МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Комплексна кваліфікаційна робота бакалавра

ІНФОРМАЦІЙНА СИСТЕМА КЕРУВАННЯ ПІЦЕРІЄЮ. ІНФОРМАЦІЙНЕ І ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ЧАТ-БОТУ БЕЗСЕРВЕРНОГО ДОДАТКУ

Здобувач освіти гр. ІН – 82 Анастасія ФЕДОРЧАК

Науковий керівник,
кандидат технічних наук, доцент,
доцент кафедри комп'ютерних наук Ігор ШЕЛЕХОВ

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую
Зав. кафедри Анатолій ДОВБИШ
"_____" _____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-82 спеціальності «122 – Комп'ютерні науки» денної форми навчання Федорчак Анастасії Федорівни.

Тема: «ІНФОРМАЦІЙНА СИСТЕМА КЕРУВАННЯ ПІЦЕРІЄЮ. ІНФОРМАЦІЙНЕ І ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ЧАТ-БОТУ БЕЗСЕРВЕРНОГО ДОДАТКУ»

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) інформаційний огляд; 2) постановка завдання; 3) визначення методів рішення завдання; 4) проектування інформаційної системи; 5) програмна реалізація; 6) аналіз результатів роботи.

Дата видачі завдання «_____»_____ 2022 р.
Керівник роботи _____ Ігор ШЕЛЕХОВ
Завдання прийняла до виконання _____ Анастасія ФЕДОРЧАК

РЕФЕРАТ

Записка: 41 стор., 21 рис., 1 додаток, 11 джерел.

Об’єкт дослідження — процес проєктування програмного забезпечення чат-боту безсерверного додатку.

Мета роботи — розробка Facebook боту для замовлення піци.

Методи дослідження — методи проєктування інформаційних систем, методи обробки природніх мов, хмарні технології

Результати — розроблений Facebook чат-бот для замовлення піци з використанням бази даних та технології обробки природніх мов.

ЗАМОВЛЕННЯ ПІЦИ, FACEBOOK, БОТ, CLAUDIA.JS, NLP,
AWS, JAVASCRIPT

ЗМІСТ

ВСТУП

Ви хочете замовити їжу у місцевому ресторані, для цього вам потрібно: зайти на сайт магазину, знайти те, що ви шукаєте, замовити та очікувати на замовлення. Але що, якщо в цьому магазині є чат-бот? Потрібно було б тільки написати повідомлення ресторану через Telegram або Facebook та сказати їм, що ми хочемо придбати у них. І якщо у вас були сумніви щодо складу, порції, якості, то ви могли б отримати відповіді на свої запитання в одну мить.

Однією з величезних і головних переваг чат-ботів є те, що, на відміну від додатків, їх не треба завантажувати та встановлювати на мобільний телефон або на персональний комп'ютер, як результат вони зовсім не займають зайве місце телефону та ПК, але також не потрібно стежити за останніми оновленнями, як ми звикли це робити. Крім того добре, що в той самий чат можна інтегрувати більше одного бота одночасно. Таким чином, ми можемо уникнути переходу від одного додатка до іншого, що займає деяку кількість нашого часу, залежно від того, що нам потрібно в кожний момент. Ще більшим плюсом є те, що запитувати можна в будь-який час дня, простіша та миттєва взаємодя зі службою підтримки клієнтів. То що ж таке чат-бот?

Чат-бот – це комп’ютерна програма, яка моделює людську розмову за допомогою голосових команд або текстових чатів, або обох одночасно. Чат-бот, скорочення від chatterbot, — це функція штучного інтелекту (AI), яку можна вбудувати та використовувати через будь-яку велику програму обміну повідомленнями. Існує ряд синонімів для чат-бота, включаючи «бот для розмови», «бот», «бот для обміну повідомленнями», «інтерактивний агент» або «штучний об’єкт розмови». [1]

Пристрій використовує штучний інтелект та машинне навчання, щоб можна було автоматично відповідати на ці запитання. З’явище віртуальних помічників спричинив розвиток штучного інтелекту та машинного навчання та як результат цього розвитку, всього за перші кілька років у Facebook Messenger було додано понад 300 тисяч віртуальних співрозмовників. Чат-бот нерідко називають одним із найпередовіших та багатообіцяючих способів

взаємодії людини з машиною. Віртуальний помічник є лише еволюцією системи відповідей на питання за наданими шаблонами, який використовує переваги обробки природної мови (NLP). Але головне питання чи варто використовувати їх у своєму бізнесі, чи вони спростять роботу і допоможуть розвивати бізнес?

Користувачі як серед «business-to-business» (B2B), і у середовищі «business-to-consumer» (B2C) дедалі частіше почали використовувати віртуальних помічників для виконання простих в розумінні і однотипних завдань. Разом з тим немало важливо, що при додаванні чат-ботів знижуються витрати на персонал, краще використовується час співробітників служби гарячої лінії та дозволяється організаціям обслуговувати клієнтів як у нічний час, так і у вихідні, коли робочий персонал недосяжний для роботи. А ось і кілька прикладів відомих технології чат-ботів, які використовують по всьому світу: Google Assistant, Amazon Alexa, а також програми для обміну повідомленнями між людьми, такі як WeChat, Telegram і Facebook Messenger.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Сучасні чат-боти

У наш час вже багато підприємців почали використовувати чат-боти для спрощення роботи та економії грошей, але мало хто з них знає, коли саме почали створювати віртуальних помічників. Перший у світі чат-бот був показаний світові ще до запуску персональних комп'ютерів. Він був розроблений Джозефом Вейценбаумом у 1966 році в лабораторії штучного інтелекту Массачусетського технологічного інституту та отримав назву ELIZA. Здатність бота до спілкування була обмежена, цей помічник порівнював питання з даними зразками і схему вибору відповідей, які він повинен дати на основі даних йому шаблонів. Оскільки тоді не використовувався штучний інтелект, його знання були обмежені, і тому міг обговорювати певні області тем, на які йому дали відповідей.

Наступним чат-ботом був Parry, написаний психіатром Кеннетом Колбі зі Стенфордського університету у спробі змоделювати людину з параноїдальною шизофренією. Потім з'явилася A.L.I.C.E, розроблена в 1995 Річардом Уоллесом. Хоча A.L.I.C.E тричі вигравала приз Лебнера, вона не змогла пройти тест Тюрінга. Тест Тюрінга перевіряє, чи здатна машина мислити розумно, як люди.[2]

Після прориву в цій області вже були запущені різні віртуальні помічники, такі як Siri, Google Assistant, Cortana. То що ж можуть сучасні чат-боти, чого не могли до цього? Віртуальні помічники останніми роками стали надзвичайно популярними завдяки досягненню в галузі машинного навчання, а разом з тим досягнень в обробці природної мови. Сьогоденні віртуальні співрозмовники розумніші і можуть обробляти не тільки письмове повідомлення, а втім голосове повідомлення також. Отже, давайте подивимося на статистику запитів у Google за останніх 5 років:

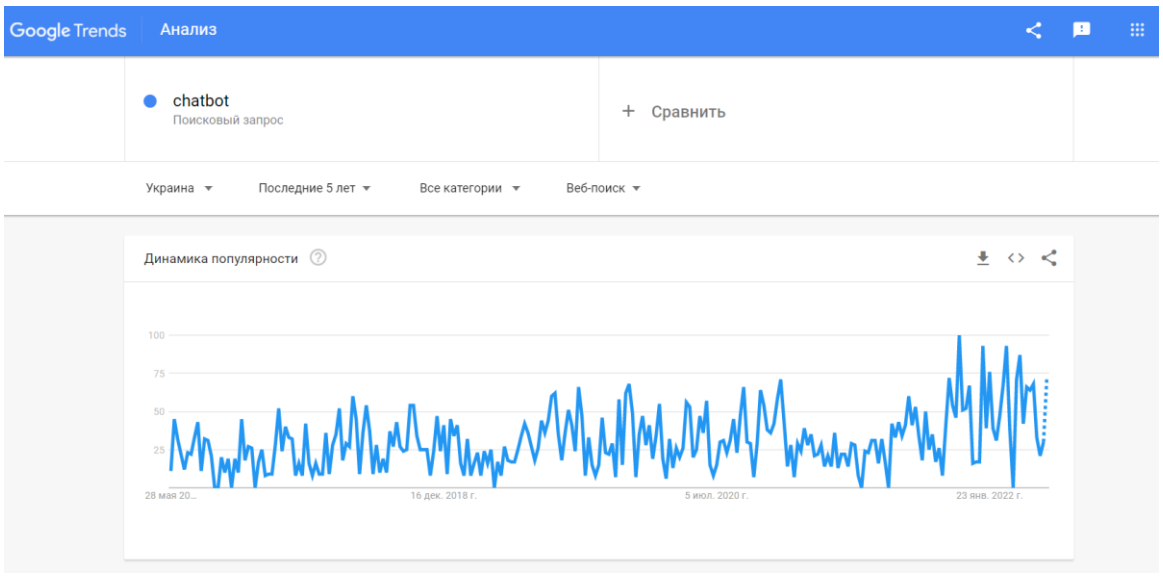


Рисунок 1. 1 – Динаміка популярності чат-ботів за словом «chatbot»[3]

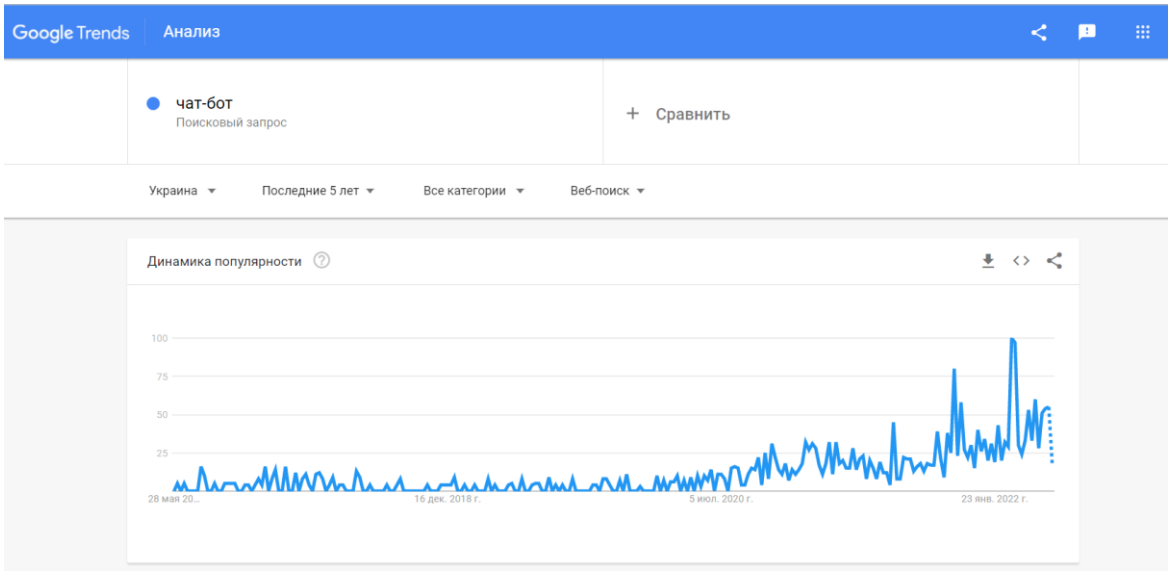


Рисунок 1. 2 – Динаміка популярності чат-ботів за словом «чат-бот»[3]

Як ми можемо побачити з графіків(Рисунок 1. 1, Рисунок 1. 2), динаміка пошуку чат-ботів в Україні росте з кожним роком. Міжнародні технологічні компанії вважають, що використання віртуальних помічників є подальшим колосальним кроком на світовому ринку. Боти зараз мають приблизно 1,5 мільярда користувачів по всьому світу, вони зараз змагаються із соціальними мережами, як-от Facebook, Telegram, Instagram, як основний інтерфейс користувача в Інтернеті. А ось пару доказів, наскільки корисні чат-боти можуть бути. Під час президентських виборів у США у 2016 році було підраховано, що більше однієї п'ятої твітів з використанням основних

виборчих хештегов було створено машинними агентами, які потенційно впливають на результат виборів. Уряди почали вивчати помічників як полегшення голосування під час виборів. Такі компанії, як Taco Bell та Domino's Pizza, використовують віртуальних помічників як агенти бронювання столиків у їхніх закладах у програмах для обміну повідомленнями. Так само є і медичні чат-боти, наприклад, Cardea, дають поради зі здоров'я та списки лікарів, які можуть допомогти.

Зараз можливість використовувати віртуальних співрозмовників можна у всіх сферах, як-от від розваг до покупки речей та будівництва свого персонального бізнесу. За допомогою віртуальних помічників бізнес може масштабуватися і бути активним цілодобово, що є важливою відмінністю від найму персоналу. Покладаючись виключно на найнятий персонал, люди не можуть обслуговувати декілька людей, тільки одну-дві людини. В підсумку, вони дозволяють підприємствам взаємодіяти з величезною кількістю клієнтів у персональному порядку і можуть масштабуватися вгору чи вниз із попитом та потребами ринку та бізнесу. Досить велика кількість досліджень споживачів показують, що чат-боти все частіше стають кращими для кінцевого споживача, методом зв'язку з підприємствами. Віртуальні помічники, які розмовляють з клієнтами через платформи обміну повідомленнями, забезпечують ліпший рівень обслуговування та зручності, які у багатьох випадках перевершують можливості людини.

Додавання чату до відділу обслуговування, реклами або продажу вимагає мінімального програмування та знань у цій галузі. У наш час багато інформації, документації, готових шаблонів та розробок, які можна використати. Однак більшість доступних чат-ботів не задовольняють потреби користувачів через незрозумілі цілі, абсурдні відповіді або некомфортабельність у використанні. І це призводить до наслідку, що клієнт не бажає користуватися віртуальним помічником і витрати зростають, отже, якщо підприємець хоче, щоб чат-бот допомагав бізнесу та приносив користь, розробка нової інтерактивної технології вимагає глибоких знань та розуміння

покупців та користувачів, їх потреб, мети для використання чат-ботів, для того щоб дизайнер міг створити додаток без проблем у використанні.

Немало важливим кроком реалізації чат-бота є й вибір відповідного механізму обробки природної мови (NLP). Якщо людина працює з віртуальним співрозмовником за допомогою голосових повідомлень, очевидно, помічнику потрібний механізм розпізнавання живої мови. Підприємці мають вирішити: хочуть вони структурованого, або неструктурованого спілкування. Віртуальні помічники створені для структурованих розмов, містять багато можливих сценаріїв розмови, що полегшує програмування, але обмежує теми спілкування з ним. У середовищі B2B (business-to-business) чат-боти зазвичай створюються для відповіді на запитання, що часто задаються, або виконання простих повторюваних завдань.

Декілька прикладів відомих чат-ботів для замовлення їжі і не тільки:

1. Ресторан Піца Доміно (Domino's Pizza) по всьому світу та навіть в Україні став відомим завдяки своїй цілодобовій доступності та смачній піці. Він може бронювати столик, оформлювати замовлення, ділитися контентом через розсилки чат-бота: страва дня, акції, новинки меню, фотозвіти.

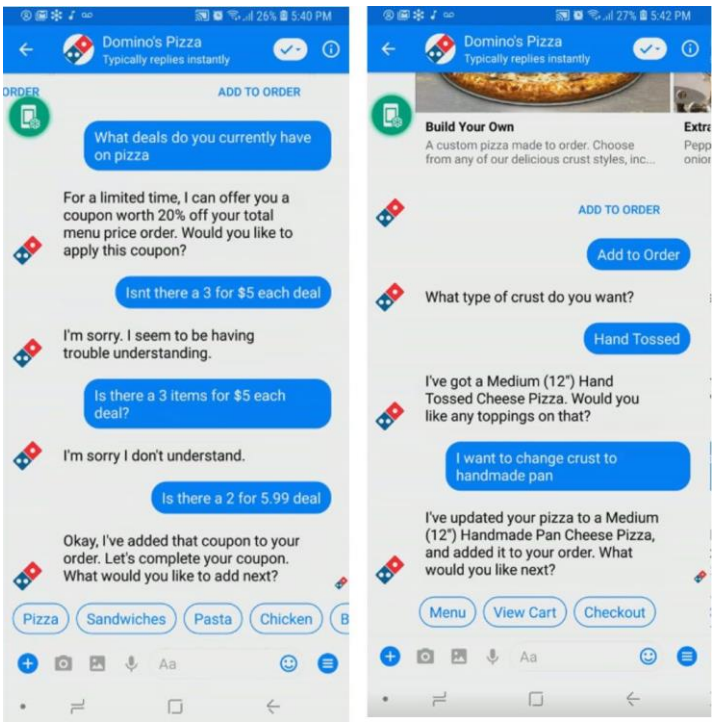


Рисунок 1. 3 – Взаємодія з чат-ботом Domino's Pizza

2. Через Wingstop чат-бот користувачі можуть знайти магазин, зробити замовлення, зв'язатися зі своїм обліковим записом Wingstop або поспілкуватися з представником служби підтримки клієнтів. Бот використовує місцезнаходження, щоб знайти магазин, щоб користувач міг зробити замовлення або отримати маршрут.

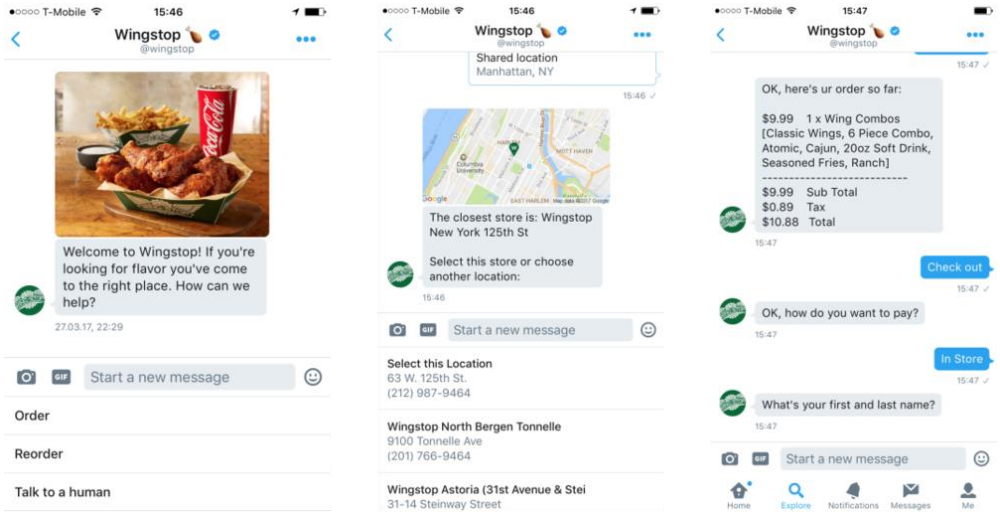


Рисунок 1. 4 – Взаємодія з чат-ботом Wingshop

1.2.1 Node.js

Відповідно до вимог завдання дипломного проекту, потреби активного мережного обміну даними між створеною системою та серверами Facebook, а також потреби швидкої розробки та простого налагодження, було зроблено рішення використовувати мову програмування JavaScript для створення системи управління чатом, а саме – платформу Node.js. То що ж таке Node.js та чому саме він?

Node.js — це среда виконання JavaScript з відкритим кодом і міжплатформною средою. Це популярний інструмент практично для будь-яких проектів. Node.js запускає двигун JavaScript V8, ядро Google Chrome, поза браузером. Це дозволяє Node.js бути дуже продуктивним.[4]

Суть технології полягає в гнучкому налаштованому серверному движку, що використовує неблокуючу модель введення/виводу на основі подій. Це свого роду переклад JS на машинну мову, який забезпечує підвищену

продуктивність та гнучкість. В результаті ми отримуємо середовище виконання, де JS-код швидко переміщується у напрямку від сервера до клієнта. З Node.js JavaScript розширив свої можливості від простого створення інтерактивних веб-сайтів до ширшого спектру варіантів використання. Наприклад, ми хочемо дізнатися відомості про клієнта1 та клієнта2, для цього ми запитуємо у серверної бази даних про них, а потім виводимо їх на екран. Очевидно, що відповідь на цей запит займає деякий час, але плюс в тому, що обидва запити даних можуть виконуватись незалежно один від одного та одночасно, як наслідок, підвищення продуктивність веб-застосунків. Іншими словами, розробка веб-додатків на Node.js забезпечує стабільну і безпечну модель вводу-виводу, що не блокує, гарно спрощуючи код. Досить важливо у використанні, що система пакетів Node.js, npm, є найбільшою у світі системою бібліотек з відкритим вихідним кодом, що робить його зручним у використанні.

1.2.2 Amazon DynamoDB

Amazon DynamoDB – повністю керована безсерверна база даних NoSQL на основі пар «ключ-значення», створена для запуску високопродуктивних програм у будь-якому масштабі. DynamoDB пропонує вбудований захист, безперервне резервне копіювання, автоматичну реплікацію у кількох регіонах, кешування в пам'яті та інструменти експорту даних.[5]

Amazon DynamoDB добре підходить для безсерверних програм за допомогою AWS Lambda. Тому саме цю базу даних було обрано для цієї роботи. Lambda може забезпечити автоматичне масштабування, обчислення без збереження стану у відповідь тригери подій. DynamoDB доступний через HTTP API і виконує автентифікацію та авторизацію через ролі IAM, що робить його ідеальним для створення безсерверних програм.

Сучасні веб-додатки найчастіше мають проблеми зі збільшенням бази даних, коли кількість користувачів постійно зростає, але в той же час також зростає кількість трафіку і даних. За допомогою використання Amazon

DynamoDB розробники можуть почати з найменшого, використовуючи тільки необхідну для роботи ємність бази даних, а потім поступово збільшувати обсяг запитів для таблиць. Що не мало важливо в роботі, це те, що їх таблиці також можуть зростати без обмежень, оскільки їх користувачі зберігають все більші обсяги даних. У фоновому режимі Amazon DynamoDB автоматично розподіляє дані та трафік для таблиці за кількістю серверів, достатньою для задоволення запитів розробника. Amazon DynamoDB пропонує низькі та передбачувані затримки за будь-якого масштабу. Amazon DynamoDB зберігає дані на накопичувачах (SSD) і синхронно реплікує їх у кількох зонах доступності AWS у регіоні AWS, щоб забезпечити вбудовану високу доступність та надійність даних.

1.2.3 Facebook Messenger API

API (Application Programming Interface) - це програмний посередник, що дозволяє двом програмам взаємодіяти один з одним. Завжди, коли ми надсилаємо повідомлення один одному і використовуємо будь-які месенджери, такі як Telgram, Facebook, Viber, дивимося розклад пар на своєму телефоні, ми щоразу використовуємо API.

То як же влаштована робота API? Коли ви використовуєте програму на своєму смартфоні через Інтернет, то вона відправляє дані на сервер. Після чого сервер отримує ці дані та ітерпретує їх, на наступному кроці вона виконує необхідні дії та врешті-решт відправляє назад на мобільний пристрій. Знову програма інтерпретує ці дані та представляє потрібну інформацію в читальному для нас вигляді, все це відбувається через API. Досить важливо при роботі розробником, їм не потрібно знати, як реалізований API, можна просто використовувати інтерфейс для зв'язку з іншими продуктами та послугами. Використання API зросло за останнє десятиліття до невід'ємної частини розробки, де багато популярних сьогодні веб-додатків були б неможливі без API.

Існує багато різних API-інтерфейсів чат-ботів, які можуть допомогти в роботі, і багато мереж, заснованих на чатах, надають API-інтерфейси для своєї платформи. Наприклад, Facebook Messenger API, Amazon Lex, Microsoft Bot framework, Slack Bot API, Telegram Bot API та багато інших. Багато API-інтерфейсів віртуальних помічників можна використовувати абсолютно безкоштовно, але як частина платформи соціального чату, як у нашому випадку. Також надано можливість використовувати більш автономні сервіси, рішення з відкритим вихідним кодом, які дозволяють моментально створювати віртуальних помічників та інтегрувати їх у групові чати, SMS-повідомлення, електронну пошту та багато інших середовищ.

У цій роботі використовується Facebook Messenger API, тому що цей API знаходиться у вільному доступі для використання та простий у роботі, оскільки надається багато документації у вільному доступі. Це дозволяє створювати інструменти, як і для надсилання, так і для отримання повідомлень через дану платформу. За допомогою цього API можна створювати чат-боти, які можуть автоматично спілкуватися з групою людей не тільки в особистих повідомленях, але в чатах також. Facebook Messenger API допоможе створити бота, який зможе відповідати на запитання моментально, тобто в режимі реального часу, виконувати різний вид діяльності, обробляти будь-які запити, які описані в полі для коментарів.

1.2.4 Claudia.js

Claudia спрощує розгортання проектів Node.js на AWS Lambda і API Gateway. Він автоматизує всі пов’язані з помилками завдання розгортання та налаштування, а також налаштовує все так, як очікують розробники JavaScript із коробки. Це означає, що ви можете легко розпочати роботу з мікросервісами Lambda і зосередитися на вирішенні важливих бізнес-проблем замість того, щоб мати справу з робочими процесами розгортання AWS.[6] Claudia працює тільки для JavaScript/Node.js, але робить це дуже добре. Оскільки Claudia орієнтована на Node.js, Claudia автоматично встановлює шаблони для

перетворення параметрів та результатів на об'єкти, які JavaScript може легко використовувати, і змушує все працювати так, як очікують розробники JavaScript.

1.3 Постановка задачі

Результати проведеного аналітичного огляду доводять актуальність задачі впровадження чат-ботів в сучасні безсерверні додатки. Метою роботи є розробка і реалізація чат-боту Facebook Messenger, за допомогою якого здійснюється замовлення їжі з ресторану, та інтеграція його з вже існуючею API.Отже, необхідно реалізувати наступні задачі:

1. Формування вхідного математичного опису чат-боту
2. Розробка інформаційної моделі чат-боту безсерверного додатку
4. Розробка структури даних для зберігання питань користувачів та відповідей на них.
5. Програмна реалізація чат-боту.
6. Тестування чат-боту.

У результаті реалізації буде можливим спілкуватися с чат-ботом, переглядати список піц та робити замовлення.

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Інформаційна модель чат-боту безсерверного додатку

Велика кількість сучасних програм створюються саме за безсерверною технологією. Чому ж так популярно почало використовувати цю технологію та в чому її плюси? Для початку потрібно з'ясувати, що таке безсерверна архітектура. Безсерверна архітектура – це спосіб створення, а також запуску програм та сервісів без необхідності керування інфраструктурою. Програма буде працювати на серверах, але керування цими серверами AWS повністю бере на себе. З використанням даної архітектури розробникам більше не доведеться займатися такими справами як масштабування, при масштабуванні ваших проектів практично не потрібно зусиль для управління ємністю, виділенням ресурсів, баз даних та систем зберігання даних.[7]

Найвідомішими і найбільш сучасними інфраструктурами є безсерверний контейнер Amazon AWS Lambda, Microsoft Azure Functions і Google Cloud Functions. У цій роботі використовується AWS Lambda, тому що це найрозвинена з доступних безсерверних інфраструктур, а також вона має стабільний API і багато успішних історій у використанні.

AWS Lambda – це безсерверний, керований подіями обчислювальний сервіс, який дозволяє виконувати код практично для необмеженого типу програми або сервісу без надання серверів та їх обслуговування. Ви можете включити Lambda з понад 200 сервісів та програм, що надаються за моделлю ПЗ як послуга (SaaS), оплачуючи тільки ті ресурси, які використовуєте.[8]

Функції Lambda прості у використанні та розумінні, але важким етапом є розгортання. Розгорнути безсерверну програму в AWS Lambda можна кількома способами: за допомогою інтерфейсу командного рядка з терміналу AWS, за допомогою AWS API; за допомогою візуального інтерфейсу консолі AWS Lambda. Розгортання безсерверної програми простіше, ніж розгортання традиційної програми.

Чат-боти Facebook – це програми підтримки сторінок Facebook, тобто вони не є окремими та незалежними програмами, такими як програми, які можна завантажити з Google Play або App Store. Чат-боти для Facebook Messenger створюються в чотири етапи:

1. створення сторінки у Facebook для майбутнього віртуального помічника;

2. створення програми у Facebook для розробників, яка буде обслуговувати бота і підключитися до головної сторінки;

3. впровадження та розгортання чат-бота;

4. підключення чат-бота до програми Facebook.

То як же можна буде взаємодіяти з нашим чат-ботом? Для початку користувач повинен відкрити сторінку на платформі Facebook та надіслати повідомлення. Програма, яка пов'язана зі сторінкою Facebook, отримає повідомлення та надішле запит чат-бота з повідомлення користувача. Наступним етапом віртуальний помічник отримає та опрацює повідомлення та поверне відповідь у програму Facebook, яка виведе її на сторінку у Facebook. Зобразимо це на графіку (Рисунок 2.1):

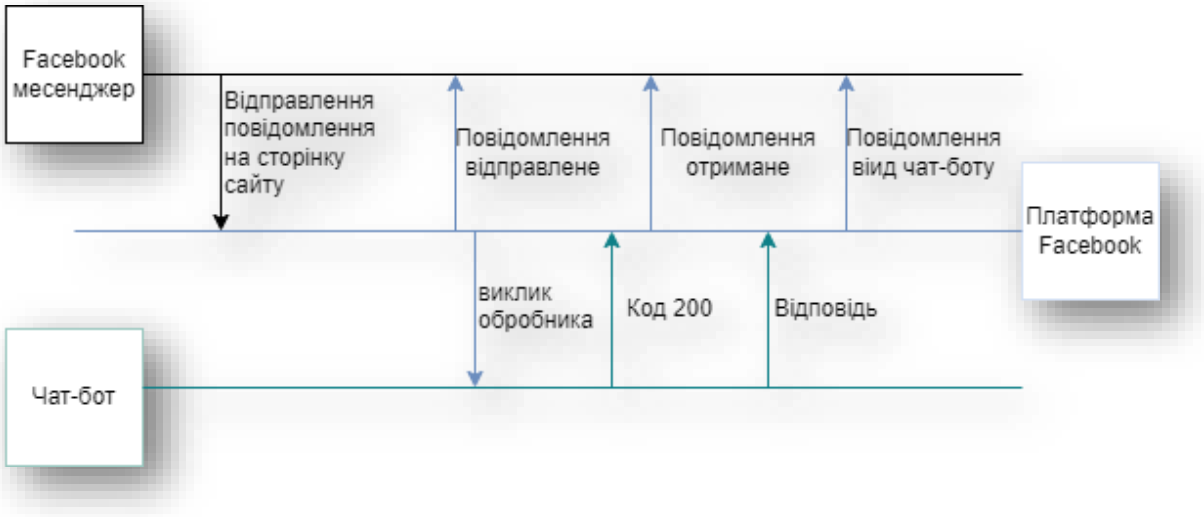


Рисунок 2.1 – Процес надсилання повідомлення чат-боту та отримання відповіді

Створити чудового віртуального помічника складно, тому що для більшості кінцевих користувачів незвичний текстовий інтерфейс, крім цього, чат-бот зобов'язаний забезпечувати хоча б якусь обробку природної мови та штучний інтелект, які складно налаштувати. Ця проблема актуальна для розробників, як результат багато платформ віртуальних співрозмовників пропонують підтримку елементів інтерфейсу, подібних до графічного інтерфейсу, таких як кнопки, квитанції та списки, щоб інтуїтивно та зручно було користуватися.

Facebook Messenger є однією з таких платформ, і її елементи інтерфейсу користувача називаються шаблонами. Facebook Messenger пропонує кілька шаблонів, у нашому випадку ми будемо використовувати шаблон списку та загальний шаблон.

Claudia Bot Builder — це бібліотека розширень для Claudia.js, яка допомагає створювати ботів для Facebook Messenger, Telegram, Skype, команд Slack, Twilio, Kik і GroupMe. Ключова ідея проекту полягає в тому, щоб видалити весь шаблонний код і загальні інфраструктурні завдання, щоб ви могли зосередитися на написанні дійсно важливої частини бота – робочих процесів нашого бізнесу.[9]

Отже, основна мета використання Claudia Bot Builder для нас – це абстрагувати структурну характеристику для платформи Facebook та дозволити отримувати та надсилати повідомлення за допомогою API. Воно використовує Claudia API Builder для створення точок входу для кожної підтримуваної платформи. Claudia Bot Builder підтримує 10 платформ, однією з них Facebook Messenger.

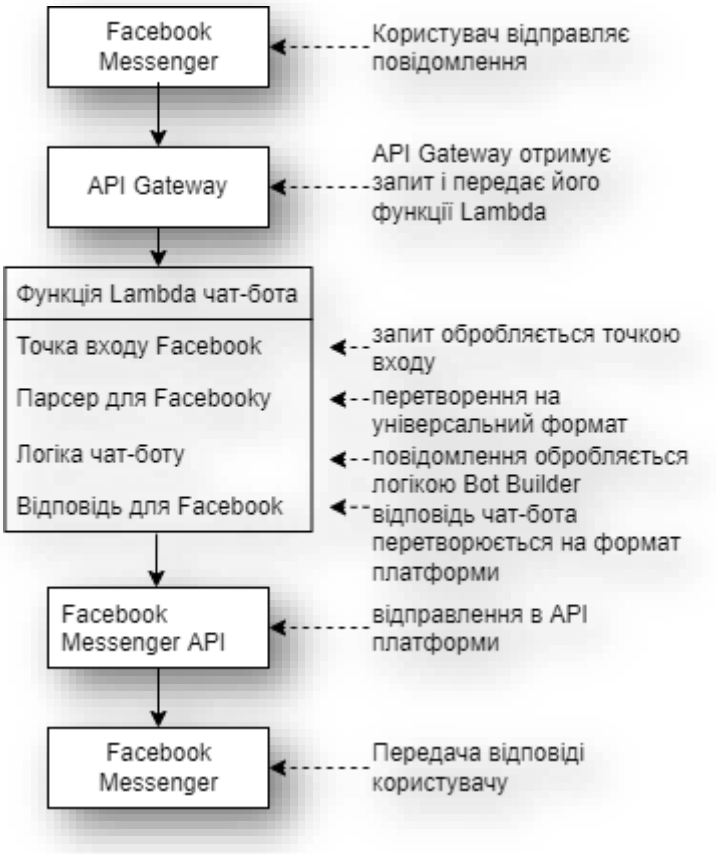


Рисунок 2.2 – Процес конвертації оновлень у Claudia Bot Builder

Як показано на Рисунок 2.2, життєвий цикл Claudia Bot Builder виглядає так:

1. Користувач надсилає повідомлення за допомогою платформи Facebook Messenger;
2. платформа надсилає повідомлення до точки входу через API Gateway, який вказували в налаштуваннях;
3. шлюз API запускає нашу лямбда-функцію, передаючи запит до кінцевої точки API, що залежить від платформи;
4. запит перетворюється в універсальний формат за допомогою парсера, що відповідають платформі;
5. конвертоване повідомлення передається в логіку бота;
6. відповідь чат-бота конвертується у формат для платформи Facebook;
7. claudia Bot Builder викликає API платформи та надсилає відповідь;

8. API платформи повертає відповідь до програми обміну повідомленнями.

Повернути відповідь на повідомлення з Claudia Bot Builder так же просто, як з Claudia API Builder: щоб відповісти текстовим повідомленням, треба повернути рядок. Також можна відповісти шаблоном для обраної платформи, використовувати обробником повідомлення з даних шаблонів або повернути об'єкт JSON.

2.2 Додавання інтерактивності до чат-боту

Спілкування – природний спосіб комунікації для кожної людини. Якщо графічний інтерфейс сайту для замовлення їжі може вимагати деякого часу, щоб розібратися, але щоб зробити замовлення за допомогою розмовного інтерфейсу клієнт може зробити інтуїтивно. При доволі просторому різноманіттi можливостей інтерфейсу чат-ботів важливо знайти баланс між розмовною складовою та іншими елементами, такими як, зображення, списки, кнопки, щоб розмовляти було досить комфортно з помічником. Тому було вирішено, що відображення списку з наявними піцами – відмінна ідея для елементарної та зручної відповідді.

Для зручності у використанні нашого помічника було віришено використовувати універсальний шаблон. Зручніший у використанні шаблон списку, але він має обмеження за розміром, він зображує список не менше ніж з двома і не більше ніж із чотирма елементами, що для нас є негідний, адже не вигідно мати лише 2-4 піц у бізнесі, особливо коли бізнес хоче розвиватися. Тому ідеальний варінат - універсальний шаблон, він більш гнучкий, може відображати від одного до десяти елементів. Кожен елемент включає зображення, заголовок, опис та кнопки, які самі можемо налаштувати. Кнопки в універсальному шаблоні можуть виконувати різні дії, такі як відкриття URL-адреси або надсилання відповіді клієнта до точки входу. Оскільки наша головна мета не тільки продивитися наявні піци, а ще дати можливість замовити піцу, ми зробимо наступне:

1) додамо кнопку Order (Буде виконувати фунцію замовлення);

2) реалізуємо оформлення замовлення піци, зберігши інформацію для замовлення в базі даних після натискання кнопки Order;

3) додамо в бот обмежену обробку тексту природною мовою, щоб кінцевому користувачу було зручно спілкуватися з нашим помічником.

Для того щоб чат-боти працювали ефективно для нашого бізнесу, він повинен вміти переводити людську мову в зручний для комп'ютера формат. Це складний процес розпізнавання та аналізу людської мови, а також генерування відповідей. Алгоритми NLP на основі штучного інтелекту визначають значення тексту , а потім повертають відповідь. Завдяки технології NLP боти можуть імітувати живого співрозмовника і спілкуватися з користувачем.

Алгоритм процесу обробки запитів у чат-боті розділиться на три гілки (Рисунок 2.3):

- користувач може переглянути подробиці замовлення;
- користувач може замовити замовлення;
- у будь-яких інших випадках чат-бот повинен повернути початкове повідомлення з меню.

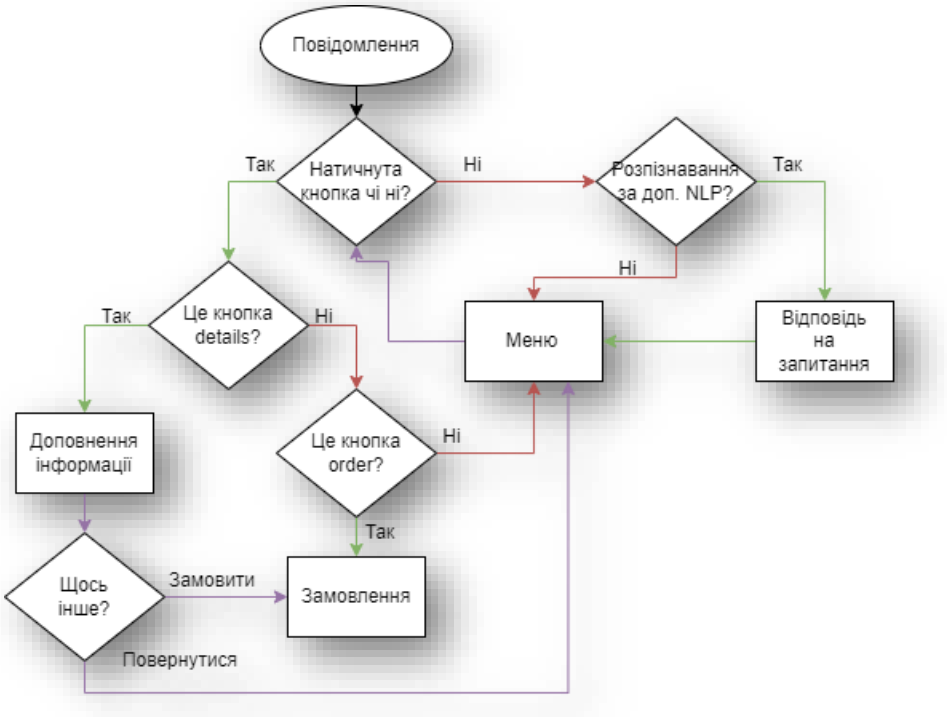


Рисунок 2.3 – Алгоритм роботи чат-бота

Отже, треба розуміти як це буде працювати в дії. Якщо повідомлення є відповіддю користувача, то потрібно перевірити, чи бажає користувач побачити детальний опис замовлення або замовити його (назвемо це дією), і отримати ідентифікатор вибраного товару. Щоб зберегти обидва значення, ми можемо оновити значення натиснутої кнопки, згенерувати рядок із переліком дії та ідентифікатора товару, розділених символом вертикальної лінії (|).Ми можемо зберегти рядок у форматі action | id, де action – це назва виконаної дії, а id – ідентифікатор замовлення.

Отримавши дію та ідентифікатор, потрібно перевірити, яка дія була вибрана – замовлення чи деталі. Коли чат-бот отримає дію details, нам потрібно скласти список інформації, і повернути цей список клієнту. Після отримання цієї інформації звичайний користувач, не зрозуміє, що робити далі, і в результаті цей клієнт до нас більше не повернеться. Для вирішення цієї дії можна встановити відповіді за замовчуванням, але неможливо передбачити всі варінти запитань, які може задати користувач.

Коли справа доходить до обробки природної мови, розробники можуть навчити бота взаємодіям і розмовам у різних сферах, а також надати деякі приклади відповідей, з яким він буде стикатися, коли не знає, що відповісти. Найпростіший спосіб направити користувача до наступної дії — показати меню з доступними параметрами. Це не дає повну гарантію, що клієнт натисне одну з кнопок, але меню допоможе отримати більш точні результати, ніж просте запитання. Нарешті, якщо повідомлення не є відповіддю користувача (тобто воно не є результатом натискання кнопки) або якщо вказано дію, відмінну від details та order, можна повернути загальну відповідь.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Короткий опис програмної реалізації

Для налаштування нашого помічника у Facebook Messenger потрібно виконати наступні кроки:

1. створити сторінку «Diploma Pizza» Facebook;
2. створити програму «pizza_diploma» у Meta for Developers;
3. створити чат-бота для Facebook Messenger;
4. підключити бота до бази даних;
5. увімкнути обробку природної мови (Natural Language Processing).

3.1.1 Налаштування Facebook Messenger

Чат-боти у Facebook Messenger обов'язково прив'язані до сторінки, тому створила публічну сторінку «Diploma Pizza» (Рисунок 3. 1). Можна у свою чергу створити окрему сторінку заздалегідь, прив'язати бота до існуючої публічної сторінки, або створити її пізніше, коли потрібно буде прикріплювати програму.

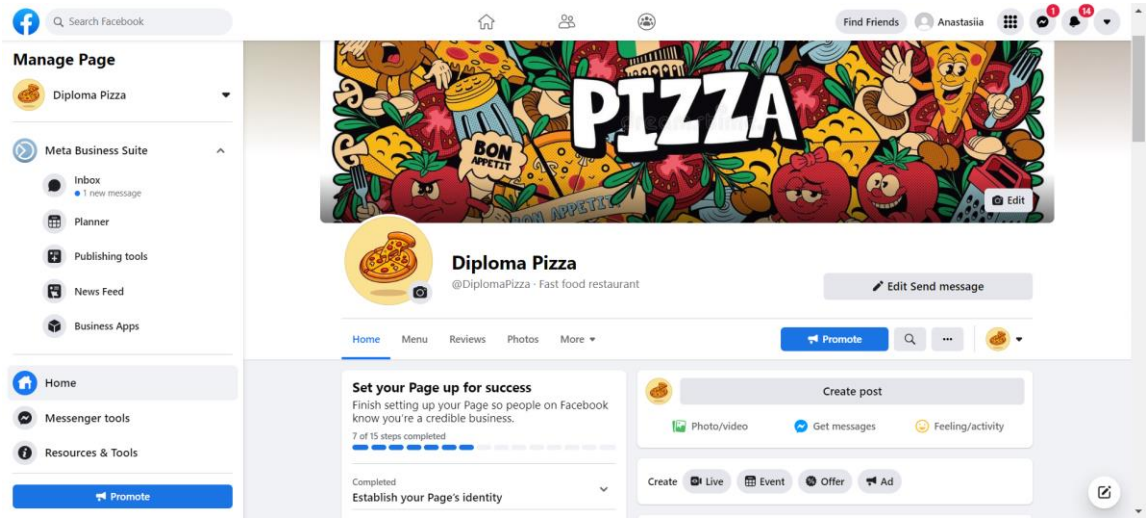


Рисунок 3. 1 – Сторінка Facebook для піцерії

Наступний крок – створення програми Facebook. При створенні програми необхідно визначити тип програми, від типу програми залежить, які саме продукти та API будуть йому доступні. Було вирішено обрати Business

type, так як це найзручніший тип, який має все необхідне для розвитку власного бізнеса. Як тільки програма була створена, додала продукт Messenger, через те що наш помічник спілкуватиметься за допомогою цього засобу. Крім цього, щоб бот знав, що йому пишуть, треба підписатися на події бота, для цього завантажила Webhooks (перед цим у вас на сервері вже був розміщений скрипт який і отримуватиме всі події та надсилатиме відповіді на них) (Рисунок 3. 2).

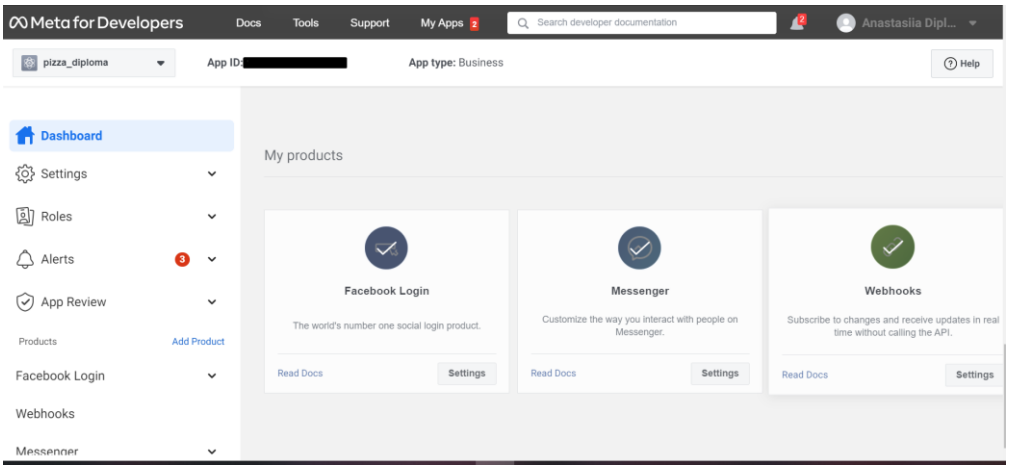


Рисунок 3. 2 – Список доданих продуктів

Тепер, коли маю готову програму, можна приступати до фактичної реалізації чат-бота.

3.1.2 Встановлення та налаштування Claudia

Для завантаження Claudia потрібно, щоб вже налаштовані були програми:

- AWS аккаут з доступумом до IAM та Lambda

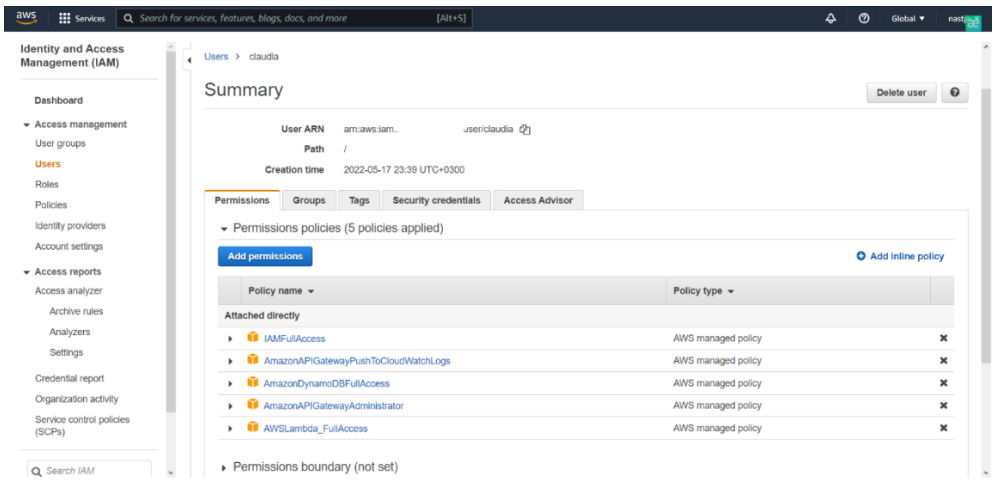


Рисунок 3. 3 – AWS аккаут

- Node.js
- NPM

```
C:\Users\User\diploma>npm -v
8.5.5

C:\Users\User\diploma>node -v
v16.15.0
```

Рисунок 3. 4 – Версії npm та node.js

Claudia доступна на NPM, і найпростіший спосіб її використання — встановити її як глобальну утиліту. Команда для встановлення: npm install claudia –g. [10]

Бібліотека Claudia API Builder доступна як пакет NPM. Вона не вимагає будівництва, тому для установки достатньо зберегти її як залежність у проекті Node.js, виконавши команду: npm install claudia-api-builder --save

З бібліотекою Claudia Bot Builder аналогічно як з Claudia API Builder. Використовуючи Bot Builder, Claudia автоматично налаштує правильні веб-хуки для всіх підтримуваних платформ і проведе через налаштування доступу. Для установки виконала команду: npm install claudia-bot-builder --save.

Модуль Bot Builder – це функція, яка у першому аргументі приймає функцію-обробник повідомлень та повертає екземпляр Claudia API Builder. Функція-обробник повідомлень – це функція, яка має викликатись при

отриманні повідомлення нашим помічником. Щоб створити функцію AWS Lambda та налаштувати чат-бот, нам потрібно виконати команду:

claudia create --region es-east-1 --api-module bot --configure-fb-bot [11], де

--region – потрібно вказати свій регіон або найближчий

--api-module - визначає шлях до головного файлу з вихідним кодом

--configure-fb-bot – забезпечує автоматичне настроювання чат-бота для Facebook Messenger

Розгортання завершилось успіхом, команда запропонувала ввести ключ доступу до нашої сторінки у Facebook, а потім вивела URL-адресу точки входу з вашим ключем верифікації, як і очікувалось(Рисунок 3. 5). Після введення ключа доступу до сторінки «Diploma Pizza» віртуальний помічник готовий та доступний для тестування та удосконалення.

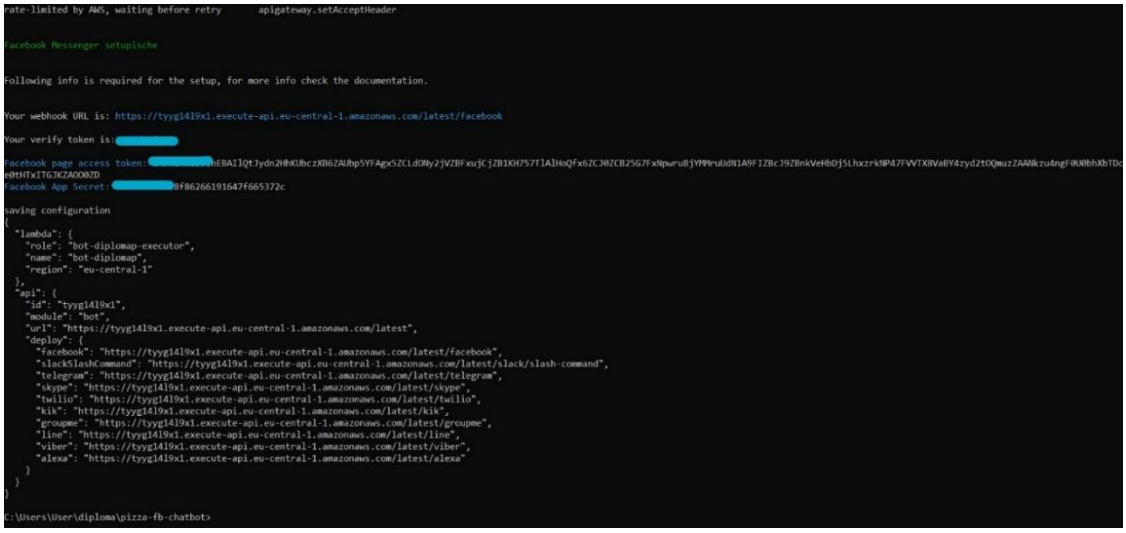


Рисунок 3. 5 – Розгортання чат-бота

Наразі наш помічник може лише відповісти привітанням на наше повідомлення(Рисунок 3. 6), оскільки ми повертаємо єдиний текстовий рядок: const api = botBuilder(() => { return `Привіт, друже! У нас ти можеш замовити піци.`}

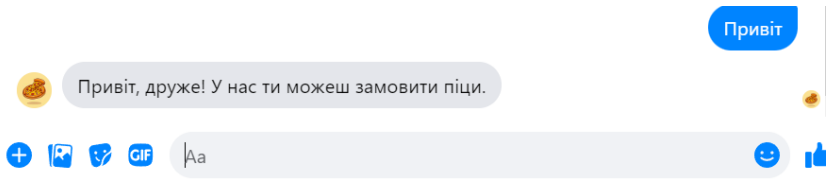


Рисунок 3. 6 – Тестове повідомлення від чат-боту

Facebook Messenger має чудову функцію, що дозволяє користувачам повідомляти про своє поточне місце розташування. Після відправлення геолокації клієнта чат-бот отримає поточні координати: широту та довготу. Для обробки координат створила нову функцію-обробник. Цей обробник приймає параметри userId та coordinates і використовує їх для зміни замовлення в базі даних. Щоб отримати можливість змінити замовлення, імпортую AWS SDK, створила екземпляр DocumentClient та виконати такі дії:

- за допомогою методу DocumentClient.scan знайти в базі даних замовлення зі статусом in-progress;
- використати метод DocumentClient.update та отримане значення orderId, щоб змінити статус замовлення.

Окрім запиту на геолокацію було додано зображення та список піц (Рисунок 3. 7), інгредієнтів та кнопок «Замовити», «Інгредієнти», «Показати піци» (Рисунок 3. 8)(Додаток А).

```
function pizzaMenu() {
 const message = new fbTemplate.Generic()
 pizzas.forEach(pizza => {
 message.addBubble(pizza.name)
 .addImage(pizza.image)
 .addButton('Інгредієнти', 'DETAILS|${pizza.id}')
 .addButton('Замовити', 'ORDER|${pizza.id}')
 })
 return message.get()
}
```

Рисунок 3. 7 – Реалізація універсального шаблону з піцами та кнопками «Замовити», «Інгредієнти»

```
function pizzaDetails(id) {
 const pizza = pizzas.find(pizza => pizza.id == id)
 return [
 '${pizza.name} Інгредієнти:'  + pizza.ingredients.
 join(', '),
 new fbTemplate.Button('Що я можу зробити?')
 .addButton('Замовити', 'ORDER|${pizza.id}')
 .addButton('Показати піци', 'ALL_PIZZAS')
 .get()
 ]
}
```

Рисунок 3. 8 – Додавання кнопок «Замовити», «Показати піци» та список інгредієнтів

Для того, щоб все працювало треба виконати команду claudia update без аргументів. Оновлення завершилось успіхом (Рисунок 3. 9), тому можемо переглядати та замовляти піци(Рисунок 3. 14-Рисунок 3. 19).

Рисунок 3. 9 – Успішне оновлення файлу з новим функціоналом

3.1.3 Підключення чат-бота до бази даних DynamoDB

Для зручного користування та мониторингу заказів клієнтів, було вирішено створити таблицю, де буде зберігатися адресу, айді замовлення, номер піци, статус замовлення та з якого додатку було відправлене повідомлення. За допомогою DocumentClient ми підключемося до DynamoDB. Далі викликала метод docClient.put, щоб зберегти замовлення у таблиці DynamoDB. Тепер потрібно створити політику, яка дозволить користувачеві, що викликає функцію Lambda, взаємодіяти з базою даних DynamoDB. Для

цього створила роль, яка містить дозволи, що дозволяють користувачам сканувати, читати, додавати та змінювати елементи в DynamoDB:

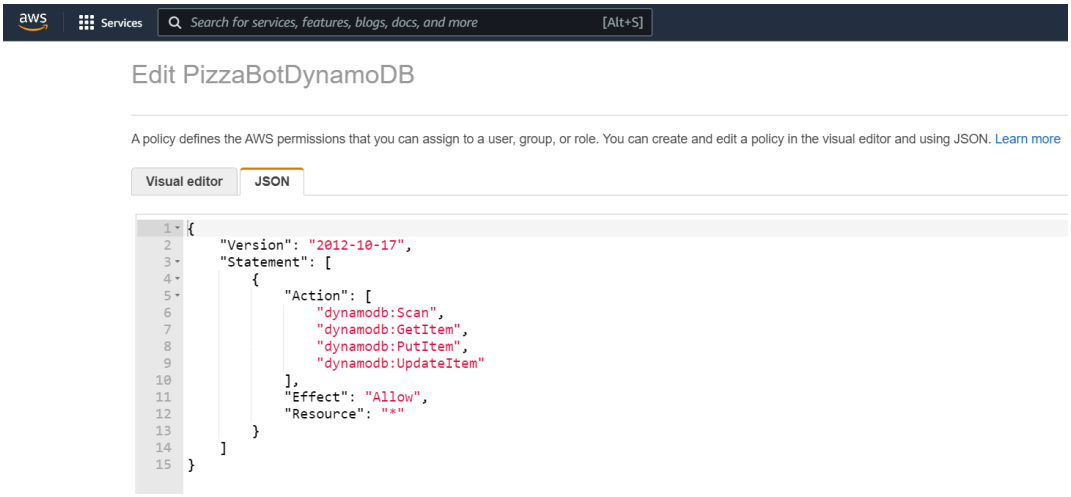


Рисунок 3. 10 – Політика DynamoDB

Щоб додати політику доступу до DynamoDB для чат-бота виконала команду, яка зображена на Рисунок 3. 11. Після успішного завершення ця команда повернула пусту відповідь, що і очікувалось.

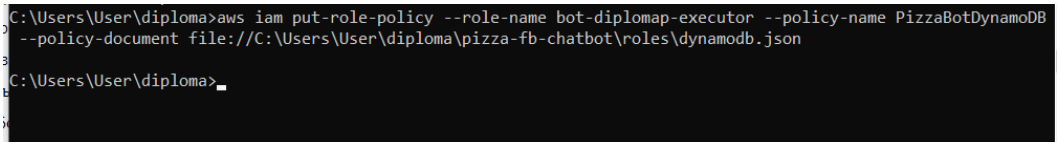
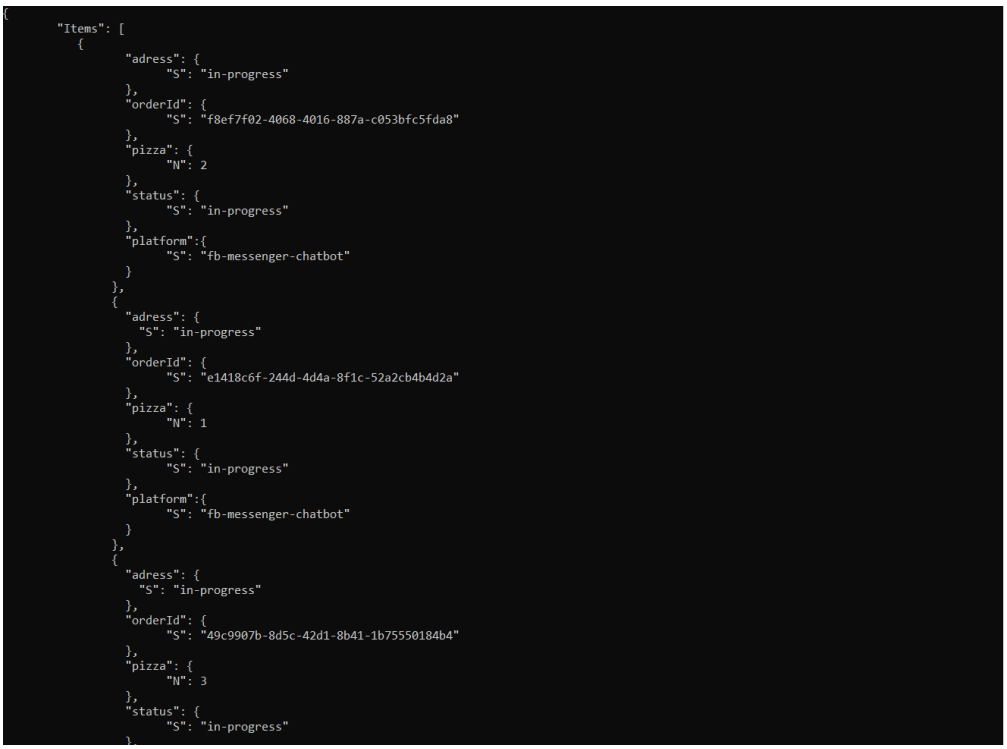


Рисунок 3. 11 – Додавання політики доступу до DynamoDB для чат-боту

Тепер чат-бот під'єднаний до бази даних та готовий до розгортання. Для усвідомлення, що все працює як треба, було виконані замовлення через чат-бот. Щоб перевірити таблицю була виконана команда aws dynamodb scan \ --table-name pizza-order \ --region es-east-1 \ --output json. Як ми можемо побачити, все працює добре, дані наявні в таблиці (Рисунок 3. 12).



```
{
    "Items": [
        {
            "adress": {
                "S": "in-progress"
            },
            "orderId": {
                "S": "f8ef7f02-4068-4016-887a-c053bfc5fda8"
            },
            "pizza": {
                "N": 2
            },
            "status": {
                "S": "in-progress"
            },
            "platform":{
                "S": "fb-messenger-chatbot"
            }
        },
        {
          "adress": {
            "S": "in-progress"
          },
          "orderId": {
                "S": "e1418c6f-244d-4d4a-8f1c-52a2cb4b4d2a"
          },
          "pizza": {
                "N": 1
          },
          "status": {
                "S": "in-progress"
          },
          "platform":{
                "S": "fb-messenger-chatbot"
          }
        },
        {
          "adress": {
            "S": "in-progress"
          },
          "orderId": {
                "S": "49c9907b-8d5c-42d1-8b41-1b75550184b4"
          },
          "pizza": {
                "N": 3
          },
          "status": {
                "S": "in-progress"
          },
```


Рисунок 3. 12 – Таблиця pizza-order

3. 1.4 Додавання простої обробки природної мови

Зараз багато доступних бібліотек, які можно інтегрувати у чот-боти, такі як Facebook's Wit.ai, Google Dialogflow, Amazon Lex, IBM Watson Assistant. У Facebook Messenger теж є вбудована бібліотека NLP, але вона пропонує тільки найпростіші інструменти. З її допомогою можна навчити чат-бот розпізнавати координати, вітання, прощання, визначати дати, час, номери телефонів, адреси електронної пошти, температуру, URL та інш. Оскількі вона вже вбудована, було вирішено використовувати саме цю бібліотеку. Головна мета, щоб наш помічник міг відповісти на «дякую».

Для цього потрібне виконати наступні дії:

1. встановити та налаштувати вбудовану бібліотеку NLP;
2. додати перевірку у файл з головним кодом, чи є повідомлення відповіддю клієнта. Якщо ні, ми не будемо використовувати засоби NLP;
3. якщо повідомлення є відповіддю клієнта, використовуємо бібліотеку NLP. Якщо подяка має місце, відповімо повідомленням «Вам дуже

дякую, за те що у нас замовили. Гарного дня!», інашке повернемося до початвого повідомлення.

Вбудована бібліотека NLP додає опізнані сутності до атрибуту nlp об'єкта message. Сутності повертаються як масив, і кожна сутність має оцінку достовірності в атрибуті confidence і значення в атрибуті value. Оцінка достовірності визначає ступінь упевненості парсера в пізнанні і має значення діапазоні від 0 до 1. Атрибут value визначає значення пізнаної сутності. У випадку з сутністю «дякую», якщо вона є в повідомленні, цей атрибут завжди матиме значення true. Ми перевіримо існування сутності «дякую», і якщо її оцінка впевненості перевищує 80%, то повернемо відповідь.

```
if (
message.originalRequest.message.nlp &&
message.originalRequest.message.nlp.entities &&
message.originalRequest.message.nlp.entities['дякую'] &&
message.originalRequest.message.nlp.entities['дякую'].length &&
message.originalRequest.message.nlp.entities['дякую'][0].confidence > 0.8
) {
return 'Вам дуже дякую, за те що у нас замовили. Гарного дня!'
}
return [
'Привіт, друже! Ось такі піци у нас є в наявності:',
pizzaMenu()
]
}, {
```

Рисунок 3. 13 – Відповідь на повідомлення з подякою

Програмна реалізація чат-боту завернеша, можемо переходити відповідно до тестування.

3. 2 Тестування роботи Facebook боту для замовлення піци

Нижче наведені скріншоти основного функціоналу Facebook чат-бота:

- Початок роботи з ботом:

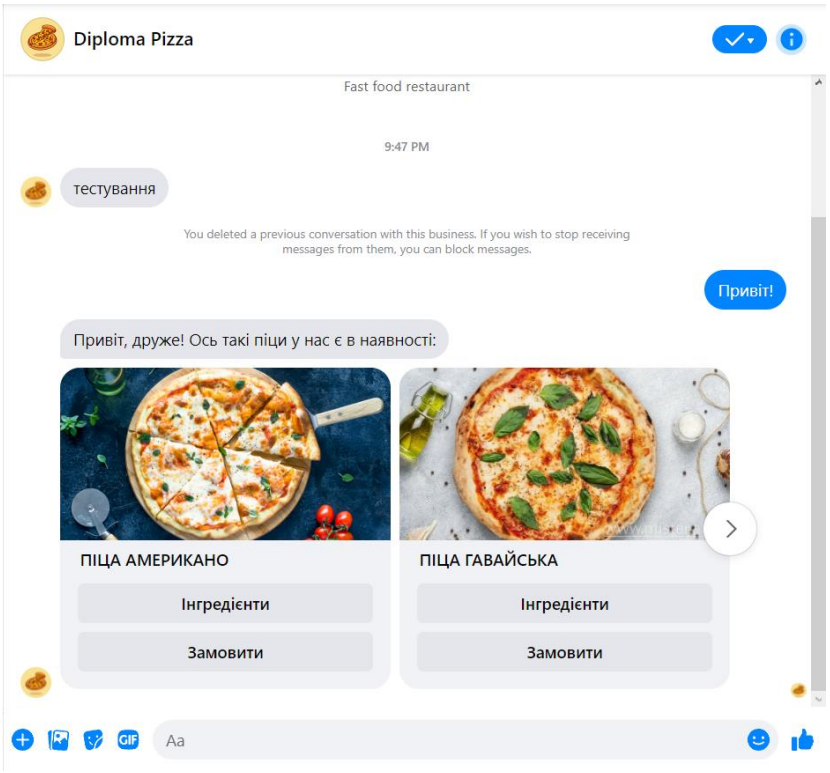


Рисунок 3. 14 – Початок роботи з ботом

- Перевірка функіоналу перегляду піц:

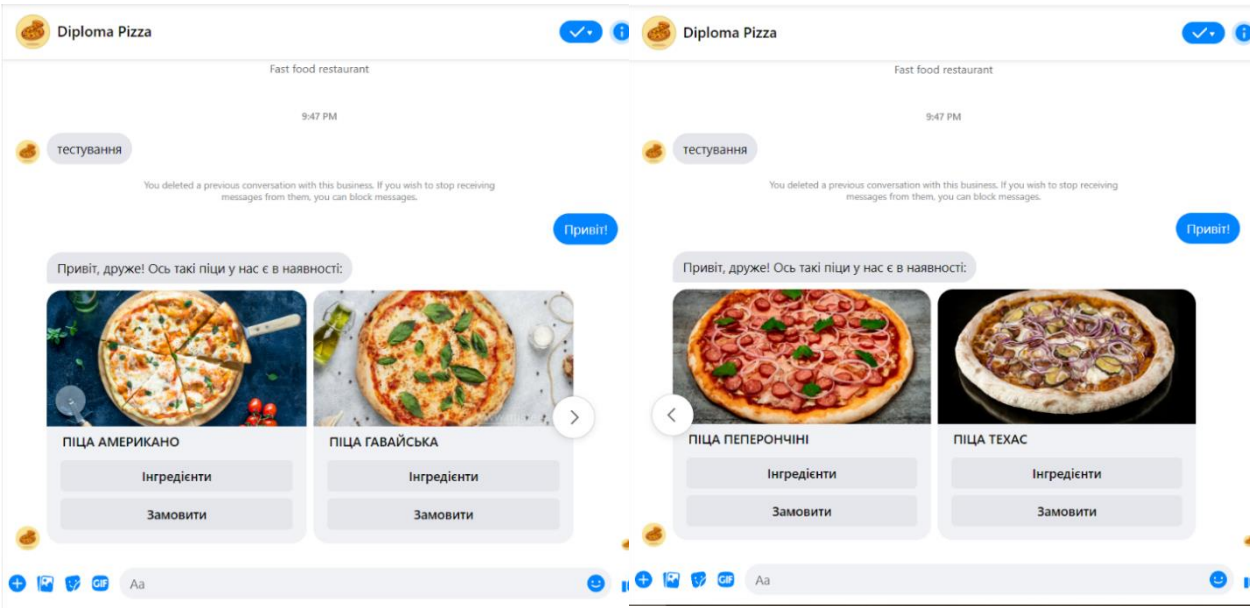


Рисунок 3. 15 – Перегляд піц

- Перевірка функціонування кнопки «Інгредієнти»:

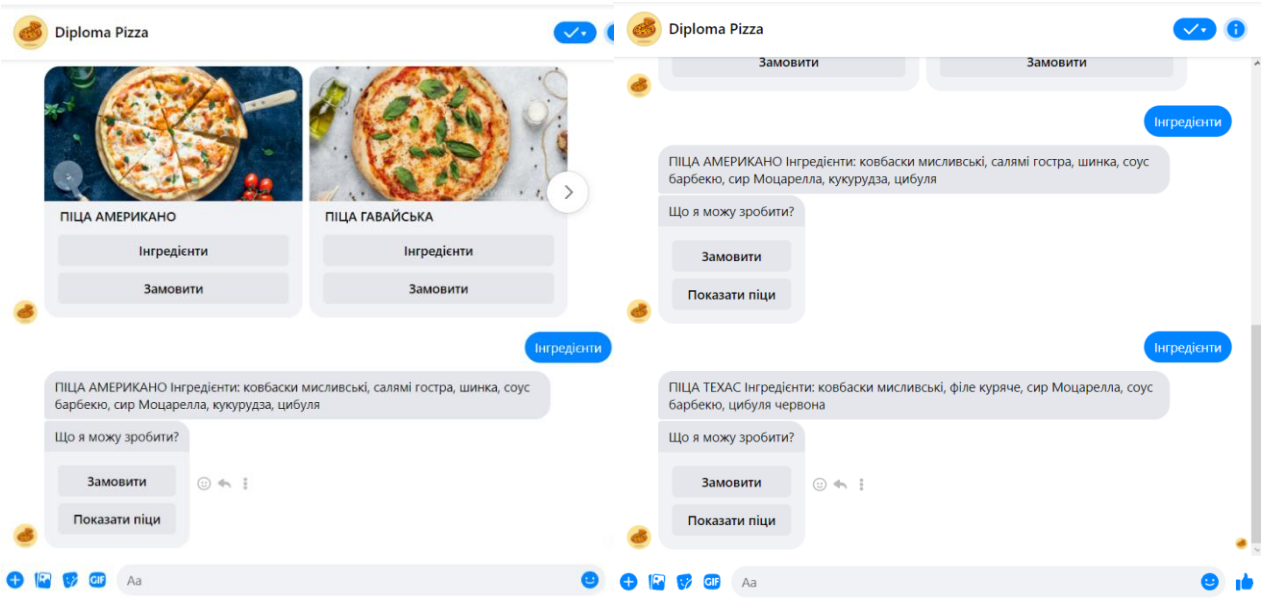


Рисунок 3. 16 – Функціонування кнопки «Інгредієнти»

- Перевірка функціонування кнопки «Замовити»:

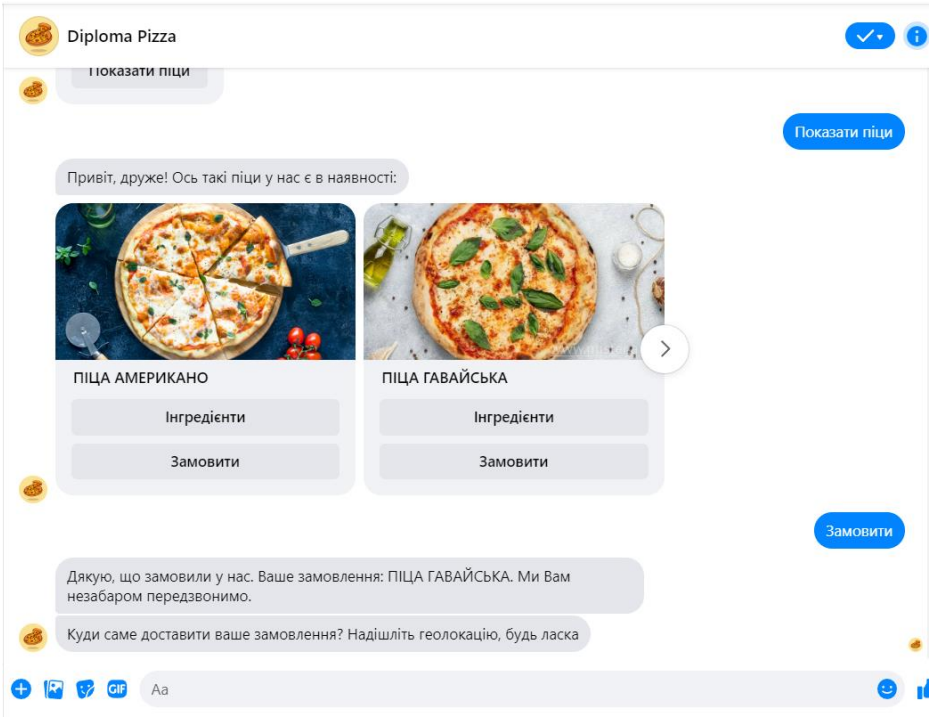


Рисунок 3. 17 – Функціонування кнопки «Замовити»

- Перевірка функціонування геолокації:

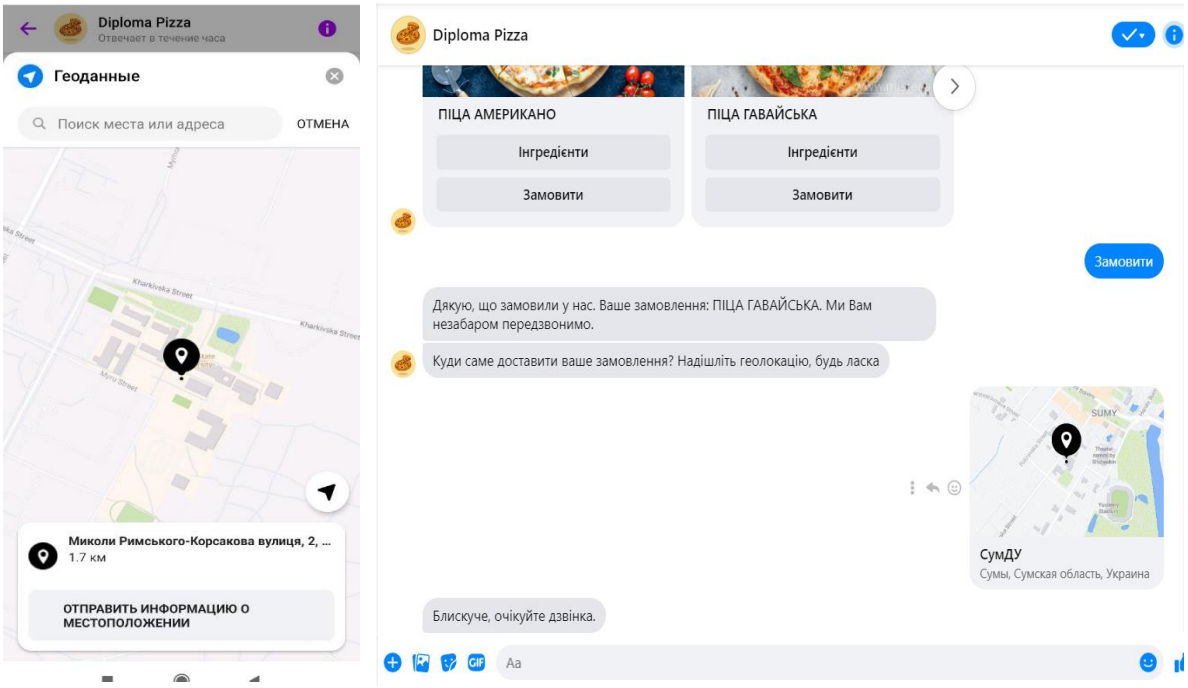


Рисунок 3. 18 – Відправлення геолокації

- Тестування NLP:

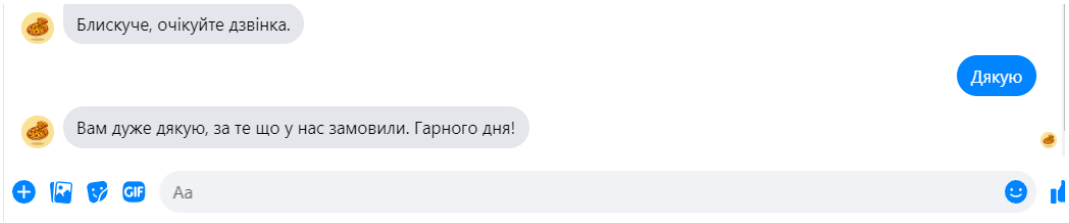


Рисунок 3. 19 – Тестування NLP

Таким чином, проведене тестування підтверджує працездатність розробленого інформаційного і програмного забезпечення чат-боту безсерверного додатку.

ВИСНОВКИ

У недалекому майбутньому чат-боти можуть стати кращим інтерфейсом користувача для багатьох видів діяльності. Використання віртуальних помічників дозволяє обслуговувати більше клієнтів з меншою кількістю працівників, економити час, підвищувати ефективність та точніть процесу. Прибуток підприємства буде зростати, а витрати на персонал - ні.

У результаті виконаної кваліфікаційної бакалаврської роботи був реалізован чат-бот «Diploma Pizza» для замовлення піц. Було досліджено методи машинного навчання для вдосконалення ботів, та їх взаємодію з користувачем. Вимоги з постановки задачі виконані. Facebook бот «Diploma Pizza» підтримує можливість оформлення замовлення, надання списку наявних піц, отримання адреса доставки, привітання та прощання з клієнтом. Комунікація з ботом відбувається за допомогою чату через Facebook Messenger.

За час виконання роботи поглибила знання у програмуванні мовою JavaScript, а також навчилася створювати безсерверних чат-ботів. Поглибила навички користування AWS та Node.js.

З можливих майбутніх покращень можна виділити:

- Додавання розпізнавання голосових повідомлень;
- Удосконалення та розвиток NLP;
- Можливість додати більше однієї позиції до замовлення:
- Збереження історії замовлень.

СПИСОК ЛІТЕРАТУРИ

1. Chatbot [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/c/chatbot.asp#citation-2
2. The History and Evolution of Chatbots [Електронний ресурс] – Режим доступу до ресурсу: https://insights.daffodilsw.com/blog/the-history-and-evolution-of-chatbots
3. Дізнайтеся, що шукають у світі [Електронний ресурс] – Режим доступу до ресурсу: https://trends.google.com/trends/?geo=UA
4. Introduction to Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.dev/learn/introduction-to-nodejs
5. Amazon DynamoDB [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/dynamodb/
6. CLAUDIA.JS [Електронний ресурс] – Режим доступу до ресурсу: https://claudiajs.com/
7. Building Applications with Serverless Architectures [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/lambda/serverless-architectures-learn-more/?nc1=h_ls
8. AWS Lambda [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/lambda/
9. Claudia bot builder [Електронний ресурс] – Режим доступу до ресурсу: https://claudiajs.com/claudia-bot-builder.html
10. INSTALLING AND CONFIGURING CLAUDIA.JS [Електронний ресурс] – Режим доступу до ресурсу: https://claudiajs.com/tutorials/installing.html
11. HELLO WORLD FROM API GATEWAY [Електронний ресурс] – Режим доступу до ресурсу: https://claudiajs.com/tutorials/hello-world-api-gateway.html

ДОДАТОК А ПРОГРАМНИЙ КОД

Bot.js

```
'use strict'
const botBuilder = require('claudia-bot-builder')
const pizzaDetails = require('./handlers/pizza-details')
const orderPizza = require('./handlers/order-pizza')
const pizzaMenu = require('./handlers/pizza-menu')
const saveLocation = require('./handlers/save-location')
const api = botBuilder((message) => {
if (message.postback) {
    const [action, pizzaId] = message.text.split('|')
if (values[0] === 'DETAILS') {
    return pizzaDetails(values[1])
} else if (values[0] === 'ORDER') {
    return orderPizza(values[1], message.sender)
 }
}
if (
message.originalRequest.message.attachments &&
message.originalRequest.message.attachments.length &&
message.originalRequest.message.attachments[0].payload.coordinates &&
message.originalRequest.message.attachments[0].payload.coordinates.lat &&
message.originalRequest.message.attachments[0].payload.coordinates.long
) {
return saveLocation(message.sender, message.originalRequest.message.
attachments[0].payload.coordinates)
}
if (
message.originalRequest.message.nlp &&
message.originalRequest.message.nlp.entities &&
message.originalRequest.message.nlp.entities['дякую'] &&
message.originalRequest.message.nlp.entities['дякую'].length &&
message.originalRequest.message.nlp.entities['дякую'][0].confidence > 0.8
```

```
    ) {
    return 'Вам дуже дякую, за те що нас замовили.
Гарного дня!'
    }
     return [
     'Привіт, друже! Ось такі піци у нас є в наявності:',
     pizzaMenu()
     ]
    }, {
     platforms: ['facebook']
    })
    module.exports = api
```

pizza-details.js

```
    'use strict'
    const pizzas = require('../data/pizzas.json')
    const     fbTemplate     =     require('claudia-bot-
builder').fbTemplate
    function pizzaDetails(id) {
     const pizza = pizzas.find(pizza => pizza.id == id)
     return [
     '${pizza.name} Інгредієнти:'  + pizza.ingredients.
     join(', '),
     new fbTemplate.Button('Що я можу зробити?')
     .addButton('Замовити', 'ORDER|${pizza.id}')
     .addButton('Показати піци', 'ALL_PIZZAS')
     .get()
     ]
    }
    module.exports = pizzaDetails
```

order-pizza.js

```
    'use strict'
    const AWS = require('aws-sdk')
    const docClient = new AWS.DynamoDB.DocumentClient()
    const pizzas = require('../data/pizzas.json')
    const pizzaMenu = require('./pizza-menu')
    const uuid = require('uuid/v4')
    function orderPizza(pizzaId, sender) {
     const  pizza  =  pizzas.find(pizza  =>  pizza.id  ==
pizzaId)
     return docClient.put({
     TableName: 'pizza-orders',
```

```
      Item: {
      orderId: uuid(),
      pizza: pizzaId,
      orderStatus: 'in-progress',
      platform: 'fb-messenger-chatbot',
      user: sender
      }
      }).promise()
      .then((res) => {
      return 'Куди саме доставити ваше замовлення?
Надішліть геолокацію, будь ласка'
      })
     }
     module.exports = orderPizza
```

save-location.js

```
     'use strict'
     const AWS = require('aws-sdk')
     const docClient = new AWS.DynamoDB.DocumentClient()
     function saveLocation(userId, adress) {
      return docClient.scan({
      TableName: 'pizza-orders',
      Limit: 1,
      FilterExpression: `user = :u, orderStatus: :s`,
      ExpressionAttributeNames: {
      ':u': { S: userId },
      ':s': { S: 'in-progress' }
      }
      }).promise()
      .then((result) => result.Items[0])
      .then((order) => {
      const orderId = order.orderId
      return docClient.update({
      TableName: 'pizza-orders',
      Key: {
      orderId: orderId
      },
      UpdateExpression:   'set   orderStatus   =   :s,
coords=:c',
      ExpressionAttributeValues: {
      ':s': 'pending',
      ':c': adress
      },
      ReturnValues: 'ALL_NEW'
```

```
 }).promise()
 })
}
module.exports = saveLocation
```

pizza-menu.js

```
'use strict'
const pizzas = require('../data/pizzas.json')
const fbTemplate = require('claudia-bot-builder').fbTemplate
function pizzaMenu() {
 const message = new fbTemplate.Generic()
 pizzas.forEach(pizza => {
 message.addBubble(pizza.name)
 .addImage(pizza.image)
 .addButton('Інгредієнти', `DETAILS|${pizza.id}`)
 .addButton('Замовити', `ORDER|${pizza.id}`)
 })
 return message.get()
}
module.exports = pizzaMenu
```

pizzas.json

```
[
 {
 "id": 1,
 "name": "ПІЦА АМЕРИКАНО",
 "image": "https://roll-club.kh.ua/wp-content/uploads/2014/08/mjasnoj-bum.jpg.webp",
 "ingredients": [
 "ковбаски мисливські", "салямі гостра", "шинка", "соус барбекю", "сир Моцарелла", "кукурудза", "цибуля"
 ]
 },
 {
 "id": 2,
 "name": "ПІЦА ГАВАЙСЬКА",
 "image": "https://roll-club.kh.ua/wp-content/uploads/2014/08/mjasnoj-bum.jpg.webp",
 "ingredients": [
 "філе куряче", "сир Моцарелла", "соус Пілаті", "томати чері", "ананас", "кукурудза", "базилік"
 ]
 },
```

```
      {
      "id": 3,
      "name": "ПІЦА ПЕПЕРОНЧІНІ",
      "image":                "https://roll-club.kh.ua/wp-
content/uploads/2014/08/mjasnoj-bum.jpg.webp",
      "ingredients": [
      "салямі гостра", "сир Моцарелла", "соус Пілаті",
"перець солодкий", "томати", "базилік"
      ]
      },
      {
      "id": 4,
      "name": "ПІЦА ТЕХАС",
      "image":                "https://roll-club.kh.ua/wp-
content/uploads/2014/08/mjasnoj-bum.jpg.webp",
      "ingredients": [
      "ковбаски мисливські", "філе куряче", "сир
Моцарелла", "соус барбекю", "цибуля червона"
      ]
      }
    ]
```