ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ТЕЛЕКОМУНІКАЦІЙ

Пояснювальна записка

до бакалаврської роботи на тему:

на тему: **«ЧАТ-БОТ ДЛЯ МОНІТОРИНГУ ФАКТИЧНОЇ ПОГОДИ НА АЕРОДРОМІ»**

Виконав: студент 4 курсу, групи
САД- 41
спеціальності
124 Системний аналіз
(шифр і назва спеціальності)
Добровольський О.В.
(прізвище та ініціали)

Керівник Дакова Л.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Нормоконтроль_____

КИЇВ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ

Кафедра Системного аналізу

Ступінь вищої освіти Бакалавр

Спеціальність 124 Системний аналіз
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри Системного аналізу
_____ Т.Б. Гордієнко
_____ _____ 2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

студенту Добровольському Олексію Вікторовичу

Тема роботи: «Чат-бот для моніторингу фактичної погоди на аеродромі», керівник роботи Дакова Лариса Валеріївна, к.т.н., затверджені наказом вищого навчального закладу від 16.02.2022 р. №22

1. Строк подання студентом роботи 22.05.2023 року.
2. Вихідні дані до роботи:
   1. Розробити програмний продукт для зручного та швидкого доступу до погодної інформації працівниками авіаційної галузі.
   2. Науково-технічна література.

3. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   3. Аналіз існуючих програмних засобів для моніторингу фактичної погоди на аеродромі.
   4. Вимоги до чат-боту для моніторингу фактичної погоди на аеродромі.
   5. Структура чат-боту для моніторингу фактичної погоди на аеродромі.
   6. Прототип чат-боту для моніторингу фактичної погоди на аеродромі.

4. Перелік графічного матеріалу (назва слайдів презентації):
   1. Переваги та недоліки існуючих програмних засобів для моніторингу фактичної погоди на аеродромі.
   2. Вимоги до чат-боту для моніторингу фактичної погоди на аеродромі.
   3. Засоби та технології для розробки чат-боту для моніторингу фактичної погоди на аеродромі.

4. Структура чат-боту для моніторингу фактичної погоди на аеродромі.
5. Робота прототипу чат-боту для моніторингу фактичної погоди на аеродромі.

5. Дата видачі завдання 18.02.2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1.	Підбір науково-технічної літератури	10.03.2023	Виконано
2.	Аналіз існуючих програмних засобів для моніторингу фактичної погоди на аеродром	25.03.2023	Виконано
3.	Вимоги до чат-боту для моніторингу фактичної погоди на аеродромі	06.04.2023	Виконано
4.	Структура чат-боту для моніторингу фактичної погоди на аеродромі	09.04.2023	Виконано
5	Прототип чат-боту для моніторингу фактичної погоди на аеродромі	02.05.2023	Виконано
6.	Висновки	10.05.2023	Виконано
7.	Оформлення пояснювальної записки	12.05.2023	Виконано
8.	Підготовка презентації	20.05.2023	Виконано

Студент _____ Добровольський О.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дакова Л.В.
(підпис) (прізвище та ініціали)

Р Е Ф Е Р А Т

Текстова частина дипломної роботи 49 с., 22 рис., 1 табл., 11 джерел.

Об`єкт дослідження – чат-бот для моніторингу фактичної погоди на аеродромі.

Мета роботи – надання швидкого доступу до погодної інформації працівниками авіаційної галузі та усунення недоліків існуючих програмних засобів для моніторингу фактичної погоди на аеродромі.

Предмет дослідження: розробкачат-бот для моніторингу фактичної погоди на аеродромі.

Було проаналізовано предметну область та визначено, що погода має важливий вплив на роботу авіації. Проведено огляд існуючих застосунків різних типів та виявлено їх переваги та недоліки. Розроблюваний чат-бот має мету усунути недоліки цих застосунків.

Проведено аналіз вимог до чат-боту, побудовано діаграму варіантів використання, що допомогла наглядно показати різних способи взаємодії користувача з чат-ботом. З використанням діаграми потоків даних було продемонстровано як кожен процес перетворює свої вхідні дані у вихідні.

Розроблено прототип чат-боту для моніторингу фактичної погоди на аеродромі та розміщено його на онлайн-порталі Azure для підключення різних каналів. Продемонстровано роботу розроблюваного прототипу. Отриманий чат-бот простий у використанні, не потребує встановлення, доступний на різних пристроях та надає швидкий доступ до погодної інформації.

Галузь використання – сучасні системи телекомунікацій України.

ПОГОДА, АЕРОДРОМ, ЧАТ-БОТ, ДІАЛОГОВИЙ ІНТЕРФЕЙС КОРИСТУВАЧА, METAR, МИТТЄВІ ПОВІДОМЛЕННЯ, БЕЗПЕКА ПОЛЬОТІВ.

ЗМІСТ

ВСТУП

Погода – один з головних факторів ризику в авіації. Авіаційні прогнози принципово відрізняються від повсякденних прогнозів погоди. Вони складаються кожну годину, а в складних випадках – кожні 15 хвилин. Прогноз погоди повинен містити точні дані по широкому спектру параметрів: швидкість і напрям вітру, характер хмарності, ймовірність турбулентності, обледеніння на різних висотах, тиск, вертикальна і горизонтальна видимість і т. д. Крім складання точних прогнозів, принципове значення має швидкість поширення інформації, оперативність оповіщення. Тому авіаційна метеорологія фактично є інформаційною галуззю і передбачає роботу з новітніми засобами інформаційних технологій.

Для моніторингу фактичної погоди на аеродромі розроблені різноманітні програмні засоби, що являють собою застосунки різних видів. Всі вони мають свої переваги та недоліки. Однак сьогодні існує відносно новий вид застосунків – чат-боти. Такі застосунки імітують розмову з користувачами за допомогою текстових повідомлень у чаті та швидко набирають популярності, оскільки все більше і більше людей починають користуватися месенджерами – застосунками для миттєвого обміну повідомленнями.

Месенджери доступні для сучасних мобільних пристроїв, персональних комп’ютерів, а також в браузері. Це означає, що інформацію про погоду можна буде отримати використовуючи більшість сучасних пристроїв.

Перевагами такого застосунку є простота у використанні, доступність на різних пристроях, надання лише необхідної інформації та відсутність потреби завантаження та встановлення на пристрій користувача.

Так, чат-бот для моніторингу фактичної погоди на аеродромі дозволить швидко поширювати погодну інформацію, а також вчасно сповіщати користувачів про оновлення даних. Отримана погодна інформація допоможе підвищити безпеку польотів, а також розумно планувати рейси літаків.

1. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ МОНІТОРИНГУ ФАКТИЧНОЇ ПОГОДИ НА АЕРОДРОМІ

1.1. Вплив погодних умов на авіацію

Погода значним чином впливає на роботу авіації. Точний прогноз погоди так само важливий, як і перевірка літака ліцензованим механіком. Авіаційна метеорологія – це надзвичайно важливий елемент складної системи управління повітряним рухом.

На відміну від наземного транспорту авіація взагалі більш залежна від погодних умов. Найчастіше скасування рейсів відбуваються саме через погану погоду. На організацію польотів на літаку впливають різні фактори пов’язані з погодою: звичайні параметри, пов’язані з напрямком і силою вітру, а також більш конкретні явища, такі як сніг та туман.

Погода може істотно вплинути на роботу літаків. Низька хмарність, туман і дощ можуть ускладнювати видимість в аеропорту або навколо нього, а грози і блискавки можуть серйозно порушити розклад польотів.

Грози та супроводжуючі їх повітряні потоки, що швидко піднімаються або падають, можуть зробити політ незручним для пасажирів і складним для пілотів, що керують літаком. Літаки не можуть злітати або приземлятися під час шторму, і їх часто перенаправляють навколо штормових осередків або відхиляють від місця призначення. Грози і удари блискавок біля аеропортів також можуть зупинити наземні операції до тих пір, поки вони не пройдуть [8].

Хоча сучасні літаки стають все більш всепогодними, в порівнянні з тими літальними апаратами, які були на зорі розвитку авіації, увагу до прогноза погоди і моніторинг поточних метеорологічних умов і сьогодні залишається невід’ємною частиною планування польотів будь-яких видів авіації.

Досвідчені пілоти літаків достатньо добре знайомі з небезпеками обмерзання (рис. 1.1), висоти основи хмар і проблем з видимістю. Якщо пілоти не

інформовані про погодні умови навколо них, то ймовірність виникнення авіакатастроф зростає.

Рис. 1.1. Обмерзання на крилі літака

Погода, зокрема швидкість і напрям вітру, зазвичай є основним фактором, що визначає, які злітно-посадкові смуги (ЗПС) використовувати в аеропорту, в якому напрямку будуть злітати і приземлятися повітряні судна та які маршрути польоту використовуються.

Літак повинен злітати і приземлятися проти вітру або з мінімальним попутним вітром. Це означає, що поточний і прогнозований напрямок вітру визначає вибір використовуваних ЗПС у будь-який час. Напрямок вітру може змінитися в найкоротші терміни, що може вплинути на траєкторію польоту [8].

Вітер, що дме через злітно-посадкову смугу, називається бічним вітром. Як правило, літак може злітати або приземлятися при слабкому бічному вітрі, зазвичай до швидкості близько 28 км / год. Бічний вітер вище цієї швидкості може змусити використовувати іншу ЗПС.

Градієнт вітру – це раптова зміна напряму або швидкості вітру, зазвичай пов’язана з грозою. Зрушення вітру може бути вертикальним або горизонтальним і може мати значний вплив на управління повітряним судном під час зльоту і посадки та, через можливу втрату контролю, є значним фактором ризику в авіації.

Туман – одна з найбільших небезпек для авіації через значне погіршення видимості (рис. 1.2). Можливість працювати в тумані, залежить від навичок пілота, обладнання літака, приладів на аеродромі та критеріїв посадки/зльоту аеропорту. Наземні операції під час туману будуть значно уповільнені або припинені повністю, що призводить до затримок [8].

Рис. 1.2. Посадка літака в тумані

Як бачимо, погодні умови значним чином впливають на авіацію. Надійна інформація про погоду дозволяє забезпечити безпечну навігацію літаків між різними географічними точками. Погодної інформація може бути корисною і для інших цілей. Наприклад для економії палива, шляхом розумного використання прогнозів вітру для прискорення руху літаків в польоті.

Саме тому важливо бути проінформованим про погодні умови. Сьогодні існують різноманітні програмні засоби для моніторингу фактичної погоди на аеродромі. Далі розглянемо деякі з них.

1.2. Аналіз застосунків різних видів для моніторингу фактичної погоди на аеродромі

Застосунки можуть бути створені різними способами, орієнтованими на різні платформи. Визначення правильного виду застсунку має вирішальне значення для успіху проекту – як з точки зору вартості, так і з точки зору сприйняття користувачами.

Серед існуючих застосунків для моніторингу фактичної погоди на аеродромі існують десктопні, веб та мобільні застосунки. Проаналізуємо їх особливості та виділимо основін перваги та недоліки.

Десктопний застосунок – це програмне забезпечення, яке може бути запущене на автономному комп'ютері для виконання певного завдання кінцевим користувачем. Ключовими особливостями десктопних додатків є широка ефективність програми, а також їх гнучкість і можливість налаштування відповідно до вимог користувача. Застосунок має графічний інтерфейс користувача та потребує завантаження та встановлення на пристрій користувача.

Metar Monitor – це десктопний застосунок для відображення детальної інформації про фактичну погоду на місцевому або будь-якому іншому аеродромі [9]. На рис. 1.3 наведено інтерфейс користувача Metar Monitor.

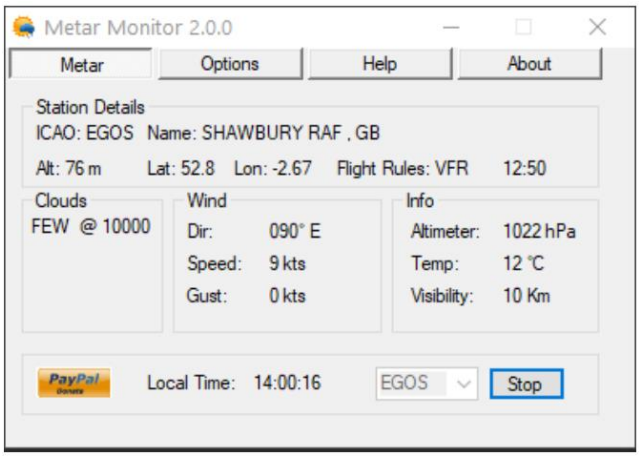

Рис. 1.3. Інтерфейс користувача Metar Monitor

Виділимо переваги та недоліки десктопного застосунку Metar Monitor.

Переваги Metar Monitor:

- простий у використанні;
- зручний інтерфейс користувача;
- безкоштовний (є можливість фінансово підтримати розробників).

Недоліки Metar Monitor:

- доступний лише для персонального комп'ютера;
- потребує встановлення.

Веб-застосунок – програмне забезпечення створене для роботи в веб-браузерах. Його можна охарактеризувати гнучкістю, тобто можливістю підлаштовуватися під розмір екрану, включаючи телефони та планшети.

The Aviation Weather Center (AWC) являє собою багатофункціональний веб-застосунок. Однією з можливостей AWC є надання цілісної, своєчасної та точної інформації про погоду для світової системи повітряного простору [10]. На рис. 1.4 наведено інтерфейс користувача AWC.

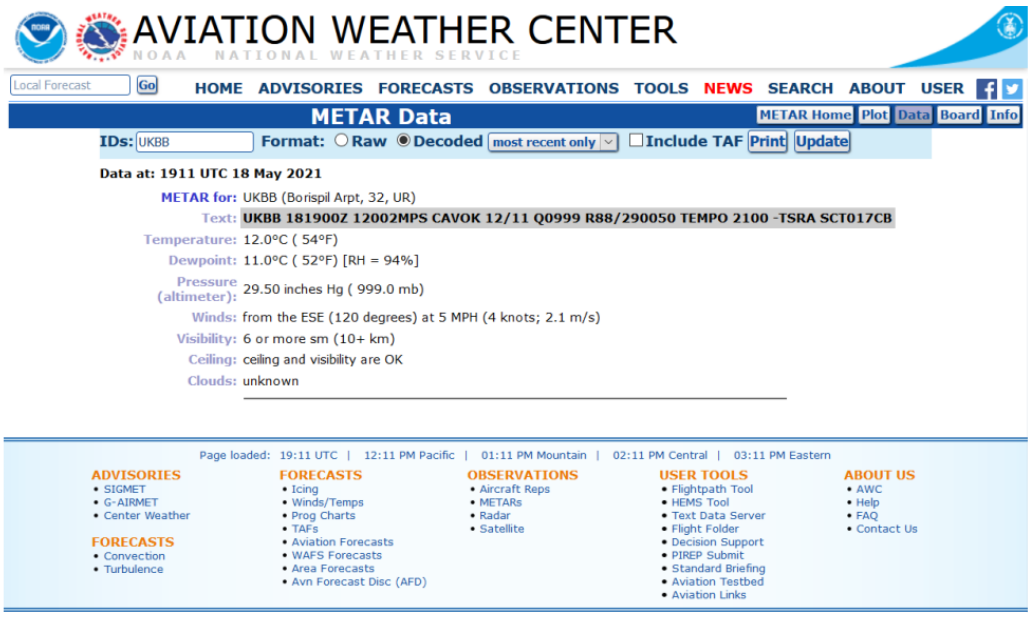

Рис. 1.4. Інтерфейс користувача AWC

Виділимо переваги та недоліки веб-застосунку AWC.

Переваги AWC:

•зручний інтерфейс користувача;

• можливість використання карти для вибору аеродрому;

• безкоштовний;

• доступний з різноманітних пристроїв;

• не потребує встановлення.

Недоліки AWC:

• потребує завантаження веб-сторінки, а не лише погодної інформації.

Мобільний застосунок – програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях, розроблене для конкретної платформи. Застосунок має графічний інтерфейс користувача та потребує встановлення на пристрій користувача.

З ростом кількості мобільних додатків, доступних в магазинах додатків, і поліпшенням можливостей смартфонів, люди завантажують на свої пристрої все більше додатків. Мобільні додатки стають все більш популярними серед користувачів мобільних телефонів.

Avia Weather – надійний та простий мобільний застосунок Avia Weather для пілотів та любителів авіакосмічній галузі. Застосунок надає інформацію для більш ніж 9500 аеропортів [11]. На рис. 1.5 наведено інтерфейс користувача Avia Weather. Виділимо переваги та недоліки мобільного застосунку Avia Weather.

Переваги Avia Weather:

• простий у використанні;

• зручний інтерфейс користувача;

• безкоштовний (має декілька платних можливостей).

Недоліки Avia Weather:

• доступний лише для мобільних пристроїв;

• потребує встановлення.

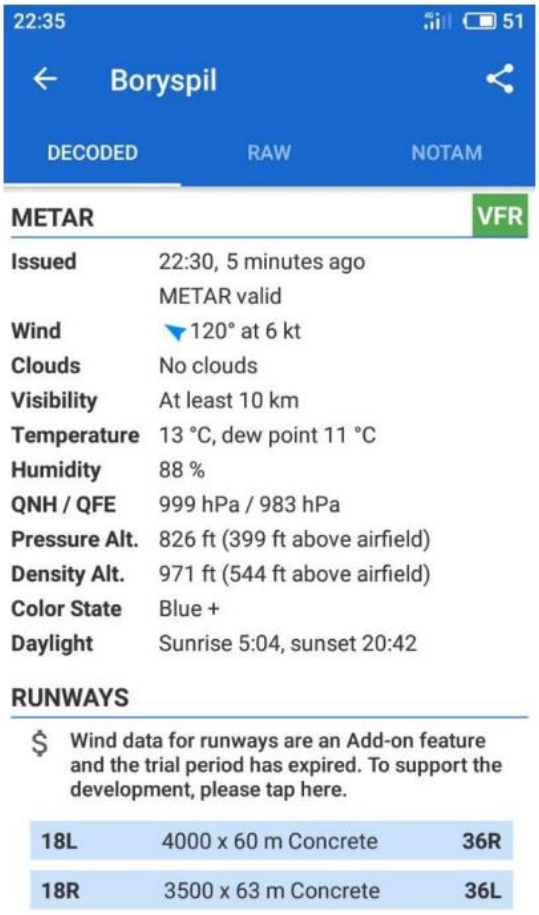

Рис. 1.5. Інтерфейс користувача Avia Weather

З різних причин вартість розробки мобільних додатків часто перевищує вартість веб-розробки. Крім того, кількість користувачів, які можуть отримати доступ до додатку, значно збільшується, якщо це веб-додаток. Також, випускати оновлення з веб-додатком набагато простіше, так як мобільні і настільні додатки повинні бути упаковані, а потім завантажені і встановлені користувачем.

Всі ці додатки являють собою різні види застосунків (десктопний, веб та мобільний) та мають свої переваги та недоліки. Можна врахувати вищеперераховані недоліки та розробити новий застосунок, в якому їх можна усунути.

1.3. Усунення недоліків існуючих програмних засобів для моніторингу фактичної погоди на аеродромі шляхом розробки чат-боту

Вище було розглянуто переваги та недоліки застосунків різних видів для моніторингу фактичної погоди на аеродромі. Варти знайти максимально просте рішення для усунення наявних недоліків.

Сьогодні існує відносно новий вид застосунків – чат-боти. Що таке чат-бот? Існує безліч думок, але немає єдиного загальноприйнятого формального визначення чат-бота. У кількох словниках згадуються боти, але не зовсім ясно, що таке сучасний чат-бот. Наступне визначення може не потрапити в словник, хоча воно містить опис того, що таке чат-бот, що він може робити, і тип платформи, на якій він знаходиться.

Чат-бот – це застосунок, часто доступний через платформи обміну повідомленнями (месенджери) та використовує деяку форму інтелекту, яка взаємодіє з користувачем через діалоговий інтерфейс користувача (CUI – Conversational User Interface) [1].

Більш детальне вивчення цього визначення показує, що чат-бот – це додаток, тобто програма, яку пише розробник. По суті, чат-бот приймає вхідні дані від користувача, обробляє їх та надає відповідь користувачеві.

Чат-боти використовують текстовий підхід. Взаємодія з користувачем відбувається у так званих месенджерах (наприклад, Telegram або Facebook Messenger) – системах обміну миттєвими повідомленнями.

Додатки для обміну повідомленнями – одне з найпопулярніших місць для взаємодії людей один з одним. Тому логічно, що додатки у вигляді чат-ботів доступні там, де зазвичай спілкуються люди. Люди спілкуються з чат-ботами за допомогою тексту і, можливо, навіть голосу.

Чат-боти використовують CUI, а не графічні інтерфейси користувача (GUI – Graphical User Interface). GUI – це інтерфейс, що відображає на екрані графічні елементи і зображення для взаємодії за допомогою миші або дотику. CUI – це інтерфейс, що використовує текст, який користувач вводить в додаток для

обміну повідомленнями. Багато з доступних сьогодні чат-ботів в основному текстові, але чат-бот також може використовувати голос для взаємодії з користувачем [1].

Може здатися, що CUI – це інтерфейс командного рядка (CLI – Command Line Interface). Однак є величезна різниця, оскільки CUI взаємодіє з користувачем імітуючи розмову, а CLI – за допомогою введення команд [1]. На рис. 1.6 наведено приклад CUI.

Є й інші причини того, чому чат-боти є хорошим вибором платформи для створення додатків, в тому числі: простота впровадження, універсальність пристрою і незалежність від платформи.

Десктопні та мобільні застосунки необхідно скачати та встановити. Проблема з веб-застосунками в тому, що вони іноді не працюють в певному браузері. Ці проблеми відсутні в чат-ботах, взаємодія з якими відбувається в додатках для обміну повідомленнями, які люди вже використовують. Люди просто говорять «Привіт» чат-боту, і він починає взаємодіяти з ними.

Рис. 1.6. Приклад діалогового інтерфейсу користувача

Враховуючи вищевказану інформацію можна визначити переваги та недоліки потенційного чат-бот для моніторингу фактичної погоди на аеродромі.

Переваги:

- простий у використанні;
- не потребує встановлення;
- доступний на різних пристроях;
- надає лише необхідну інформацію;
- швидкий доступ до необхідної інформації.

Недоліки:

- потребує наявності месенджеру у користувача.

Як бачимо, впровадження чат-боту дозволяє позбутися недоліків, що присутні в наведених вище застосунках. Для взаємодії з чат-ботом користувач повинен використовувати месенджери. Однак їх використання не є проблемою, оскільки в сучасному світі месенджерами починають користуватися все більше людей, серед них і працівники авіаційної галузі.

Таким чином користувачі зможуть отримати доступ до інформації про погоду використовуючи свій улюблений месенджер, будь то Telegram, Facebook Messenger або інший месенджер з підтримкою чат-ботів. При цьому месенджери доступні для сучасних мобільних пристроїв, персональних комп’ютерів, а також доступні в браузері.

Висновок

В першому розділі було описано вплив погодних умов на авіацію. Зокрема, було визначено, що погода значним чином впливає на роботу авіації, тому необхідно мати засіб, що може швидко та якісно надати інформацію про погоду.

Проведено аналіз існуючих програмних засобів для моніторингу фактичної погоди на аеродромі, а також виявлено їх переваги та недоліки. Після цього розглянуто, як усунути наявні недоліки існуючих засобів шляхом розробки чат-боту, що дозволить швидко та без зайвих деталей надати інформацію про фактичну погоду на будь-якому сучасному пристрої з доступом до месседжерів.

2. ВИМОГИ ДО ЧАТ-БОТУ ДЛЯ МОНІТОРИНГУ ФАКТИЧНОЇ ПОГОДИ НА АЕРОДРОМІ

2.1. Загальний опис інженерних вимог

Розробники програмного забезпечення прикладають значні зусилля для збору та документування вимог до програмного забезпечення, а також управління ними. Недостатній обсяг інформації, що надходить від користувачів, вимоги, сформульовані не в повному обсязі, їх кардинальна зміна – основні причини, через які найчастіше командам, які працюють в області інформаційних технологій, не вдається вчасно і в рамках бюджету надати клієнтам всю заплановану функціональність. [3]

Перш ніж приступати до розробки будь-якої системи необхідно спочатку визначити, для чого ця система потрібна. Якщо ж мета створення системи не визначена, то абсолютно незрозуміло який система повинна бути і неможливо навіть припустити влаштує розроблена система її користувачів.

Думати про потенційне вирішення проблеми можна лише тільки після того, як розроблений повний набір призначених для користувача вимог, оскільки це означає, що вже визначено саме те, що користувач зможе отримати від розроблюваної системи.

Інженерія вимог є широко поширеним і звичним терміном, однак, не дивлячись на це, для значної частини людей він до сих пір залишається незрозумілим. Це вже саме по собі є причиною того, що вимоги бувають дуже часто некоректно сформовані, а такими вимогами достатньо складно управляти [5].

Інженерія вимог – це процес виявлення потреб та бажань зацікавлених сторін та їх перетворення в узгоджений набір детальних вимог, які можуть слугувати основою для всіх наступних етапів розробки.

Вимоги до ПЗ – задокументований, узгоджений та затверджений набір основних властивостей створюваного програмного забезпечення, який забезпечує досягнення цілей розробки програмного забезпечення.

2.2. Функціональні вимоги

Функціональні вимоги – це можливості продукту, які розробники повинні реалізувати, щоб користувачі змогли використовувати програмне забезпечення для вирішення задач, для яких воно розробляється. Тому важливо чітко визначити їх як для команди розробників, так і для зацікавлених сторін проекту. Як правило, функціональні вимоги описують поведінку системи в певних умовах.

Функціональні вимоги допомагають команді розробників рухатися в правильному напрямку, а також перевірити, чи надає програмне забезпечення можливості, які очікує від нього користувач.

Сформуємо функціональні вимоги до чат-боту для моніторингу фактичної погоди на аеродромі, тобто вимоги, що визначають рівень функціональності застосунку:

- отримання інформації про погоду за кодом аеропорту;
- отримання інформації про погоду за назвою регіону та назвою аеропорту;
- можливість підписатися на отримання актуальних погодних даних з аеродрому;
- можливість відписатися від отримання актуальних погодних даних з аеродрому;
- можливість встановити частоту отримання повідомлень про погоду.

Функціональні вимоги можна описати за допомогою історій користувача (User Stories). Історії користувача – це опис можливостей програмного забезпечення з точки зору кінцевого користувача. Історії користувача описують, що саме користувач хоче від системи.

Історії користувача – швидкий спосіб оперування вимогами користувача, без необхідності застосування занадто формалізованих документів, та виконання адміністративних задач пов'язаних з опрацюванням цих документів. Наміром з яким використовують історії користувача є швидше та менш накладне реагування на швидко змінювані вимоги реального світу.

Опишемо історії користувача чат-боту для моніторингу фактичної погоди на аеродромі:

- як користувач, я хочу отримати актуальну інформацію про погоду за кодом аеропорту, щоб бути проінформований про погодні умови на аеродромі;
- як користувач, я хочу отримати актуальну інформацію про погоду за назвою регіону та назвою аеропорту, щоб бути проінформований про погодні умови на аеродромі;
- як користувач, я хочу підписатися на отримання актуальної погодної інформації з аеродрому, щоб проводити моніторинг фактичної погоди на аеродромі;
- як користувач, я хочу відписатися від отримання актуальної погодної інформації з аеродрому, бо більше не маю потреби в моніторингу фактичної погоди на аеродромі;
- як користувач, я хочу встановити частоту отримання повідомлень про погоду, щоб отримувати погодну інформацію з заданою періодичністю.

2.3. Діаграма варіантів використання

Для демонстрації різних способів взаємодії користувача з чат-ботом використаємо UML (Unified Modeling Language) діаграму варіантів використання (Use Case Diagram). Ключовою концепцією моделювання варіантів використання є те, що вона допомагає нам проектувати систему з точки зору кінцевого користувача. Це ефективний метод передачі поведінки системи в термінах користувача шляхом опису всієї видимої поведінки системи.

Для побудови діаграми використаємо такі компоненти:

• варіанти використання (прецеденти) – зовнішні специфікації послідовності дій, які система або інша сутність можуть виконувати в процесі взаємодії з акторами, без розгляду внутрішньої структури цієї системи;

• актори (діючі особи) – будь-які об’єкти, суб’єкти або системи, що взаємодіють з чат-ботом;

• відношення – семантичний зв’язок між окремими елементами моделі.

Варіанти використання мають різні види відношень. Відношення між двома варіантами використання в основному моделюють залежність між ними. На рис. 2.1 показано діаграму варіантів використання чат-боту для моніторингу фактичної погоди на аеродромі.

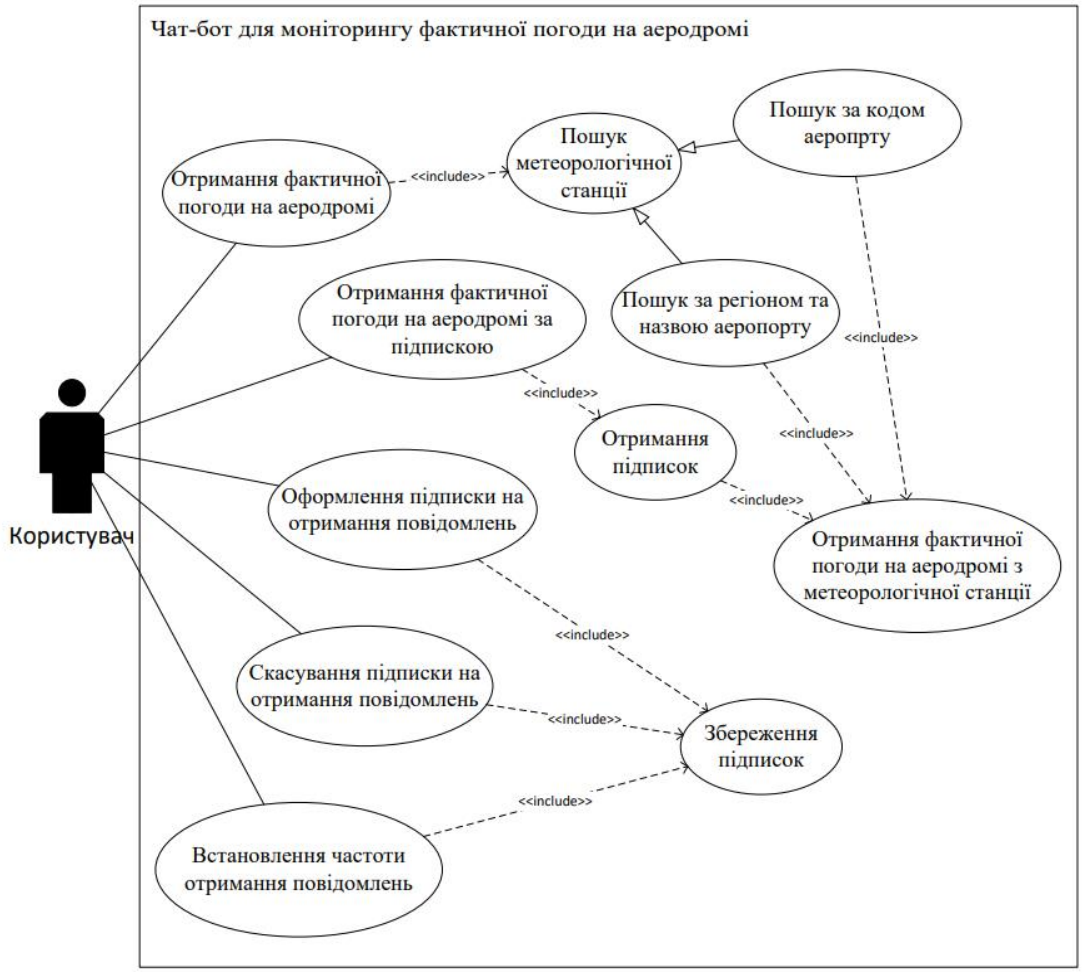

Рис. 2.1. Діаграма варіантів використання чат-боту для моніторингу фактичної погоди на аеродромі

2.4. Діаграма потоків даних

Діаграми потоків даних (Data Flow Diagrams) є одним із засобів моделювання функціональних вимог. Вони використовуються для розбиття вимог на функціональні компоненти (процеси) та представлення їх як мережі, пов’язаної потоками даних. Основна мета таких інструментів – продемонструвати, як кожен процес перетворює свої вхідні дані у вихідні, а також виявити взаємозв’язок між цими процесами.

Основними компонентами діаграм потоків даних є:

- зовнішні сутності - матеріальні об’єкти або особи, які є джерелом або одержувачем інформації;
- процеси - діяльність з перетворення вхідних потоків даних у вихідні за певним алгоритмом;
- сховище даних – абстрактний пристрій, що призначений для зберігання інформації;
- потоки даних – механізми моделювання передачі інформація від джерела до одержувача.

На рис. 2.2. наведено діаграму потоків даних нульового рівня. Діаграма має зіркоподібну топологію, в центрі якої знаходиться основний процес, підключений до приймачів та джерел інформації, за допомогою яких користувачі та інші системи взаємодіють із системою.

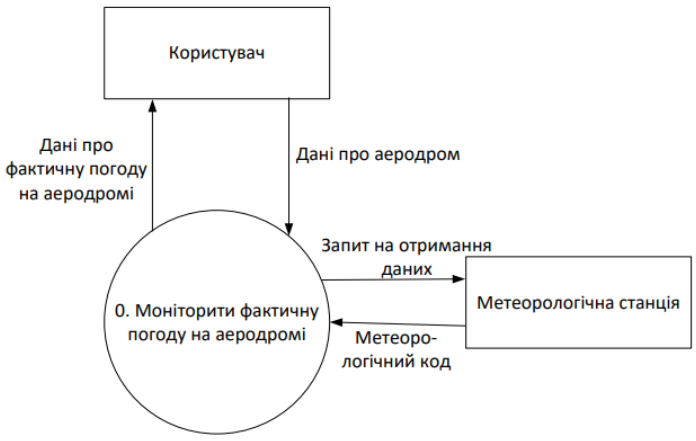

Рис. 2.2. Діаграма потоків даних нульового рівня

Після опису основного процесу проводиться декомпозиція, зокрема визначаються процеси, що складають основний процес. Потім процеси поділяються на підпроцеси до необхідного рівня деталізації. Так будується ієрархія діаграм потоків даних. На рис. 2.3 наведено діаграму потоків даних першого рівня.

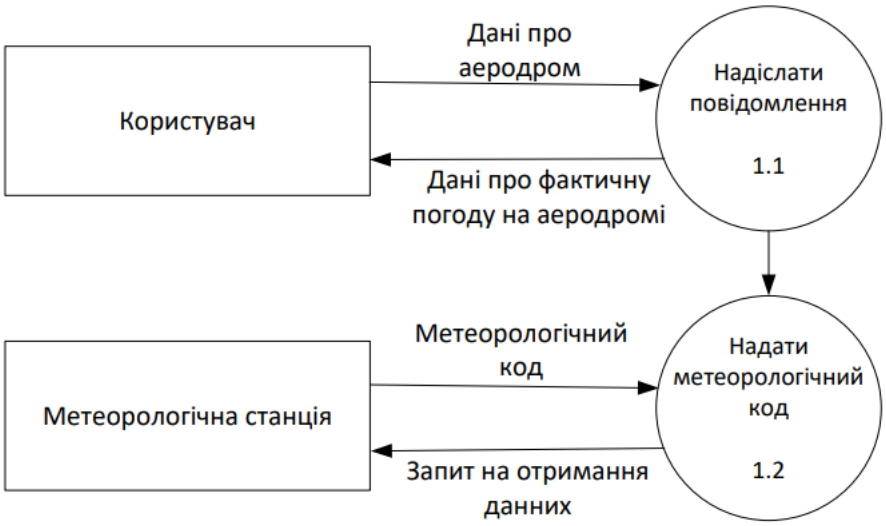

Рис. 2.3. Діаграма потоків даних першого рівня

Декомпозицію процесів можна проводити й далі, однак зупинимося на першому рівні, оскільки діаграма достатнього наглядно демонструє перетворення вхідних даних у вихідні.

2.4. Нефункціональні вимоги

Сформуємо нефункціональні вимоги до чат-боту для моніторингу фактичної погоди на аеродромі, тобто вимоги, що визначають якість функціональності, а також умови реалізації необхідної функціональності застосунку:

- простота використання;
- інтерфейс, доступний різними мовами;

- час відповіді на повідомлення користувача повинен бути не більше 5 секунд;
- розгортання на сервері незалежно від операційної системи;
- обробка не коректного вводу користувача;
- ведення журналу подій, що відбулися в системі.

Висновок

В другому розділі проведено інженерію вимог до чат-боту для моніторингу фактичної погоди на аеродромі. Визначено функціональні вимоги та представлено їх за допомогою історій користувача. Побудовано діаграму варіантів використання розроблюваного чат-боту, що демонструє різні способи взаємодії користувача з системою. Для кращого розуміння діаграми було здійснено вербальний опис варіантів використання. Для наглядної демонстрації перетворення процесами вхідних даних у вихідні побудовано діаграму потоків даних.

Також, описано нефункціональні вимоги, що допомогли визначити властивості та обмеження системи. Так, визначено властивості програмного забезпечення які забезпечують задоволення потреб користувачів.

3. СТРУКТУРА ЧАТ-БОТУ ДЛЯ МОНІТОРИНГУ ФАКТИЧНОЇ ПОГОДИ НА АЕРОДРОМІ

3.1. Вибір засобів та технологій для розробки

Чат-боти призначені для вирішення загальних задач, використовуючи природний стиль спілкування у поєднанні з машинним навчанням для вдосконаленого інтелекту. Для більш вдалого архітектурного проектування застосунку варто визначити засоби та технології, що будуть використовуватися для розробки чат-боту для моніторингу фактичної погоди на аеродромі.

.NET – це безкоштовна платформа для операційних систем Windows, Linux і macOS для розробників з відкритим вихідним кодом, розроблена компанією Microsoft. Прагнення Microsoft надати розробникам єдину платформу для вирішення будьяких проблем було реалізовано за допомогою. NET. З використанням платформи .NET вже пару десятиліть розробляють веб-додатки, десктопні та мобільні додатки різної складності.

.NET підтримує Common Language Infrastructure, тому вихідний код застосунку компілюється в Common Intermediate Language незалежно від використовуваної мови програмування. Це гарантує відмінну сумісність між мовами платформи.

.NET включає CLR (Common Language Runtime) – це середовище виконання, яка включає підтримку Windows, macOS і Linux. CLR відповідає за виділення пам’яті та управління нею. CLR також є віртуальною машиною, яка не тільки виконує додатки, але також генерує і компілює код за допомогою JIT-компілятора [6]. Останньою версією на сьогоднішній день є .NET 5.

C# – це сучасна об’єктно-орієнтована мова програмування загального призначення, розроблений Microsoft під час розробки .NET Framework.

C# має ряд особливостей, таких як лямбда-вирази і анонімні типи даних, які традиційно присутні в різних функціональних мовах [6]. Більш того, з появою

Language Integrated Query (LINQ) C# підтримує ряд конструкцій, які роблять її унікальним в середовищі програмування.

Нижче наведено список деяких важливих особливостей C#:

- логічні вирази;
- автоматична очистка пам’яті;
- управління версіями збірок;
- властивості;
- делегати та події;
- прості у використанні узагальнення;
- індексатори;
- умовна компіляція;
- проста багатопоточність;
- LINQ та лямбда-вирази;
- механізм рефлексії. Останньою версією на сьогоднішній день є C# 9.

Microsoft Bot Framework

Microsoft Bot Framework – це фреймворк для створення інтелектуальних і інтуїтивно зрозумілих чат-ботів, які будуть доступні через звичні засоби зв’язку, такі як Telegram, Skype, Slack, Teams та інші популярні засоби, без будь-яких додаткових зусиль [4].

Платформа забезпечує підтримку управління користувачами, управління сеансами, управління станом, аутентифікації і розмовних моделей, таких як діалоги, дії, картки або вкладення. Пакет SDK для ботів має відкритий вихідний код.

Microsoft Bot Framework містить наступні компоненти:

- портал розробника Bot Framework – служба, яка підключає чат-бота до засобів зв’язку.
- bot Builder SDK – бібліотека для однієї з підтримуваних мов програмування, яка надає потужні засоби для створення чат-ботів. На даний момент засоби для розробки з використанням Microsoft Bot Framework доступні для C#, JavaScript, Typescript та Python. Також активно ведеться розробка засобів для Java.

• каталог ботів – каталог ботів, до яких можна підключитися за допомогою Bot Connector [2]. Сьогодні, останньою версією SDK для мови програмування C# є 4.0.

Entity Framework Core

Entity Framework Core – фреймворк для об'єктно-реляційної проекції (ORM – Object-relational mapping). Entity Framework Core надає розробникам автоматизований механізм доступу і зберігання даних в базі даних.

Фреймворк дозволяє розробникам .NET працювати з реляційними даними за допомогою об'єктів. Це усуває необхідність в написанні значної частини коду для доступа до даних, що зазвичай необхідно розробникам.

Останньою версією на сьогоднішній день є Entity Framework Core 5.0.

На рис. 3.1 показано принцип роботи Entity Framework Core.

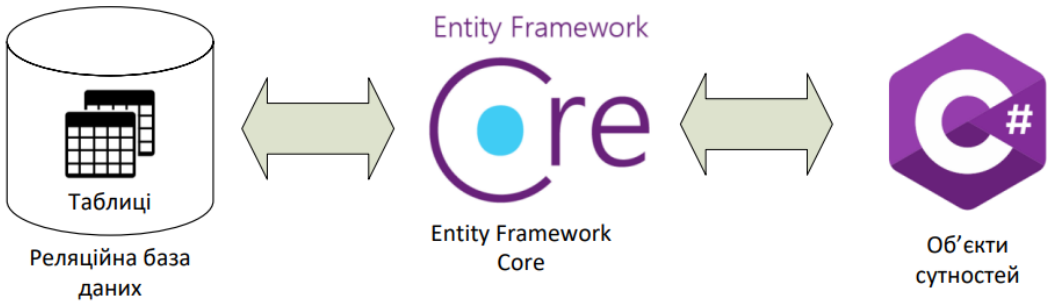

Рис. 3.1. Ппринцип роботи Entity Framework Core

Visual Studio Visual Studio – це інтегроване середовище розробки, що використовується для редагування, налагодження і побудови коду, а потім для публікації додатка. Це багатофункціональне програмне забезпечення, що використовується для багатьох аспектів розробки програмного забезпечення.

Крім стандартного редактора, що надається більшістю IDE, Visual Studio включає в себе компілятори, інструменти автозаповнення коду, графічні конструктори і багато інших функцій, що спрощують процес розробки

програмного забезпечення. Останньою версією на сьогоднішній день є Visual Studio 2019 16.0.

Bot Framework Emulator – автономний додаток, який не тільки надає інтерфейс чату, але також надає інструменти для налагодження, які допомагають зрозуміти, як і чому бот поводиться так, як поводиться. Емулятор можна запускати локально разом із чат-ботом, що розробляється [1]. Використовуючи емулятор, можна взаємодіяти зі своїм ботом і перевіряти повідомлення, які ваш бот відправляє і отримує. Емулятор відображає повідомлення так, ніби вони відображаються в інтерфейсі веб-чату, і реєструє запити і відповіді у JSON-форматі, коли ви обмінюєтеся повідомленнями з ботом.

Azure – це онлайн-портал, який дозволяє отримувати доступ до хмарних служб та ресурсів, що надаються Microsoft, а також керувати ними. Ці послуги і ресурси включають в себе зберігання даних і їх перетворення в залежності від вимог розробників. Щоб отримати доступ до цих ресурсів і службам, все, що потрібно – це активне підключення до Інтернету і можливість підключитися до портала Azure. Маючи більш ніж 200 сервісів та численні переваги, Microsoft Azure, безсумнівно, є найбільш швидкозростаючою платформою хмарних обчислень.

3.2. Опис архітектури

При розробці чат-боту важливо знати, як його компоненти поєднуються один з одним, якими шляхами проходить повідомлення, з чим взаємодіє бот або які сервіси доступні.

Архітектура чат-боту для моніторингу фактичної погоди на аеродромі достатньо проста: основними будівельними блоками є канали (месенджери), Bot Connector, бот-застосунок (рис. 3.2).

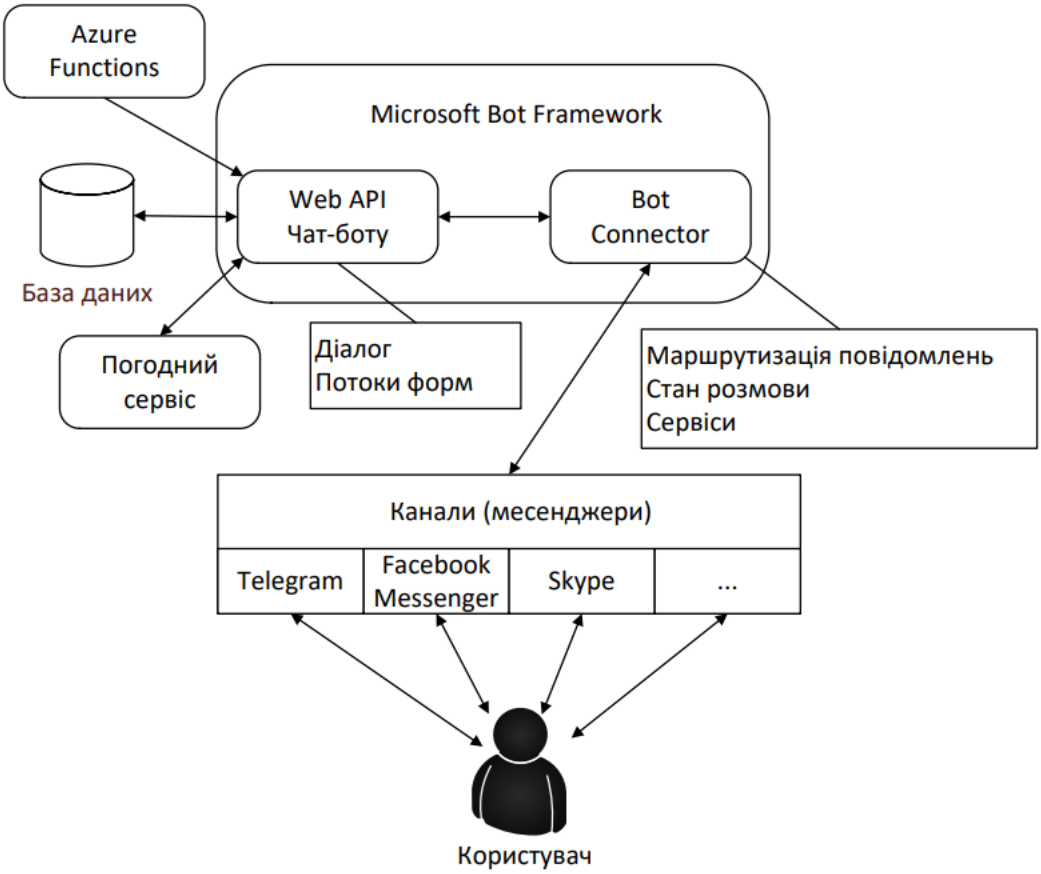

Рис. 3.2. Архітектура чат-боту для моніторингу фактичної погоди на аеродромі

Кожен з цих компонентів виконує свої функції. Варто детально розглянути кожен з них. Канали – це додатки, за допомогою яких користувач взаємодіє з чат-ботом. Канали часто асоціюються з додатками для обміну повідомленнями, наприклад Telegram, Skype або Facebook Messenger. Хоча це й дійсно так, каналом також може бути будь-яка інша програма, що відправляє та приймає повідомлення в Bot Connector. Крім додатків для обміну повідомленнями, Bot Framework підтримує електронну пошту, SMS і веб-сайти. Bot Connector – це хмарний компонент Microsoft, який відправляє повідомлення у канали і отримує повідомлення від них. Потім Bot Connector відправляє і отримує повідомлення з чату [2]. Однак є ще декілька особливостей, що стосуються Bot Connector.

Bot Connector зберігає стан, тобто настроюється інформацію, яку чат-бот може зберегти. Чат-бот може зберігати дані про розмову, користувача або користувача, що бере участь в розмові. Bot Connector діє як адаптер між різними каналами та чатботом, забезпечуючи аутентифікацію бота, серіалізацію з широким спектром каналів.

Чат-бот взаємодіє з Bot Connector через повідомлення, що називаються діями (activities). Ці дії можуть бути текстом між користувачем і чат-ботом, або іншим видом повідомлення. Будь-якому випадку чат-бот отримує повідомлення у вигляді дії, що десеріалізується з JSON-рядка.

В базі даних чат-боту зберігаються дані про підписку на отримання погодної інформації. Такими даними є ідентифікатор користувача, код аеропорту та частота сповіщень.

Зазвичай чат-бот відправляє повідомлення користувачу безпосередньо у відповідь на отримання повідомлення від користувача. Однак, у випадку сповіщень, чат-боту може знадобитися відправити проактивне повідомлення – повідомлення у відповідь на певну подію, що зазвичай ініціюється зовнішньою системою [7]. В якості такої системи практично буде використати Azure Functions, що є зручними та простими у використанні.

Azure Functions використовуються для виконання невеликих фрагментів коду, які називаються функціями, незалежно від інфраструктури додатки. З їх допомогою відбувається відправка сповіщень користувачу. Функція автоматично спрацьовує при виконанні певного типу події. Тобто, у заданий користувачем час буде спрацьовувати функція, що відправляє сповіщення користувачу.

Обмін даними між каналами, Bot Connector і чат-ботом є частиною архітектури чат-боту для моніторингу фактичної погоди на аеродромі. На рис. 3.3 показано шляхи взаємодії між каналом, Bot Connector, чат-ботом та погодним сервісом.

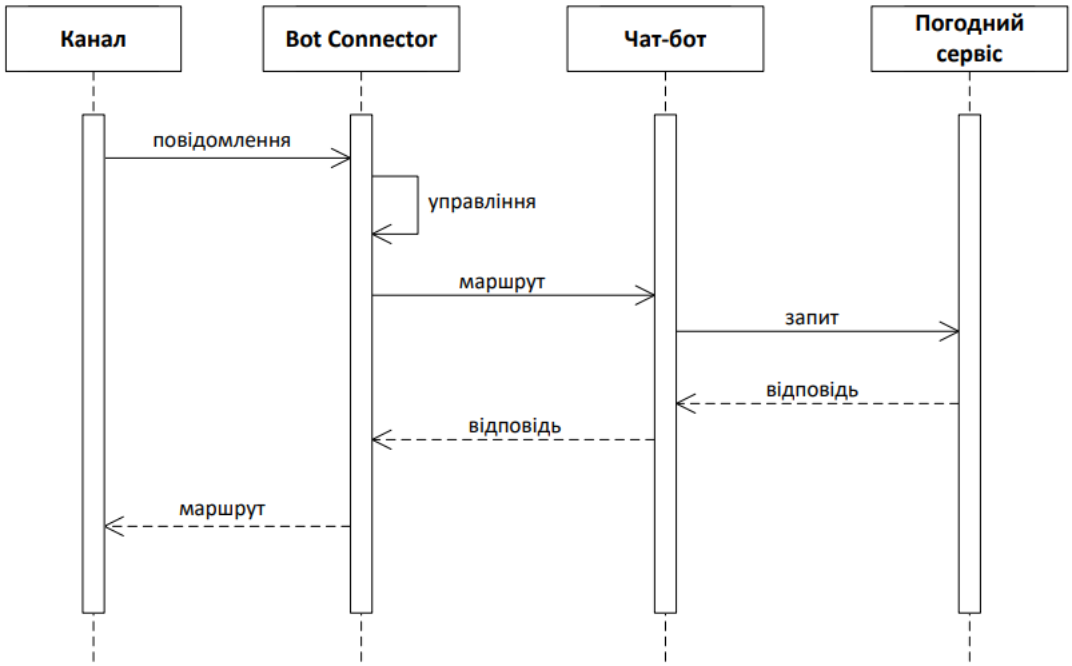

Рис. 3.3. Шляхи взаємодії між каналом, Bot Connector, чат-ботом та погодним сервісом

Кожна зі стрілок між об’єктами представляє повідомлення, передане між компонентами, а, більш конкретно, кожна з цих стрілок представляє обмін даними через Інтернет. Якщо у користувача повільне з’єднання з Інтернетом, то його робота з чат-ботом погіршується, і, навпаки, покращується якщо підключення до Інтернету хороше [2]. Як бачимо, архітектура розроблюваного чат-боту достатньо проста, хоча й потребує зусиль для її реалізації.

3.3. Опис взаємодії класів

Важливо визначити які класи буде мати чат-бот, які зв’язки будуть між ними та як вони будуть взаємодіяти між собою. Для цього використовується діаграма класів UML, що описує структуру системи, показуючи класи системи, їх атрибути, операції та взаємозв’язки між об’єктами.

Необхідно визначити як саме чат-бот буде обробляти метеорологічний код, вилучати з нього погодну інформацію та надавати її у читабельному для людини вигляді. Для цього коротко розглянемо, що являє собою метеорологічний код METAR, що буде використовуватися для передачі інформації.

METAR (METeorological Aerodrome Report –) авіаційний метеорологічний код для передачі зведень про фактичну погоду на аеродромі. Рядок METAR – найпоширеніший у світі формат передачі спостережних даних про погоду. Він високо стандартизований через Міжнародну організацію цивільної авіації (ICAO), що дозволяє зрозуміти його у більшості країн світу [12]. Прикладом такого рядка є: UKBB 031130Z 01009MPS CAVOK 20/04 Q1019 R88/CLRD// TEMPO 03010G16MPS, де, наприклад, UKBB – код міжнародного аеропорту «Бориспіль»), а 01009MPS показує напрям вітру 10° та швидкість 9 м/с. Кожен сегмент містить в собі певну погодну інформацію або дані про спостереження. Необхідно реалізувати можливість зчитати кожен сегмент та представити його у вигляді програмних об’єктів для подальшої роботи з ними. На рис. 3.4. наведено діаграму класів, що відповідають за обробку рядка METAR.

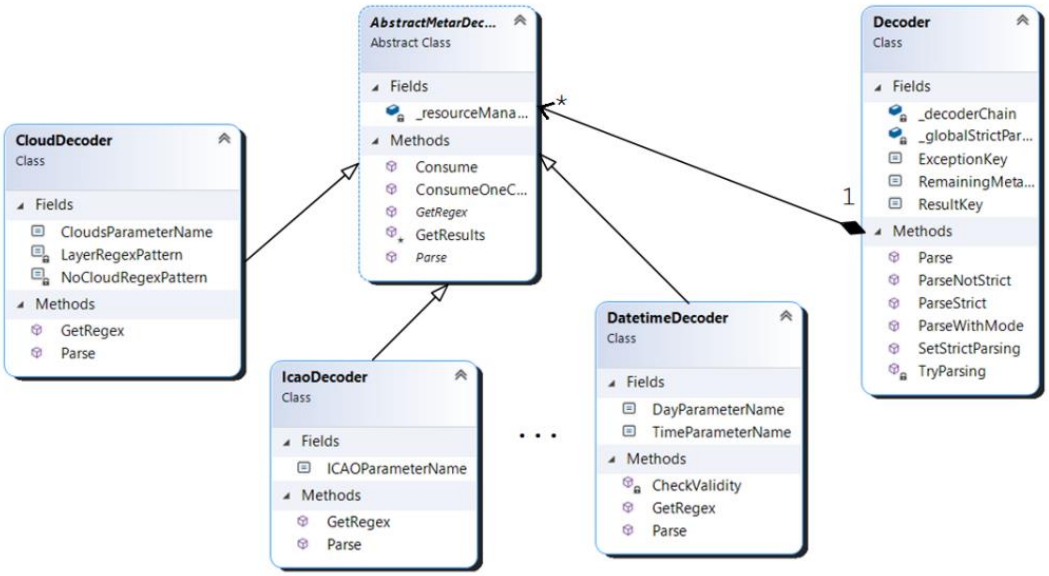

Рис. 3.4. Діаграма класів, що відповідають за обробку рядка METAR

Як видно з діаграми класи декодерів наслідуються від абстрактного класу AbstractMetarDecoder. На діаграмі наведено лише три класи-декодера, для її кращого розуміння. Однак для кожного сегмента варто реалізувати окремий декодер. Клас Decoder використовує класи декодерів для повного декодування рядку METAR та в подальшому використовується для представлення погодної інформації у зручному для людини вигляді.

Далі необхідно реалізувати класи для представлення погодної інформації користувачу. На рис. 3.5. наведено діаграму класів, що формують текст повідомлення, що буде містити інформацію про погоду на аеродромі.

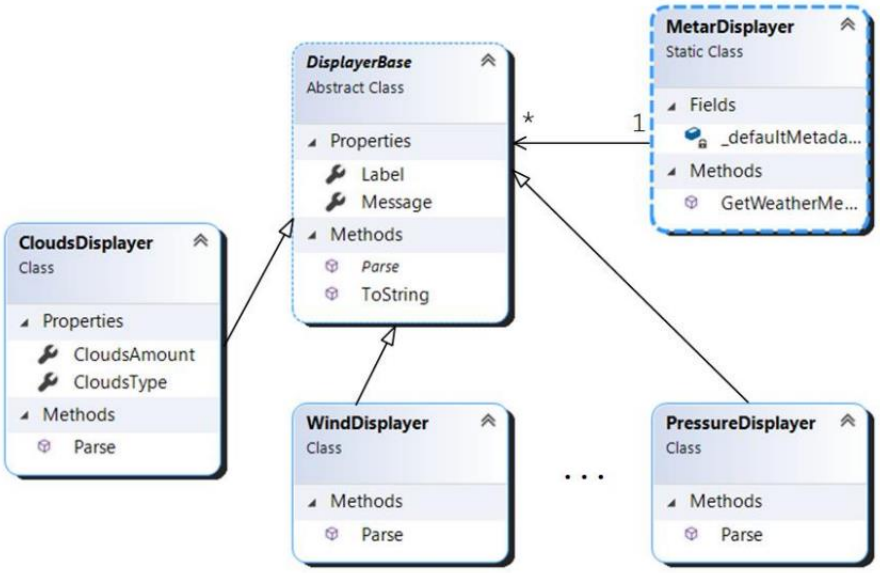

Рис. 3.5. Діаграма класів, що відповідають за відображення погодної інформації

Класи, що відповідають за відображення погодної інформації наслідуються від абстрактного класу DisplayerBase. Для кращого розуміння на діаграмі наведено лише три таких класи. Обов’язковим методом є метод Parse, який власне проводить відображення конкретної погодної інформації у зручний для людини вигляд. Статичний клас MetarDisolayer використовує ці класи для подальшого надання декодованої погодної інформації.

Для реалізацій чат боту необхідно створити класи API-контролерів, адаптера, діалогів, сервісів та роботи з базою даних. При цьому варто прагнути використовувати більш слабкий зв’язок між класами: композицію замість наслідування та агрегацію замість композиції. На рис. 3.6 наведено діаграму класів чат-боту.

Рис. 3.6. Діаграма класів чат-боту

На діаграмі бачимо два API-контролери, що наслідуються від абстрактного класу ControllerBase, який для заощадження місця на діаграмі позначено без полів та методів, оскільки вже реалізований спеціалістами

Microsoft. BotController призначений для отримання повідомлень від користувача, а NotifyController – для отримання сповіщень.

WeatherBot використовує класи діалогів, що наслідуються від класу ComponentDialog, які є об’єктно орієнтованим представленням діалогів з користувачем. В діалогах відбувається звернення до відповідних сервісів. Для забезпечення слабких зв’язків відповідні класи сервісів реалізують інтерфейси IMetarService та IAirportDataServicve. Таким чином в майбутньому легко можна буде замінити реалізації цих сервісів.

Для доступа до бази даних використовується патерн Репозиторій, зокрема інтерфейс ISubscriptionRepository та клас EFSubscriptionRepository. В середині цього класу відбувається звернення до класу-контексту бази даних BotDbContext. Клас Subscription є класом-сутністю Entity Framework Core, що співставляється відповідній таблиці в реляційній базі даних.

Висновок

В третьому розділі розглянуто структуру чат-боту для моніторингу фактичної погоди на аеродромі. Проведено аналіз засобів та технологій, які будуть використані для розробки застосунку.

Також описано архітектуру розроблюваного чат-боту. Показано як різні компонети застосунку взаємодіють між собою та як повідомлення передаються від користувача до чат-боту і навпаки. Така архітектура дозволить розробити легко розширюваний чат-бот, доступ до якого користувачі зможуть отримати через різні канали.

Проведено опис взаємодій класів застосунку та побудовано діаграми класів для кращого розуміння їх взаємодії. З такою структурою класів можна легко додавати новий код без необхідності редагування існуючого.

4. ПРОТОТИП ЧАТ-БОТУ ДЛЯ МОНІТОРИНГУ ФАКТИЧНОЇ ПОГОДИ НА АЕРОДРОМІ

4.1. Прототипування програмного забезпечення

Прототипування програмного забезпечення – процес створення прототипів програмного забезпечення, які відображають функціональні можливості продукту, що розробляється та можуть не відповідати точній логіці вихідного програмного забезпечення.

Прототипування програмного забезпечення набуває популярності як модель розробки програмного забезпечення, оскільки дозволяє зрозуміти вимоги клієнтів на ранній стадії розробки. Це допомагає отримати цінні відгуки від клієнтів і допомагає зрозуміти, що саме очікується від продукту, що розробляється.

Розробники можуть створити своє власне бачення продукту, але потенційні користувачі можуть по-справжньому зрозуміти продукт тільки тоді, коли вони побачать робочу версію програмного забезпечення. Вони зможуть вказати, що їм подобається, а що ні, а також запропонувати новий функціонал. Прототип може також допомогти ідентифікувати програмні компоненти або засоби для розробки. Можна виявити, що технологія, яку планувалося використовувати не підходить і потрібно обрати інші доступні технології.

Метою є розробка робочої версії програмного забезпечення, яка може використовуватися для демонстрації його основних функцій. Розробники повинні прагнути того, щоб демонстраційна система була запущена і працювала через чотиришість тижнів. Звичайно, для цього потрібно вести розробку дуже швидко, тому розробники можуть ігнорувати проблеми з надійністю та продуктивністю, а працювати з елементарним призначеним для користувача інтерфейсом. Бажано розробляти прототип таким чином, щоб його можна було без зайвих зусиль перетворити в повноцінне програмне забезпечення.

4.2. Розробка та публікація прототипу чат-боту

Для розробки прототипу чат-боту для моніторингу фактичної погоди на аеродромі використаємо засоби та технології, описані в попередньому розділі: платформу .NET, мову програмування C#, Microsoft Bot Framework, Entity Framework Core, середовище розробки Visual Studio, Bot Framework Emulator та онлайн-портал Azure.

Використовуючи діаграми класів з попереднього розділу створимо класи. При цьому класи для перетворення рядку METAR у читабельний для людини вигляд помістимо в окрему бібліотеку класів. На рис. 4.1 наведено структуру проектів в Visual Studio.

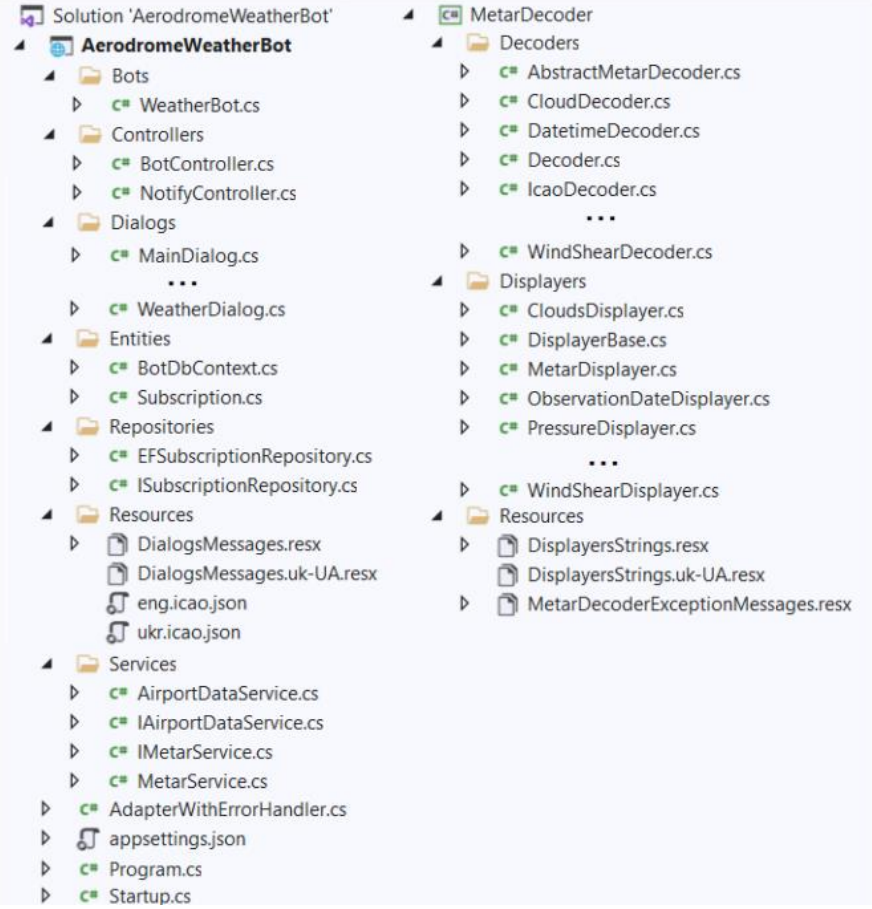

Рис. 4.1. Структура проектів в Visual Studio

Проект «AerodromeWeatherBot» являє собою застосунок чат-бот, а проект «MetarDecoder» – бібліотеку класів. Всі ці класи проектів описані в попередньому розділі. Варто звернути увагу на папку Resources. Вимоги до чат-боту включають інтерфейс користувача, доступний різними мовами. Тому в прототипі вже реалізовано українську та англійську локалізацію. Таким чином чат-бот вже може спілкувати двома мовами. У таблиці 4.1 наведено частину ресурсів українською мовою. В майбутньому можна буде додати нові мови в чат-бот, просто додавши відповідні файли ресурсів.

Таблиця 4.1

Частина ресурсів українською мовою

Назва	Значення
NotificationFrequencyChangedICAO	Я змінив частоту сповіщень на {0} год для аеропорту з кодом ICAO {1}.
PleaseEnterTheAirportCode	Введіть код аеропорту
PleaseEnterTheRegion	Будь ласка, введіть назву регіону
PleaseSelectTheAirportName	Оберіть назву аеропорту
UnsubscribedICAO	Я скасував Вашу підписку на аеропорт за кодом ICAO {0}
WhatElseCanIDoForYou	Що ще я можу зробити для Вас?
AlreadySubscribed	Ви вже підписалися. Гарного настрою.
RegionNotFound	На жаль, я не можу знайти регіон {0}. Що я можу зробити для вас?
WeatherInformationNotFoundICAO	На жаль, я не можу знайти інформацію про погоду для {0}
SetUpNotifications	Налаштування сповіщень

Після завершення розробки, опублікуємо його на онлайн-порталі Azure для подальшого підключення чат-боту до різних каналів та надання до нього доступу користувачам.

Після публікації можна встановити зображення профілю, ім’я та опис бота. Також можна протестувати бота у веб-чаті Azure. Підключимо бота до Facebook Messenger, Skype та Telegram (рис. 4.2). Налаштування підключення відрізняється в залежності від каналу, але основна ідея однакова.

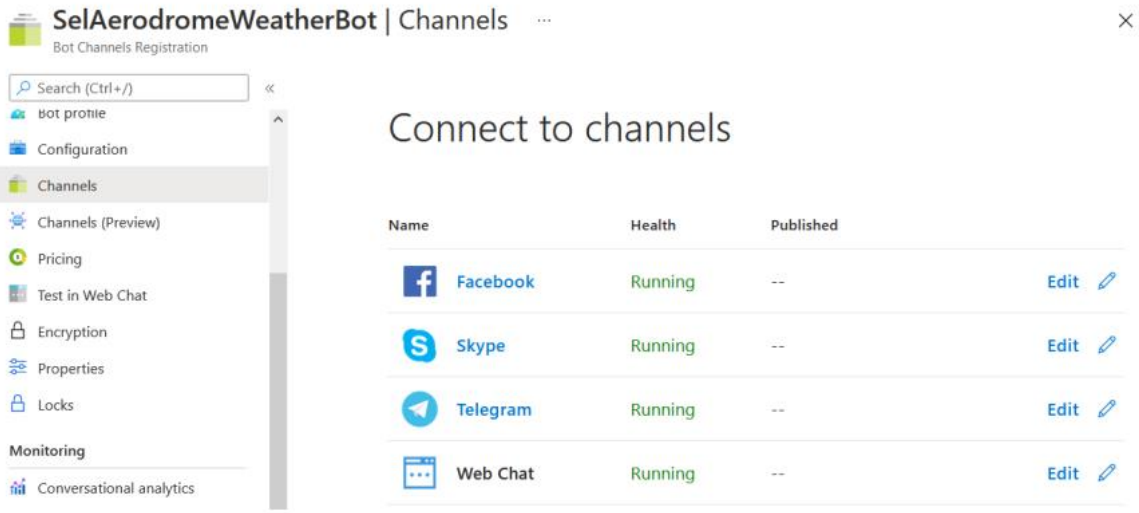

Рис. 4.2. Підключення чат-боту до декількох каналів

4.3. Демонстрація роботи прототипу чат-боту

Продемонструємо роботу прототипу чат боту для моніторингу фактичної погоди на аеродромі.

1. Початок розмови. Коли користувач розпочинає розмову, чат бот надсилає у відповідь повідомлення з привітанням, а також пропонує користувачу можливі послуги. На рис. 4.3 наведено початок розмови в Facebook Messenger.

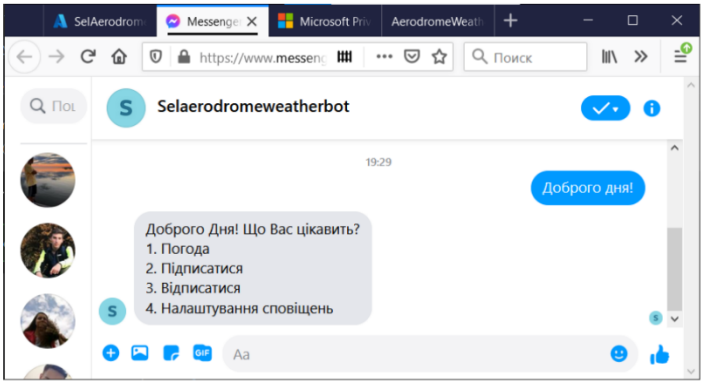

Рис. 4.3. Початок розмови з чат-ботом в Facebook Messenger в браузері

2. Отримання погодної інформації. Якщо користувач запитує в чат-бота погодну інформацію, пропонується обрати один із способів отримання даних про аеропорт: за кодом аеропорту або за назвою регіону та назвою аеропорту.

2а. Отримання погодної інформації за кодом аеропорту. Якщо користувач обирає отримання інформації за кодом ICAO, то чат-бот запитує цей код, після цього надає погодну інформацію. Якщо аеропорт не знайдено, то чат-бот відправляє відповідне повідомлення. На рис. 4.4. показано отримання погодної інформації в Skype.

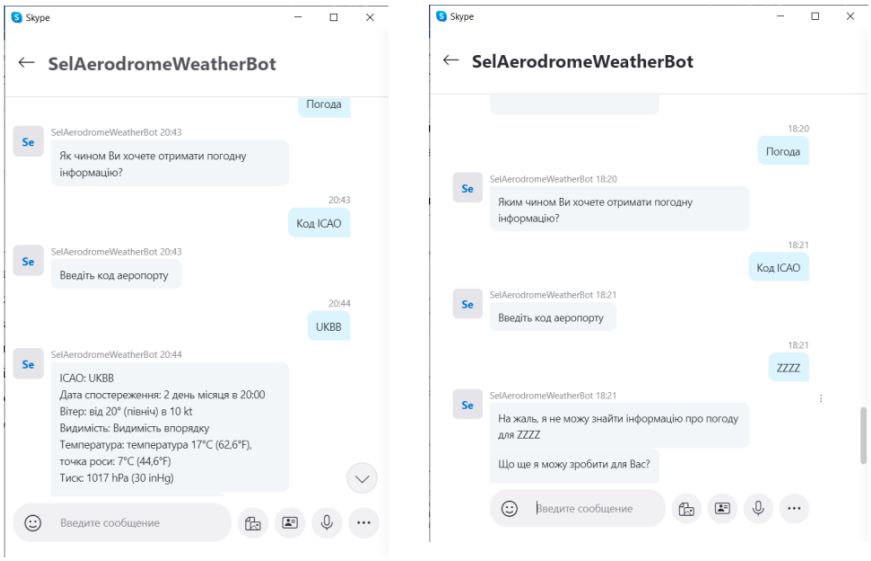

Рис. 4.4. Отримання погодної інформації за кодом аеропорту в Skype

2б. Отримання погодної інформації за назвою регіону та назвою аеропорту. Якщо користувач обирає отримання інформації за назвою регіону та назвою аеропорту, то чат-бот запитує назву регіону та назву аеропорту, після цього надає погодну інформацію. Якщо аеропорт регіон або аеропорт не знайдено, то чат-бот відправляє відповідне повідомлення. На рис. 4.5. показано отримання погодної інформації в Веб-чаті.

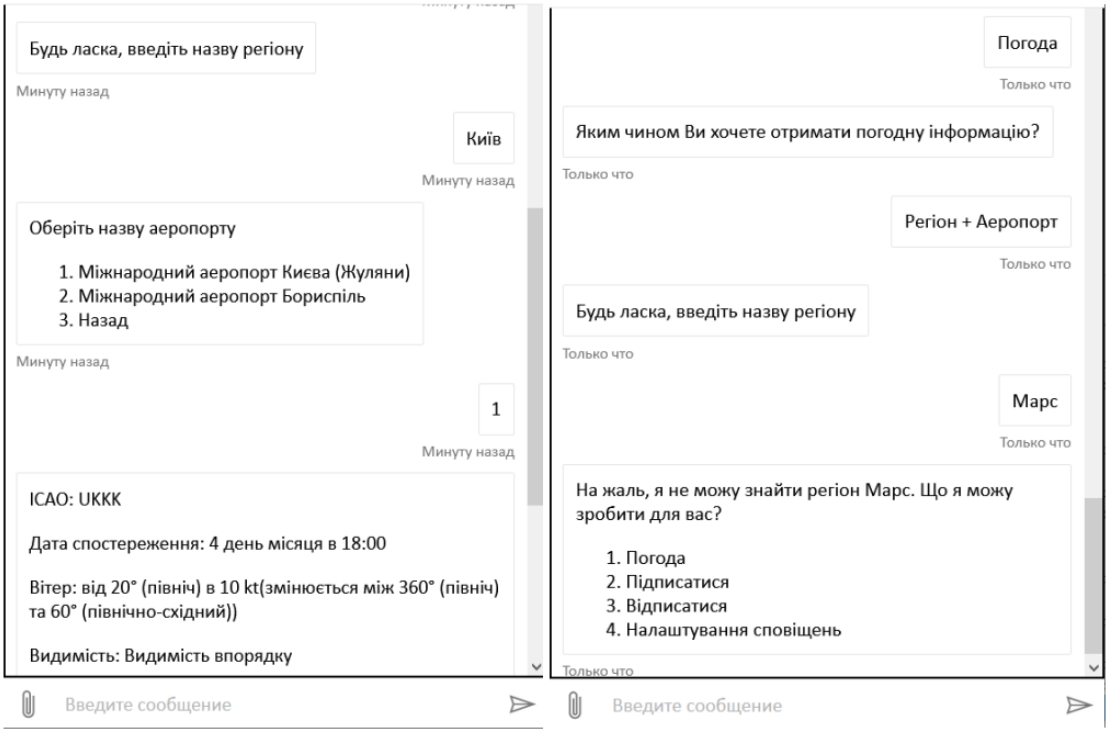

Рис. 4.5. Отримання погодної інформації за назвою регіону та аеропорту

3. Оформлення підписки на отримання актуальних погодних даних з аеродрому. Якщо користувач просить чат-бот підписатися на отримання фактичної погоди на аеродромі, то аналогічно отриманню погоди, чат-бот пропонує оформити підписку за кодом аеропорту або за назвою регіону та назвою аеропорту. Після цього пропонується задати частоту отримання сповіщень. Якщо підписка вже оформлена, то чат-бот інформує користувача про це.

Оскільки логіка пошуку аеропорту однакова для отримання погоди, оформлення підписки, скасування підписки та налаштування сповіщень, то тут і далі не будуть наводитись знімки екранів, на яких продемонстровано не коректний ввід коду аеропорту, назви регіону або назви аеропорту. На рис. 4.6 показано оформлення підписки на отримання фактичної погоди на аеродромі в Facebook Messenger з англійською локалізацією.

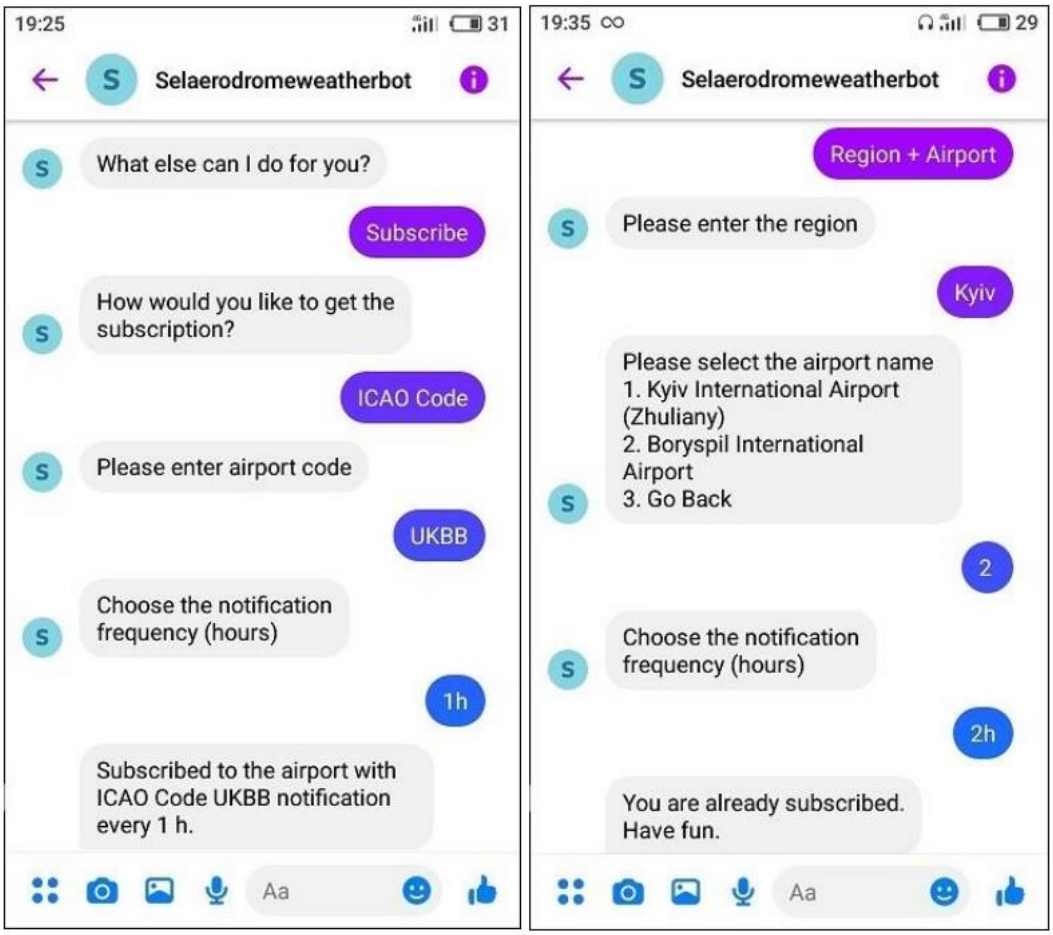

Рис. 4.6. Оформлення підписки на моніторинг фактичної погоди на аеродромі в Facebook Messenger з англійським інтерфейсом

4. Скасування підписки на отримання актуальних погодних даних з аеродрому. Якщо користувач просить відписатися від отримання фактичної погоди на аеродромі, то чат-бот пропонує знайти потрібний аеропорт за кодом або за назвою

регіону та назвою аеропорту. Після цього надсилає повідомлення про успішну відписку. Якщо підписку не знайдено, то чат-бот надсилає відповідне повідомлення. На рис. 4.7 показано скасування підписки у Skype.

5. Налаштування частоти сповіщень. Якщо користувач просить змінити налаштування сповіщень то чат-бот пропонує знайти потрібний аеропорт за кодом або за назвою регіону та назвою аеропорту. Після цього пропонує встановити нову частоту сповіщень. На рис. 4.8 показано налаштування частоти сповіщень в Facebook Messenger з англійським інтерфейсом.

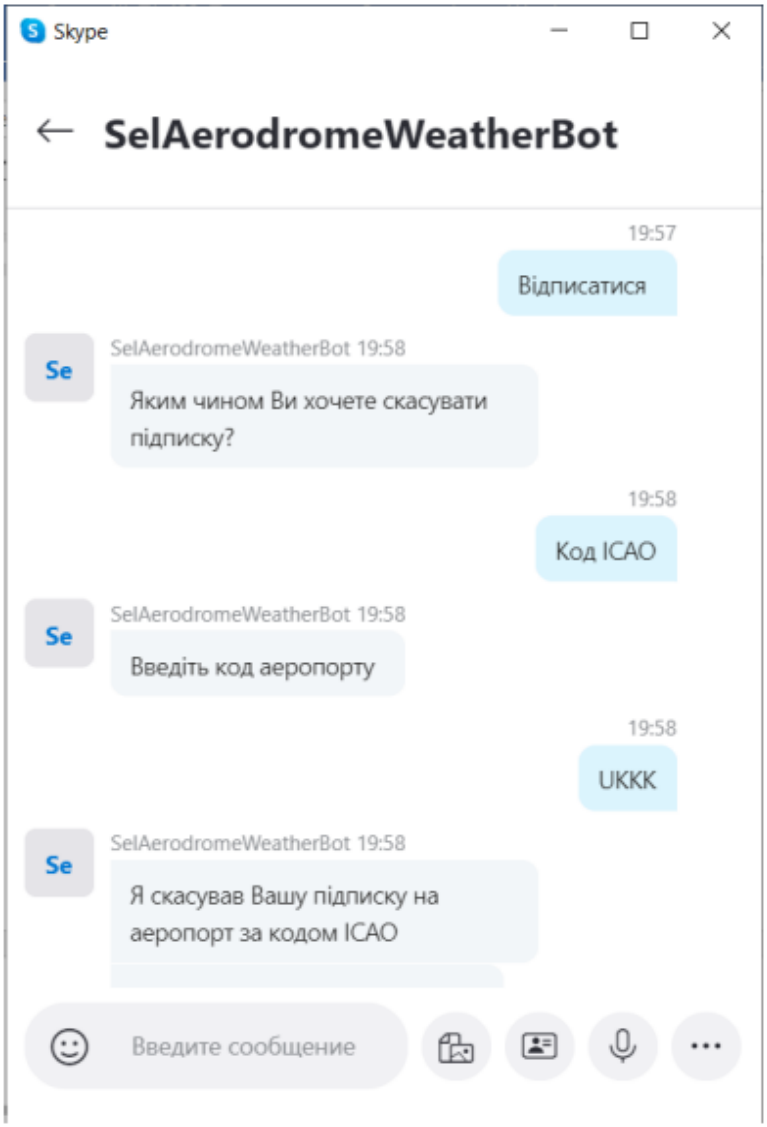

Рис. 4.7. Скасування підписки на моніторинг фактичної погоди на аеродромі в Skype

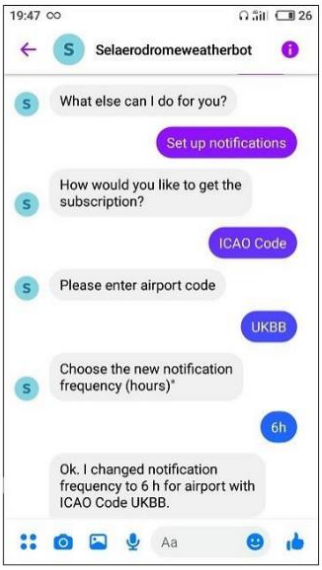

Рис. 4.8. Налаштування частоти сповіщень в Facebook Messenger з англійським інтерфейсом

6. Отримання фактичної погоди на аеродромі за підпискою. Зі встановленою користувачем частотою сповіщень, чат-бот надсилає повідомлення користувачу з актуальною погодною інформацією. На рис. 4.9 показано нове повідомлення в Skype, що отримане за підпискою.

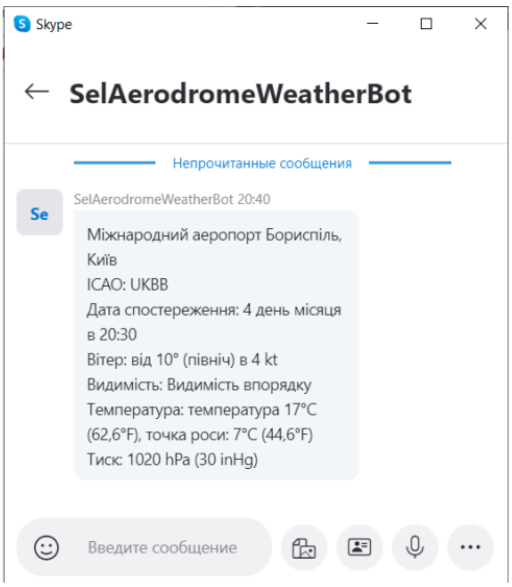

Рис. 4.9. Нове повідомлення для користувача в Skype, що отримане за підпискою

Висновок

У четвертому розділі було розглянуто що таке прототип програмного забезпечення та його призначення. Розроблено прототип чат-боту для моніторингу фактичної погоди на аеродромі. Прототип дозволяє отримувати погодну інформацію за кодом аеропорту або за назвою регіону та назвою аеропорту.

Крім цього є можливість підписатися на моніторинг фактичної погоди на аеродромі, а також скасувати підписку або змінити частоту сповіщень. Продемонстровано роботу прототипу розроблюваного чат-боту в різних каналах: Facebook Messenger, Skype та Web-чаті з українським та англійським інтерфейсами користувача.

ВИСНОВКИ

Метою дипломного проекту була розробка чат-боту для моніторингу фактичної погоди на аеродромі, що дозволив би максимально швидко та просто отримати інформацію про погоду на аеродромі працівникам авіаційної галузі. Було проаналізовано предметну область та визначено, що погода має важливий вплив роботи авіації. Проведено огляд існуючих застосунків різних типів та виявлено їх переваги та недоліки. Розроблюваний чат-бот має мету усунути недоліки цих застосунків.

Проведено аналіз вимог до чат-боту, побудовано діаграму варіантів використання, що допомогла наглядно показати різних способи взаємодії користувача з чат-ботом. З використанням діаграми потоків даних було продемонстровано як кожен процес перетворює свої вхідні дані у вихідні.

Визначено засоби та технології, що необхідні для розробки чат-боту: платформа .NET, мова програмування C#, Microsoft Bot Framework, Entity Framework Core, середовище розробки Visual Studio, Bot Framework Emulator та онлайн-портал Azure. Таким чином, здобуто досвід їх використання та покращено навички роботи з ними.

Описано архітектуру та показано, як різні компоненти програмного забезпечення взаємодіють між собою. Для демонстрації взаємодії між класами застосунку було побудовано діаграми класів.

Розроблено прототип чат-боту для моніторингу фактичної погоди на аеродромі та розміщено його на онлайн-порталі Azure для підключення різних каналів. Продемонстровано роботу розроблюваного прототипу. Отриманий чат-бот простий у використанні, не потребує встановлення, доступний на різних пристроях та надає швидкий доступ до погодної інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Joe Mayo. Programming the Microsoft Bot Framework. A Multiplatform Approach to Building Chatbots – USA, Microsoft Press 2018, – 361 с.

2. Srikanth Machiraju, Ritesh Modi. Developing Bots with Microsoft Bots Framework – USA, Apress 2018, – 283 с.

3. Elizabeth Hull, Kenneth Jackson, Jeremy Dick. Requirements Engineering, Third edition. – UK, Springer-Verlag London 2011, – 207 с.

4. Bot Framework SDK documentation - Bot Service | Microsoft Docs: [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/azure/bot-service/index-bf-sdk

5. Elizabeth Hull, Kenneth Jackson, Jeremy Dick. Requirements Engineering, Third edition. – UK, Springer-Verlag London 2011, – 207 с.

6. Mark J. Price. C# 9 and .NET 5 – Modern Cross-Platform Development - Fifth Edition. – USA, Packt Publishing Ltd. 2020, – 822 с.

7. Send proactive notifications to users - Bot Service | Microsoft Docs: [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-proactive-message

8. The Importance of an Aviation Meteorologist | AIRTUG: [Електронний ресурс] – Режим доступу: https://airtug.com/blog/importance-aviation-meteorologist/

9. Nics Site: [Електронний ресурс] – Режим доступу: http://www.nicstorey.co.uk/planeplotter/metarmonitor.php

10. AWC – METeorological Aerodrome Reports (METARs): [Електронний ресурс] – Режим доступу: https://www.aviationweather.gov/metar/

11. Avia Weather – METAR & TAF – додатки в Google Play: [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.mytowntonight.aviationweather/ 12. METAR – Вікіпедія: [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/METAR

ДОДАТОК А
Лістінг програми


```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "appId": {
      "type": "string",
      "metadata": {
        "description": "Active Directory App ID, set as MicrosoftAppId in the Web App's Application Settings."
      }
    },
    "appSecret": {
      "type": "string",
      "metadata": {
        "description": "Active Directory App Password, set as MicrosoftAppPassword in the Web App's Application Settings. Defaults to \"\"."
      }
    },
    "botId": {
      "type": "string",
      "metadata": {
        "description": "The globally unique and immutable bot ID. Also used to configure the displayName of the bot, which is mutable."
      }
    },
    "botSku": {
```

```
        "defaultValue": "F0",
        "type": "string",
        "metadata": {
            "description": "The pricing tier of the Bot Service Registration. Acceptable values are F0 and S1."
        }
    },
    "newAppServicePlanName": {
        "type": "string",
        "defaultValue": "",
        "metadata": {
            "description": "The name of the new App Service Plan."
        }
    },
    "newAppServicePlanSku": {
        "type": "object",
        "defaultValue": {
            "name": "S1",
            "tier": "Standard",
            "size": "S1",
            "family": "S",
            "capacity": 1
        },
        "metadata": {
            "description": "The SKU of the App Service Plan. Defaults to Standard values."
        }
    },
    "appServicePlanLocation": {
```

```
        "type": "string",
        "metadata": {
          "description": "The location of the App Service Plan."
        }
      },
      "existingAppServicePlan": {
        "type": "string",
        "defaultValue": "",
        "metadata": {
          "description": "Name of the existing App Service Plan used to create the Web App for the bot."
        }
      },
      "newWebAppName": {
        "type": "string",
        "defaultValue": "",
        "metadata": {
          "description": "The globally unique name of the Web App. Defaults to the value passed in for \"botId\".
    },
    "variables": {
      "defaultAppServicePlanName": "[if(empty(parameters('existingAppServicePlan')), 'createNewAppServicePlan', parameters('existingAppServicePlan'))]",
      "useExistingAppServicePlan": "[not(equals(variables('defaultAppServicePlanName'), 'createNewAppServicePlan'))]",
      "servicePlanName": "[if(variables('useExistingAppServicePlan'), parameters('existingAppServicePlan'), parameters('newAppServicePlanName'))]",
      "resourcesLocation": "[parameters('appServicePlanLocation')]",
```

```
        "webAppName": "[if(empty(parameters('newWebAppName')), parameters('botId'), parameters('newWebAppName'))]",
        "siteHost": "[concat(variables('webAppName'), '.azurewebsites.net')]",
        "botEndpoint": "[concat('https://', variables('siteHost'), '/api/messages')]"
    },
    "resources": [
        {
            "comments": "Create a new App Service Plan if no existing App Service Plan name was passed in.",
            "type": "Microsoft.Web/serverfarms",
            "condition": "[not(variables('useExistingAppServicePlan'))]",
            "name": "[variables('servicePlanName')]",
            "apiVersion": "2018-02-01",
            "location": "[variables('resourcesLocation')]",
            "sku": "[parameters('newAppServicePlanSku')]",
            "properties": {
                "name": "[variables('servicePlanName')]"
            }
        },
        {
            "comments": "Create a Web App using an App Service Plan",
            "type": "Microsoft.Web/sites",
            "apiVersion": "2015-08-01",
            "location": "[variables('resourcesLocation')]",
            "kind": "app",
            "dependsOn": [
                "[resourceId('Microsoft.Web/serverfarms', variables('servicePlanName'))]"
            ],
```

```
      "name": "[variables('webAppName')]",
      "properties": {
        "name": "[variables('webAppName')]",
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('servicePlanName'))]",
        "siteConfig": {
          "appSettings": [
            {
              "name": "WEBSITE_NODE_DEFAULT_VERSION",
              "value": "10.14.1"
            },
            {
              "name": "MicrosoftAppId",
              "value": "[parameters('appId')]"
            },
            {
              "name": "MicrosoftAppPassword",
              "value": "[parameters('appSecret')]"
            }
          ],
          "cors": {
            "allowedOrigins": [
              "https://botservice.hosting.portal.azure.net",
              "https://hosting.onecloud.azure-test.net/"
            ]
          },
          "webSocketsEnabled": true
    {
```

```
        "apiVersion": "2017-12-01",
        "type": "Microsoft.BotService/botServices",
        "name": "[parameters('botId')]",
        "location": "global",
        "kind": "bot",
        "sku": {
          "name": "[parameters('botSku')]"
        },
        "properties": {
          "name": "[parameters('botId')]",
          "displayName": "[parameters('botId')]",
          "endpoint": "[variables('botEndpoint')]",
          "msaAppId": "[parameters('appId')]",
          "developerAppInsightsApplicationId": null,
          "developerAppInsightKey": null,
          "publishingCredentials": null,
          "storageResourceId": null
        },
        "dependsOn": [
          "[resourceId('Microsoft.Web/sites/', variables('webAppName'))]"
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "groupLocation": {
      "type": "string",
      "metadata": {
```

```
            "description": "Specifies the location of the Resource Group."
        }
    },
    "groupName": {
        "type": "string",
        "metadata": {
            "description": "Specifies the name of the Resource Group."
        }
    },
    "appId": {
        "type": "string",
        "metadata": {
            "description": "Active Directory App ID, set as MicrosoftAppId in the Web App's Application Settings."
        }
    },
    "appSecret": {
        "type": "string",
        "metadata": {
            "description": "Active Directory App Password, set as MicrosoftAppPassword in the Web App's Application Settings."
        }
    },
    "botId": {
        "type": "string",
        "metadata": {
            "description": "The globally unique and immutable bot ID. Also used to configure the displayName of the bot, which is mutable."
```

```
        }
      },
      "botSku": {
        "type": "string",
        "metadata": {
          "description": "The pricing tier of the Bot Service Registration. Acceptable values are F0 and S1."
        }
      },
      "newAppServicePlanName": {
        "type": "string",
        "metadata": {
          "description": "The name of the App Service Plan."
        }
      },
      "newAppServicePlanSku": {
        "type": "object",
        "defaultValue": {
          "name": "S1",
          "tier": "Standard",
          "size": "S1",
          "family": "S",
          "capacity": 1
        },
        "metadata": {
          "description": "The SKU of the App Service Plan. Defaults to Standard values."
        }
```

```
        },
        "newAppServicePlanLocation": {
            "type": "string",
            "metadata": {
                "description": "The location of the App Service Plan. Defaults to \"westus\"."
            }
        },
        "newWebAppName": {
            "type": "string",
            "defaultValue": "",
            "metadata": {
                "description": "The globally unique name of the Web App. Defaults to the value passed in for \"botId\"."
            }
        }
    },
    "variables": {
        "appServicePlanName": "[parameters('newAppServicePlanName')]",
        "resourcesLocation": "[parameters('newAppServicePlanLocation')]",
        "webAppName": "[if(empty(parameters('newWebAppName')), parameters('botId'), parameters('newWebAppName'))]",
        "siteHost": "[concat(variables('webAppName'), '.azurewebsites.net')]",
        "botEndpoint": "[concat('https://', variables('siteHost'), '/api/messages')]",
        "resourceGroupId": "[concat(subscription().id, '/resourceGroups/', parameters('groupName'))]"
    },
    "resources": [
```

```
{
    "name": "[parameters('groupName')]",
    "type": "Microsoft.Resources/resourceGroups",
    "apiVersion": "2018-05-01",
    "location": "[parameters('groupLocation')]",
    "properties": {}
},
{
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2018-05-01",
    "name": "storageDeployment",
    "resourceGroup": "[parameters('groupName')]",
    "dependsOn": [
      "[resourceId('Microsoft.Resources/resourceGroups/',
parameters('groupName'))]"
    ],
    "properties": {
      "mode": "Incremental",
      "template": {
        "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
        "contentVersion": "1.0.0.0",
        "parameters": {},
        "variables": {},
        "resources": [
          {
            "comments": "Create a new App Service Plan",
            "type": "Microsoft.Web/serverfarms",
```

```
                "name": "[variables('appServicePlanName')]",
                "apiVersion": "2018-02-01",
                "location": "[variables('resourcesLocation')]",
                "sku": "[parameters('newAppServicePlanSku')]",
                "properties": {
                    "name": "[variables('appServicePlanName')]"
                }
            },
            {
                "comments": "Create a Web App using the new App Service
Plan",
                "type": "Microsoft.Web/sites",
                "apiVersion": "2015-08-01",
                "location": "[variables('resourcesLocation')]",
                "kind": "app",
                "dependsOn": [
                    "[concat(variables('resourceGroupId'),
'/providers/Microsoft.Web/serverfarms/', variables('appServicePlanName'))]"
                ],
                "name": "[variables('webAppName')]",
                "properties": {
                    "name": "[variables('webAppName')]",
                    "serverFarmId": "[variables('appServicePlanName')]",
                    "siteConfig": {
                        "appSettings": [
                            {
                                "name": "WEBSITE_NODE_DEFAULT_VERSION",
}
```

```
using Microsoft.Bot.Builder;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Schema;
using System.Collections.Concurrent;
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
namespace AerodromeWeatherBot.Bots
{
   public class WeatherBot<T> : ActivityHandler where T : Dialog
   {
      private readonly Dialog _dialog;
      private readonly BotState _conversationState;
      private readonly BotState _userState;
      private readonly ConcurrentDictionary<string, ConversationReference>
_conversationReferences;
      public WeatherBot(ConversationState conversationState, UserState userState,
T dialog,
         ConcurrentDictionary<string, ConversationReference>
conversationReferences)
      {
         _conversationState = conversationState;
         _userState = userState;
         _dialog = dialog;
         _conversationReferences = conversationReferences;
      }
      public override async Task OnTurnAsync(ITurnContext turnContext,
         CancellationToken cancellationToken = default)
```

```
        {
            await base.OnTurnAsync(turnContext, cancellationToken);
            await _conversationState.SaveChangesAsync(turnContext, false,
cancellationToken);
            await _userState.SaveChangesAsync(turnContext, false,
cancellationToken);
        }
        protected override async Task
OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
            CancellationToken cancellationToken)
        {
            AddConversationReference(turnContext.Activity as Activity);
            await _dialog.RunAsync(turnContext,
                _conversationState.CreateProperty<DialogState>("DialogState"),
cancellationToken);
        }
        //protected override async Task
OnMembersAddedAsync(IList<ChannelAccount> membersAdded,
        //    ITurnContext<IConversationUpdateActivity> turnContext,
CancellationToken cancellationToken)
        //{
        //    await _dialog.RunAsync(turnContext,
        //        _conversationState.CreateProperty<DialogState>("DialogState"),
cancellationToken);
        //}
        protected override Task OnConversationUpdateActivityAsync(
            ITurnContext<IConversationUpdateActivity> turnContext,
CancellationToken cancellationToken)
        {
```

```
        AddConversationReference(turnContext.Activity as Activity);
        return base.OnConversationUpdateActivityAsync(turnContext,
cancellationToken);
    }
    private void AddConversationReference(Activity activity)
    {
        var conversationReference = activity?.GetConversationReference();
        _conversationReferences?.AddOrUpdate(conversationReference.User.Id,
            conversationReference, (key, newValue) => conversationReference);
    }
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ