КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавра
за спеціальністю 121 Інженерія програмного забезпечення

на тему:

РОЗРОБКА ВЕБ-СЕРВІСУ «ПРОГНОЗ ПОГОДИ» З ВИКОРИСТАННЯМ БІБЛІОТЕКИ ANGULAR

Виконав студент 4-го курсу
Олег МОСЬПАН _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Лариса КАТЕРИНИЧ _____
(підпис)

Засвідчую, що в цій роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри інтелектуальних
програмних систем
«____» _____ 2021 р.,
протокол № ___
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2021

РЕФЕРАТ

Дипломна робота: 51 сторінка, 31 рисунок, 3 таблиці, 12 джерел та 2 додатки.

Ключові слова: ANGULAR, API, OPENWEATHERMAP, ВЕБ-СЕРВІС, ВЕБ-ДОДАТОК, ПОГОДА, ПРОГНОЗ, СЛУЖБА, ФРЕЙМВОРК.

Об’єкт дослідження: процес розробки веб-сервісу «Прогноз погоди» з використанням фреймворку Angular та TailwindCSS. Предметом роботи є веб-сервіс для зручного відображення даних про погоду у будь-якій точці світу.

Мета роботи: реалізація веб-сервісу для відображення інформативних даних про погоду у будь-якій точці світу.

Інструменти розробки: Angular 11.2.14, TailwindCSS, мова програмування TypeScript, відкриті API OpenWeatherMap та HERE.

Рекомендації щодо використання роботи: для дослідження інструментів та технологій розробки веб-сервісів.

Сфера застосування: повсякденне життя, застосування у метеорологічних станціях.

Значимість роботи: наведено порівняння різних методів розробки веб-сервісів, порівняння найпопулярніших фреймворків та використання відкритих API .

Висновки: були проаналізовані методи розробки веб-сервісів, їх ефективність при розробці веб-сервісу, описані основні технології розробки, порівняно існуючі фреймворки для розробки.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

API – Application Programming Interface, програмний інтерфейс застосунку;

CLI – Command Line Interface, інтерфейс командного рядка;

CSS – Cascading Style Sheets, каскадні таблиці стилів;

DOM – Document Object Model, об'єктна модель документа;

HTML – HyperText Markup Language, мова розмітки гіпертексту;

HTTP – HyperText Transfer Protocol, протокол передачі гіпертекстових документів;

IDE – Integrated development environment, інтегроване середовище розробки;

IIS – Internet Information Services

JRE – Java Runtime Environment, середовище виконання для Java;

JS – JavaScript

JSON – JavaScript Object Notation, запис об'єктів JavaScript;

OWM – OpenWeatherMap

SOA – Service-oriented architecture, сервісно-орієнтована архітектура;

SOAP – Simple Object Access Protocol, простий протокол доступу до об’єктів;

SPA – Simple Page Application, односторінковий застосунок;

URL – Uniform Resource Locator, уніфіко́ваний лока́тор ресу́рсів;

WORA – Слоган «Write once, run anywhere»

WSDL – Web Services Description Language, мова опису вебсервісів;

XML – Extensible Markup Language, розширювана мова розмітки;

БЄМ – Блок Елемент Модифікатор

УФ – Ультрафіолет

ВСТУП

В сучасному світі, мережа інтернет стала загальновизнаним фактором ділового та суспільного життя. За роки існування, широка розповсюдженість та зростаюча пропускна здатність створюють умови, при яких вигідно вирішувати більшість задач за допомогою інтернет-технологій. Людська потреба завжди бути в курсі справ, виводять інформаційні технології на стежку вдосконалення вже наявних гаджетів та застосунків так і на створення нових додатків та гаджетів. Можна дійти висновку, що інтернет об'єднує у собі багато різних платформ, а інформація зберігається в різноманітних джерелах даних. Тому, досить актуальною є проблема зв'язання таких різнобічних даних, а також створення засобів, які дозволять отримувати їх у зручному вигляді для подальшої обробки.

Концепція веб-сервісів покликана розв'язати цю задачу об'єднання, інтеграції різнобічних систем на основі відкритих стандартів. Нині, величезна кількість людей користуються послугами веб-сервісів. На мою думку, веб-сервіси прогнозу погоди, мають попит серед населення. Вони є найпопулярнішими у світі, адже майже кожна людина, маючи доступ до мережі інтернет, переглядає прогноз погоди на сьогодні, на завтра, на тиждень чи на цілий місяць.

Сьогодні ви не можете згадати термін "веб-сервіс", відразу ж не отримавши посилання на веб-служби Amazon або веб-службу Google, Google Cloud Platform. Однак на це є причина. Ці технічні гіганти підняли планку, звернувши увагу на необхідність розробки додатків. А масштаб на зразок Amazon та Google - це саме те, що робить існування сучасних веб-сервісів можливим. Однак, попри всі технологічні досягнення, тестування веб-сервісів і процес розробки все ще є складною задачею. Оскільки вони залежать від операційних систем для управління програмами, найменші відхилення можуть призвести до багатогранних робочих збоїв при спробі переміщення даних між

сервером та хмарою. Але за допомогою API веб-розробники можуть інтегрувати розширені функції в додатки, що дозволяє значно покращити процес налаштування та гнучкість. Кінцевим результатом таких маніпуляцій є покращення якості використання для користувачів та розробників.

Метою цієї робити є створення зручного веб-сервісу для поглибленого перегляду даних про погоду у будь-якому місті нашої планети.

Об’єктом даної роботи є методи розробки сучасних веб-сервісів та технології розробки функціональних веб-застосунків.

Предмет цієї роботи – застосування технологій розробки веб-сервісів та веб-застосунків.

Для досягнення визначеної мети необхідно виконати наступні **завдання**:

- Розглянути базові підходи до розробки веб-сервісів;
- Розглянути сучасні технології які використовуються для створення веб-сервісів та веб-застосунків;
- Розробити веб-сервіс «Прогноз погоди» з набутими знаннями;
- Провести опитування серед населення;
- Мануальне тестування;
- Висновки щодо виконаної роботи;

РОЗДІЛ 1 БАЗОВІ ПІДХОДИ ДО РОЗРОБКИ ВЕБ-СЕРВІСІВ

1.1 Основні поняття розробки веб-сервісів

Новітні застосунки використовують різноманітні платформи для розробки веб-застосунків. Деякі можуть бути розроблені на JavaScript, Angular JS, інші – на .Net, Node.js і т.д. Частіше за все ці додатки потребують певну взаємодію один з одним. Оскільки вони розроблені з використанням різних мов програмування, моделей та технологій, задача забезпечення точного зв'язку між ними стає дійсно складною. Тут на допомогу приходять веб-сервіси. Вони надають спільну платформу, яка дозволяє декільком додатків, написаних на різних мовах програмування мати цю можливість взаємодіяти один з одним.

Існує досить багато визначень поняття «web-сервіс». Але найбільш точно та логічно він визначається як «програмна система, розроблена для підтримки взаємодії машина-машина через мережу». Він має інтерфейс, описаний у машинно-обробному форматі (зокрема, WSDL). Інші системи взаємодіють з веб-сервісом способами, передбаченими їх описом, із використанням повідомлень SOAP, як правило, переданих за допомогою HTTP із серіалізацією XML у поєднанні з іншими веб-стандартами.

Досить часто люди плутаються в поняттях: веб-сервіс та веб-сайт. Головною відмінністю є те, що веб-сервіс надає певні послуги. Наприклад, такий веб-сервіс як прогноз погоди пропонує користувачу повний спектр даних щодо температури, опадів, хмарності тощо, у місті яким він цікавиться. Web-сайт у свою чергу – це сторінка на якій знаходиться інформація, з якою користувач має змогу ознайомитися.

Інтерфейс веб-сервісу грає дуже важливу роль у залученні уваги користувачів мережі інтернет. Усі елементи навігації повинні бути простими

та зрозумілими – це дуже важливо для простоти використання користувачем та просування веб-сервісу у пошукових системах.

Основним компонентом веб-служби є дані, які передаються між клієнтом і сервером, а саме XML. XML (розширювана мова розмітки) є аналогом HTML вона проста в розумінні проміжної мови, яку у свою чергу розуміють багато мов програмування. Отже, коли програми розмовляють між собою, вони насправді розмовляють у форматі XML. Це забезпечує загальну платформу для додатків, розроблених на різних мовах програмування, аби вони мали змогу спілкуватися між собою.

Веб-сервіси використовують SOAP (Simple Object Access Protocol) для передачі даних XML між програмами. Дані надсилаються через звичайний HTTP. Дані, що надсилаються з веб-служби до програми, називаються SOAP-повідомленнями. Повідомлення SOAP - це ні що інше, як XML-документ. Оскільки документ написаний у форматі XML, клієнтська програма, що викликає веб-службу, може бути написана будь-якою мовою програмування.

1.2 Основні типи веб-сервісів

В загальному випадку, виділяють два основних типи веб-сервісів [1]:

- Веб-сервіс SOAP
- Веб-сервіс RESTful

Для того аби служба була повністю функціональна, необхідна наявність певних компонентів. Вони повинні бути наявними в незалежності від мови програмування використаної при розробці веб-сервісу. Розглянемо ці компоненти більш детально.

1.2.1 Веб-сервіс SOAP

SOAP відомий як транспортно-незалежний протокол обміну повідомленнями. Основною функцією є передача даних XML у вигляді повідомлень SOAP. В кожному такому повідомленні є XML-документ, при чому передається тільки структура XML-документа за певним шаблоном, але не його зміст. Перевагою цього компоненту є те, що для відправлення він використовує стандартний веб-протокол HTTP.

Ось з чого складається повідомлення SOAP (рисунок 1.1):

- Кожен документ SOAP повинен містити у собі кореневий елемент, відомий як «Конверт». Кореневий елемент – це перший елемент в XML-документі.
- «Конверт» у свою чергу ділиться на 2 основні частини. Перша – це заголовок, а друга – тіло.
- Заголовок містить данні маршрутизації, які є інформацією для XML-документа, якому клієнту він повинен бути відправленим.
- Тіло зберігає у собі фактичне повідомлення.

Рисунок 1.1 – Структура SOAP повідомлення

1.2.2 Веб-сервіс RESTful

RESTful – це легка служба, з можливістю масштабування та підтримки, побудована на архітектурі REST. Вона надає API клієнту, що її викликав, у безпечній, рівномірній манері без збереження стану. Викликаючий клієнт, може виконувати завчасно визначені операції за допомогою служби RESTful.

RESTful методи (рисунок 1.2):

- **GET**: цей метод використовується для отримання інформації, яка відправляється на сервер, з залученням таких методів як PUT або POST. У цього метода немає тіла запиту.
- **POST**: цей метод використовується для створення документа або запису з використанням тіла запиту, вказаної URL-адреси, ключа документа, ключа контексту й т.д. Те ж саме можна отримати за допомогою метода GET.
- **PUT**: цей метод використовується для оновлення будь-якого документа або запису, які вже існують.
- **DELETE**: цей метод використовується для видалення будь-якого вже наявного документа або запису.

Рисунок 1.2 – Методи сервісу RESTful

Порівняння SOAP та RESTful

Хоча ці два типи веб-сервісів використовуються для виконання запиту та отримання відповіді, вони повністю відрізняються за способом роботи.

Нижче наведений список основних відмінностей:

- Конверт SOAP можна використовувати в REST, але не навпаки.
- SOAP не має вбудованого методу, а REST має GET, PUT, POST тощо.
- SOAP має збереження стану, тоді як REST - ні.
- Тіла запитів та відповідей у SOAP підтримують лише формат даних XML. У REST тіла запиту та відповіді підтримують багато форматів даних, таких як JSON, XML, звичайний текст тощо.
- SOAP повільніший за REST, оскільки обробка запитів у SOAP займає більше часу через формат даних XML. REST використовує JSON, який є дуже легким і, отже, робить це швидшим.
- SOAP, як правило, безпечніший, за REST-послуги, оскільки SOAP-послуги надають не тільки SSL, а і WSS. Цей SSL присутній як у SOAP, так і в REST.

1.3 Архітектура веб-сервісів

Архітектура веб-сервісів складається з трьох основних компонентів: сервіс запиту, провайдера та реєстрації. Кожен з наведених компонентів має певну роль в окремих фазах циклу розробки [2]. Ролі постачальника та запитувача – це логічні конструкції, тому, що сервіс може виступати як в ролі служби запиту, так і служби провайдера.

Приклад повного сценарію, неперервного життєвого циклу може починатися зі створення та публікації інтерфейсу сервісу (збірка), далі приступаємо до створення та розгортання мережі (розгортання), на останньому кроці переходимо до публікації реалізованої служби та завершаємо процес, викликом веб-сервісу службою запиту (запуск). Життєвий цикл розробки охоплює наступні фази: збірка, розгортання, запуск, та керування.

Розглянемо усі етапи циклу, більш детально:

Збірка. Цей етап охоплює розробку та тестування веб-сервісу, визначення опису інтерфейсу та реалізації служби. Пошук наявного визначення інтерфейсу також є збірко-часовою функцією. Реалізація веб-сервісів може бути представлена шляхом створення нових веб-сервісів, перетворення наявних додатків у веб-сервіси, а також складання нових веб-сервісів із вже заведених. Реалізація веб-сервісу передбачає використання мов програмування та моделей які підходять під середовище постачальника послуг. Перетворення чинних додатків у веб-сервіси охоплює створення сервісних інтерфейсів та оболонок для представлення відповідної бізнес-функції додатку. Розробка нових веб-сервісів з вже наявних, передбачає наявність послідовності та засобів керування потоками повідомлень між програмами напряму або з використанням технологій документообороту.

Розгортання. Завдання, пов'язані з фазою розгортання життєвого циклу розробки, включають публікацію інтерфейсу служби та визначення реалізації служби, розгортання коду виконання для веб-сервісу та інтеграцію із застарілими системами. Для веб-сервісів, що представляють трансформовані програми, розгортання може включати лише веб-обгортку сервісу, оскільки програма може бути вже розгорнута. Для потоків послуг, розгортання містить в собі налаштування менеджера робочого процесу та менеджера бізнес-процесів, а також використання та контроль нових потоків.

Запуск. На етапі життєвого циклу запуску веб-сервіс повністю розгорнутий і працює. Під час цього, запитувач послуги може знайти

визначення послуги та викликати усі визначені сервісні операції. Функції виконання включають статичну та динамічну прив'язку, сервісне обслуговування взаємодії таке як, наприклад – функція обміну повідомленнями та взаємодії за допомогою протоколу SOAP із застарілими системами.

Управління. Етап управління життєвим циклом веб-служби охоплює постійне управління та адміністрування веб-сервісу. Безпека, доступність, продуктивність, якість обслуговування та всі бізнес-процеси повинні бути вирішені.

Таким чином, основними принципами при розробці веб-сервісів є: простота реалізації, незалежність від мови програмування та платформи, пошук веб-сервісів та інтеграції застосунків з їх допомогою. Найбільш популярні та потрібні мови програмування та платформи розглянемо в параграфі 1.4.

1.4 API як засіб інтеграції застосунків

Веб-служба, на відміну від API, працює більше як ресурс, доступний за допомогою Інтернету. Мережевий ресурс можна застосувати до конкретних завдань, але для їх функціонування потрібна мережа. Це означає, що всі веб-служби є API, але лише деякі API є веб-службами.

Веб-служба працює, підтримуючи взаємозв’язок «машина-машина» за допомогою мережі. Як такі, веб-служби, як правило, пов’язані з SOA або сервісно-орієнтованою архітектурою. Це дозволяє розділити різні функції, а потім зробити їх доступними як різні послуги в мережі (рисунок 1.3).

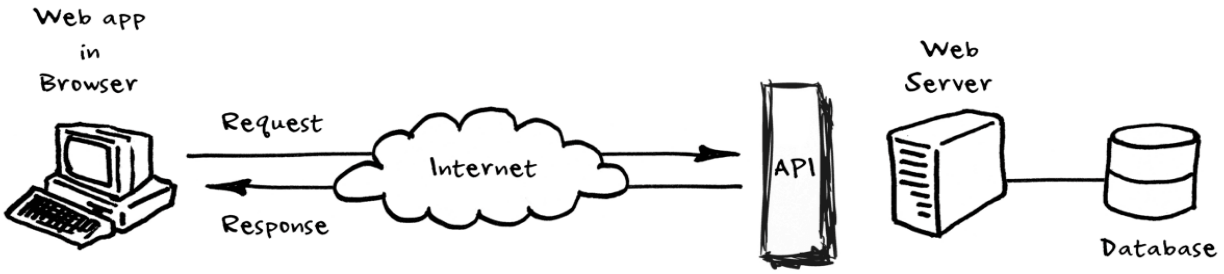

Рисунок 1.3 – Як працює API

API розшифровується як Інтерфейс програмування застосунків [8]. Цей інтерфейс дозволяє людям використовувати функціонал та дані іншої програми. Можна зрозуміти їх як будівельні блоки, з яких можна зробити майже все, оскільки їх можна знайти у всьому - від Spotify до Yahoo Finance.

Структури API дозволяють розробникам виконувати завдання, які мало чим відрізняються від повсякденних подій. Наприклад, подумайте про те, щоб замовити сервер, той сервер вкласти ваше замовлення, а потім повернути замовлення, коли воно буде готове. Цей покроковий процес повертає бажаний результат: смачну їжу (в цьому випадку). Веб-прикладом може бути хтось, хто підписався на новий сайт електронної комерції, використовуючи свій обліковий запис Facebook.

По суті, API допомагають сайтам спілкуватися в Інтернеті та розуміти інформацію (незалежно від мов програмування), щоб полегшити процеси. Запити протоколу HTTP дозволяють надсилати дані та отримувати дані [11]. Єдине застереження полягає в тому, що кожен API вимагає постійного тестування для забезпечення стабільної роботи. Якщо ви не напишете кожен рядок коду з нуля, ви будете взаємодіяти із зовнішніми програмними компонентами, кожен зі своїм API. Навіть якщо ви пишете щось з нуля, добре розроблений програмний додаток матиме внутрішні API, які допоможуть вам упорядкувати ваш код і забезпечити повторне використання компонентів.

Існує багато загальнодоступних API, які дозволяють скористатися перевагами функцій, розроблених в інших місцях Інтернету.

API визначається як специфікація можливої взаємодії з програмним компонентом. Що це, власне, означає? Ну, уявіть, що автомобіль був програмним компонентом. Його API міститиме інформацію про те, що він може робити - прискорюватись, гальмувати, включати радіо тощо. Він також міститиме інформацію про те, як ви можете змусити його це робити. Наприклад, для прискорення ви кладете ногу на педаль газу і натискаєте.

API не повинен пояснювати, що відбувається всередині двигуна, коли ти ставиш ногу на акселератор. Ось чому, якщо ви навчились керувати автомобілем з двигуном внутрішнього згоряння, ви можете сісти за кермо електромобіля, не вивчаючи цілком новий набір навичок. Інформація про те, що і як поєднується у визначенні API, яке є абстрактним та окремим від самого автомобіля.

Треба мати на увазі одне, що назва деяких API часто використовується як для специфікації взаємодій, так і для фактичного програмного компонента, з яким ви взаємодієте. Наприклад, фраза "Twitter API" не тільки стосується набору правил програмної взаємодії з Twitter, але загалом розуміється як річ, з якою ви взаємодієте, як у "Ми робимо аналіз твітів, які ми отримали від API Twitter ".

На сьогодні люди використовують чотири різні API.

- Складені API. Це злиття API служб і даних. Серія завдань працює синхронно завдяки виконанню, а НЕ завдяки запитам завдань. Ці API можуть прискорити процес виконання, а також покращити продуктивність прослуховування веб-інтерфейсу.
- API партнерів. Для цього потрібні ліцензії або спеціальні права на доступ, оскільки вони загалом недоступні для публічних розробників.

- Відкриті API. На відміну від попередніх, відкриті або “загальнодоступні” API не мають обмежень доступу і можуть бути доступними для всіх.
- Внутрішні API. Як випливає з назви, вони працюють як “приватні” API у внутрішніх системах. Вони можуть бути використані серед внутрішніх команд в одній компанії для вдосконалення послуг або продуктів.

Деякі API також вимагають ключів для автентифікації, перш ніж дозволити поєднання інформації. Саме такими я і скористався у цій дипломній роботі.

Тепер, коли ми знаємо, що це за елемент, нам тепер потрібно зрозуміти різницю між веб-API та веб-службами. Однією з найбільш очевидних відмінностей є те, що веб-сервісам, на відміну від API, потрібна мережа для функціонування. API можуть функціонувати в Інтернеті або в автономному режимі.

Основні різниці між веб-службами та веб-API:

Веб-служба:

- Послуга на основі SOAP і повертає дані у форматі XML.
- Підтримує лише протокол HTTP.
- Не має відкритого коду, але може бути використана будь-яким клієнтом, який розуміє XML.
- Для отримання та надсилання даних через мережу потрібен протокол SOAP, тому вона не є полегшеною архітектурою.

Веб-API:

- Підтримує протокол HTTP.
- Може розміщуватися в межах програми або IIS.
- Має відкритий код, і ним може користуватися будь-який клієнт, який розуміє JSON або XML.

– Має просту архітектуру і добре підходить для пристроїв з обмеженою пропускною здатністю, таких як мобільні пристрої.

Крім того, веб-служби не є агностичними щодо протоколів, як API. API можуть використовувати будь-який стиль дизайну або протокол, а веб-служби обмежені переважно SOAP або Simple Object Access Protocol.

Загальнодоступні API часто також мають відкритий код і більш прозоро ставляться до своєї документації. Веб-служби жертвують цією прозорістю для більш конкретних даних, партнерів та безпеки. Однак безпека API залишається проблемою.

РОЗДІЛ 2 АНАЛІЗ ТЕХНОЛОГІЙ ТА ОСОБЛИВОСТІ ЇХ ВИКОРИСТАННЯ

2.1 Аналіз онлайн-сервісу надання API OpenWeatherMap

Для отримання даних про погоду та геолокацію треба скористатися онлайн-сервісами які надають ці данні, в нашому випадку це OpenWeatherMap та HERE. Розглянемо обидва сервіси більш детально.

OpenWeatherMap – онлайн-сервіс, який надає платні (є безкоштовна ліценція з обмеженим функціоналом) API для доступу до даних про погоду, прогноз погоди та історичних даних [3]. В якості джерел, сервіс, використовує офіційні дані з метеостанцій аеропортів, метеорологічних служб різних держав та дані з приватних метеостанцій.

Після отримання даних сервісом, вони оброблюються та на їх основі будується прогноз погоди та погодні мапи. Основною ідеєю сервісу OWM є використання приватних метеостанцій, адже це дозволяє збільшити точність вхідних даних, а як наслідок, точність самих прогнозів погоди.

OWM використовує платний API, аби надавати дані про погоду, та карти з погодними явищами, такими як хмари, вітер, тиск та опади. Усі дані можуть бути отриманими в форматах JSON, XML або HTML.

Розглянемо типи даних детальніше:

а) **Поточні погодні дані** Можуть бути знайдені за назвою міста (у сервісі доступно більш як 200 000 міст) або по географічних координатах. Дані оновлюються кожні 10 хвилин.

б) **Прогнози** Також можуть бути знайдені за назвою міста або по географічних координатах. OWM надає наступні види прогнозних даних:

1) Хвилинний прогноз на 1 годину

2) Годинний прогноз на 4 дні

3) Денний прогноз на 16 днів

4) Кліматичний прогноз на 30 днів

в) **Пошук** Система геокодування дозволяє знайти міста за їх назвою, країною, поштовому індексу або географічними координатами. Пошук також можливий за частиною назви міста. Для того, аби результати пошуку були більш точними, назва міста на країни повинні бути розділені комою.

г) **Погодні мапи** OWM надає велику кількість погодних мап таких як: мапи опадів, хмарності, атмосферного тиску, температури, вітру та багато інших.

Підключення метеостанцій до сервісу відбувається в 3 кроки:

- Реєстрація в проєкті OpenWeatherMap
- Посилання даних згідно з API
- Після підключення метеостанції та посилання даних, необхідно перевірити їх правильність на персональній сторінці сервісу

При посиланні метеоданих до сервісу методом POST, до HTTP заголовку, для авторизації, необхідно додати заголовок x-api-key зі значенням API ключа який можна знайти в особистому кабінеті (Додаток А).

Нижче наведена таблиця з назвою певних параметрів та їх розшифруванням (таблиця 2.1):

Параметр	Одиниця виміру	Опис
wind_dir	Градуси	Напрям вітру
wind_speed	Метри в секунду	Швидкість вітру
wind_gust	Метри в секунду	Швидкість поривів вітру
temp	Градуси Цельсію	Температура повітря
humidity	Проценти	Відносна вологість
pressure		Атмосферний тиск

rain_1h	Міліметри	Опади за останню годину
rain_24h	Міліметри	Опади за останні 24 години
rain_today	Міліметри	Опади з опівночі
snow	Міліметри	Сніг за останні 24 години
lum	Ватт на квадратний метр	Яркість
lat	Десятичні градуси	Широта
long	Десятичні градуси	Довгота
alt	Метри	Висота
radiation		Радіація
dewpoint	Градуси цельсію	Точка роси
uv		UV - індекс
name	Рядок	Назва метеостанції

Таблиця 2.1 – Таблиця параметрів наданих метеостанціями

З переліченого вище можемо зробити висновок, що цей сервіс повністю задовільняє усі потреби для розробки веб-сервісу «Прогноз погоди».

2.2 Особливості використання фреймворків Angular та TailwindCSS

2.2.1 Фреймворк Angular

Фреймворк JavaScript (або сторона клієнта), це технологія, яка надає нам правильні інструменти для створення веб-додатків, одночасно визначаючи, як його слід розробляти та як слід організовувати код.

Більшість фреймворків JS сьогодні є значущими, а це означає, що вони мають власну філософію щодо створення веб-програми, і вам, можливо, доведеться витратити трохи часу на вивчення основних концепцій [10]. Інші рішення, такі як Backbone, не вказують розробникам, як створювати проєкт, а деякі навіть називають такі технології лише бібліотеками, а не фреймворками.

Насправді фреймворки JavaScript з’явилися не так давно [9]. Раніше, веб-сайти будувались із погано структурованим кодом JS (у багатьох випадках на основі jQuery). Однак користувацькі інтерфейси на стороні клієнта стають дедалі складнішими, і JavaScript втрачає репутацію "мови іграшок". Сучасні веб-сайти значною мірою покладаються на JS, виникла необхідність правильно організувати та протестувати код. Тому фреймворки на стороні клієнта стали популярними, і сьогодні їх існує щонайменше десяток.

Раніше AngularJS був "золотою дочірньою структурою" фреймворків JavaScript, який вперше був представлений корпорацією Google у 2012 році [6]. Він був побудований з урахуванням концепції Model-View-Controller, хоча автори фреймворку часто називали його "Model-View-*" або навіть "Model-View-Whatever".

Фреймворк, написаний на чистому JavaScript, був призначений для відключення логіки програми від маніпуляцій DOM і спрямований на динамічне оновлення сторінок. Проте, це було не дуже нав'язливо: ви могли мати лише частину сторінки, керовану AngularJS. Цей фреймворк представив безліч потужних функцій, які дозволяють розробнику досить легко створювати багаті односторонні програми.

Зокрема, було введено цікаву концепцію прив'язки даних, яка означала автоматичне оновлення дисплея при зміні моделі (даних), і навпаки. Крім того, була представлена ідея директив, які дозволяли винаходити власні теги HTML, реалізовані за допомогою JavaScript. Наприклад, ви можете написати:

<calendar> </calendar>

Це спеціальний тег, який буде оброблений AngularJS і перетворений у повноцінний календар відповідно до інструкцій базового коду. Звичайно, вашою роботою буде кодування правильної директиви.

Ще однією досить важливою річчю була програма Dependency Injection, яка дала змогу з'єднувати компоненти програми разом таким чином, щоб уможливлювався багаторазовий та тестований код.

AngularJS дуже швидко став популярним і зміцнів. Проте, супровідники вирішили зробити крок далі і продовжили розробку нової версії, яка спочатку називалася Angular 2 (пізніше просто Angular без частини "JS"). Не випадково фреймворк отримав нову назву: насправді він був переписаний та перероблений, тоді як багато концепцій були переоцінені.

Перший стабільний випуск Angular 2 був випущений у 2016 році, і відтоді AngularJS почав втрачати свою популярність на користь нової версії [5]. Однією з головних особливостей Angular 2 була здатність розвиватися для декількох платформ: веб, мобільних та власних робочих столів (тоді як AngularJS не має мобільної підтримки нестандартно).

Щоб зробити речі ще складнішими, Angular 4 вийшов наприкінці 2016 р. Як пояснюється в офіційному дописі в блозі, супровідники вирішили дотримуватися семантичної версії, починаючи з Angular 2.

Дотримуючись цього принципу, зміна основної версії (наприклад, "2.x.x" на "3.x.x") означає, що були внесені деякі зміни руйнування. Проблема полягає в тому, що компонент Angular Router вже був на версії 3. Щоб виправити цю виправлену помилку, було вирішено повністю пропустити Angular 3. На щастя, перехід від Angular 2 до 4 був менш болісним, ніж від AngularJS до Angular 2, хоча багато розробників все ще були досить розгублені з приводу цього безладу.

Angular 5 був випущений в листопаді 2017 року. Він також сумісний із попередніми версіями Angular.

Angular підтримується на різних платформах (веб, мобільних, настільних), він потужний, сучасний та має приємну екосистему.

Нижче наведені переваги використання цього фреймворку [4]:

- Angular надає не тільки інструменти, а й шаблони дизайну для побудови проєкту в ремонтопридатному режимі. Коли Angular додаток створено правильно, ви не отримаєте безліч класів і методів, які важко змінити й ще важче перевірити. Код

структурований зручно, і вам не доведеться витрачати багато часу на розуміння того, що відбувається.

- Це JavaScript, але краще. Angular будується за допомогою TypeScript, що, своєю чергою, залежить від JS ES6. Вам не потрібно вивчати зовсім нову мову, але ви все одно отримуєте такі функції, як статичне написання, інтерфейси, класи, простори імен, декоратори тощо.
- Не потрібно винаходити велосипед наново. З Angular ви вже маєте безліч інструментів, щоб негайно розпочати створення програми. У вас є вказівки щодо надання елементам HTML динамічної поведінки. Ви можете увімкнути форми за допомогою FormControl і ввести різні правила перевірки. Ви можете легко надсилати асинхронні HTTP-запити різних типів. Ви можете налаштувати маршрутизацію з найменшою кількістю клопоту.
- Компоненти від'єднані. Angular намагався усунути тісний зв’язок між різними компонентами програми. Ін'єкція виконується в стилі NodeJS, і ви можете легко замінити різні компоненти.
- Усі маніпуляції з DOM відбуваються там, де мають відбуватися. За допомогою Angular ви не пов’язуєте тісну презентацію та логіку програми, що робить ваш вибір набагато чистішим та простішим.
- Тестування – це серце. Angular призначений для ретельного тестування, і він підтримує як пристрій, так і наскрізне тестування за допомогою таких інструментів, як Jasmine та Protractor.
- Angular є мобільним та готовим до настільних комп'ютерів, а це означає, що у є одна структура для декількох платформ.
- Angular активно підтримується і має велику спільноту та екосистему. Ви можете знайти багато матеріалів про цей фреймворк, а також багато корисних сторонніх інструментів.

Треба сказати, що, на жаль, Angular - це досить великий і складний фреймворк із власною філософією, який новачкам може бути складно зрозуміти та звикнути до нього. Однак вивчення концепцій фреймворку - не єдине завдання; крім цього, вам також має бути комфортно з декількома додатковими технологіями:

- Рекомендується кодувати програми Angular у TypeScript. Можна написати код за допомогою сучасного JavaScript (ES6).
- TypeScript - це надмірна кількість JavaScript.
- Гарно. практикою є використання Angular CLI для подальшого прискорення процесу розробки.
- Менеджер пакунків npm Node широко використовується для встановлення самого Angular та інших компонентів
- Навчання налаштування програми запуску завдань, таких як Gulp або Grunt, може бути дуже корисним, оскільки може бути багато речей, які потрібно зробити, перш ніж програма дійсно розподілиться для виробництва.
- Використання мініфайєрів (UglifyJS) та пакетів (Webpack) також дуже поширене в наші дні.
- Під час розробки програми важливо мати можливість налагоджувати код, тому ви повинні знати, як працювати з такими інструментами налагодження, як Augury.

Звичайно, дуже важливо протестувати Angular додатки, які можуть бути дуже складними. Одним з найпопулярніших інструментів тестування є Jasmine (який є тестовою основою) та Protractor (який використовується для наскрізного тестування).

Отож, можемо дійти висновку, що є досить багато речей, яким слід навчитися перед тим, як розпочати створювати веб-додатки на стороні клієнта. В мережі Інтернет існує безліч ресурсів, які можуть допомогти вам вивчити всі ці інструменти та технології. Звичайно, вам потрібен трохи часу, щоб

отримати їх, але в результаті ви отримаєте цінний досвід і зможете впевнено створювати складні програми.

Останнє, що варто згадати, – це те, що іноді може бути занадто багато використовувати Angular для програми. Якщо у вас невеликий або середній проєкт без складних користувальницьких інтерфейсів та взаємодій, можливо, набагато кращою ідеєю буде дотримуватися звичайного старого JavaScript. Тому дуже важливо врахувати всі вимоги, особливості нового додатка, а також врахувати терміни, перш ніж приймати рішення, використовувати фреймворк JavaScript чи ні.

Тож можна сказати, що Angular – це не просто фреймворк, а скоріше платформа, яка надає розробникам можливість створювати додатки для Інтернету, мобільних пристроїв та настільних ПК.

2.2.2 Фреймворк TailwindCSS

Відповідно до офіційної документації, Tailwind CSS – це CSS-фреймворк утиліт для швидкого створення користувацьких інтерфейсів [7]. Мені подобається думати, що це крутий спосіб визначення вбудованих стилів і створення приголомшливого інтерфейсу без написання жодного рядка власного CSS.

На мій погляд, одна річ, яку більшість розробників можуть знайти непривабливою в Tailwind CSS, це те, що розмітка виглядає набагато більш завантаженою, ніж ви могли б бажати. Tailwind - не перша CSS-бібліотека утиліт, але вона найбільш популярна на цю мить.

TailwindCSS – це CSS-бібліотека, яка спрощує стилізацію HTML, тим же шляхом, як це робить Bootstrap, додаючи величезну кількість різноманітних класів. Але, на відміну від Bootstrap, який додає вже готові до вживання компоненти, такі як кнопки, ALERT і навбари, класи TailwindCSS націлені на конкретну властивість. У TailwindCSS немає заздалегідь написаної кнопки, її ти повинен зробити сам.

Відмінністю TailwindCSS від всіх інших CSS-фреймворків, на чолі яких Bootstrap, є саме класи як властивості, а не класи як компоненти. І якщо зовні TailwindCSS на них схожий, за своєю суттю, він щось зовсім інше – новий підхід до написання CSS. Він має добре розроблений CSS і не створює проблем з готовністю та кольором класу.

Він добре працює з усіма препроцесорами. Але якщо вам потрібно опублікувати базовий стиль, ви можете запустити цикл і TailwindCSS ніяк цьому не завадить

TailwindCSS прискорює роботу на БЄМ. Не потрібно додавати модифікатор для будь-якої частини, достатньо визначити основний стиль і прописати тип, який не є новим. Якщо модифікований блок використовується не в одному місці, а в багатьох, є можливість створити модифікатор. З TailwindCSS різні шляхи не можуть бути розділені окремим класом.

Можемо дійти висновку, що TailwindCSS легко впроваджується, легко поєднується з БЄМ та css-in-js підходами, не обмежує в жорстких рамках по стилю (як Bootstrap та аналоги), але задає загальні правила дизайну системи.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА РОЗРОБКИ ВЕБ-СЕРВІСУ

3.1 Модельні уявлення об’єкту розробки

Метою роботи є розробка веб-сервісу для поглибленого перегляду даних про погоду, він буде створений на основі алгоритму (рисунок 3.1), в ньому описана послідовність дій яку повинен зробити користувач аби успішно скористатись веб-сервісом.

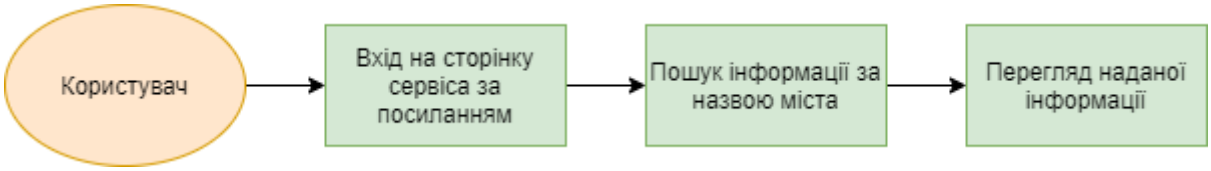

Рисунок 3.1 – Алгоритм роботи веб-сервісу

Користувач має доступ до сервісу за посиланням, сервіс викладений на хостинговому сервісі Netlify.

Після виконання алгоритму (рисунок 3.1), користувач отримає інформацію щодо погоди у шуканому місті.

Веб-сервіс складається з однієї сторінки, тобто він є SPA, на цій сторінці розташовані блоки в яких буде відображатися певна інформація. Даний веб-сервіс має наступну структуру:

Головна сторінка веб-сервісу – на головній сторінці розташовані основні блоки, серед яких, блок тижневого прогнозу погоди, індексу ультрафіолетового випромінювання, швидкості вітру, відчувається як, вологості, напрямку вітру, тиску та блок часу сходу та заходу сонця. Також окремий блок для відображення короткої інформації про погоду на зараз який складається з іконки поточної погоди взятою за API с сервісу OpenWeatherMap, стислого опису погоди, температури, дати та повної назви шуканого міста в залежності від країни, також можуть бути показані, назви

штатів (Сполучені Штати Америки) та почтові індекси. Поле пошуку знаходиться під логотипом сервісу (рисунок 3.2);

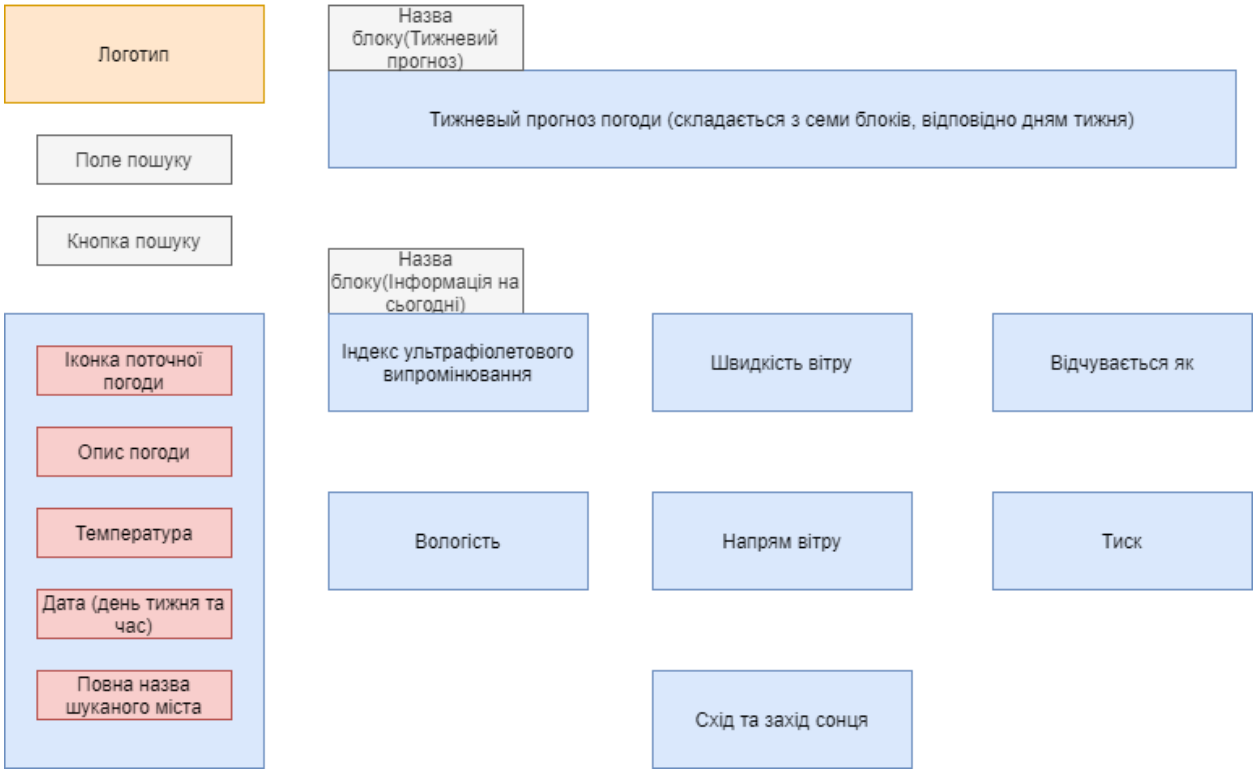

Рисунок 3.2 – Макет головної сторінки веб-сервісу

Розглянемо кожен блок детальніше:

Тижневий прогноз погоди – тут розташована коротка інформація про погоду на сім днів яка містить у собі іконку погоди, назву дня тижня, число, максимальну та мінімальну температуру. Цей блок розділений на сім частин відповідно до днів тижня (рисунок 3.3).

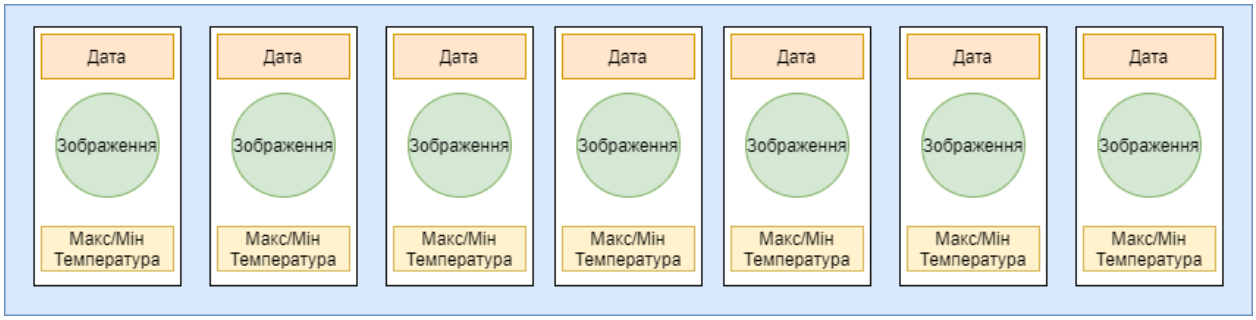

Рисунок 3.3 – Блок тижневого прогнозу погоди

Коротка інформація про погоду на сьогодні – цей блок складається з п’яти частин, таких як: іконка поточної погоди, короткого опису погоди, температури в цей момент часу, дати (день тижня та число) та повної назви шуканого міста (рисунок 3.4).

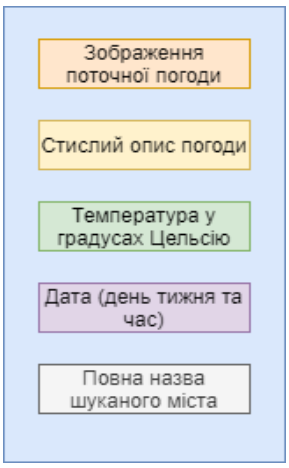

Рисунок 3.4 – Блок короткої інформації про погоду

Найбільшу частину сторінки займає блоком повної інформації на сьогодні, він містить таку інформацію: індекс ультрафіолетового випромінювання, швидкість вітру, відчувається як, вологість, напрямок вітру, тиск та час сходу та заходу сонця. Розглянемо детальніше макети кожного блоку інформації:

Індекс ультрафіолетового випромінювання – це показник, що характеризує рівень ультрафіолетового випромінювання. Він приймає значення від 0 до 11 (рисунок 3.5). Чим вище значення, ти вище потенціальна небезпека для шкіри та очей людини та здоров’я в цілому, тому він обов’язково повинен бути наявним у всіх сервісах прогнозу погоди, мій не є виключенням.

Рисунок 3.5 – Значення УФ-індексу

Сам блок складається з назви, значення UV-index та зображення. При наведенні відображається коротка інформація про рівень небезпеки (рисунок 3.6);

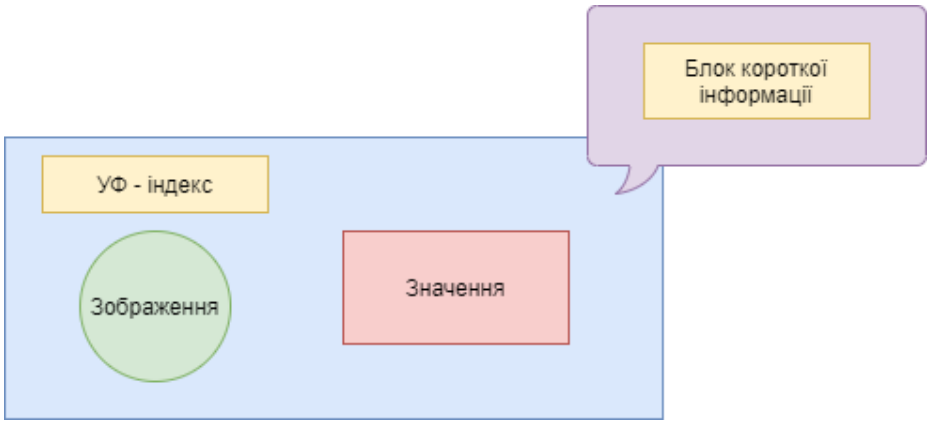

Рисунок 3.6 – Блок УФ-індексу

Швидкість вітру – важливий показник у прогнозах погоди. Одиниця виміру – м/с. При наведені на блок відображається коротка інформація про значення швидкості вітру (таблиця 3.1).

Сила вітру в балах Бофорта	Швидкість вітру, м/с	Назва вітру	Дія вітру
0	1	Штиль	Дим підіймається в гору

3	4	Слабкий	Хитаються невеликі гілки дерев
6	11	Сильний	Хитаються середні стовбури дерев
9	20	Шторм	Вітер ламає великі дерева
12	29	Ураган	Вітер завдає великих спустошень

Таблиця 3.1 – Швидкість вітру

Блок складається з назви, значення швидкості вітру в м/с та зображення. При наведені відображається коротка інформація про рівень небезпеки (рисунок 3.7);

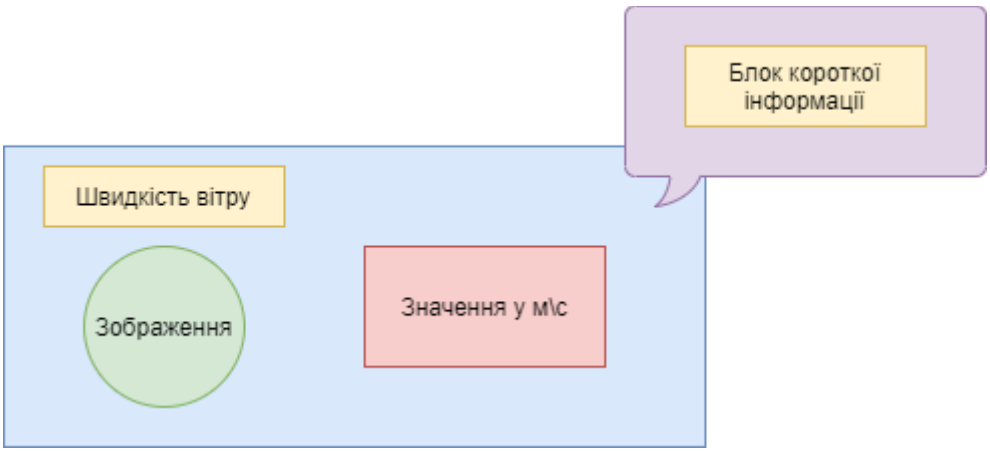

Рисунок 3.7 – Блок швидкості вітру

Відчувається як – складається з назви блоку, зображення та температури у градусах Цельсію. Це значення показує як буде відчуватися температура повітря людиною одягнутою за сезоном.

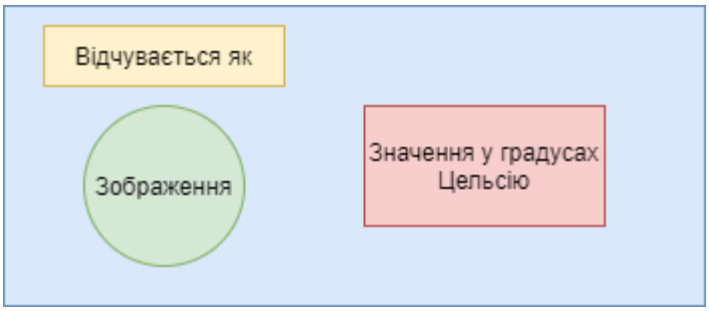

Рисунок 3.8 – Блок фактичної температури

Вологість – блок складається з назви, зображення та значення вологості у відсотках (рисунок 3.9);

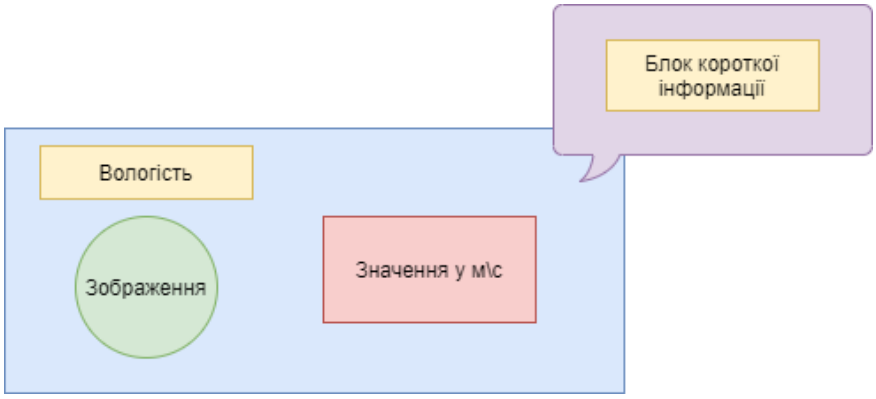

Рисунок 3.9 – Блок вологості

Значення вологості повітря, наведені в таблиці нижче (таблиця 3.2):

Значення вологості повітря у відсотках	Опис рівня вологості
Більш як ≥ 70 %	Занадто високий рівень вологості
≥ 60 та < 70 %	Високий рівень вологості
≥ 30 та < 60 %	Ідеальний рівень вологості
≥ 25 та < 30 %	Малий рівень вологості
Менш як < 25 %	Занадто малий рівень вологості

Таблиця 3.2 – Рівні вологості

Напрям вітру – важливий показник, для пілотів, яхтсменів, парапланувальників, повітроплавців та усіх інших, У сервісі він представлений в градусах (рисунок 3.10);

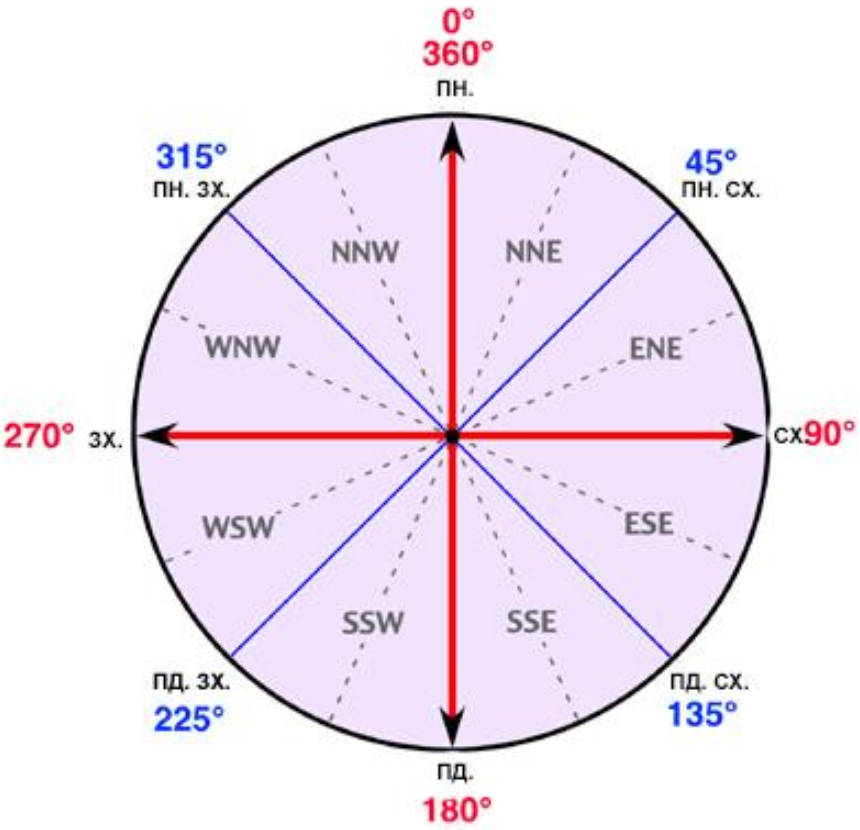

Рисунок 3.10 – Відповідність градуса та напряму вітру

Блок складається з зображення, назви та значення напрямку вітру у градусах (рисунок 3.11);

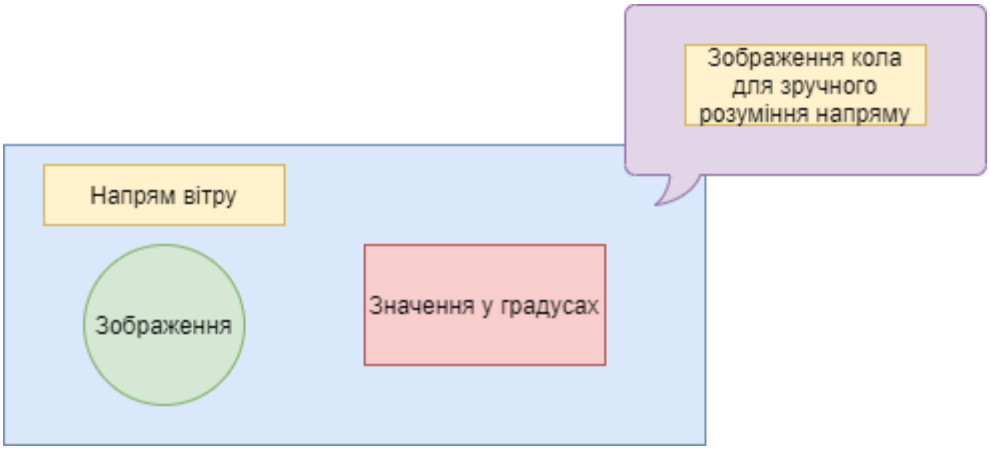

Рисунок 3.11 – Блок напрямку вітру

Рівень тиску – надзвичайно важливий параметр будь-якого прогнозу погоди. Блок складається з назви, зображення та значення атмосферного тиску у Паскалях (рисунок 3.12);

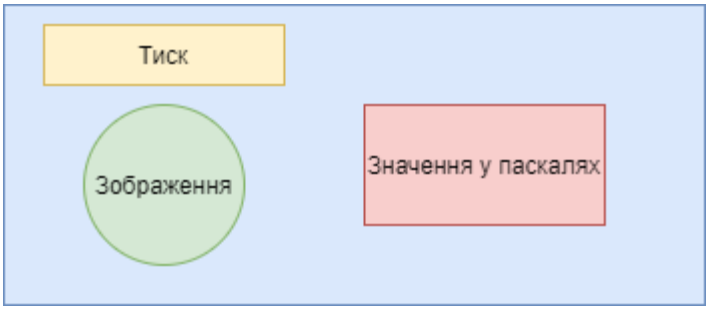

Рисунок 3.12 – Блок рівня атмосферного тиску

Схід та захід сонця – цей блок містить в собі два зображення, назву та два значення часу відповідно (рисунок 3.13);

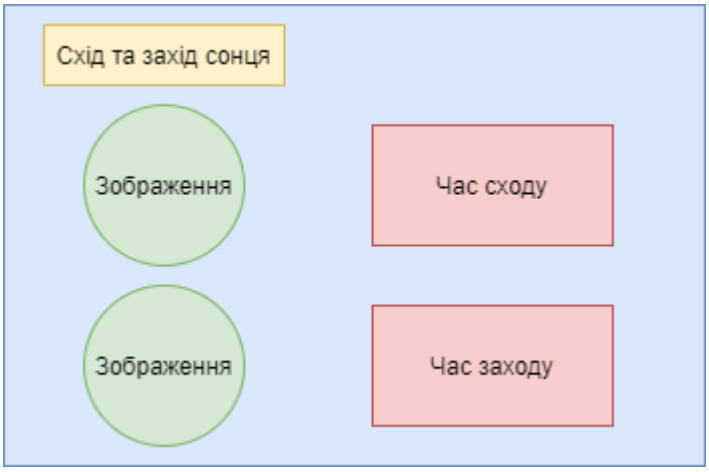

Рисунок 3.13 – Блок сходу та заходу сонця

Після створення макетів, поглиблення знань в основних параметрах прогнозу погоди, процес розробки веб-сервісу став простішим.

3.2 Архітектура системи

Фреймворк складається з певних бібліотек, деякі з них є основними, а інші доповненнями. Складаючи HTML шаблони, створюючи класи компонентів для керування цими шаблонами, додаючи певну логіку додатків у сервіси та зв’язуючи компоненти та служби в модулях, пишеться Angular-додаток.

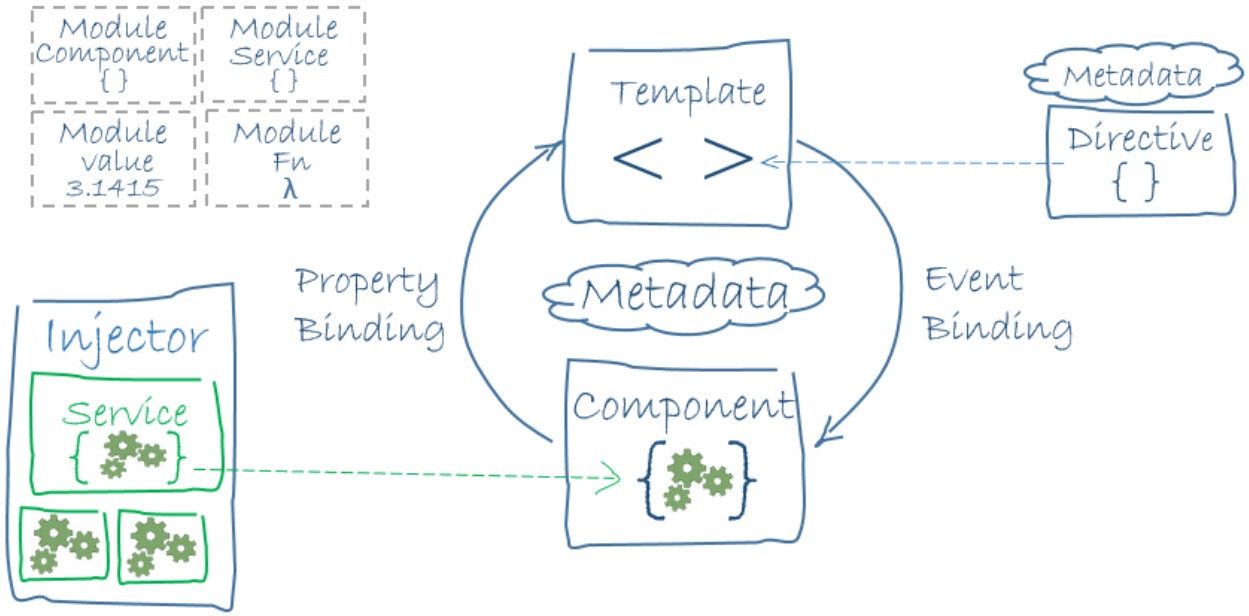

Рисунок 3.14 Вісім основних блоків Angular додатку

Схема архітектури (рисунок 3.14) визначає вісім основних блоків Angular додатку [12]:

- Модулі
- Компоненти
- Шаблони
- Метадані
- Прив'язка даних
- Директиви
- Послуги
- Ін’єкція залежності

3.3 Результат розробки

Виходячи з модельного представлення та розроблених макетів блоків для веб-сервісу, був розроблений веб-сервіс для детального перегляду прогнозу погоди.

Цей веб-сервіс надає користувачам повний спектр інформації щодо температури, вологості, УФ-індексу, напрямку вітру, сили вітру, тиску та часу сходу та заходу сонця.

На головній сторінці представлена інформація про усі характеристики погоди в цей момент у обраному місті (рисунок 3.15);

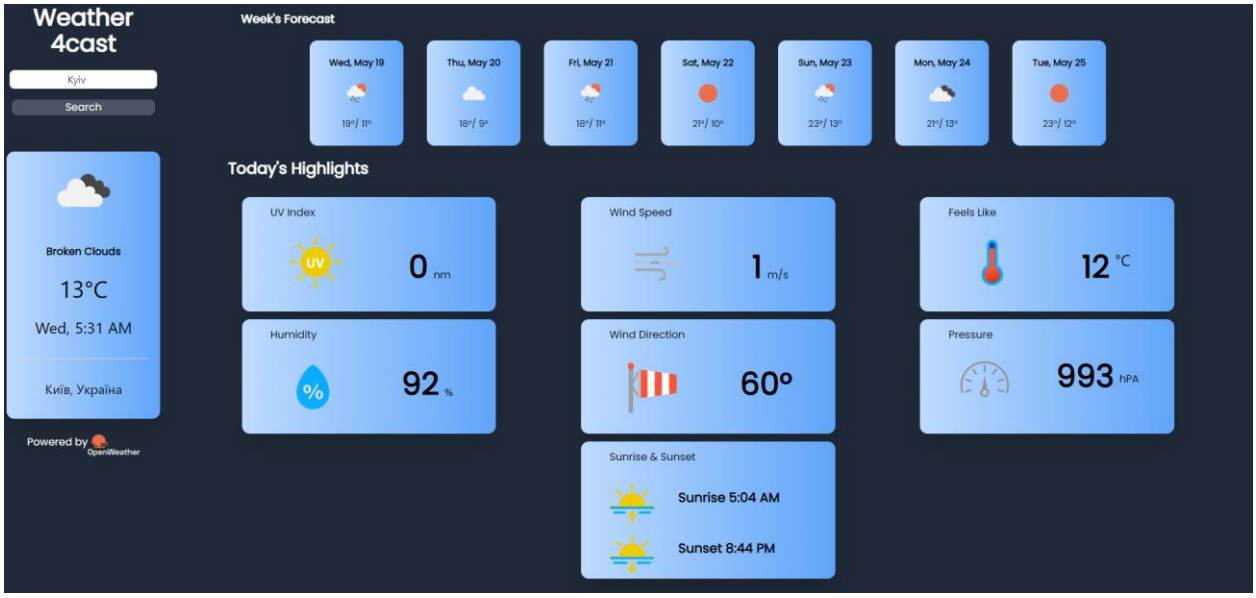

Рисунок 3.15 – Основна сторінка веб-сервісу у декстопній версії

Розглянемо роботу кожного блоку детальніше. Для прикладу візьмемо інформацію про погоду у місті Києві на 19 травня 2021 року. Аби отримати інформацію про погоду, користувачу необхідно ввести назву міста, або обрати його зі списку найпопулярніших міст (рисунок 3.16). Вводити інформацію необхідно в поле «City», яке розташоване у верхньому лівому куті (Рисунок 3.16)

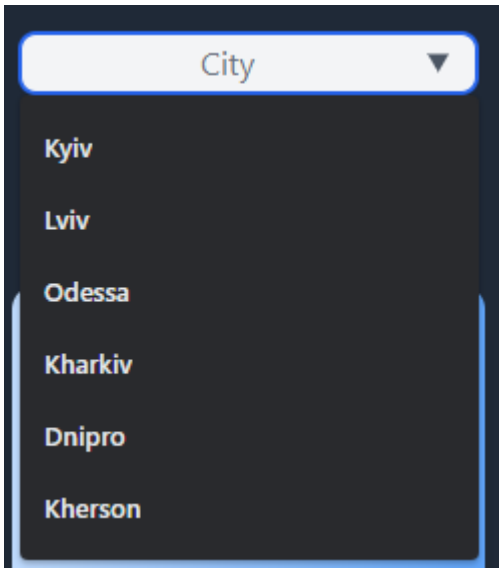

Рисунок 3.16 – Поле вводу міста та список популярних запитів

Після вибору міста, у сервісі з’явиться детальна інформація про погоду у цьому регіоні. В блоці короткої інформації про погоду з’являться значення температури, іконка погоди, стислий опис, дата та назва регіону (рисунок 3.17). Можемо побачити, що у Києві хмарне небо, 13 градусів тепла та 5 година ранку.

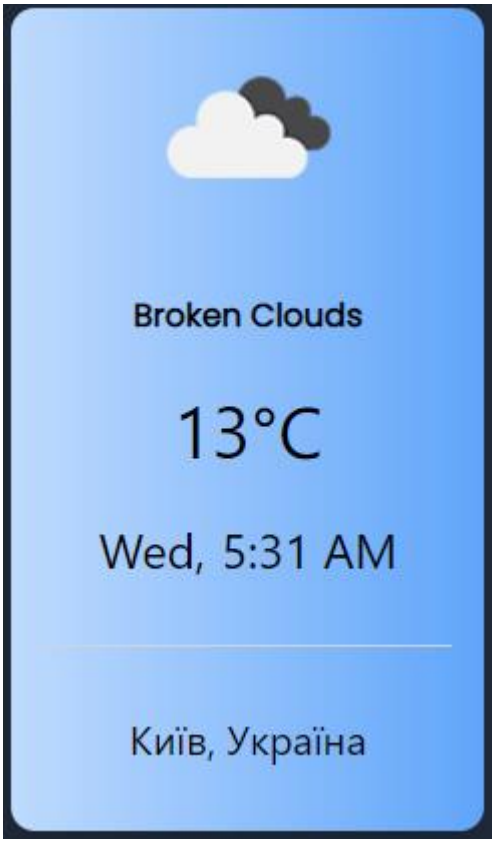

Рисунок 3.17 – Надана інформація у блоці короткого опису погоди

Розглянемо блок прогнозу погоди на неділю. В ньому відображається найголовніша інформація у вигляді іконок з зображеннями дощу, хмар, сонця, блискавиць та дані про максимальну та мінімальну температуру впродовж дня (рисунок 3.18);

Рисунок 3.18 – Надана інформація у блоці тижневого прогнозу

У блоці сьогоднішніх основних подій, відображена детальна інформація щодо основних характеристик погоди сьогодні (рисунок 3.19);

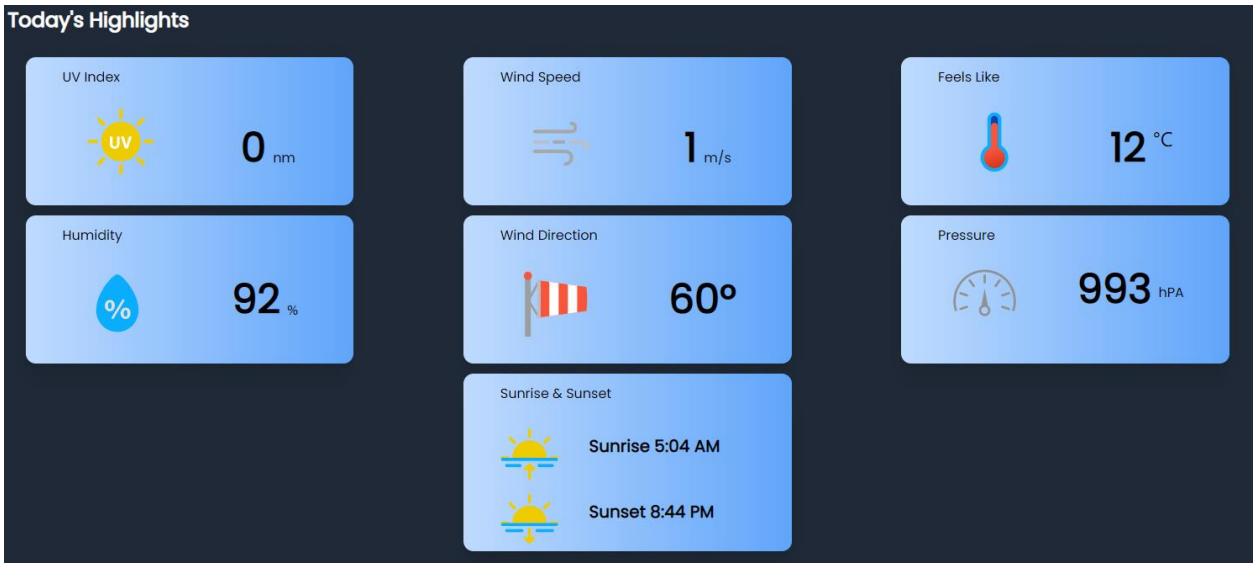

Рисунок 3.19 – Надана інформація у блоці сьогоднішніх основних подій

Можемо побачити, що в кожному блоці з’явилися певні значення які повністю відповідають усім характеристикам у підрозділі 3.1, розшифруємо отримані значення:

УФ-індекс – значення 0, що відповідає низькому рівню ультрафіолетового випромінювання, тобто захист не потрібен.

Швидкість вітру – значення 1 м/с, що відповідає вітру «Штиль», 0 балів за шкалою Бофорта.

Відчувається як – значення 12 градусів за Цельсієм, що відповідає фактичному значенню отриманому у блоці короткого опису погоди.

Вологість – значення 92 %, що відповідає високому рівню вологості.

Напрям вітру – значення 60 градусів, що відповідає Сх. Пн. Сх. напрямку

Тиск – значення 993 Паскаль, що відповідає нормальному атмосферному тиску.

Схід та захід сонця – час сходу 05:04 ранку, час заходу 08:44.

3.4 Результати опитування

За допомогою сервісу «Google Forms» було створено опитування (Додаток Б). Для проходження опитування, респондентам необхідно було перейти за посиланням, ввести свою електронну пошту, для запобігання багаторазових відповідей, та відповісти на 9 питань. В результаті опитування 70 осіб, були отримані наступні результати: (рисунок 3.20, 3.21, 3.22, 3.23, 3.24, 3.25, 3.26, 3.27, 3.28).

Візуальну привабливість веб-сервісу:

- 21,4% (15 осіб) респондентів оцінили, як надзвичайно привабливий,
- 48,6% (34 особи) респондентів оцінили, як дуже привабливий,
- 24,3% (17 осіб) респондентів оцінили, як дещо привабливий,
- 5.7% (4 особи) респондентів оцінили, як не дуже привабливий,
- 0% (0 осіб) респондентів оцінили, як взагалі не привабливий.

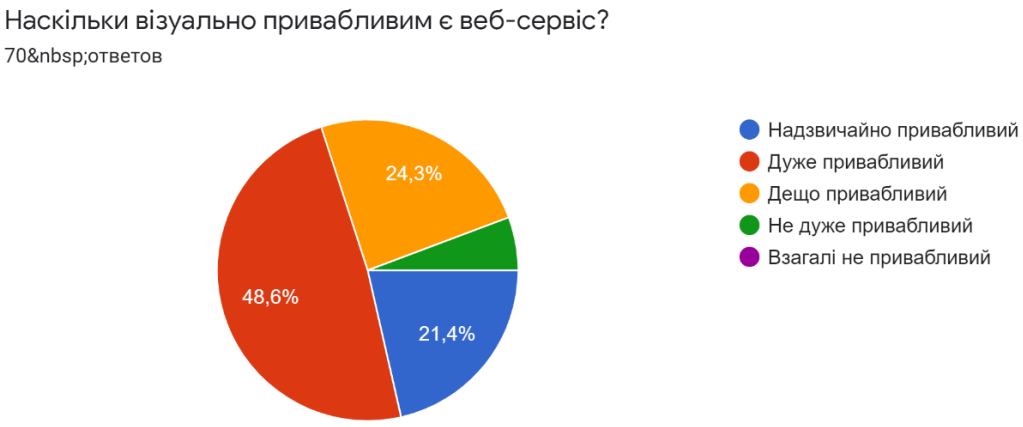

Рисунок 3.20 – Статистика першого питання

Простоту розуміння інформації:

- 42,9% (15 осіб) респондентів оцінили, як надзвичайно легко,
- 44,3% (34 особи) респондентів оцінили, як дуже легко,
- 10% (17 осіб) респондентів оцінили, як дещо легко,
- 2,9% (4 особи) респондентів оцінили, як не дуже легко,
- 0% (0 осіб) респондентів оцінили, як взагалі не легко.
-

Рисунок 3.21 – Статистика другого питання

Роботу веб-сервісу за шкалою від 1 до 10:

- 35,7% (25 осіб) респондентів оцінили на 10 балів,
- 32,9% (23 особи) респондентів оцінили на 9 балів,
- 17,1% (12 осіб) респондентів оцінили на 8 балів,
- 8,6% (6 осіб) респондентів оцінили на 7 балів,
- 4,3% (3 особи) респондентів оцінили на 6 балів,
- 1,4% (1 особа) респондентів оцінили на 5 балів.
- 0% (0 осіб) респондентів оцінила на 4, 3, 2, 1 бали.
-

Рисунок 3.22 – Статистика третього питання

Простим у використанні, веб-сервіс, за шкалою від 1 до 10, був для 65,2% (45 осіб) респондентів, вони вважають сервіс надзвичайно простим у використанні.

Рисунок 3.23 – Статистика четвертого питання

Задоволеними кількістю інформації представленою у веб-сервіс залишились 84,3% (59 осіб) респондентів, а не задоволеними 5,7% (4 особи).

Рисунок 3.24 – Статистика п’ятого питання

62,9% (44 особи) респондентів використовували би цей сервіс у повсякденному житті, 18,6% (13 осіб) – ні.

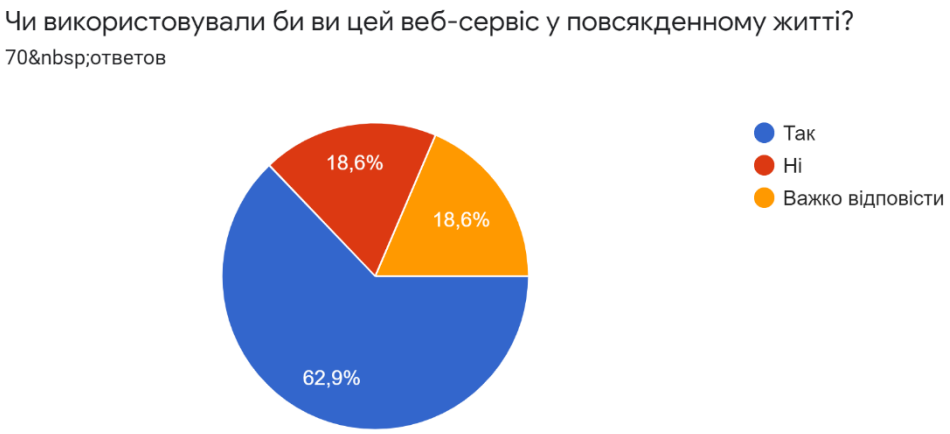

Рисунок 3.25 – Статистика шостого питання

72,9% (51 особа) респондентів, вважають, що сервіс необхідно доповнити певними функціями, 27,1 (19 осіб) іншої думки.

Рисунок 3.26 – Статистика сьомого питання

Цей веб-сервіс, 74,3% (52 особи) респондентів порекомендували би своїм знайомим, 8,6% (6 осіб) – ні.

Рисунок 3.27 – Статистика восьмого питання

Повністю довіряють інформації у веб-сервісі – 41,4% (29 осіб) респондентів, довіряють майже повністю – 31,4% (22 особи) респондентів, помірно довіряють – 24,3% (17 осіб) респондентів, майже не довіряють – 2,9% (2 особи) респондентів.

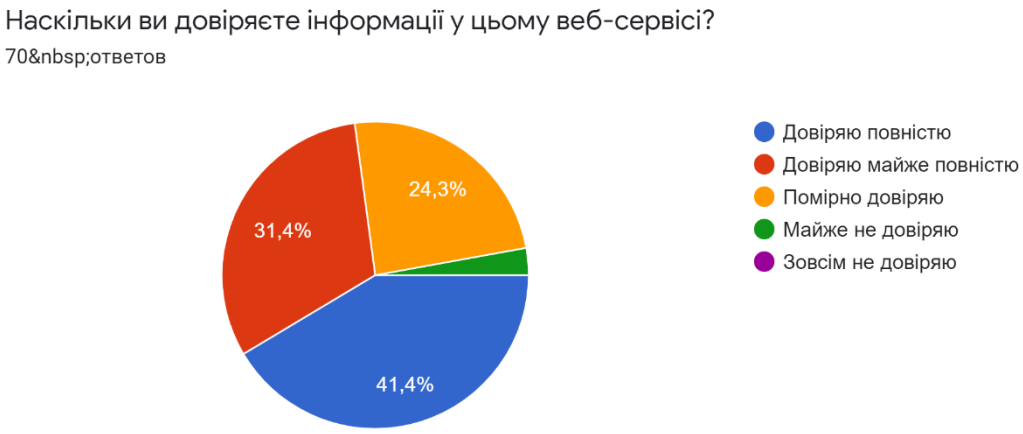

Рисунок 3.28 – Статистика дев’ятого питання

Також респондентам було запропоноване десяте питання, де кожен мав змогу написати свої ідеї, які можна втілити у майбутньому для покращення веб-сервісу.

ВИСНОВКИ

Використання веб-сервісів є одним з найперспективніших напрямів та форм ведення бізнесу та надання інформації різного типу. Використання веб-сервісу надання прогнозу погоди допомагає людям швидко та зручно отримати усю детальну інформацію про погоду у будь-якому місті.

В даній роботі була поставлена наступна мета: розробити зручний веб-сервіс для надання інформативних погодних даних у будь-якому місті планети.

В процесі досягнення поставленої мети були виконанні наступні задачі:

- Вивчено базові принципи розробки веб-сервісів;
- Вивчено базові принципи роботи API;
- Розглянуто переваги та недоліки популярних фреймворків для розробки інтерфейсу веб-сервісу;
- Розроблено веб-сервіс прогноз погоди з застосуванням усіх вивчених технологій;
- Проведено опитування серед різних верств населення, за результатами якого можна зробити висновки про успішність розробленого веб-сервісу;

Веб-сервіс цілком можливо впровадити як повноцінний сервіс для перегляду прогнозу погоди після деяких доопрацювань.

Таким чином слід вважати, що результати розробки повністю відповідають усім вимогам технічного завдання, поставлена мета досягнута. Робота має закінчений характер.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Lafon Y. Web Services Activity [Електронний ресурс] / Yves Lafon // W3C Working Group. – 2008. – Режим доступу до ресурсу: https://www.w3.org/2002/ws/.
2. Web Services Architecture [Електронний ресурс] / [D. Booth, H. Haas, F. McCabe та ін.] // W3C Working Group. – 2004. – Режим доступу до ресурсу: https://www.w3.org/TR/ws-arch/.
3. OpenWeatherMap [Електронний ресурс] // OpenWeather. – 2021. – Режим доступу до ресурсу: https://openweathermap.org/technology.
4. "Angular Docs" [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/guide/architecture
5. Angular, version 2: proprioception-reinforcement [Електронний ресурс] – Режим доступу до ресурсу: http://blog.angularjs.org/2016/09/angular2-final.html
6. AngularJS and Angular 2+: a Detailed Comparison [Електронний ресурс] / Manjunath M. – 2018. – Режим доступу до ресурсу: https://www.sitepoint.com/angularjs-vs-angular/.
7. Getting started with Tailwind CSS [Електронний ресурс]:[Веб-сайт] – Режим доступу до ресурсу: https://tailwindcss.com/docs
8. Biehl M. RESTful API Design / Matthias Biehl., 2016. – 299 с.
9. Saks E. JavaScript Frameworks: Angular vs React vs Vue. / Elar Saks., 2019. – 47 с.
10. Cherny B. Programming TypeScript: Making Your JavaScript Applications Scale / Boris Cherny., 2019. – 265 с.
11. Maurer F. A Study of the Effectiveness of Usage Examples in REST API Documentation / F. Maurer, C. Anslow, M. P. Robillard., 2017. – 61 с.
12. Architecture Overview [Електронний ресурс] – Режим доступу до ресурсу: https://v2.angular.io/docs/ts/latest/guide/architecture.html

ДОДАТОК А

Файл api.interceptor.ts. Введення API ключів.

```
import { environment } from '../../environments/environment';
import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/co
mmon/http';
import { Observable } from 'rxjs';

export class APIInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>
 {
    let requestURL = req.url;
    if (requestURL.indexOf('@api-x') !== -1) {
      requestURL = requestURL.replace('@api-x', environment.geoCodeAPI);
      console.log( requestURL)
      const cloneReq = req.clone({
        url: requestURL,
        params: req.params.set(
          'apiKey',
          'oyiLCKwrHuef9AC3obarCMYzT6N4YyRONfc7IZdPTYQ'
        )
      });
      return next.handle(cloneReq);
    }

    else if (requestURL.indexOf('@api-y') !== -1) {
      requestURL = requestURL.replace('@api-y', environment.weatherAPI);
      const cloneReq = req.clone({
        url: requestURL,
        params: req.params.set(
          'appid',
          '2144ffced487ef08539ab2aa49cf0b1d'
        )
      });

      return next.handle(cloneReq);
    }
  }
}
```

ДОДАТОК Б

Питання в опитуванні «Вебсервіс прогноз погоди»

1) Наскільки візуально привабливим є веб-сервіс?

a) Надзвичайно привабливий
b) Дуже привабливий
c) Дещо привабливий
d) Не дуже привабливий
e) Взагалі не привабливий

2) Наскільки просто зрозуміти інформацію у веб-сервісі?

a) Надзвичайно легко
b) Дуже легко
c) Дещо легко
d) Не дуже легко
e) Взагалі не легко

3) Як ви оцінюєте роботу веб-сервісу за шкалою від 1 до 10?

a) Надзвичайно жахливо – 0
b) Надзвичайно чудово – 10

4) Наскільки простим для вас був веб-сервіс у використанні за шкалою від 1 до 10?

a) Надзвичайно складний – 0
b) Надзвичайно простий – 10

5) Чи задоволені ви кількістю інформації представленної у веб-сервісі?

a) Так
b) Ні
c) Важко відповісти

6) Чи використовували би ви цей веб-сервіс у повсякденному житті?

a) Так
b) Ні

c) Важко відповісти

7) Чи варто доповнити сервіс певними функціями та інформацією?

a) Так

b) Ні

8) Чи порекомендували би ви цей веб-сервіс своїм знайомим?

a) Так

b) Ні

c) Важко відповісти

9) Наскільки ви довіряєте інформації у цьому веб-сервісі?

a) Довіряю повністю

b) Довіряю майже повністю

c) Помірно довіряю

d) Майже не довіряю

e) Зовсім не довіряю

10) Чи є у вас ідеї чим можна доповнити існуючий веб-сервіс?