування програмного засобу для моніторингу метеорологічних умов.
3. Провести аналіз інструментальних засобів для моніторингу метеорологічних даних: OpenWeatherMap API, WeatherStack API, AccuWeather API.
4. Розробити алгоритм роботи застосунку.
5. Реалізувати програмно ключові функціональні можливості.
6. Провести функціональне та модульне тестування додатку.
7. Визначити сферу використання застосунку як надання актуальної інформації про метеорологічні умови.

3

АНАЛІЗ АНАЛОГІВ

Критерій	Meteofor	AccuWeather	Dark Sky	Meteored	Windy	WeatherHub
Розширена система пошуку	+	+	-	+	+	+
Оцінка проміжних і кінцевих результатів	+	+	-	+	-	+
Голосовий пошук	-	+	-	-	+	+
Виведення поточної дати та часу	+	+	+	+	+	+
Швидкість завантаження сторінок	500 мс	400 мс	450 мс	550 мс	350 мс	300 мс
Зміна мови	+	+	-	+	+	+
Зміна шкал вимірювання	+	+	+	+	+	+
Чат-бот для виведення інформації	-	+	-	-	-	+

4

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги:

1. Додаток повинен візуалізувати зібрані метеорологічні дані у зручному для користувача форматі, наприклад, за допомогою графіків, діаграм або карт.
2. Додаток має надавати можливість прогнозувати погодні умови на основі зібраних даних за допомогою алгоритмів статистичних методів.
3. Додаток повинен мати функцію сповіщення користувачів про екстремальні погодні умови, такі як шторми, повені або посухи.

Нефункціональні вимоги:

1. Додаток повинен бути оптимізований, особливо при обробці великих обсягів даних від багатьох метеостанцій.
2. Веб-інтерфейс додатку має коректно відображатися та функціонувати в різних веб-браузерах та на різних пристроях.
3. Інтерфейс додатку повинен бути інтуїтивно зрозумілим та легким у використанні для користувачів різного рівня технічної грамотності.

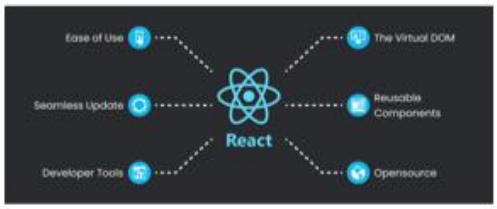
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

OpenWeather API

mongoDB.

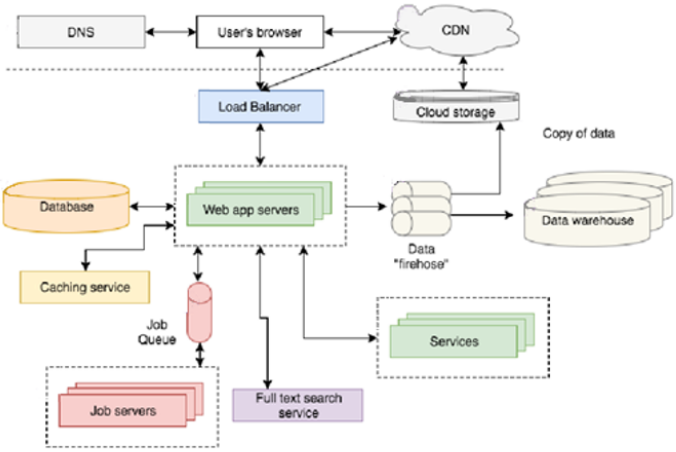
JavaScript



OpenCage

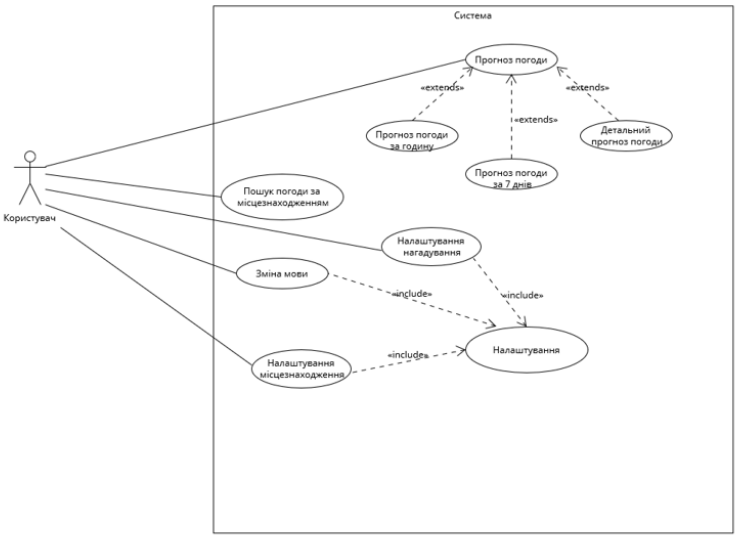
6

Архітектура додатку процесу моніторингу метеорологічних умов



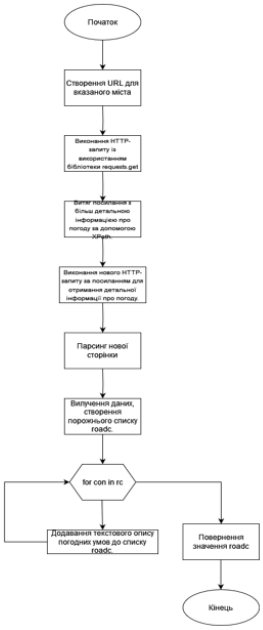
7

Діаграма варіантів використання роботи програми



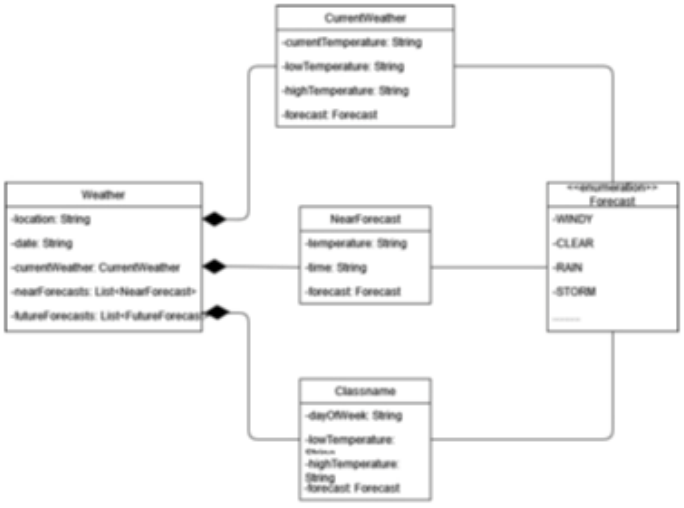
8

Блок-схема алгоритму роботи додатку



9

Діаграма класів для визначення погоди в місцевості



10

Мапа сайту

11

ЕКРАННІ ФОРМИ

Головна форма додатку

12

ЕКРАННІ ФОРМИ

13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Ружанський Д. О. Розгляд аналогів програмного забезпечення для моніторингу метеорологічних умов. Матеріали Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез 10.14.2024, ДУТ, м. Київ. с. 38 - 39
2. Ружанський Д. О. Розробка програмного засобу для моніторингу метеорологічних умов мовою JavaScript. Збірник матеріалів IV Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті» 01.05.2024, ДУТ, м. Київ. с. 116 – 118

15

ВИСНОВКИ

1. Визначено вимоги розробки застосунку для моніторингу метеорологічних умов: розробка архітектури, визначення frontend/backend-частини, проектування класів.
2. Проаналізовано алгоритми та методи спискової структури для задачі формування програмного засобу для моніторингу метеорологічних умов.
3. Проведено аналіз інструментальних засобів для моніторингу метеорологічних даних: OpenWeatherMap API, WeatherStack API, AccuWeather API.
4. Розроблено алгоритм роботи застосунку.
5. Реалізовано програмно функціональні можливості.
6. Проведено функціональне та модульне тестування додатку.
7. Визначено сферу використання застосунку як надання актуальної інформації про метеорологічні умови.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Основний модуль index.js:

```
// api key : 82005d27a116c2880c8f0fcb866998a0


// SELECT ELEMENTS

const iconElement = document.querySelector(".weather-icon");

const tempElement = document.querySelector(".temperature-value p");

const descElement = document.querySelector(".temperature-description p");

const locationElement = document.querySelector(".location p");

const notificationElement = document.querySelector(".notification");


// App data

const weather = {};


weather.temperature = {

    unit : "celsius"

}


// APP CONSTS AND VARS

const KELVIN = 273;

// API KEY

const key = "82005d27a116c2880c8f0fcb866998a0";


// CHECK IF BROWSER SUPPORTS GEOLOCATION

if('geolocation' in navigator){

navigator.geolocation.getCurrentPosition(setPosition, showError);

}else{

    notificationElement.style.display = "block";

    notificationElement.innerHTML = "<p>Browser doesn't Support Geolocation</p>";

}


// SET USER'S POSITION

function setPosition(position){

    let latitude = position.coords.latitude;

    let longitude = position.coords.longitude;


    getWeather(latitude, longitude);

}


// SHOW ERROR WHEN THERE IS AN ISSUE WITH GEOLOCATION SERVICE

function showError(error){

    notificationElement.style.display = "block";

    notificationElement.innerHTML = `<p> ${error.message} </p>`;

}


// GET WEATHER FROM API PROVIDER

function getWeather(latitude, longitude){

    let api = `http://api.openweathermap.org/data/2.5/weather?lat=${latitude}&lon=${longitude}&appid=${key}`;
```

```
    fetch(api)
        .then(function(response){
            let data =
response.json();
            return data;
        })
        .then(function(data){
            weather.temperature.value
= Math.floor(data.main.temp - KELVIN);
            weather.description =
data.weather[0].description;
            weather.iconId =
data.weather[0].icon;
            weather.city = data.name;
            weather.country =
data.sys.country;
        })
        .then(function(){
            displayWeather();
        });
}

// DISPLAY WEATHER TO UI
function displayWeather(){
    iconElement.innerHTML = `<img
src="icons/${weather.iconId}.png"/>`;
    tempElement.innerHTML =
`${weather.temperature.value}°<span>C<
/span>`;
    descElement.innerHTML =
weather.description;
    locationElement.innerHTML =
`${weather.city}, ${weather.country}`;
}

// C to F conversion
function
celsiusToFahrenheit(temperature){
    return (temperature * 9/5) + 32;
}

// WHEN THE USER CLICKS ON THE
TEMPERATURE ELEMENET
tempElement.addEventListener("click",
function(){
    if(weather.temperature.value ===
undefined) return;

    if(weather.temperature.unit ==
"celsius"){
        let fahrenheit =
celsiusToFahrenheit(weather.temperatur
e.value);
        fahrenheit =
Math.floor(fahrenheit);

        tempElement.innerHTML =
`${fahrenheit}°<span>F</span>`;
        weather.temperature.unit =
"fahrenheit";
    }else{
        tempElement.innerHTML =
`${weather.temperature.value}°<span>C<
/span>`;
        weather.temperature.unit =
"celsius"
    }
});

// Обробник повідомлення /start
// Він видає загальну інформацію про
бота та перелік доступних команд
MyBot.onText(/\/start/, msg => {
      const chat_id = msg.chat.id
      MyBot.sendMessage(
            chat_id,
            `Welcome to <b>Meteo
Search System</b>, thanks for using!

It is possible to use such commands:
/now <b>city</b> - get meteo info in
the city.
/tomorrow <b>city</b> - get tomorrow
meteo info in the city.
```

```
/week <b>city</b> - get weekly meteo info in the city.

/lang <b>lang_code</b> - set new language in database (two letters, for example UA, RU, etc)

/set <b>city</b> - set base city in database.

/w - get meteo info for base city for 3 days.

/location - to set actual location using Telegram function (works only for mobile versions).

/help - check commands.
    `,
            { parse_mode: 'HTML' }
      )
})


// Обробник повідомлення /set
MyBot.onText(/\/set (.+)/, (msg, match) => {
      const chat_id = msg.chat.id
      const user_id = msg.from.id
      const city = match.input.split(' ')[1]
      if (city === undefined) {
            MyBot.sendMessage(chat_id, `Enter your base city name`)
            return
      }
      User.findOneAndUpdate({ user_id }, { city }, (err, res) => {
            if (err) {
                  MyBot.sendMessage(
                        `Sorry, there is some error:\n\r${err}`
                  )
            } else if (res === null) {
                  const new_user = new User({
                        user_id,
                        city,
                  })
                  new_user
                        .save()
                        .then(() =>
      MyBot.sendMessage(
      chat_id,
      `${msg.from.first_name}, information was saved`
                              )
                        )
                        .catch(() => {
      MyBot.sendMessage(
      chat_id,
      `${msg.from.first_name}, sorry, information was not saved`
                              )
                        })
            } else {
                  MyBot.sendMessage(
                        chat_id,
      `${msg.from.first_name}, information was updated`
                  )
            }
            return
      })
})


// Обробник повідомлення /lang
MyBot.onText(/\/lang (.+)/, (msg, match) => {
      const chat_id = msg.chat.id
      const user_id = msg.from.id
      const lang = match.input.split(' ')[1]
      if (lang === undefined) {
```

```
            MyBot.sendMessage(chat_id,
`Please provide lang code (UA, RU, EN,
...)`)
            return
      }


      User.findOneAndUpdate({ user_id
}, { lang }, (error, res) => {
            if (error) {
                  MyBot.sendMessage(
                        `Sorry, there
is some error:\n\r${error}`
                  )
            } else if (res === null) {
                  MyBot.sendMessage(
                        chat_id,

      `${msg.from.first_name}, firstly
provide /set command`
                  )
            } else {
                  MyBot.sendMessage(
                        chat_id,

      `${msg.from.first_name},
language was updated`
                  )
            }
            return
      })
})


const getInfo = (chat_id, user_id,
city, index) => {
      if (city) {
            Axios
                  .get(

      `https://api.opencagedata.com/ge
ocode/v1/json?q=${city}&key=${OpenCage
Key}&pretty=1`
                  )
                  .then(
                        resp => {
                              const {
lat, lng } =
resp.data.results[0].geometry

      getWeather(chat_id, lat, lng,
null, index, city)
                        },
                        error => {

      console.log('error', error)

      MyBot.sendMessage(chat_id,
`Sorry, can not find latitude and
longitude.`+error)
                        }
                  )
      } else {
            User.findOne({ user_id })
                  .then(doc => {
                        if (doc) {
                              Axios

      .get(

      `https://api.opencagedata.com/ge
ocode/v1/json?q=${doc.city}&key=${Open
CageKey}&pretty=1`
                                    )

      .then(

      resp => {

            const { lat, lng } =
resp.data.results[0].geometry

            getWeather(chat_id, lat,
lng, doc.lang, index, doc.city)

      },

      error => {

            console.log('error',
error)
```

```
            MyBot.sendMessage(

                  chat_id,

                  `Sorry, can not find
weather for this query.`

            )

      }
                                    )
                        } else {

      MyBot.sendMessage(

      chat_id,

      `Can not check this info.
            \n\rPlease, firstly
provide /set [city] command.`
                                    )
                              }
                  })
                  .catch(error => {

      MyBot.sendMessage(
                                 chat_id,
                                 `Sorry,
there is some error:\n\r${error}`
                              )
                  })
      }
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Weather App -
JavaScript</title>
    <link rel="stylesheet"
href="font/Rimouski.css">
    <link rel="stylesheet"
href="style.css">
</head>
<body>

    <div class="container">
        <div class="app-title">
            <p>Weather</p>
        </div>
        <div class="notification">
</div>
        <div class="weather-
container">
            <div class="weather-icon">
                <img
src="icons/unknown.png" alt="">
            </div>
            <div class="temperature-
value">
                <p>-
°<span>C</span></p>
            </div>
            <div class="temperature-
description">
                <p> - </p>
            </div>
            <div class="location">
                <p>-</p>
            </div>
        </div>
    </div>

    <script src="app.js"></script>
</body>
</html>
```