МЕЛІТОПОЛЬСЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО

факультет інформатики, математики та економіки

кафедра інформатики і кібернетики

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: «**Розробка та розгортання Slack-бота для отримання даних про погоду**»

Виконав: здобувач вищої освіти
групи М314-і
Спеціальності 122 Комп'ютерні науки
ОП Комп'ютерні науки
Золотарьов Павло Романович

Керівник: доктор філософії,
Сіциліцин Юрій Олександрович

Рецензент: д.т.н., професор
кафедри «Комп'ютерні науки»
ТДАТУ імені Дмитра Моторного
Малкіна Віра Михайлівна

Запоріжжя – 2025 року

ЗОЛОТАРЬОВ ПАВЛО РОМАНОВИЧ

РОЗРОБКА ТА РОЗГОРТАННЯ SLACK-БОТА ДЛЯ ОТРИМАННЯ ДАНИХ ПРО ПОГОДУ

Анотація. Кваліфікаційна робота присвячена розробці та розгортанню Slack-бота для отримання актуальних даних про погоду з використанням технологій JavaScript та платформи Node.js. Такий підхід є особливо актуальним у сучасних умовах, коли корпоративні комунікаційні інструменти вимагають швидкого доступу до оперативної інформації в межах робочого середовища. Розроблений бот забезпечує автоматизоване отримання метеорологічних даних і надання їх користувачам у зручному форматі безпосередньо в Slack, що сприяє оптимізації робочих процесів та підвищенню ефективності взаємодії команд.

У роботі здійснено аналіз існуючих рішень для інтеграції ботів у Slack та огляд популярних сервісів погодних даних, таких як OpenWeatherMap і WeatherAPI. Розглянуто їхні переваги, недоліки та особливості використання. Обґрунтовано вибір технологічного стеку, до якого входять Node.js, офіційний фреймворк Slack Bolt, а також зовнішній погодний API. Використання Bolt.js забезпечує зручний інтерфейс для обробки подій, команд і повідомлень, що значно спрощує розробку, тестування та підтримку програмного продукту.

Основні функції розробленого бота включають обробку користувацьких запитів у Slack, отримання поточної погоди та прогнозу для вибраного міста, форматування відповіді у вигляді інтерактивних блоків Slack, а також обробку помилок мережевих запитів. У роботі описано процес розгортання застосунку, зокрема використання Socket Mode, налаштування OAuth-токенів та можливість контейнеризації засобами Docker.

Структура роботи складається з трьох розділів. Перший розділ містить аналіз предметної області, огляд інструментів для створення ботів і хмарних сервісів погодних даних. У другому розділі розглядаються особливості вибору технологій та описуються ключові компоненти стеку Node.js і фреймворку Bolt. Третій розділ присвячений реалізації програмного продукту, включаючи розробку архітектури, програмування основних модулів і тестування бота в реальному середовищі Slack.

Результати роботи можуть бути використані для подальшої розробки інтелектуальних ботів, інтеграції зовнішніх сервісів та розширення можливостей корпоративних платформ комунікації.

Ключові слова: Slack-бот, JavaScript, Node.js, Bolt, автоматизація, погодні дані, OpenWeatherMap, інтеграція, корпоративні комунікації, веб-розробка.

ZOLOTAROV PAVLO

DEVELOPMENT AND DEPLOYMENT OF A SLACK-BOT FOR OBTAINING WEATHER DAT

Annotation. The thesis is dedicated to the development and deployment of a Slack bot for retrieving up-to-date weather information using JavaScript and the Node.js platform. This approach is particularly relevant in modern conditions, where corporate communication tools require quick access to operational information within the workspace. The developed bot provides automated retrieval of meteorological data and delivers it to users in a convenient format directly in Slack, which helps optimize workflows and increase the efficiency of team communication.

The work includes an analysis of existing solutions for integrating bots into Slack and a review of popular weather data services such as OpenWeatherMap and WeatherAPI. Their advantages, disadvantages, and specific usage features are examined. The choice of the technological stack-Node.js, the official Slack Bolt framework, and an external weather API-is justified. The use of Bolt.js provides a convenient interface for handling events, commands, and messages, significantly simplifying the development, testing, and maintenance of the software product.

The main functions of the bot include processing user requests in Slack, obtaining current weather conditions and forecasts for a selected city, formatting responses using Slack's interactive block components, and handling network request errors. The work also describes the deployment process of the application, including the use of Socket Mode, OAuth token configuration, and the option to containerize the bot with Docker.

The structure of the work consists of three sections. The first section contains an analysis of the subject area, an overview of tools for creating bots, and cloud services for weather data. The second section discusses the selection of technologies and describes the key components of the Node.js stack and the Bolt framework. The third section is devoted to the implementation of the software product, including the development of architecture, programming of the main modules, and testing of the bot in a real Slack environment.

The results of the work can be used for further development of intelligent bots, integration of external services, and expansion of the capabilities of corporate communication platforms.

Keywords: Slack bot, JavaScript, Node.js, Bolt, automation, weather data, OpenWeatherMap, integration, corporate communication, web development

ЗМІСТ

ЗМІСТ.....2
РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ.....6
1.1 Основні методи та підходи до розробки Slack-ботів для отримання погодних даних.....6
1.2 Аналіз існуючих Slack-ботів для отримання даних про погоду.....10
1.3 Побудова концептуальної моделі та архітектури Slack-боту для отримання погодних даних.....13
2. Архітектура Slack-боту.....14
РОЗДІЛ 2 ВИБІР ТЕХНОЛОГІЧНОГО СТЕКУ РОЗРОБКИ ДОДАТКУ.....17
2.1. Вибір інструментів та технологій.....17
2.2. Вибір серверної частини, API та інших технологій.....24
2.3. Інструменти для отримання та обробки погодних даних у Slack-боті.....26
РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОДЕЛЮВАННЯ ВИТРАТ В ПРИВАТНІЙ КОМПАНІЇ.....32
3.1. Технічне завдання на розробку Slack-боту для отримання даних про погоду.....32
3.2. Реалізація основних модулів.....37
3.3. Опис роботи програмного засобу.....40
ВИСНОВКИ.....49
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....51

ВСТУП

Актуальність. З розвитком корпоративних комунікаційних платформ та активним переходом компаній до цифрової взаємодії зростає потреба в інструментах, що забезпечують швидкий доступ до оперативної інформації безпосередньо у робочому середовищі. Сучасні команди працюють у динамічних умовах, де актуальні дані - зокрема інформація про погодні умови - можуть впливати на планування, логістику, координацію проєктів та прийняття рішень. Отримання таких даних за допомогою традиційних веб-сервісів або мобільних застосунків потребує додаткових дій, що ускладнює робочі процеси та знижує їх ефективність.

Одним із найбільш зручних інструментів для автоматизації отримання оперативних повідомлень є чат-боти, інтегровані у корпоративні платформи, такі як Slack. Вони дозволяють миттєво отримувати необхідну інформацію в межах звичного комунікаційного середовища, без перемикання між додатками. Проте більшість існуючих рішень або мають обмежений функціонал, або є комерційними продуктами з високою вартістю підписки. Це ускладнює їх використання малими командами, стартапами та організаціями, які прагнуть мати кастомізований інструмент для власних потреб.

Створення Slack-бота для отримання погодних даних вирішує проблему оперативного доступу до метеорологічної інформації, автоматизує процес її запиту та представлення у зручному форматі. Використання JavaScript, Node.js та офіційного фреймворку Slack Bolt дозволяє розробити гнучкий, масштабований та інтерактивний інструмент, який легко інтегрується у робочі процеси команди. Такий бот може забезпечувати не лише базову інформацію про поточну погоду, але й прогноз, попередження про несприятливі умови та інші дані, що сприяють підвищенню ефективності планування.

Об’єкт: Slack-бот для отримання даних про погоду/

Предмет: розробка та розгортання Slack-бота для отримання даних про погоду.

Мета дослідження: розробити бот для корпоративного месенджера Slack, що дозволить користувачу швидко отримати дані про погоду в конкретному місті.

Сформована мета, об’єкт і предмет дослідження дозволили поставити такі завдання:

- Проаналізувати основні методи та підходи до розробки ботів у Slack
- Розробити архітектуру.
- Порівняти існуючі рішення для даного домену
- Вибрати стек технологій для розробки Slack-боту.
- Реалізувати Slack-бот для отримання даних про погоду.

Створити користувацьку інструкцію для роботи з ботом.

Теоретичне завдання: полягає в аналізі методів інтеграції ботів у корпоративні платформи комунікацій, таких як Slack, дослідженні існуючих підходів до автоматизації отримання та обробки оперативних даних у робочому середовищі, а також вивченні інструментів для розробки програмних рішень на основі обраного технологічного стеку (зокрема JavaScript, Node.js та фреймворку Slack Bolt). Особлива увага приділяється методам взаємодії зі зовнішніми API, обробці подій і повідомлень у Slack.

Практичне завдання: виражається у розробці програмного додатку - Slack-бота, який автоматизує процес отримання актуальної погодної інформації та надає її користувачам безпосередньо в середовищі Slack. Бот дозволяє отримувати поточні погодні умови, прогноз та інші метеорологічні показники на основі даних зовнішніх погодних сервісів (наприклад,

OpenWeatherMap). Додаток розробляється з використанням JavaScript, Node.js та офіційного SDK Bolt, що забезпечує інтерактивну взаємодію з користувачами, обробку команд, подій та повідомлень

Структура роботи: Перший розділ присвячений огляду предметної області, зокрема основним методам взаємодії ботів зі Slack API, принципам роботи корпоративних комунікаційних платформ, а також аналізу існуючих рішень для отримання погодних даних. Розглянуто переваги та недоліки популярних сервісів, таких як OpenWeatherMap, WeatherAPI та AccuWeather, а також існуючих ботів у Slack, що надають погодну інформацію та обґрунтовано необхідність створення власного додатку.

Другий розділ зосереджений на виборі технологічного стеку для розробки Slack-бота. У ньому детально описано інструменти та технології, які застосовуються для реалізації погодного бота, зокрема мову програмування JavaScript, платформу Node.js, офіційний SDK Slack Bolt, а також роботу з API зовнішніх погодних сервісів. Розглянуто переваги обраного стеку з точки зору швидкості розробки, підтримки подієвої моделі, інтеграції з хмарними платформами та можливості розгортання у Socket Mode без необхідності у публічному веб-сервері.

Третій розділ присвячено безпосередній реалізації програмного продукту. У ньому описано архітектуру бота, модуль обробки подій Slack (app_mention, message, slash-команди), модуль взаємодії з погодним API, модуль форматування відповідей та модуль тестування. Детально розглянуто алгоритми роботи кожного компонента, наведено приклади коду на JavaScript із використанням Bolt.js, описано процес налаштування токенів, підписів та розгортання бота у робочому середовищі Slack.

РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні методи та підходи до розробки Slack-ботів для отримання погодних даних

У сучасних умовах цифровізації та автоматизації інформаційних процесів все більшого поширення набувають чат-боти, які забезпечують швидкий та зручний доступ до необхідної інформації. Чат-боти активно використовуються у сфері бізнесу, освіти, державного управління та інформаційних сервісів. Однією з популярних платформ для створення та впровадження чат-ботів є Slack - корпоративний месенджер, що дозволяє інтегрувати сторонні програмні сервіси та автоматизовані системи. Slack-боти є ефективним інструментом для отримання довідкових даних, зокрема інформації про погодні умови.

Розробка Slack-бота для отримання погодних даних передбачає застосування комплексу методів і підходів, які охоплюють питання архітектури програмного забезпечення, інтеграції з зовнішніми сервісами, обробки користувацьких запитів та забезпечення надійності й безпеки системи. Ключовим аспектом є правильна організація взаємодії між платформою Slack та погодними сервісами, що надають актуальні метеорологічні дані. [40]

Основним інструментом для розробки Slack-ботів є Slack API - набір програмних інтерфейсів, які дозволяють створювати власні додатки, керувати ботами та обробляти події у робочому просторі Slack. Slack API підтримує кілька підходів до взаємодії з користувачами, зокрема Events API, Slash Commands та інтерактивні повідомлення. Для ботів, призначених для отримання погодної інформації, найбільш доцільним є використання команд зі слешем, які дозволяють користувачам надсилати запити у зручному форматі, наприклад /weather Lviv. Такий підхід забезпечує простоту використання та швидкість отримання результатів.

Важливим етапом розробки є вибір та інтеграція зовнішнього погодного API. Оскільки Slack не надає власних метеорологічних даних, бот повинен звертатися до спеціалізованих погодних сервісів. Найпоширенішими серед них є OpenWeatherMap, WeatherAPI, AccuWeather та Meteostat. Дані сервіси надають доступ до поточної погоди, прогнозів, історичних даних та додаткових метеорологічних параметрів. Отримання інформації зазвичай здійснюється шляхом надсилання HTTP-запитів та обробки відповідей у форматі JSON. Вибір конкретного сервісу залежить від вимог до точності даних, частоти оновлення та умов ліцензування.

Наступним важливим підходом є проєктування архітектури Slack-бота. Найчастіше застосовується клієнт-серверна архітектура, у якій бот реалізується як серверний застосунок. У такій архітектурі Slack виступає клієнтом, що надсилає запити, а серверна частина бота виконує обробку команд, взаємодіє з погодним API та формує відповідь. Цей підхід дозволяє централізувати бізнес-логіку, забезпечити масштабованість та спростити обслуговування системи. Серверна частина може бути реалізована з використанням різних мов програмування, таких як Python, JavaScript (Node.js) або Java, що дозволяє адаптувати рішення до наявних технічних ресурсів.

Особливу увагу під час розробки Slack-бота для погодних даних слід приділяти методам обробки користувацьких запитів. Користувачі можуть вводити назви міст різними мовами, з помилками або у скороченому вигляді. Тому бот повинен використовувати механізми перевірки та нормалізації введених даних. Для цього можуть застосовуватися методи попередньої обробки тексту, такі як приведення до стандартного формату, перевірка коректності символів та обробка виняткових ситуацій. У разі некоректного запиту бот має інформувати користувача про помилку та надавати підказки щодо правильного формату введення.

Ще одним важливим аспектом є форматування та подання результатів. Slack надає можливість використовувати структуровані повідомлення з використанням Block Kit, що дозволяє створювати інформативні та візуально зрозумілі відповіді.

Завдяки цьому погодні дані можуть бути представлені у вигляді структурованих блоків із зазначенням температури, погодних умов, швидкості вітру та інших параметрів. Такий підхід значно покращує зручність сприйняття інформації та підвищує користувацький досвід.

Важливим методом забезпечення стабільності роботи Slack-бота є обробка помилок та виняткових ситуацій. До таких ситуацій належать відсутність відповіді від погодного сервісу, перевищення лімітів API-запитів або проблеми з мережею. Реалізація механізмів повторних запитів, логування помилок та інформування користувачів про тимчасові проблеми дозволяє підвищити надійність системи та уникнути збоїв у роботі бота.

Окрему увагу необхідно приділити питанням безпеки. Slack-боти використовують токени доступу та ключі API, які повинні зберігатися у захищеному вигляді. Найбільш поширеним підходом є використання змінних середовища або спеціалізованих сервісів для зберігання конфіденційних даних. Також важливо мінімізувати права доступу бота, надаючи лише ті дозволи, які необхідні для його функціонування.

Розробка Slack-ботів - це процес створення програмних агентів, здатних автоматично обробляти повідомлення користувачів, взаємодіяти з API та виконувати дії в межах корпоративного середовища Slack. Основною метою такого аналізу є оцінка архітектури, можливостей інтеграції та ефективності роботи бота для забезпечення швидкого доступу до актуальної інформації, зокрема даних про погоду. Основними елементами проєктування та розробки Slack-ботів є [1]:

- Slack API та подієва модель взаємодії. Slack надає розгалужену інфраструктуру для створення інтерактивних ботів, що включає Web API, Events API та Socket Mode. Аналіз цих компонентів дозволяє визначити, який механізм інтеграції найбільш ефективний для конкретного застосування.
- Використання зовнішніх API (OpenWeatherMap, WeatherAPI). Основою будь-якого погодного бота є інтеграція з API, що надають метеорологічні дані.Також важливо враховувати правила кешування,

обмеження запитів і можливість використання декількох джерел даних для підвищення надійності.

– Структура обробки команд і повідомлень. Slack-боти працюють за принципом обробки вхідних команд і подій. Основними підходами є:

- Slash-команди (наприклад, /weather Kyiv)
- Обробка ключових слів у повідомленнях
- Обробка згадувань бота
- Інтерактивні кнопки та блоки (Block Kit)

Правильне структурування команд дозволяє покращити взаємодію користувачів з ботом та забезпечити інтуїтивність роботи [2].

– Форматування повідомлень та візуалізація інформації. Slack дозволяє створювати структуровані та візуально привабливі повідомлення з використанням Block Kit. Основними принципами формування повідомлень є:

- структурованість інформації (температура, вітер, вологість, опис),
- використання іконок, кольорів, блокових елементів,
- можливість швидкого сприйняття інформації,
- дотримання рекомендацій Slack щодо форматування [3].

Коректна візуалізація значно підвищує зручність і корисність бота.

– Інструменти розробки Slack-ботів. Сучасні технології дозволяють створити потужний бот з мінімальними витратами часу:

- Node.js - основна платформа для розробки подієво орієнтованих застосунків;
- Slack Bolt - офіційний фреймворк для створення ботів із мінімальним обсягом коду;
- Axios / fetch API - для HTTP-запитів до погодних API;
- Docker - для контейнеризації та розгортання;
- GitHub / GitLab - для керування версіями [4].

Кожен із цих інструментів сприяє масштабованості та гнучкості програмного рішення.

– Тестування та налагодження. Тестування є важливим етапом розробки. Використання інструментів на кшталт ngrok, Slack Event Tester значно спрощує процес налагодження [5].

1.2 Аналіз існуючих Slack-ботів для отримання даних про погоду

Slack-боти погоди є корисними інструментами для автоматичного отримання прогнозу погоди безпосередньо в робочому просторі Slack. Вони дозволяють користувачам швидко отримувати актуальні дані про температуру, опади, вітер та інші метеоумови, що особливо зручно для планування робочого дня, подорожей або заходів на відкритому повітрі. Існує кілька популярних рішень, які можна розділити на готові комерційні боти та приклади відкритого коду для самостійної інтеграції [6].

Forecast Slack Bot - один із найпопулярніших ботів для отримання прогнозу погоди. Він дозволяє користувачам отримувати поточну температуру, погодні умови та прогноз на декілька днів прямо в Slack. Бот інтегрується з зовнішніми погодними API, такими як OpenWeatherMap, і надає простий інтерфейс для запитів через команду /weather [6]. Інтерфейс зображено на Рис. 1.1:

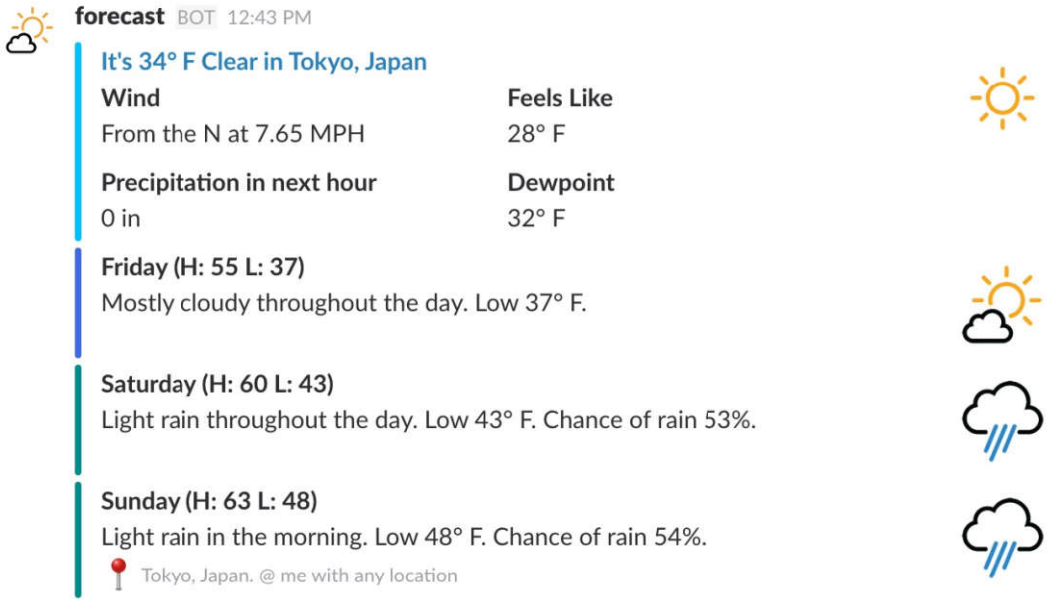


Рис. 1.1 - Інтерфейс боту Forecast Slack Bot

Основні функції:

- Поточна погода та прогноз на 5 днів;
- Можливість вибору міста або геолокації користувача;
- Автоматичні повідомлення про зміни погоди.

Переваги:

- Простота використання та інтеграції;
- Швидке отримання актуальних даних;
- Підтримка багатьох мов.

Недоліки:

- Обмежений функціонал без платного API;
- Неможливість гнучкого налаштування повідомлень [6].

Slack Weather Bot (Postman Template) - шаблон для налаштування Slack-бота за допомогою Postman і Webhook, що дозволяє відправляти користувачам повідомлення з прогнозом погоди. Він добре підходить для розробників, які хочуть створити власного погодного бота з кастомними функціями [7]. Інтерфейс зображено на Рис. 1.2:



Рис. 1.2 Інтерфейс Slack Weather Bot with Postman Template

Основні функції:

- Надсилання прогнозу погоди за запитом;
- Інтеграція з Slack через Webhook;

- Можливість автоматичних повідомлень за розкладом.

Переваги:

- Гнучкість налаштувань і кастомізації;
- Легко інтегрується з будь-яким API;
- Безкоштовна реалізація для тестування.

Недоліки:

- Вимагає базових знань Postman та Slack API;
- Не надає готового користувацького інтерфейсу [7].

Weather Bot by franciskim - приклад простого Slack-бота на Node.js та Express, який отримує дані про погоду через сторонній API. Бот дозволяє інтегрувати повідомлення про погоду у власні канали Slack та налаштовувати команди [8].

Основні функції:

- Поточна погода за містом;
- Використання команд Slack для отримання прогнозу;
- Можливість розширення функцій за допомогою Node.js.

Переваги:

- Відкритий код, можливість модифікації;
- Простий і легкий у використанні;
- Добре підходить для навчальних та тестових проєктів.

Недоліки:

- Обмежений функціонал без додаткової розробки;
- Потребує знань JavaScript та Node.js [8].

Таким чином, існуючі Slack-боти погоди надають широкий спектр функцій від базового прогнозу до спеціалізованих повідомлень. Вибір конкретного рішення залежить від потреб користувачів, необхідного рівня автоматизації та можливостей інтеграції з API.

1.3 Побудова концептуальної моделі та архітектури Slack-боту для отримання погодних даних

Проєктування Slack-боту для отримання погодних даних починається з побудови концептуальної моделі, яка визначає основні компоненти системи, їх взаємодію та потоки даних. Концептуальна модель дозволяє чітко окреслити функціональні можливості бота, логіку його роботи та взаємодію з користувачами та сторонніми сервісами [9].

1. Концептуальна модель

Концептуальна модель Slack-боту для погоди включає такі основні компоненти:

- Інтерфейс користувача (Slack) - основний канал взаємодії, через який користувачі відправляють команди боту і отримують повідомлення. Бот реагує на команди, такі як /weather <місто> або /forecast <місто>[10].
- Обробник команд (Command Handler) - модуль, що приймає вхідні повідомлення, визначає тип запиту та передає його на відповідний сервіс обробки даних.
- Сервіс погодних даних (Weather API) - зовнішній сервіс (наприклад, OpenWeatherMap або WeatherAPI), що надає актуальну інформацію про температуру, опади, швидкість вітру та прогноз на декілька днів [11].
- Логіка обробки даних (Data Processor) - компонент, що форматує дані для відправки користувачу, обчислює середні значення, відображає попередження про екстремальні погодні умови та генерує повідомлення для Slack.
- База даних (Database, опціонально) - використовується для зберігання історії запитів, персональних налаштувань користувачів, геолокації та інших даних, що дозволяє реалізувати персоналізовані функції [12].

Концептуальна модель також передбачає можливість автоматичної відправки повідомлень про зміну погодних умов у заданий час або при

досягненні певних порогових значень (наприклад, температура нижче 0°C або сильний вітер).

2. Архітектура Slack-боту

Архітектура бота базується на багаторівневій моделі, яка забезпечує гнучкість, масштабованість та інтеграцію з зовнішніми сервісами. Основні рівні архітектури включають [13]:

1. Рівень інтерактивності (Interaction Layer) - взаємодія з користувачем через Slack API. Цей рівень відповідає за отримання повідомлень, обробку команд та відправку відповідей.

2. Рівень логіки (Application Layer) - містить бізнес-логіку бота, включаючи обробку команд, формування повідомлень, планувальник автоматичних сповіщень та обробку винятків.

3. Рівень доступу до даних (Data Access Layer) - взаємодія з базою даних для збереження історії запитів, налаштувань користувачів та кешування результатів API для зменшення кількості запитів до сторонніх сервісів.

4. Рівень інтеграції з зовнішніми API (Integration Layer) - модуль для отримання погодних даних через REST API сторонніх сервісів та обробки JSON/ XML відповідей [14].

Архітектура забезпечує високу модульність, що дозволяє додавати нові функції, такі як:

- прогноз на декілька днів;
- підтримка декількох локацій;
- сповіщення про екстремальні погодні умови;
- персоналізовані налаштування для користувачів.

3. Основні потоки даних

Потік даних у Slack-боті для погоди виглядає наступним чином:

1. Користувач надсилає команду через Slack;

2. Command Handler розпізнає команду і формує запит до Weather API;

3. Weather API повертає дані про погоду у структурованому форматі;

4. Data Processor обробляє отримані дані, формує повідомлення для користувача;

5. Повідомлення надсилається користувачу через Slack [15].

Отже, здійснивши огляд предметної області, пов'язаної з розробкою Slack-ботів для отримання погодних даних, встановлено, що чат-боти є зручним і ефективним інструментом для автоматизованого доступу до інформації в умовах цифровізації та активного використання корпоративних месенджерів. Платформа Slack надає широкі можливості для інтеграції сторонніх сервісів і створення прикладних рішень для роботи з довідковими даними.

Проаналізувавши можливості Slack API, зокрема використання Slash-команд, подій та інтерактивних повідомлень, визначено, що для задач отримання погодної інформації найбільш доцільним є застосування Slash-команд, які забезпечують простоту взаємодії користувачів із ботом та швидке отримання результатів.

Розглянувши зовнішні погодні сервіси, такі як OpenWeatherMap та WeatherAPI, встановлено, що інтеграція з метеорологічними API є необхідною умовою реалізації Slack-бота. Застосування таких сервісів дозволяє отримувати актуальні погодні дані у зручному форматі та забезпечує гнучкість у виборі джерел інформації.

Проаналізувавши підходи до проєктування архітектури Slack-ботів, визначено доцільність використання клієнт-серверної та багаторівневої архітектури, яка забезпечує структурованість, масштабованість і можливість подальшого розвитку функціоналу. Розглянуто основні компоненти системи та їхню взаємодію під час обробки користувацьких запитів.

Проаналізувавши існуючі рішення для отримання погодних даних у Slack, виявлено, що готові боти часто мають обмежений функціонал або потребують використання платних сервісів. У зв'язку з цим власна розробка

дозволяє краще адаптувати бот до потреб користувачів і забезпечити необхідну гнучкість.

Таким чином, здійснивши огляд і аналіз предметної області, сформовано основу для подальшої реалізації Slack-бота з отримання погодних даних. Отримані висновки дозволяють обґрунтувати вибір архітектурних рішень і підходів до інтеграції зовнішніх сервісів, що буде використано в наступних розділах роботи.

РОЗДІЛ 2 ВИБІР ТЕХНОЛОГІЧНОГО СТЕКУ РОЗРОБКИ ДОДАТКУ

2.1. Вибір інструментів та технологій

Під час розробки Slack-бота для отримання погодних даних важливим етапом є вибір інструментів та технологій, які забезпечують надійну інтеграцію з платформою Slack, ефективну обробку користувацьких запитів та стабільну взаємодію з зовнішніми погодними сервісами. Обґрунтований вибір технологічного стеку дозволяє спростити процес розробки, підвищити продуктивність системи та забезпечити можливість подальшого розширення функціоналу.

Основною платформою для реалізації чат-бота обрано корпоративний месенджер Slack, який широко використовується для командної комунікації та підтримує створення власних інтегрованих додатків. Slack надає розвинений програмний інтерфейс Slack API, що дозволяє обробляти події, команди користувачів та надсилати повідомлення у канали або приватні чати. Для взаємодії з користувачами в межах погодного бота доцільним є використання Slash Commands, які забезпечують швидкий доступ до функціоналу шляхом введення коротких текстових команд.

Вибір технологічного стеку є одним із ключових етапів під час створення програмного забезпечення, оскільки саме він визначає продуктивність, масштабованість та зручність подальшої підтримки розробленого рішення. Для реалізації Slack-бота, який надає користувачам актуальні погодні дані, було обрано мову програмування JavaScript та платформу Node.js, а також низку спеціалізованих бібліотек, що забезпечують взаємодію зі Slack API та сервісами отримання погодної інформації. Такий вибір зумовлений гнучкістю технологій, широкою підтримкою спільноти та ефективністю роботи з мережевими запитами в реальному часі [16].

JavaScript, у поєднанні з Node.js, є одним із найпопулярніших рішень у сфері розробки сучасних інтеграційних сервісів і ботів. Node.js дозволяє ефективно працювати з асинхронними операціями, що має ключове значення для обробки HTTP-запитів до зовнішніх API та взаємодії з платформою Slack. Простота синтаксису JavaScript та велика екосистема бібліотек значно прискорюють процес розробки, роблячи його більш зрозумілим і структурованим навіть для команд з різним рівнем досвіду [17].

Для комунікації зі Slack було обрано бібліотеку @slack/bolt, яка надає простий та гнучкий програмний інтерфейс для створення інтерактивних застосунків. Bolt спрощує обробку команд, подій і взаємодій у робочих просторах Slack, дозволяючи легко реалізувати як базову, так і розширену функціональність. Завдяки модульності бібліотеки, розробник може поступово додавати нові можливості, такі як інтерактивні блоки, кнопки, повідомлення з форматуванням та автоматизовані сповіщення [18]. Ключовим інструментом для розробки Slack-бота є фреймворк Bolt for JavaScript, який є офіційно рекомендованим рішенням від Slack. Bolt значно спрощує процес створення ботів, надаючи готові механізми для обробки подій, команд та інтерактивних елементів. Використання Bolt дозволяє зосередитися на бізнес-логіці застосунку, зменшуючи кількість допоміжного коду для роботи зі Slack API. Крім того, фреймворк підтримує інтеграцію з вебсервером та легко масштабується для складніших сценаріїв використання.

Для отримання погодних даних використовується зовнішнє API, наприклад, OpenWeatherMap або WeatherAPI, які забезпечують доступ до актуальної та історичної інформації про погоду, включаючи температуру, атмосферний тиск, швидкість вітру та прогноз на декілька днів. Ці API мають просту структуру запитів, широкий функціонал та потужну документацію, що дозволяє легко інтегрувати їх у Slack-бота й забезпечити швидке отримання точних погодних даних [19].

OpenWeather API надає можливість отримувати актуальні, прогнозні та історичні погодні дані для різних географічних регіонів у режимі реального часу.

OpenWeather API є хмарним сервісом, який працює за принципом клієнт-серверної взаємодії та використовує протокол HTTP для обміну даними. Доступ до функціоналу сервісу здійснюється за допомогою унікального API-ключа, який ідентифікує користувача та дозволяє контролювати кількість запитів. Такий підхід забезпечує безпеку доступу та дозволяє реалізувати обмеження на використання сервісу відповідно до умов ліцензування.

Однією з основних переваг OpenWeather API є широкий набір доступних даних. Сервіс надає інформацію про поточні погодні умови, зокрема температуру повітря, відчутну температуру, атмосферний тиск, вологість, швидкість і напрям вітру, а також опис погодних умов. Крім того, OpenWeather API підтримує отримання короткострокових та довгострокових прогнозів, що дозволяє розширити функціональність Slack-бота та надавати користувачам не лише поточну, а й прогнозну інформацію.

Запити до OpenWeather API можуть формуватися за різними параметрами, зокрема за назвою населеного пункту, географічними координатами або ідентифікатором міста. Така гнучкість є важливою перевагою, оскільки дозволяє обробляти різні типи користувацьких запитів. У контексті Slack-бота це дає змогу реалізувати зручний інтерфейс взаємодії, де користувач може отримати погоду, вказавши лише назву міста.

Відповіді OpenWeather API повертаються у форматі JSON, який є стандартним для вебзастосунків та легко обробляється в середовищі JavaScript. Формат JSON дозволяє структуровано представити погодні дані, що спрощує їх подальше форматування та відображення у повідомленнях Slack. Завдяки цьому серверна частина бота може швидко обробляти отримані дані та формувати зрозумілу відповідь для користувача.

Важливою характеристикою OpenWeather API є наявність безкоштовного тарифного плану, який дозволяє виконувати обмежену кількість запитів за одиницю часу. Це робить сервіс зручним для навчальних, дослідницьких та дипломних проєктів. Водночас OpenWeather API підтримує масштабування та перехід на платні плани у разі збільшення навантаження або потреби в розширеному функціоналі.

Окрему увагу при роботі з OpenWeather API необхідно приділяти обробці помилок та обмежень сервісу. До таких обмежень належать перевищення ліміту запитів, некоректні параметри запиту або тимчасова недоступність сервісу. У межах Slack-бота ці ситуації повинні оброблятися на серверному рівні з метою інформування користувача про виникнення проблеми та забезпечення стабільної роботи системи.

Таким чином, використання OpenWeather API є обґрунтованим вибором для реалізації Slack-бота з отримання погодних даних. Даний сервіс поєднує в собі доступність, функціональність та зручність інтеграції, що дозволяє створити ефективний та надійний інструмент для автоматизованого отримання метеорологічної інформації в середовищі Slack.

Одним із ключових елементів функціонування Slack-бота для отримання погодних даних є взаємодія з зовнішніми метеорологічними сервісами через мережу Інтернет. Для обміну даними між серверною частиною бота та погодним API використовується протокол HTTP, який є стандартом для передачі гіпертекстових даних у вебсередовищі. HTTP-протокол забезпечує уніфікований механізм надсилання запитів та отримання відповідей між клієнтом і сервером, що робить його універсальним інструментом для інтеграції різних інформаційних систем.

У межах Slack-бота HTTP використовується для надсилання запитів до погодного сервісу з метою отримання актуальної метеорологічної інформації. Найчастіше застосовуються HTTP-запити типу GET, які дозволяють отримувати дані без зміни стану сервера. У параметрах запиту передаються такі дані, як назва населеного пункту, одиниці вимірювання, мова відповіді та

ключ доступу до API. У відповідь сервер погодного сервісу повертає структуровані дані у форматі JSON, що є зручним для подальшої обробки в середовищі JavaScript.

Для реалізації HTTP-запитів у середовищі Node.js доцільно використовувати бібліотеку Axios, яка є популярним інструментом для роботи з асинхронними HTTP-запитами. Axios підтримує проміси та асинхронні функції, що дозволяє реалізувати неблокуючу модель виконання та ефективно обробляти мережеві операції. Це є особливо важливим для чат-ботів, які повинні швидко реагувати на запити користувачів і обробляти декілька звернень одночасно. [36]

Однією з основних переваг бібліотеки Axios є зручний та зрозумілий інтерфейс для формування HTTP-запитів. Axios дозволяє легко передавати параметри запиту, заголовки та конфігураційні налаштування, що спрощує інтеграцію з різними API. Крім того, бібліотека автоматично перетворює відповіді сервера у формат JSON, що зменшує обсяг допоміжного коду та підвищує читабельність програмної логіки. [37]

Важливою функціональною можливістю Axios є підтримка централізованої обробки помилок. У процесі отримання погодних даних можуть виникати різні виняткові ситуації, зокрема недоступність сервера, перевищення ліміту запитів або некоректні параметри запиту. Axios дозволяє відстежувати HTTP-статуси відповідей та реалізовувати механізми обробки помилок, що дає змогу коректно інформувати користувача про проблеми та запобігати аварійному завершенню роботи бота. [38]

Ще однією важливою перевагою використання Axios є можливість налаштування глобальних параметрів HTTP-запитів, зокрема базової URL-адреси, тайм-аутів та заголовків. Такий підхід дозволяє централізувати конфігурацію доступу до погодного сервісу та підвищити зручність супроводу програмного забезпечення. У контексті Slack-бота це спрощує зміну погодного API або параметрів доступу без необхідності внесення значних змін у код. [39]

У разі потреби зберігання історії запитів або користувацьких налаштувань застосунок може бути доповнений базою даних, наприклад, MongoDB або PostgreSQL. Використання бази даних дозволяє реалізувати розширені функції: персоналізовані сповіщення, вибір улюблених локацій, налаштування періодичного надсилання прогнозів або аналіз погодної статистики [20].

У процесі розробки Slack-бота важливим етапом є налагодження та тестування взаємодії між платформою Slack і серверною частиною застосунку. Оскільки Slack використовує механізм вебхуків та HTTP-запитів для надсилання подій і команд, сервер бота повинен бути доступним з мережі Інтернет. На етапі локальної розробки це може становити певну проблему, оскільки локальний сервер зазвичай недоступний ззовні. Для розв'язання цієї задачі доцільно використовувати сервіс ngrok.[29]

Ngrok - це інструмент для створення захищених тунелів між локальним сервером розробника та глобальною мережею Інтернет. Сервіс дозволяє тимчасово опублікувати локальний вебсервер, надавши йому публічну URL-адресу, через яку зовнішні сервіси можуть надсилати HTTP-запити. У контексті розробки Slack-бота ngrok використовується для приймання запитів від Slack API, зокрема подій, Slash Commands та інтерактивних повідомлень, без необхідності негайного розгортання застосунку на хмарному сервері.

Основною перевагою використання ngrok є спрощення процесу тестування та налагодження. Завдяки ngrok розробник може запускати серверну частину бота локально, одночасно забезпечуючи її доступність для Slack. Це дозволяє оперативно вносити зміни в код, тестувати обробку подій та перевіряти коректність відповідей бота без складних процедур розгортання. Такий підхід значно скорочує час розробки та підвищує ефективність роботи. [30]

Ngrok підтримує роботу з HTTP та HTTPS-протоколами, що є важливою вимогою для інтеграції зі Slack, оскільки платформа вимагає захищене з'єднання для передавання даних. Під час запуску ngrok автоматично створює

захищений HTTPS-тунель, що дозволяє відповідати вимогам безпеки без додаткового налаштування SSL-сертифікатів. Це робить ngrok зручним інструментом для використання в навчальних та експериментальних проєктах. [31]

Ще однією важливою можливістю ngrok є моніторинг HTTP-запитів. Сервіс надає вебінтерфейс, у якому можна переглядати всі вхідні та вихідні запити, їх заголовки та тіла. Це дозволяє детально аналізувати взаємодію Slack з серверною частиною бота, виявляти помилки в запитах та покращувати логіку обробки подій. Такий функціонал є особливо корисним на етапі налагодження та тестування.[32]

Використання ngrok також сприяє гнучкості процесу розробки. Оскільки сервіс не прив'язаний до конкретної платформи або мови програмування, він може використовуватися незалежно від обраного технологічного стеку. У межах даного проєкту ngrok інтегрується з сервером Slack-бота, реалізованим на Node.js з використанням фреймворку Bolt for JavaScript, що забезпечує простоту та зручність налаштування.[33]

Водночас слід зазначити, що ngrok є інструментом переважно для етапу розробки та тестування. Безкоштовна версія сервісу має певні обмеження, зокрема тимчасовий характер URL-адреси та обмежену кількість з'єднань. Тому на етапі промислового впровадження Slack-бота доцільно використовувати повноцінні хмарні платформи для розгортання застосунку. Проте для навчальних і дипломних проєктів можливостей ngrok є цілком достатньо.[34]

Таким чином, використання сервісу ngrok є ефективним рішенням для забезпечення зовнішнього доступу до локального сервера Slack-бота на етапі розробки. Ngrok значно спрощує процес тестування, забезпечує відповідність вимогам безпеки та дозволяє швидко налагодити взаємодію між Slack та серверною частиною застосунку.[35]

Важливою перевагою обраного стеку є його масштабованість і можливість розгортання на різних платформах - від локального середовища до хмарних сервісів, таких як Heroku, Vercel або AWS Lambda. Це забезпечує гнучкість під час вибору середовища виконання та дає змогу оптимізувати витрати на інфраструктуру залежно від навантаження.

Окремо слід зазначити широку підтримку спільноти та доступність документації для всіх обраних технологій. Це значно спрощує процес розробки, дає змогу швидко знаходити рішення для можливих технічних проблем і забезпечує стабільну підтримку проекту в майбутньому.

У підсумку, поєднання Node.js, JavaScript, бібліотеки @slack/bolt та зовнішніх погодних API забезпечує повноцінний набір інструментів для створення надійного, легко масштабованого та інтерактивного Slack-бота. Обраний технологічний стек дає змогу реалізувати всі необхідні функції - від обробки команд користувачів до автоматичного надсилання прогнозів - і гарантує відповідність вимогам сучасних цифрових сервісів.

2.2. Вибір серверної частини, API та інших технологій

Коректний вибір інструментів та технологій є одним із ключових елементів у процесі створення Slack-бота для отримання погодних даних, оскільки саме він визначає стабільність роботи системи, швидкість реакції на запити та можливість подальшого розширення функціонал

У ролі серверної частини було обрано платформу Node.js, оскільки вона оптимально підходить для роботи з асинхронними операціями, які є невід’ємною частиною систем, що активно взаємодіють із мережевими API. Node.js забезпечує обробку великої кількості запитів без суттєвого навантаження на сервер, що є важливою перевагою для ботів, які працюють у режимі реального часу та повинні відповідати на команди практично миттєво. Крім того, велика екосистема npm-пакетів дозволяє розширювати функціональність системи з мінімальними витратами часу та ресурсів.[16]

Для інтеграції зі Slack використовується фреймворк Slack Bolt, який надає простий і структурований інтерфейс для обробки подій, команд, інтерактивних блоків і повідомлень. Цей фреймворк значно знижує складність взаємодії зі Slack API, дозволяючи зосередитися на логіці бота та обробці даних, а не на низькорівневих мережевих операціях. Bolt також підтримує механізми підписок на події й забезпечує автоматичну маршрутизацію обробників команд.

Для отримання погодних даних було обрано зовнішній API - такі сервіси, як OpenWeatherMap, WeatherAPI або Tomorrow.io. Вони пропонують стандартний REST-інтерфейс для доступу до температури, вологості, швидкості вітру, атмосферного тиску, опису погодних умов та прогнозів на декілька днів. Вибір конкретного постачальника може залежати від необхідної точності, доступності безкоштовного тарифу чи потреби в додаткових параметрах, таких як мапи опадів або історичні дані. Інтеграція з цими API дозволяє отримувати інформацію в режимі реального часу або за розкладом, що дає можливість реалізувати автоматизовані сповіщення для користувачів Slack.

Для розгортання серверної частини можуть використовуватися як традиційні VPS-сервери, так і безсерверні рішення - зокрема, Heroku, Vercel, Railway або AWS Lambda. Вони дозволяють автоматизувати деплой, масштабування та моніторинг, що робить підтримку бота значно простішою. Використання безсерверної архітектури є особливо вигідним для ботів, що працюють подійно та не потребують постійного навантаження на сервер.

У сукупності вибрані технології дозволяють створити надійного, масштабованого й функціонального Slack-бота. Завдяки використанню Node.js забезпечується швидка обробка запитів, Slack Bolt істотно спрощує інтеграцію з корпоративною платформою Slack а зовнішні погодні API надають точні дані. Такий технологічний стек робить систему гнучкою, адаптованою до змін і здатною масштабуватися відповідно до потреб користувачів.

2.3. Інструменти для отримання та обробки погодних даних у Slack-боті

Отримання та обробка погодних даних є ключовим етапом роботи Slack-бота, який забезпечує користувачів актуальною інформацією про погодні умови в будь-якій локації. На відміну від класичного веб-парсингу, де необхідно завантажувати та аналізувати HTML-код, Slack-бот використовує спеціалізовані зовнішні API, що забезпечують структуровані та стандартизовані погодні дані у форматі JSON. Такий підхід забезпечує надійність, високу швидкість обробки інформації та зменшує ризик виникнення помилок, пов'язаних зі зміною структури веб-сторінок.

Робота з API включає отримання поточних погодних даних, прогнозів, історичних температур, рівня опадів, швидкості вітру та інших метеорологічних характеристик. Для цього використовуються такі сервіси, як OpenWeatherMap, WeatherAPI або Tomorrow.io. Вони дозволяють інтегрувати дані без потреби в парсингу HTML та забезпечують стабільний доступ до інформації через REST-архітектуру [21].

Slack-бот, створений у межах цього проєкту, також використовує модульну систему обробки подій, що дозволяє реагувати на Slash-команди, ключові слова та інтерактивні кнопки. Для цього застосовується фреймворк Slack Bolt, який спрощує комунікацію між сервером та Slack API, а також забезпечує зручний інтерфейс для побудови логіки бота [22].

Основні задачі, які вирішує система отримання погодних даних

У контексті Slack-бота функціонал отримання та обробки прогнозів погоди виконує низку важливих завдань:

- Отримання реальних погодних даних.

Бот у режимі реального часу робить запити до API, отримуючи такі параметри:

1. температуру (фактичну і відчутну),
2. вологість,

3. швидкість і напрямок вітру,
4. атмосферний тиск,
5. рівень хмарності,
6. погодні описи (ясно, дощ, гроза, сніг тощо),
7. іконки погодних станів для візуальної подачі.

- Надання прогнозу на декілька днів.

API дає змогу отримати прогноз на:

1. 24 години,
2. 3, 5 або 7 днів (залежно від тарифу),
3. погодинні значення температури.

Це дозволяє користувачам планувати свою діяльність, отримуючи зведений прогноз у вигляді Slack-повідомлення.

- Автоматичні сповіщення.

Slack-бот може:

1. щодня надсилати прогноз у вибраний канал,
2. інформувати про критичні погодні явища (сильний вітер, інтенсивний дощ),
3. працювати за CRON-розкладом.

Для цього використовується планувальник node-cron, який дозволяє задавати автоматичні періодичні завдання [25].

- Збереження даних користувачів.

Для зручності використовується база даних (MongoDB або PostgreSQL), де зберігаються:

1. обрані локації,
2. останні виконанні команди,
3. налаштування автоматичних сповіщень,
4. історія взаємодій.

Це підвищує персоналізацію бота та робить його більш корисним для командної роботи.

У цьому проекті використовуються сучасні інструменти для інтеграції з метеорологічними сервісами, що забезпечують структуровані дані у форматі JSON.

OpenWeatherMap

OpenWeatherMap - один з найпопулярніших API для отримання погодної інформації. Він надає доступ до широкого набору даних, включаючи:

1. координати,
2. температуру,
3. швидкість вітру,
4. погодні описи,
5. прогнози.

Приклад запиту:

```
const axios = require("axios");
async function getWeather(city) {
   const apiKey = process.env.WEATHER_API_KEY;
const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&lang=ua&appid=${apiKey}`;
   const response = await axios.get(url);
   return response.data;
}
```

OpenWeatherMap також підтримує прогноз на 5 днів, погодинні дані, історичні параметри та інші функції [23].

WeatherAPI

WeatherAPI пропонує більш точні прогнози, а також дані про:

1. ультрафіолетовий індекс,
2. астрономічні події (фази місяця, часи сходу й заходу),
3. якість повітря.

Це може розширити функціонал бота, роблячи його ще більш інформативним [24].

Slack-бот створюється на основі офіційного SDK - Slack Bolt. Він підтримує:

- Slash-команди (напр.: /weather Kyiv)
- Обробку подій (message, app_mention)
- Інтерактивні кнопки та блоки
- Роботу з Web API (chat.postMessage, views.open)

Базовий приклад Slack-бота:

```
const { App } = require("@slack/bolt");
const axios = require("axios");
const app = new App({
  token: process.env.SLACK_BOT_TOKEN,
  signingSecret: process.env.SLACK_SIGNING_SECRET
});

app.command("/weather", async ({ command, ack, say }) => {
    await ack();
    const city = command.text.trim();
    const weather = await getWeather(city);
    await say(`Погода в *${city}*: ${weather.main.temp}°C, $
{weather.weather[0].description}`);
});

(async () => {
  await app.start(3000);
  console.log("Slack bot is running");
})();
```

Планування та автоматичні сповіщення

Slack-бот може надсилати повідомлення за розкладом. Використовується бібліотека node-cron:

```
const cron = require("node-cron");
```

```
cron.schedule("0 8 * * *", async () => {
  const users = await UserConfig.find({ notificationsEnabled: true });
  for (const user of users) {
    const weather = await getWeather(user.defaultCity);
    await app.client.chat.postMessage({
      channel: user.userId,
      text: `Щоденний прогноз: ${weather.main.temp}°C`
    });
  }
});
```

Аналіз платформи Slack показав доцільність її використання як середовища для реалізації чат-бота, оскільки вона надає розвинений програмний інтерфейс, підтримує Slash-команди, події та інтерактивні елементи, що дозволяє створювати зручні інструменти для командної роботи. Застосування фреймворку Slack Bolt for JavaScript значно спрощує процес взаємодії зі Slack API, зменшуючи обсяг допоміжного коду та дозволяючи зосередитися на реалізації бізнес-логіки застосунку.

Вибір мови програмування JavaScript та платформи Node.js є обґрунтованим з огляду на їхню підтримку асинхронної обробки запитів, що є критично важливим для роботи з зовнішніми API у режимі реального часу. Використання бібліотеки Axios забезпечує зручну реалізацію HTTP-запитів, обробку відповідей у форматі JSON та централізовану роботу з помилками, що підвищує стабільність і надійність функціонування бота.

Для отримання погодних даних було проаналізовано можливості зовнішніх метеорологічних API, зокрема OpenWeatherMap та WeatherAPI. Їх використання дозволяє отримувати актуальні, прогнозні та додаткові метеорологічні параметри з мінімальними витратами на інтеграцію. Формат даних JSON та гнучкість параметрів запитів забезпечують зручну адаптацію сервісів до потреб користувачів Slack-бота.

Окрему увагу приділено інструментам налагодження та тестування, зокрема сервісу ngrok, який дозволяє забезпечити зовнішній доступ до локального сервера на етапі розробки. Це значно спрощує процес тестування інтеграції зі Slack API та прискорює цикл розробки застосунку.

Таким чином, обраний технологічний стек, що включає Node.js, JavaScript, фреймворк Slack Bolt, бібліотеку Axios та зовнішні погодні API, забезпечує надійну основу для створення функціонального, масштабованого та зручного у використанні Slack-бота. Використані інструменти є доцільними для реалізації поставлених завдань і відповідають сучасним вимогам до розробки інтеграційних програмних рішень.

РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОДЕЛЮВАННЯ ВИТРАТ В ПРИВАТНІЙ КОМПАНІЇ

3.1. Технічне завдання на розробку Slack-боту для отримання даних про погоду

3.1.1. Загальні положення

3.1.1.1. Найменування проекту:

Розробка та розгортання Slack-бота для отримання даних про погоду

3.1.1.2. Підстави для розробки:

Це технічне завдання розроблено на основі потреби своєчасної актуалізації погодних даних у корпоративній середі Slack

3.1.1.3. Замовник:

[Назва компанії або організації, що замовляє проект].

3.1.1.4. Розробник:

[Назва компанії або групи розробників, що будуть займатися проектом].

3.1.1.5. Цільове призначення:

Програмний додаток призначений для швидкого доступу до актуальних погодних даних в корпоративній мережі Slack

3.1.1.6. Етапи розробки:

Розробка технічного завдання.

Планування.

Проектування архітектури системи.

Розробка програмного забезпечення.

Тестування, верифікація та валідація.

Впровадження та навчання користувачів (Delivery & UAT).

Супровід та технічна підтримка.

3.1.2. Призначення розробки

3.1.2.1. Мета створення додатку:

Створення Slack-боту для автоматизованого отримання, обробки та надання погодних даних користувачам безпосередньо у робочих каналах Slack. Метою розробки є забезпечення швидкого доступу до точних метеорологічних показників, підвищення ефективності командної роботи та спрощення процесу моніторингу погодних умов у реальному часі.

3.1.2.2. Основні задачі:

- Отримання поточних погодних даних для заданої локації.
- Надання прогнозу погоди на кілька днів.
- Надсилання автоматичних сповіщень у Slack за розкладом.
- Обробка Slash-команд і взаємодія з користувачами через інтерактивні компоненти Slack.
- Інтеграція з одним або кількома зовнішніми погодними API.

3.1.3. Функціональні вимоги

3.13.1. Основні функції додатку:

- Отримання поточної погоди:
Бот повинен забезпечувати отримання даних про температуру, вологість, вітер, атмосферний тиск, видимість і погодний стан.
- Прогноз погоди:
Надання прогнозу на 24 години та на 3–7 днів, залежно від можливостей обраного API.
- Інтерактивні Slack-команди:
Підтримка Slash-команд типу /weather, /forecast, а також обробка кнопок та повідомлень у блоках.
- Автоматичні сповіщення:
Можливість надсилати щоденні погодні зведення або попередження про негоду у вказаний час.

- Збереження налаштувань користувачів:

1. вибір міста за замовчуванням;
2. вибір мови виведення;
3. налаштування часу сповіщень.

3.2. Додаткові функції:

- Мультиплатформенність:

Бот має коректно працювати на локальному сервері, хмарному хостингу або в Docker-контейнері.

- Підтримка декількох API:

Можливість перемикання між OpenWeatherMap, WeatherAPI та Tomorrow.io.

- Підтримка групових та приватних чатів.

3.3. Технічні вимоги

3.3.1. Мова програмування

JavaScript/TypeScript (Node.js).

3.3.2. Фреймворк для серверної частини

Slack Bolt Framework (Node.js).

3.3.3. База даних

MongoDB (для динамічних JSON-подібних структур).

3.3.4. Інструменти для отримання погодних даних

Axios або Fetch для HTTP-запитів до API. Підтримувані погодні API:

- OpenWeatherMap
- WeatherAPI
- Tomorrow.io

3.3.5. API

RESTful API для зовнішніх сервісів та внутрішніх мікросервісів (за потреби).

3.3.6. Інтерфейс користувача

Slack Block Kit: інтерактивні повідомлення, кнопки, меню, картки.

3.3.7. Система контролю версій

Git.

3.3.8. Система контейнеризації

Docker (для простого розгортання та масштабування).

3.4. Вимоги до безпеки

3.4.1. Безпека даних

- Використання HTTPS з боку API та вебхуків Slack.
- Шифрування токенів і ключів у змінних середовища.
- Верифікація підписів Slack (Slack Signing Secret).

3.4.2. Аутентифікація та авторизація

- Обмеження доступу через OAuth2 Slack.
- Контроль дозволів додатку (scopes).

3.5. Вимоги до продуктивності

3.5.1. Швидкість обробки

- Час відповіді на команду - не більше 1–2 секунд.
- Підтримка одночасної обробки декількох запитів.
- Оптимізація роботи через кешування API-даних.

3.5.2. Масштабованість

- Можливість горизонтального масштабування бота через кілька контейнерів.
- Підтримка збільшення обсягів запитів зі Slack-робочих просторів.

3.6. Тестування та приймання

3.6.1. Типи тестування

Функціональне тестування (модульне, інтеграційне, E2E тестування)

Тестування продуктивності.

Тестування безпеки.

3.6.2. Критерії приймання

- коректно працює зі Slack-командами;
- стабільно отримує погодні дані;
- підтримує надсилання автоматичних сповіщень;
- пройшов усі зазначені етапи тестування;
- документовано відповідно до технічних вимог.

3.7. Документація

3.7.1. Користувацька документація

- Опис встановлення бота у робочий простір Slack.
- Інструкції з використання команд та налаштування сповіщень.

3.7.2. Технічна документація

Документація для розробників, що включає опис архітектури програми, інструкції з налаштування та розгортання.

3.8. Етапи виконання робіт та графік

3.8.1. Етапи

Аналіз вимог та проектування - [вказати строки].

Розробка програмного забезпечення - [вказати строки].

Тестування - [вказати строки].

Впровадження - [вказати строки].

3.8.2. Графік виконання

Зазначити конкретні дати початку та завершення кожного етапу.

3.9. Вимоги до технічної підтримки та супроводу

3.9.1. Технічна підтримка

Надання технічної підтримки на період не менше 3 років після впровадження Slack-боту.

3.9.2. Оновлення

- Регулярне оновлення для врахування змін у Slack API та зовнішніх погодних API.
- Розширення функціоналу відповідно до потреб користувачів.

3.2. Реалізація основних модулів

У цьому розділі розглядається реалізація ключових модулів проєкту Slack-бота, який надає користувачам актуальну інформацію про погоду. Система складається з декількох основних компонентів: інтеграції зі Slack API, серверної частини для обробки команд та запитів, модулів взаємодії з погодними сервісами та модуля форматування й відправлення відповідей користувачу. Далі подано детальний опис структури та функціоналу кожної частини системи.

Основні файли, що відповідають за роботу бота, включають:

- main.js – головний серверний файл, який обробляє взаємодію зі Slack та виклики погодного API.
- weatherService.js – модуль для отримання погодних даних з зовнішнього API.
- slackController.js – логічний модуль, який обробляє команди користувачів.
- formatter.js – модуль форматування повідомлень для Slack.
- config.json – файл налаштувань, де містяться ключі API, токени Slack та параметри сервера.

У Slack інтеграція реалізована через Slash-команди та Events API, що дозволяє користувачам викликати бота за допомогою команд типу:

- /weather <місто>
- /forecast <місто>

Користувач може:

- отримати поточну погоду;
- переглянути прогноз на 2 дні;
- дізнатися точну температуру, швидкість вітру, вологість та інші атмосферні параметри;

Усі відповіді відображаються у вигляді структурованих блоків Slack Block Kit, що робить взаємодію з ботом зручною та візуально зрозумілою.

Файл main.js реалізує серверну частину, яка:

- приймає запити зі Slack;
- виконує валідацію команд;
- викликає модулі для отримання погоди;
- формує відповіді і надсилає їх назад у Slack.

Основні маршрути включають:

app.command('/slack/commands'):

Це основний endpoint, який приймає Slash-команди. Він розпізнає текст команди, визначає тип запиту (поточна погода, прогноз, допомога), а також місто, яке ввів користувач.

Після отримання команд бот виконує такі дії:

- відправляє запит до weatherService.js;
- отримує дані з погодного API;
- формує структуровану відповідь у форматі блоків Slack;
- повертає її користувачу.

Функція команди (/weather, /forecast) також містить логіку взаємодії з API погоди. У ньому реалізовано:

- формування запиту до зовнішнього погодного API (OpenWeatherMap, WeatherAPI або інший);
- обробка JSON-відповіді;
- перетворення даних у зручний формат;
- виявлення потенційних помилок (місто не знайдено, неправильний ключ API, відсутність даних).

Основні дії модуля:

- отримання поточної погоди;
- отримання прогнозу;
- конвертація температури з Кельвінів у Цельсії (якщо потрібно);
- визначення погодних умов за кодами (сонячно, хмарно, дощ, туман тощо).

Також у команді передбачувано форматування та валідація:

- створення блоків Slack Block Kit;
- оформлення заголовків, списків, іконок погоди;
- визначення яку команду викликав користувач;
- перевіряє коректність введених даних (місто, параметри);
- повертає результат у Slack.

Бот також може мати можливість виконувати автоматичний аналіз та надсилати регулярні сповіщення:

- щоденний прогноз о 7:00;
- попередження про різке погіршення погоди;
- сповіщення про дощ чи мороз;
- персональні налаштування користувача щодо міста та часу надсилання прогнозу.

База даних (за потреби, наприклад PostgreSQL або SQLite) зберігає:

- історію запитів користувачів;
- налаштування повідомлень;
- вибране місто користувача;
- лог помилок API.

Структура таблиць може містити поля:

- user_id
- default_city
- notify_time
- last_forecast
- errors_log

Для візуалізації прогнозу бот може використовувати:

- іконки погодних умов;
- температурні індикатори;
- короткі поради (теплий одяг, дощовик, висока вологість).

Функціонал забезпечує:

- швидку відповідь на запити;
- стабільну інтеграцію зі Slack;
- правильну обробку команд;
- надання точних погодних даних у будь-який час.

Серверна логіка бота забезпечує повний цикл обробки запитів: від отримання текстової команди від користувача до формування структурованої погодної відповіді. Використання Node.js та Slack API дозволяє легко розширювати функціонал і додавати нові можливості.

3.3. Опис роботи програмного засобу

Створений у межах даного проекту програмний засіб являє собою інтелектуальний чат-бот для платформи Slack, призначений для отримання актуальної інформації про поточні погодні умови та короткостроковий прогноз на основі зовнішнього API-сервісу OpenWeather. Основна ідея полягає в тому, щоб інтегрувати цілодобовий, простий у використанні інструмент взаємодії, доступний безпосередньо з робочих чатів, каналів або особистих повідомлень Slack. Програмний засіб реалізований на платформі Node.js з використанням фреймворку Slack Bolt, що забезпечує просту інтеграцію, стабільність роботи та гнучкість у розширенні функціоналу.

У цьому підрозділі описано повний життєвий цикл роботи бота, починаючи з моменту виклику користувачем Slash-команди й завершуючи формуванням структурованих відповідей, включно з обробкою помилок, взаємодією з API OpenWeather, тунелюванням ngrok та логікою обробки команд.

3.11.1 Загальна концепція та призначення системи

Розроблений чат-бот виконує функції швидкого доступу до даних про погоду без необхідності переходу на сторонні сайти або мобільні додатки. Достатньо ввести одну команду у вікні Slack, і система у відповідь надає:

- поточну температуру у вказаному місті;
- текстовий опис погодних умов українською мовою;
- швидкість вітру;
- рівень вологості;
- інформативне емодзі, що допомагає швидко інтерпретувати стан погоди.

Також реалізовано окрему команду для отримання прогнозу погоди на сьогодні та завтра, який будується на основі 5-денного погодного прогнозу з інтервалом у три години. На додачу до основних переваг, бот працює однаково як у приватних повідомленнях, так і в публічних каналах Slack, що робить його гнучким інструментом для командної роботи.

Система працює за принципом запиту-відповіді. Slack надсилає запит до серверної частини застосунку, де відбувається обробка даних, формування запитів до OpenWeather API та створення відповіді для користувача. Застосунок працює у режимі постійного прослуховування порту й очікування подій. Для того щоб Slack зміг взаємодіяти із сервером, який запущено локально на комп'ютері розробника, використовується тунель ngrok.

3.11.2 Архітектура та компоненти програмного засобу

Архітектура чат-бота складається з кількох ключових компонентів:

1. Серверний застосунок на Node.js, який виконує роль центрального логічного ядра.
2. Фреймворк Slack Bolt, що забезпечує зручний API для взаємодії зі Slack подіями, Slash-командами й повідомленнями.
3. Зовнішній API-сервіс OpenWeather, що повертає дані про погоду в форматі JSON.

4. Служба тунелювання ngrok, що створює публічний HTTPS-доступ до локально запущеного серверу.

5. Функціональний модуль обробки даних (функції розбору прогнозів, підбору емодзі, форматування відповіді).

6. Модуль обробки помилок, що відповідає за стійкість системи.

Усі елементи працюють у тісній взаємодії. Slack виступає у ролі інтерфейсу користувача, Node.js виконує логічні операції, OpenWeather є джерелом даних, а ngrok забезпечує глобальний доступ для Slack.

Комунікація між компонентами є асинхронною та відбувається через HTTP-запити, що дозволяє системі працювати швидко, не блокуючи основний потік виконання.

3.11.3 Механізм обробки Slash-команд у Slack

Slack забезпечує можливість створення спеціальних текстових команд, які починаються зі знаку слеш ("/"). У створеному програмному засобі використано дві такі команди: /weather та /forecast.

Процес обробки команди відбувається за наступним алгоритмом:

1. Користувач вводить команду у будь-якому каналі або приватному повідомленні.

2. Slack надсилає HTTP-запит до сервера бота за заздалегідь вказаною URL-адресою.

3. Запит містить:

- текст команди;
- назву каналу;
- ідентифікатор користувача;
- час виклику;
- підпис, що дозволяє перевірити автентичність запиту.

4. Застосунок на Node.js приймає запит і миттєво надсилає коротку відповідь-підтвердження (ack), що є обов'язковою вимогою Slack. Без цього бот буде вважатися недоступним, а команда - невдалою.

5. Після цього система виконує основну логіку - звернення до OpenWeather API, обробку даних і формування відповіді.

Важливим аспектом є структура відповіді. Slack дозволяє надсилати як приватні відповіді (видимі лише тому, хто ввів команду), так і публічні (видимі всім учасникам каналу). Це визначається параметром response_type.

3.11.4 Алгоритм роботи команди /weather

Команда /weather надає поточну погоду у місті, яке вказує користувач. Інтерфейс боту після команди показаний на Рис. 3.1:



Рис. 3.1 - виведення команди /weather

Її внутрішня логіка складається з кількох етапів.

1. Отримання тексту після команди. Якщо місто не вказане, користувач отримує повідомлення про необхідність уточнити назву.

2. Формування URL-запиту до OpenWeather API з параметрами:

- q - текст міста, при цьому застосовується encodeURIComponent, щоб уникнути проблем у випадку нестандартних символів;
- units=metric - використання метричної системи;
- lang=ua - українська локалізація описів погоди;
- appid - унікальний ключ доступу до API.

3. Відправлення GET-запиту та отримання JSON-відповіді.
4. Виділення ключових параметрів:
- main.temp - температура,
- weather[0].description - текстовий опис,
- wind.speed - швидкість вітру,
- main.humidity - відносна вологість,
- weather[0].icon - код погодної іконки.

5. Передача коду іконки у функцію getWeatherEmoji, яка визначає відповідний емодзі.
6. Побудова форматованого повідомлення для Slack, що містить:
- емодзі,
- температуру з емодзі-термометром,
- опис погоди,
- вологість,
- вітер.

7. Надсилання повідомлення користувачу.

Команда також передбачає обробку помилок, включно з неправильною назвою міста, відсутністю доступу до API або збоями у мережі.

3.11.5 Алгоритм роботи команди /forecast

Команда /forecast призначена для отримання короткострокового прогнозу, який будується за даними п’ятиденного прогнозу з розділенням по три години. Інтерфейс прогнозу показаний на Рис. 3.2:



Рис 3.2 - виведення команди /forecast

Алгоритм роботи включає такі етапи:

1. Перевірка, чи користувач передав місто як аргумент.
2. Формування запиту до ендпоінту /forecast OpenWeather API.
3. Отримання масиву list, який містить 40 записів прогнозу.
4. Визначення дат "сьогодні" та "завтра" за локальним часом.
5. Фільтрація прогнозів, що відповідають поточній даті та наступному дню.
6. Обчислення середньої температури для кожного дня.
7. Витягування першого опису та піктограми дня як представницького варіанта.
8. Формування структурованого повідомлення, що містить прогноз на два дні.
9. Кінцеве надсилання результату.

Якщо дані не можуть бути отримані або місто вказано некоректно, система повертає повідомлення про помилку

3.11.6 Структура та організація коду

Програмний код має чітку модульну структуру та складається з логічних блоків:

- Імпорт залежностей (dotenv, @slack/bolt, axios).

- Завантаження конфігураційних змінних.
- Створення екземпляра застосунку Slack Bolt.
- Функція обробки емодзі для погодних умов.
- Обробник команди /weather.
- Обробник команди /forecast.
- Запуск сервера.

Структурована організація забезпечує легку підтримку та можливість розширення функціоналу.

3.11.7 Взаємодія з API OpenWeather

OpenWeather API є джерелом ключових даних для роботи чат-бота. Застосунок звертається до двох основних ендпоінтів:

1. /weather - поточні погодні умови.
2. /forecast - прогноз на 5 днів.

Кожен запит формує JSON-структуру, яку система розбирає на окремі параметри. Українська локалізація (lang=ua) робить відповіді зручними для користувача. API-ключ береться зі змінних середовища, що гарантує безпеку зберігання конфіденційних даних.

3.11.8 Робота через ngrok

Оскільки під час розробки сервер працює на локальному комп'ютері, Slack не може напряму надсилати HTTP-запити на локальну адресу. Цю проблему вирішено шляхом використання ngrok.

Ngrok виконує такі функції:

1. Створює захищений HTTPS-тунель.
2. Встановлює зв’язок між глобальною мережею та локальним сервером.
3. Генерує унікальну публічну URL-адресу.
4. Дає змогу Slack надсилати запити у режимі реального часу.

Завдяки цьому розробник може розгортати та тестувати бота без необхідності використання зовнішнього серверного хостингу.

3.11.9 Обробка помилок

Система передбачає багаторівневу обробку помилок:

- некоректна назва міста;
- недоступність OpenWeather API;
- помилки мережі;
- невірний формат відповіді;
- внутрішні винятки Node.js.

У разі виникнення помилки користувач отримує дружнє повідомлення, яке підказує, що саме сталося. Це підвищує стабільність та зручність роботи з ботом.

Здійснивши проєктування та реалізацію програмного засобу у вигляді Slack-бота для отримання та обробки погодних даних, ми визначили його призначення, основні функціональні можливості та технічні вимоги.

Проаналізувавши вимоги до функціоналу Slack-бота, визначено перелік основних і додаткових можливостей, зокрема отримання поточних погодних даних, формування прогнозу, підтримку Slash-команд, автоматичних сповіщень і користувацьких налаштувань. Це дозволило структурувати логіку роботи програмного засобу та визначити типові сценарії його використання.

Розглянувши технічні аспекти реалізації, обґрунтовано вибір мови програмування, фреймворку та інструментів для взаємодії із зовнішніми погодними API. Проаналізувавши архітектуру чат-бота, визначено основні модулі системи, їх функціональне призначення та взаємодію між собою. Застосування Node.js, фреймворку Slack Bolt та сервісу OpenWeather забезпечує асинхронну обробку запитів і стабільну роботу бота.

Реалізувавши обробку Slash-команд /weather та /forecast, описано алгоритми формування запитів до погодного API, обробки відповідей у форматі JSON та подання результатів користувачеві з використанням

інструментів Slack Block Kit. Запропонований підхід дозволяє забезпечити зрозуміле й наочне представлення погодної інформації.

Розглянувши питання безпеки, продуктивності та масштабованості, а також засоби тестування й налагодження, зокрема використання сервісу ngrok, забезпечено коректну взаємодію Slack із серверною частиною під час розробки. Передбачивши обробку типових помилок, підвищено стабільність роботи програмного засобу.

Отже, отримані результати підтверджують можливість використання розробленого програмного засобу для автоматизованого отримання погодної інформації в корпоративному середовищі Slack.

ВИСНОВКИ

У ході виконання даної роботи було розроблено та реалізовано сучасний інтерактивний програмний засіб у вигляді чат-бота для платформи Slack, який забезпечує отримання актуальної інформації про погодні умови та прогноз на найближчі дні. Основною метою проєкту було створення зручного інструменту для швидкого доступу до метеорологічних даних без необхідності використання сторонніх ресурсів, мобільних додатків або веб-сайтів.

Розроблений бот реалізований на основі серверного застосунку Node.js з використанням фреймворку Slack Bolt, що дозволяє інтегрувати команди та події Slack, обробляти запити користувачів та формувати структуровані відповіді. У якості джерела даних обрано OpenWeather API, що гарантує точність і актуальність інформації про температуру, швидкість вітру, рівень вологості та погодні умови. Для забезпечення доступності локально запущеного сервера було використано сервіс ngrok, який створює публічний HTTPS-тунель та дозволяє тестувати та експлуатувати бот у реальному часі.

Одним із ключових елементів реалізації є забезпечення інтерактивності та наочності відповіді користувачу. Для цього була реалізована функція підбору емодзі відповідно до погодних умов, що дозволяє швидко оцінити стан погоди в обраному місті. Команди /weather та /forecast забезпечують повну інформацію про поточні умови та прогноз на сьогодні і завтра, що підвищує практичну цінність застосунку для користувачів.

Розробка включала створення алгоритмів обробки запитів, формування повідомлень, перевірки помилок і валідації введених користувачем даних. Було передбачено стійку обробку некоректних запитів, відсутності аргументів або невірного формату даних, що дозволяє уникнути аварійного завершення роботи та забезпечує стабільність системи. Всі операції виконуються асинхронно, що забезпечує швидке реагування бота та мінімальні затримки у відповіді.

Однією з важливих переваг розробленого програмного засобу є його масштабованість. Бот легко адаптується до нових команд, інтеграцій та розширень функціоналу. Можливе додавання графічних блоків (Blocks), кнопок, меню, а також інтеграція з іншими сервісами та API. Це робить систему гнучкою та придатною для використання в командних робочих просторах будь-якої організації.

Програмний засіб демонструє високий рівень практичності, оскільки дозволяє скоротити час на отримання необхідної інформації, робить процес більш зручним та інтерактивним, а також сприяє підвищенню продуктивності користувачів у Slack. Бот є універсальним рішенням, що поєднує простоту використання, надійність та можливість подальшого розвитку.

Результати виконаної роботи підтверджують, що інтеграція серверного застосунку, Slack API та OpenWeather API дозволяє створювати ефективні, швидкі та зручні інструменти для автоматизації рутинних завдань. Використання сучасних технологій, таких як Node.js, Bolt та ngrok, забезпечує гнучкість, стабільність і безпеку роботи системи. Крім того, бот може використовуватися як навчальний приклад для вивчення інтеграції зовнішніх API з платформами корпоративних комунікацій.

Таким чином, реалізований програмний засіб успішно виконує поставлені завдання, демонструє високу зручність та ефективність у роботі, а також може бути рекомендований до подальшого впровадження та розвитку у корпоративних та навчальних середовищах. Його архітектура та модульна структура дозволяють легко масштабувати систему та інтегрувати нові функції, що робить її перспективною для майбутніх розширень та адаптації до різних сценаріїв використання.

Розробка такого чат-бота також відкриває можливості для подальших досліджень у сфері інтеграції ботів з метеосервісами, використання аналітики прогнозів для планування та автоматизації рішень у бізнес-процесах. Крім того, реалізація на українській мові забезпечує локалізацію та підвищує зручність використання системи для користувачів у регіоні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бределєв І. В., Ковалюк О. О. Розробка персональної системи підтримки прийняття рішень на основі мобільних технологій. Матеріали LIII науково- технічної конференції підрозділів ВНТУ, Вінниця, 20-22 березня 2024 р. Електрон. текст. дані. 2024. URI: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20930.

2. Голоднюк В. О., Паночишин Ю. М. «Інформаційна технологія створення чат-бота для розміщення об'яв для товарів та послуг», в Матеріалах Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», Вінниця, 2024

3. Крупа А.Г. Машинне навчання і штучний інтелект як чинники удосконалення інформаційно-комп'ютерних технологій. Актуальні проблеми сучасної філософії та науки: виклики сьогодення: зб. наук. праць / редкол. М. А. Козловець, Л. В. Горохова, О. В. Чаплінська [та ін.]. Житомир: Видавничий центр ЖДУ імені Івана Франка, 2022. С. 99-102.

4. Саранча Д., Кобилін О. РОЗРОБКА БОТА ПОГОДНОГО ІНФОРМЕРА В TELEGRAM. X Міжнародна науково-технічна конференція «Поліграфічні, мультимедійні та web-технології» (Травень 14–17, 2025). Харків, 2025. с. 112–113.

5. Черес, Б. О., Белзецький, Р. С. Розробка чат-бота для отримання прогнозу погоди з використанням api погодних сервісів. Матеріали LIV науково-технічної конференції підрозділів ВНТУ. (Вінниця, 24-27 березня 2025 р). Вінниця, 2025.

6. Axios API. Axios Docs URL: https://axios-http.com/ru/docs/api_intro (дата звернення 25.11.2025)

7. Axios Docs URL: https://axios-http.com/ru/docs/req_config (дата звернення 02.11.2025)

8. Axios Into Docs. URL: https://axios-http.com/ru/docs/intro (дата звернення 05.11.2025)

9. Axios. URL: https://axios-http.com/ru/docs/example (дата звернення 20.10.2025)

10. Baby, C.J.; Khan, F.A.; Swathi, J.N. Home Automation Using IoT and a Chatbot Using Natural Language Processing. In Proceedings of the 2017 Innovations in Power and Advanced Computing Technologies (i-PACT); April 2017; pp. 1–6.

11. Bengfort, B.; Bilbro, R.; Ojeda, T. Applied Text Analysis with Python: Enabling LanguageAware Data Products with Machine Learning; O'Reilly Media, Inc., 2018; ISBN 978-1-4919-6299-2.

12. Bolt for JavaScript GitHub Repository. URL: https://github.com/slackapi/bolt-js (дата звернення 05.09.2025)

13. Bot Architecture Concepts. URL: https://developer.slack.com/overview/bot-users (дата звернення 30.09.2025)

14. Checking the weather using an automation in Slack: Automating workflows with the Slack platform - Slack Developers. URL: https://slack.dev/workshop/checking-the-weather-automating-in-slack/ (дата звернення 14.11.2025)

15. Forecast Slack Bot. URL: https://forecastslackbot.com/ (дата звернення 12.11.2025)

16. From development to production: ngrok use cases for apps and APIs. URL: https://ngrok.com/use-cases (дата звернення 15.10.2025)

17. Get started with ngrok's API gateway - ngrok documentation. URL: https://ngrok.com/docs/guides/api-gateway/get-started (дата звернення 25.11.2025)

18. Integrating External APIs with Slack Bots. URL: https://developer.okta.com/blog/2021/01/13/build-slack-bot (дата звернення 25.11.2025)

19. JavaScript Info - Modern JavaScript Tutorial. URL: https://javascript.info (дата звернення 20.08.2025)

20. Manaswi, N.K.; Manaswi, N.K.; John, S. Deep Learning with Applications Using Python; Springer, 2018.

21. MongoDB Documentation. URL: https://www.mongodb.com/docs (дата звернення 25.08.2025)

22. Ngrok API ngrok API Overview - ngrok documentation. URL: https://ngrok.com/docs/api (дата звернення 25.08.2025)

23. Ngrok keeps your applications secure with modern TLS standards | ngrok blog. URL: https://ngrok.com/blog/modern-tls-protocols (дата звернення 25.08.2025)

24. Ngrok. Test webhooks in seconds - Trigger and replay with one click. URL: https://ngrok.com/use-cases/webhook-testing (дата звернення 13.09.2025)

25. Ngrok. URL: https://ngrok.com/ (дата звернення 13.092025)

26. Node.js Official Documentation. URL: https://nodejs.org (дата звернення 14.11.2025)

27. Node-cron Scheduler. URL: https://www.npmjs.com/package/node-cron (дата звернення 5.12.2025)

28. OpenWeatherMap API. URL: https://openweathermap.org/api (дата звернення 11.10.2025)

29. OpenWeatherMap One Call API . URL: https://openweathermap.org/api/one-call-api (дата звернення 11.12.2025)

30. PostgreSQL Documentation. URL: https://www.postgresql.org/docs

31. Slack API Documentation. URL: https://api.slack.com/methods/chat.postMessage (дата звернення 20.10.2025)

32. Slack Block Kit Builder Reference. URL: https://docs.slack.dev/reference/block-kit/ (дата звернення 05.09.2025)

33. Slack Bolt Documentation. URL: https://docs.slack.dev/tools/bolt-js/ (дата звернення 25.11.2025)

34. Slack Bolt Framework Guide. URL: https://docs.slack.dev/tools/bolt-js/ (дата звернення 11.09.2025)

35. Slack Bolt Framework. URL: https://slack.dev/bolt (дата звернення 11.09.2025)

36. Slack Developer Tools & Testing Utilities. URL: https://testrigor.com/blog/slack-testing /(дата звернення 11.09.2025)

37. Slack Weather Bot (Postman Template). URL: https://www.postman.com/postman/published-postman-templates/collection/wz16l7o/slack-weather-bot (дата звернення 25.09.2025)

38. Storing User Data in Bots. URL: https://medium.com/swlh/how-to-store-user-data-in-your-slack-bot-2b3a3c9e3e0b (дата звернення 11.11.2025)

39. Tomorrow.io Weather API. URL: https://www.tomorrow.io/

40. Weather Bot by franciskim (Node.js). URL: https://gist.github.com/franciskim/13a52debd47ffb7929f5 (дата звернення 25.09.2025)

41. WeatherAPI Documentation. URL: https://www.weatherapi.com/docs (дата звернення 25.11.2025)

42. What is ngrok? - ngrok documentation. URL: https://ngrok.com/docs/what-is-ngrok (дата звернення 25.11.2025)