1 АНАЛІЗ МОЖЛИВОСТЕЙ СУЧАСНИХ ПРОГРАМНИХ ПРОДУКТІВ ТИПУ «ЧАТ-БОТ»

1.1 Історія розвитку та основні властивості програм чат-ботів

Розвиток цифрових комунікацій та технологій комп'ютерних мереж (як локальних, так і глобального характеру, в основному – мережі Інтернет) ще до появи можливості передачі обширних обсягів мультимедійних даних дозволив людям спілкуватися між собою у досить ефективному віддаленому режимі. Так, досить традиційною до початку 90-х років була можливість відправлення короткого текстового повідомлення – телеграми (зокрема, за допомогою спеціальної телеграфної мережі – телетайпу «Телекс») або навіть відправлення звичайного друкованого (рукописного) листа поштою. Обидва способи вимагали оплати відповідної послуги, вартість якої була такою, що надсилати повідомлення кожного дня (не говорячи вже про декілька повідомлень на день) було практично не можливо. Крім того, час отримання текстового повідомлення адресатом від моменту відправки складав від кількох годин до величини порядку днів.

Із появою та впровадженням найперших та найпростіших технологій комп'ютерних мереж саме обмін короткими текстовими повідомленнями був чи не найпершою функцією, яку такі мережі могли забезпечувати. Так, глобальна комп'ютерна мережа Fidonet забезпечувала обмін текстовими повідомленнями через систему мережної пошти Netmail [1]. Уже на даному зародковому етапі становлення мережних технологій деякі особи прагнули автоматизувати відправку окремих типів повідомлень, налаштовуючи програму-робота, яка могла у повністю автоматичному режимі спілкуватися із іншими «живими» абонентами мережі. Такі програми-роботи (пізніше назва скоротилася до взагалі більш короткого слова «бот», «боти») були призначені для цілей наступних типів:

а) видачі допомоги у системах надання відповідей на часто задавані питання FAQ (Frequently Asked Questions);

б) надання певної статистичної інформації на вимогу користувача;

в) надання загально необхідної інформації (наприклад, правил поведінки, допоміжної інформації про ролі різних користувачів, будь-яка корисна для всіх інформація) при появі нових користувачів у системі;

г) проведення регулярних навчальних розсилань і т.п.

Спілкування у виключно текстовому режимі, але у режимі оффлайн недостатньо активно сприяло впровадженню таких продуктів, які ми сьогодні називаємо чат-ботами. Для роботи повноцінного чат-боту має бути реалізованим режим спілкування онлайн. Першою системою онлайн спілкування стала розроблена у 1988 році Internet Relay Char (IRC) – система для спілкування великої кількості користувачів одночасно в рамках однієї віртуальної кімнати, яку називали IRC-каналом [2]. Даний протокол став дійсно популярним і у текстовому онлайн спілкуванні злилися мільйони користувачів по всьому світу. Тут практично вперше і проявилися можливості чат-ботів, які будучи по своїй суті звичайної комп'ютерною програмою, при поверхневому огляді здавалися звичайними користувачами, як і люди, що увійшли до чату. Зважаючи на те, що у кімнатах часто писали повідомлення діти та люди, що погано вміють виражати свої думки, а також допускають чимало помилок (малограмотні або неуважні), іноді звичайних користувачів обвинувачували у тому, що вони є ботами, а ботів навпаки певний час (до моменту розкриття) могли вважати людьми.

Тут слід звернутися від розгляду цільових платформ до іншого аспекту використання чат-ботів, а саме – до ступеня їх схожості з людиною. Усі чат-боти можна поділити на відкриті, про які з самого початку відомо, що це бот, а не живий співрозмовник, а також – таємні, про які така інформація приховується. У випадку відкритого бота особливих вимог щодо якості спілкування, зазвичай, не виникає (навіть вибагливі користувачі, розуміючи, що спілкуються з ботом, не звертають уваги на якість його відповідей, а просто намагаються отримати потрібний їм результат). Якщо бот прихований, то його розробник має закласти у нього досить складне інтелектуальне підґрунтя, за рахунок якого відповіді бота мають бути максимально наближені до людських. У науковій літературі з штучного інтелекту навіть існує таке поняття, як тест Т'юринга, яке означає

процедуру сліпого текстового спілкування людини із комп'ютерною програмою чат-ботом, в результаті чого людина не зможе точно сказати, що розмовляє не з живим співрозмовником. Слід відмітити, що для коротких неспецифічних розмов, які проводяться із якісно складними програмними продуктами, досить важко встановити, що спілкування відбувається з ботом. Зазвичай, для цього потрібно використовувати питання, які потребують точної конкретної відповіді, отриманої в результаті побудови аналогій, узагальнення, аналізу слабко формалізованих ситуацій, образного мислення і т.п. Таким чином, для побудови найбільш якісних ботів розробникам слід намагатися імітувати ці властивості мозку людини, що втім є зайвим при реалізації простих відкритих ботів, єдиною функцією яких є надання користувачеві різноманітної корисної інформації.

Зрозуміло, що можливості ботів могли використовувати зловмисники. Пишучи у велику кількість IRC-каналів за допомогою ботів (що неможливо було би зробити у повністю ручному режимі) зловмисник значно збільшував шанси на успіх своєї атаки, реалізованої за правилами соціальної інженерії (окрема галузь знань на стику психології та інформаційних технологій, яка допомагає проводити злом комп'ютерних систем за допомогою спілкування з людиною-користувачем). Також можливості чат-ботів могли використовувати недобросовісні спеціалісти з реклами, надаючи більші обсяги рекламної інформації більш широкій аудиторії. Звичайно, чат-боти для IRC виконували і корисні для суспільства функції, зокрема такі, що перелічені у пунктах а)-г) вище у цьому підрозділі.

Наступним кроком після IRC став тотальний розвиток персональних програм-месенджерів, які домінують на ринку спілкування і по даний час, лише додаючи нову функціональність, як, наприклад, голосові дзвінки через Інтернет та відео спілкування. Такими месенджерами спочатку були ICQ, QIP та ін., а пізніше – WhatsApp, Viber, Telegram. У багатьох з цих продуктів можуть використовуватися чат-боти, тому розглянемо їх докладніше.

а) Інтернет-пейджер ICQ, що був дуже поширеним для настільних ПК, але не отримав такого розвитку для мобільних систем. На сьогоднішній день може вважатися застарілим (принаймні в умовах української Інтернет-спільноти) та

таким, що поступився місцем більш сучасним (молодим) продуктам. Слід відмітити, що у часи власної популярності дане програмне забезпечення активно використовувало програм-ботів, але на сьогоднішній день розробляти їх немає сенсу через занепад самого месенджеру в цілому;

б) практично повний аналог ICQ – російська програма QIP, що у 2000-х та на початку 2010-х рр. була дуже широко поширена на ПК серед української Інтернет-спільноти (в першу чергу, через підтримку інших, крім ICQ, протоколів, наприклад, Jabber, а також масі іншої, додаткової у порівнянні з ICQ, функціональності), але на даний момент, у зв'язку із бурхливим розвитком мобільних месенджерів ця програма мало використовується навіть і для ПК;

в) програма для відозв'язку Skype із самого свого виникнення має функцію передачі виключно текстових повідомлень, однак через маркетингову політику виробника не асоціюється у широких мас користувачів із текстовим обміном, і цю можливість в реальності використовує порівняно мало людей. Через такі обставини чат-боти для Skype не використовуються;

г) програма-месенджер Viber, яка початково створювалася для смартфонів, але через свою шалену популярність отримала і поширену на сьогоднішній день версію для ПК. Цей продукт ізраїльського походження в умовах України на сьогоднішній день є дійсно лідером по організації в основному текстового обміну, і, в результаті саме його використання, обмін текстовими повідомленнями і став тією процедурою, що сьогодні вже зручно підходить не тільки для спілкування друзів та знайомих між собою, а й керівників високого рівня зі своїми підлеглими (тобто сучасні поняття про субординацію вже давно дозволяють такий обмін). Дане програмне забезпечення насичене рекламними продуктами, а також чат-ботами, що виконують рекламні, а також допоміжні сервісні, інформаційні та інші функції у системі. Слід відмітити, що відношення до Viber-ботів у переважної більшості користувачів системи чисто рекламне: їх не сприймають, як щось корисне, а звертаються до них тільки у випадку ;

д) майже повний аналог Viber - програмний продукт WhatsApp від компанії Facebook, що раніше був значно популярнішим, особливо у закордонних країнах

Заходу, однак після його купівлі гігантом Facebook розвивається значно гірше, ймовірно, не витримуючи внутрішньої конкуренції з Facebook Messenger. Слід відмітити, що незважаючи на те, що WhatsApp інтегрально є найпопулярнішим месенджером у світі, можливість створення ботів у ньому відсутня, тому він не підходить для цілей даної роботи (версію WhatsApp Business, яка допускає створення ботів, можна не брати до уваги у зв'язку із її відносно малою поширеністю);

е) Facebook Messenger має очевидний і дуже серйозний плюс, який полягає у тотальній інтеграції із обліковим записом Facebook, тобто усі дії, що виконуються у цій, нібито окремій програмі Facebook Messenger, одразу відбиваються у Facebook-акаунті користувача, що є дуже зручним за умови інтенсивного використання цієї соціальної мережі. Слід відмітити, що боти для Facebook Messenger є досить популярним засобом комунікації із користувачами платформи. Однак, через її надзвичайно широку популярність та, відповідно, невисоку кваліфікацію в галузі IT середнього користувача фейсбук, для створення ботів під дану платформу використовуються прості конструктори чатів типу сервісів ChattyPeople, Botsify, Chatfuel, FlowXO, BeepBoop. Ці розробки мають типові недоліки систем конструкторів: немає можливості тонких налаштувань та наявне обмеження функціональності тільки тими можливостями, які передбачені розробниками конструктора.

є) додаток Telegram надає набагато більш широкі можливості, ніж простий текстовий обмін, зокрема по організації Telegram-каналів, а також стійкі криптоалгоритми, які нібито не можуть зламати навіть державні спецслужби. Зручний інтерфейс а також інші особливості месенджеру привели до того, що саме у ньому у найбільш широкій мірі і поширилися програмні продукти типу «чат-бот». При цьому, на відміну від Viber, не можна сказати, що усі ці боти призначені для рекламних цілей. Навпаки, часто чат-боти для Telegram мають допоміжний, сервісний або освітній характер, тобто сприймаються позитивно більшістю користувачів. Відповідно, саме даний месенджер і буде взятий за основу у подальшій розробці;

ж) існують і інші засоби текстового обміну, які, однак, не отримали великої популярності у широкого загалу, тому їх навряд чи доцільно використовувати і при організації чат-боту для відстежування метеорологічних умов.

1.2 Дослідження існуючих програм-аналогів для відстежування метеорологічних умов

Перед стартом будь-якої розробки слід розглянути існуючі на ринку або у вільному доступі програмні продукти, що виконують схожі функції, тобто програмні продукти-аналоги, призначені для відстежування метеорологічних умов.

В першу чергу дізнаватися про погоду можна за допомогою відповідних сайтів. Виявити найбільш популярні із них можна за допомогою пошукового запиту в Google – рис. 1.1.

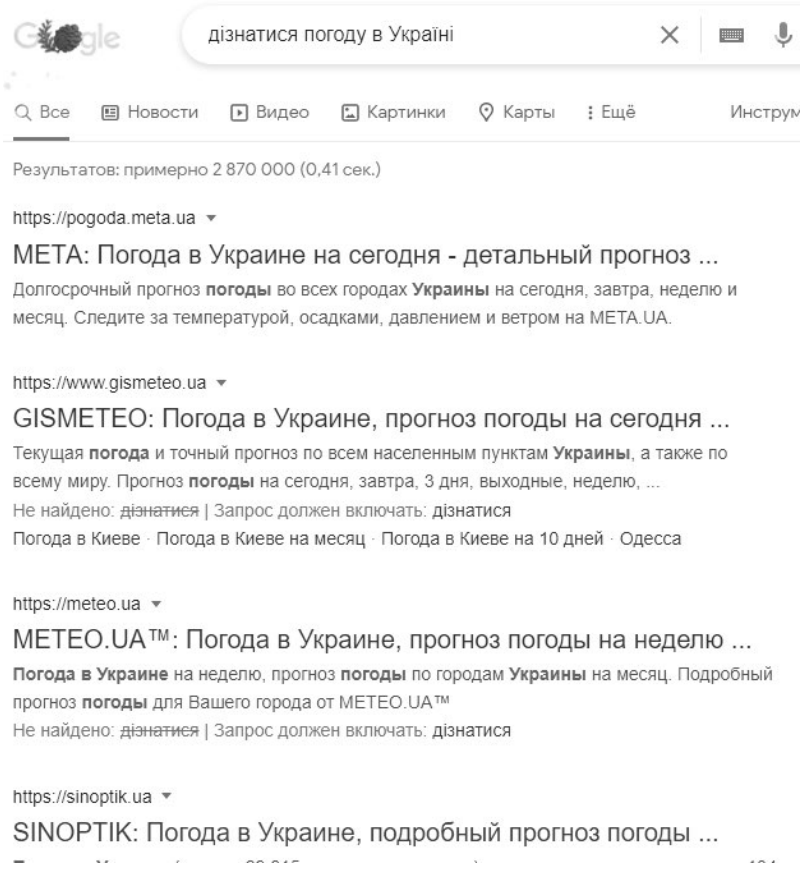

Рис. 1.1 Найбільш популярні сайти для відстежування погоди в Україні

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

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

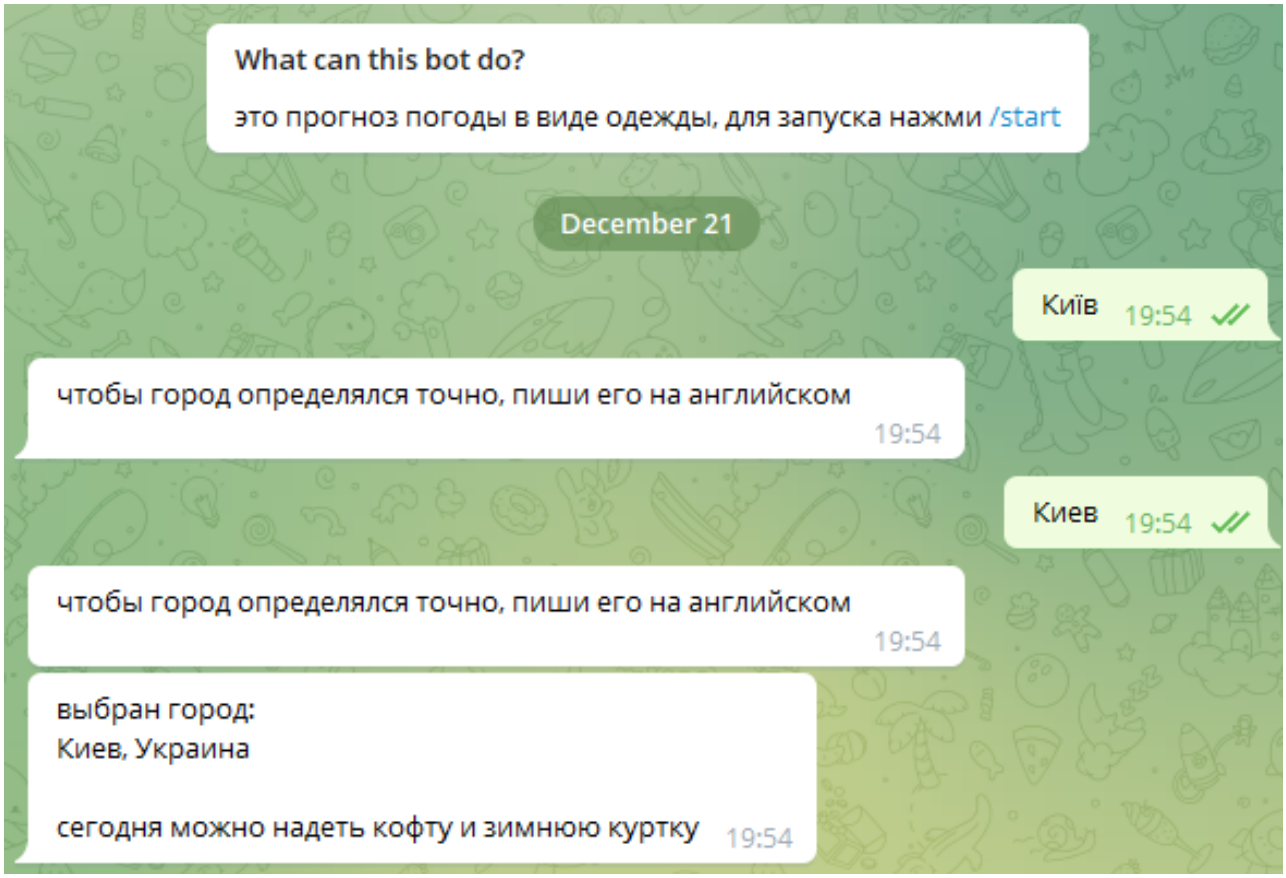

Рис. 1.2 Інтерфейс чат-боту «Прогноз одежды» для Telegram

Дане рішення має ряд недоліків:

- неможливість визначення самих метеорологічних умов, а лише відповідного до них одягу;

- відсутність вибору мови.

Наступним, більш повним рішенням є чат-бот «Погодник». Явною його перевагою є можливість вибору мови спілкування (в т.ч. української), а також досить докладна інформація, що видається користувачеві боту про погоду – рис. 1.3.

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

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

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

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

Meteobot має наступні недоліки:

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

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

- не ефективно працює з географічними місцями, зокрема, на конкретний ввід відомого міста «Київ» пропонує обирати одну з численних (до того ж не зовсім зрозумілих – рис. 1.4) альтернатив, хоча будь-яка людина при такому запиті однозначно зрозуміла би, про яке місто йде мова.

Беручи до уваги недоліки розглянутих програмних продуктів типу чат-боту для відстежування метеорологічних умов, можна зробити однозначний висновок про те, що доцільною власна розробка такого чат-боту.

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

Розглянуті особливості існуючих програмних продуктів спричинюють необхідність розробки власного рішення – чат-боту для Telegram, який би дозволяв проводити моніторинг погоди, або іншими словами, відстежування метеорологічних умов.

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

Вид продукту – чат-бот для месенджера Telegram.

Тип архітектури продукту – клієнт-серверна архітектура.

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

Цільова платформа клієнта – будь-яка актуальна версія месенджеру Telegram (настільна або мобільна).

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

Протокол взаємодії – HTTP.

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

2 РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ПРИ СТВОРЕННІ ЧАТ-БОТУ ДЛЯ ВІДСТЕЖУВАННЯ МЕТЕОРОЛОГІЧНИХ УМОВ

2.1 Обґрунтування вибору архітектури та типу додатку

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

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

Ще однією можливістю є запровадження чат-боту Telegram, який має наступні переваги:

- інформація надходить до пристрою користувача в автоматичному режимі, без прикладання додаткових зусиль з боку користувача, і миттєво надається йому при виникненні такої потреби;

- не потребує встановлення додаткового програмного забезпечення, що споживає апаратні ресурси клієнтського пристрою;

- використовує функціональність надзвичайно поширеного програмного продукту – месенджеру Telegram.

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

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

Слід сказати, що фізично усі повідомлення, що передаються у системі Telegram від кожного абонента (людини чи бота), спочатку надсилаються на анонімні сервери самої програми Telegram, які забезпечують процеси шифрування повідомлень і взагалі керують усім процесом спілкування клієнтів месенджеру між собою (в тому числі – і користувачів з ботами). Проходження повідомлень від бота до клієнта і назад через сервери Telegram відбуваються у повністю прозорому режимі, тобто відображувати цей процес якимось чином у програмному коді бота не потрібно.

Слід відмітити, що повнофункціональний чат-бот запам’ятовує користувачів, які хоча б раз до нього звернулися, для чого слід записувати ідентифікатор відповідного чату з ними до бази даних. Відповідно серверний код, який обробляє запити клієнтської частини, часто працює у зв’язці з сервером баз даних.

Отже, архітектура проектованого чат-боту буде такою, як показано на рис. 2.1.

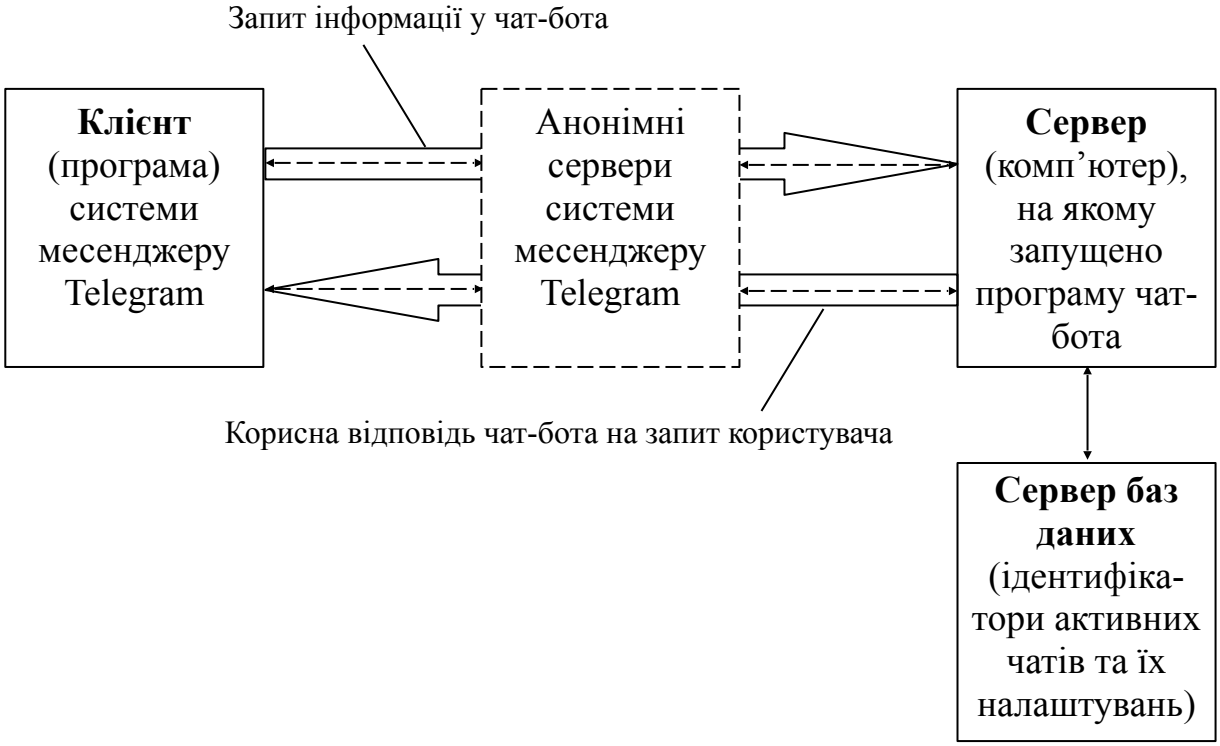

Рис. 2.1 Архітектура проектованого програмного продукту типу чат-бот для Telegram

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

2.2 Вибір основних технологій для створення додатку чат-боту

Першочерговим питанням, яке постає перед розробниками будь-якого програмного забезпечення, є вибір моделі або технології, парадигми його розробки [12]. Такими, що широко використовуються на сьогоднішній день у виробничій практиці, є технології структурного (процедурного) та об'єктно-орієнтованого програмування. Кожна з них має свої особливості, переваги і недоліки, які розглянемо докладніше.

Структурне [16], або як його ще називають практикуючі програмісти, процедурне програмування засноване на використанні окремих структурних блоків – в першу чергу, підпрограм (процедур і функцій). Говорячи більш строго, структурне програмування має на увазі побудову програми відповідно до трьох основних принципів: слідування, розгалуження, повторення.

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

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

Суть же методології об'єктно-орієнтованого програмування [15] полягає в тому, що система розглядається, як сукупність окремих сутностей – об'єктів, які мають набір якихось своїх внутрішніх параметрів – властивостей, а також можуть

взаємодіяти між собою за допомогою деяких дій – викликів методів (або більш непрямим чином – шляхом надсилання повідомлень, оброблюваних методами об’єктів; для цього необхідна присутність активної сутності, яка роздає повідомлення адресатам, як, наприклад, менеджер вікон в ОС Windows, який надсилає вікнам програм адресовані до них повідомлення).

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

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

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

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

Під інкапсуляцією мається на увазі об’єднання даних (значення властивостей класу у деякого конкретного об’єкта) та засобів їх обробки (методи класу). Це знову ж таки зручно психологічно, так як дозволяє реалізовувати окремі завершені сутності - класи, які самі обробляють свої дані. Звернення до об’єктів цих класів відбувається за допомогою методів, що утворюють інтерфейс класу.

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

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

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

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

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

2.3 Обґрунтування вибору мови програмування та відповідної програмної платформи для розробки

Для програмування чат-ботів можна використовувати будь-яку серверну мову програмування. Аналіз публікацій навчального характеру з даного напрямку свідчить про те, що часто для даних цілей використовується мова Python та JavaScript (мається на увазі – на базі серверної платформи Node.JS). Якщо не прив’язуватися до досить вузької теми створення саме чат-ботів, то взагалі кажучи, Python більше використовується для створення інтелектуальних додатків, тобто таких, що мають суттєву інтелектуальну складову. Функціонування чат-боту спирається на порівняно просту логіку типу «запит-аналіз-відповідь», тому використання Python із усіма його потужними бібліотеками та засобами розробки не є цілком обґрунтованим. У даній роботі будемо використовувати мову програмування JavaScript, за допомогою якої створена платформа Node.JS, можливості яких розглянемо докладніше.

Важливою особливістю мови JavaScript є те, що її висхідні програмні коди інтерпретуються, що є досить гнучким, але повільним рішенням. Оператори у цій, в цілому С-подібній мові, розділяються крапкою з комою. Мова чутлива до регістру, що часто випускають з виду початківці, ймовірно тому, що HTML, застосовуваний зазвичай з JavaScript спільно, не залежить від регістру (імена тегів і атрибутів HTML можна писати як малими, так і великими літерами).

Однорядковий коментар виділяється символом //, а багаторядковий - парою символів / * і * /, в чому JavaScript знову повторює С.

До даних застосовується слабкий (динамічний) контроль типів. В операторах з різнотипними даними останні автоматично приводяться до необхідного типу. Типи даних можуть бути примітивними і складеними. Примітивні типи містять прості однорідні значення, такі дані можна передавати функціям як параметри за значенням, а не за посиланням. Складені типи містять

різнорідні дані (в тому числі і складені), їх передають у функції тільки за посиланням.

Мова JavaScript об’єктно-орієнтована, проте заснована на прототипах, а не на класах. Є чотири типи об’єктів: вбудовані об’єкти, об’єкти браузера, об’єкти документа і об’єкти користувача (програміста).

Введення-виведення в основному обмежене взаємодією з документами і користувачами. За умовчанням передбачається, що доступ до локальної файлової системи заборонений. Однак браузери можуть надавати спеціальні об’єкти, за допомогою яких забезпечується робота з файловою системою користувача, хоча і з видачею попереджень про небезпеку виконання файлових операцій.

Сценарії JavaScript активно взаємодіють з об’єктами, вбудованими в Web-сторінку. Для цього вони, власне, і створюються. Але перш, ніж ця взаємодія стане можливою, слід впровадити код сценарію в текст HTML-документа. Існує кілька способів зв'язати HTML-документ з конкретним сценарієм (скриптом), але зазвичай їх просто розміщують всередині контейнерного тега <SCRIPT>, тобто між дескрипторами <script> і </script>.

Контейнер <SCRIPT> в цьому випадку буде перебувати безпосередньо в HTML-документі, причому у довільному його місці. Програмний код пишуть прямо в HTML-документі або в спеціальних текстових файлах, які можна викликати з головного HTML-документа. Для початку розглянемо перший варіант. Перш за все браузер знаходить тег <script> в тілі веб-документа, і весь наступний текст намагається обробити як скриптовий код. І так до тих пір, поки не зустріне закриваючий тег </script>. Після цього всі наступні символи будуть вважатися HTML-текстом. Будь-який HTML-документ може містити довільне число «скриптових включень», але кожне має відкриватися і завершуватися відповідним тегом. Від їх розташування у тілі HTML-документа іноді може залежати функціонування всієї Web-сторінки, але про це буде сказано пізніше.

Контейнерний тег <script> може містити атрибут SRC, який вказує ім’я або URL-адресу текстового файлу, що містить код сценарію. Цей атрибут необхідний в тому випадку, якщо сценарій розташований не безпосередньо в HTML-

документі, а в окремому файлі. Розширення файлу зі сценарієм може бути яким завгодно, але зазвичай використовують js:

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

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

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

<HTML> <HEAD>

<TITLE> Приклад застосування тега NOSCRIPT </ TITLE>

</ HEAD>

<SCRIPT TYPE = "text / JavaScript»>

<! -

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

</ SCRIPT>

<NOSCRIPT>

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

</ NOSCRIPT>

</ HTML>

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

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

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

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

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

Якщо необхідно, щоб сценарій виявився в браузері перш, ніж буде завантажено елементи HTML-документа, то його слід розташувати у верхній частині HTML-коду, а ще краще в контейнері <head> в заголовку документа.

Початково мова JavaScript була виключно браузерною, призначеною для створення динамічних веб-сторінок, але у 2009 р. з'явилася перша версія движка системи Node.JS, що перетворив JavaScript на мову загального призначення. Втім, настільні додатки на Node.JS майже відсутні, а широко даний інструмент використовується для організації бек-енд складових сучасних веб-додатків. Таким

чином, завдяки даній платформі, мова JavaScript стала досить поширеною мовою серверного програмування.

2.4 Проектування способу взаємодії користувача із проектованим чат-ботом

Взаємодія користувача з будь-яким ботом розпочинається із переходу за посиланням виду https://t.me/name_of_a_bot або шляхом пошуку (аналогічно пошуку довільного користувача системи Telegram) за ім'ям боту (@name_of_a_bot – у попередньому прикладі). Після цього дається команда START що виконується шляхом натискання відповідної кнопки у нижній правій частині вікна месенджеру, або безпосередньо задаючи текстову команду /start у полі для набору Telegram-повідомлення. У відповідь бот надсилатиме повідомлення із переліком доступних команд, які будуть наступними:

а) /now <місто> – видати інформацію про погодні умови в даний момент у заданому місті;

б) /tomorrow <місто> – видати інформацію про погодні умови завтра у заданому місті;

в) /week <місто> – видати інформацію про погодні умови на найближчі 7 днів у заданому місті;

г) /w – видати інформацію про погодні умови на найближчі 3 дні у місті, яке раніше було встановлене за основне.

Окрім команд, пов'язаних безпосередньо із погодою, також слід запровадити команди налаштувань боту, які будуть наступними:

ґ) /lang <мова> – встановлює мову спілкування боту (тобто видачі інформації ним);

д) /set <місто> – встановлює основне місто для даного користувача, погодні умови за яким будуть видаватися, якщо місто не вказано, наприклад, по команді /w;

е) /location – видає поточне основне місто.

є) /help – ще раз видає перелік доступних текстових команд боту (стандартна команда).

Даних команд буде достатньо для того, щоби реалізувати функціональність, визначену завданням на розробку та розділом постановки задачі даної роботи.

2.5 Формалізація алгоритмів роботи проектованого додатку

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

Рис. 2.2 Блок-схема алгоритму роботи чат-бота, що надає інформацію про погодні умови

Керуючись інформацією, наведеною у даному підрозділі, а також у вищенаведеному викладі, можна переходити до процесу проведення програмної реалізації чат-боту для месенджера Telegram, що і виконується у наступному розділі.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЧАТ-БОТУ ДЛЯ ВІДСТЕЖУВАННЯ МЕТЕОРОЛОГІЧНИХ УМОВ

3.1 Налаштування програмного середовища для функціонування додатку

Створенню підлягає програмний продукт типу чат-бот для месенджера Telegram, основна функціональність якого пов’язана із наданням користувачеві інформації про погоду. Перед тим, як проводити програмування такого додатку, потрібно провести певні організаційні дії, пов’язані із його налаштуваннями. В першу чергу, для створення довільного чат-бота у Telegram потрібно отримати його ідентифікатор, або за термінологією Telegram – токен (англ. Token). За допомогою даного ідентифікатору здійснюється уся взаємодія в системі «чат-бот – користувач» і без нього створити чат-бота будь-яким способом не можливо. Для отримання токена слід у Telegram звернутися у чаті до головного чат-бота BotFather і провести декілька дуже простих команд:

а) при натисненні кнопки START – рис. 3.1 – висвітлюється досить обширний перелік усіх команд по налаштуванню чат-ботів;

б) найпершою командою є /newbot, що дозволяє перейти до процесу створення нового бота;

в) в першу чергу BotFather пропонує задати ім’я для нового бота – рис. 3.2, яке відповідає імені користувача при спілкуванні в Telegram, тому це поле може бути не унікальним (тобто може співпадати з іменами існуючих користувачів та іменами інших ботів);

г) після введення імені BotFather запитує унікальне ім’я бота (або нік за термінологією Telegram), що не може співпадати з іменами інших ботів та повинно закінчуватися на літери «bot»;

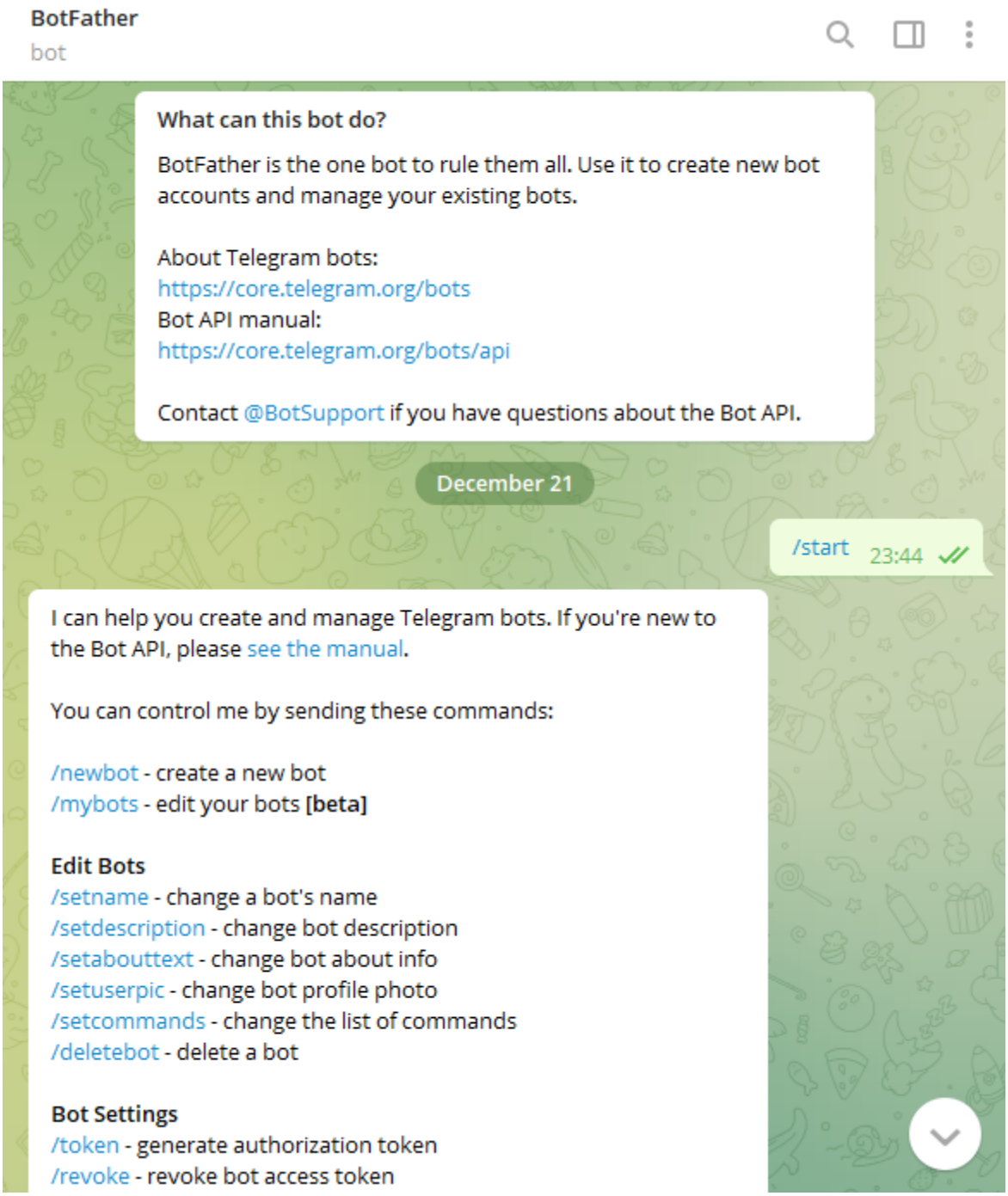

Рис. 3.1 Стартовий екран засобу BotFather, який дозволяє створювати нові чат-боти

д) у відповідь BotFather надсилає велике повідомлення, у якому найголовнішою інформацією є токен виду 270485614:

AAHfiqksKZ8WmR2zSjiQ7_v4TMAKdiHm9T0. Цей текст треба зберегти та не поширювати, оскільки він надає повний контроль над ботом.

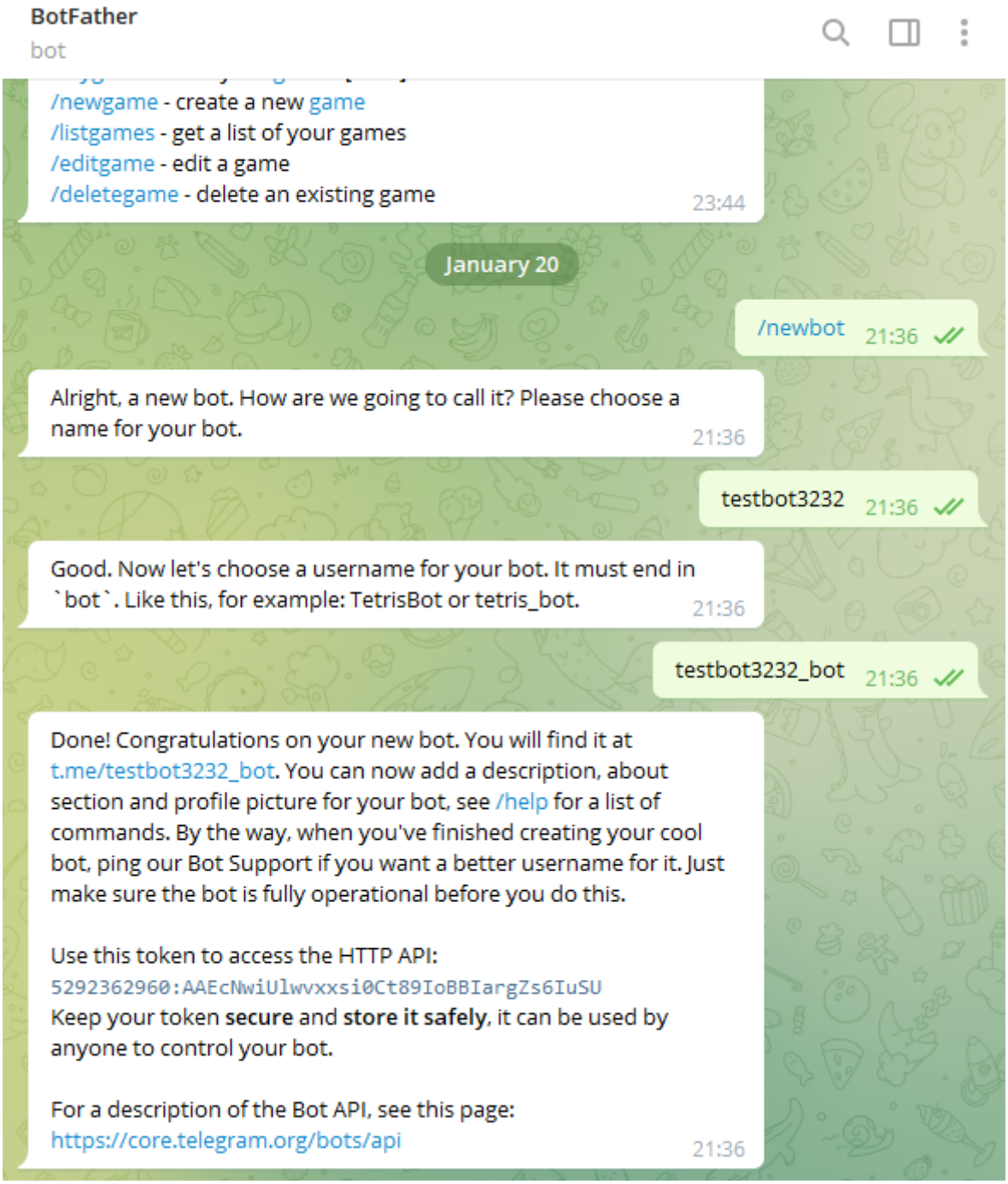

Рис. 3.2 Завдання імені та ніка для нового бота, а також отримання його токену

Для створення такої програми, як чат-бот у Telegram обов’язкових налаштувань більше не потрібно (хоча з додатковими налаштуваннями можна

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

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

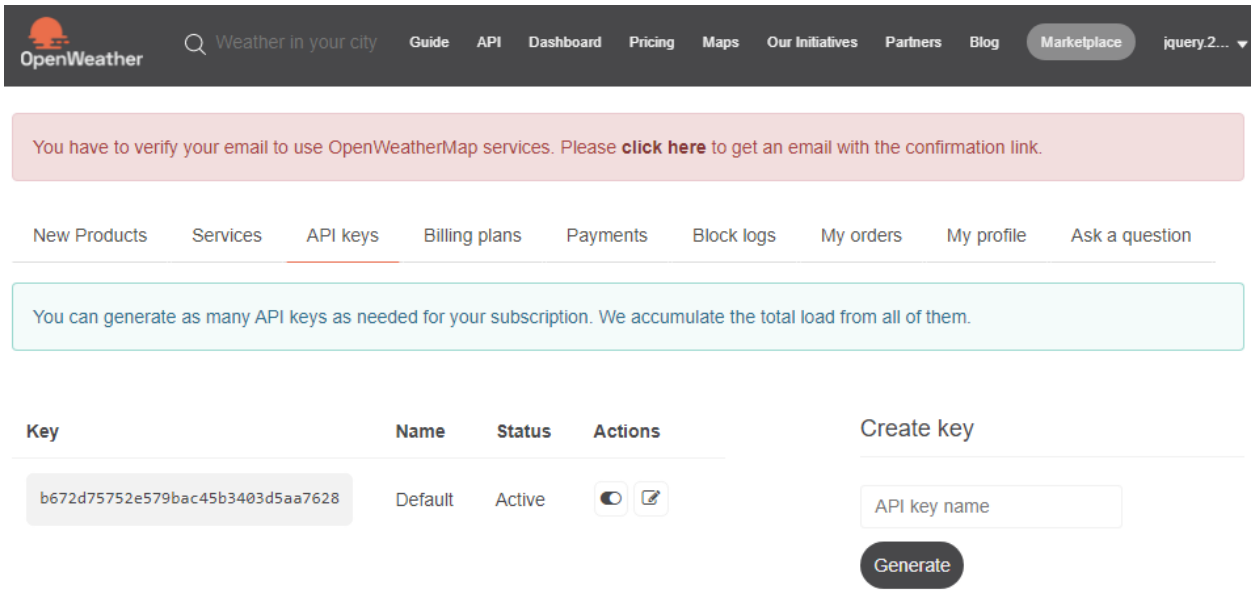

Рис. 3.3 Персональна сторінка безкоштовного тарифного плану на сервісі OpenWeatherMap

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

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

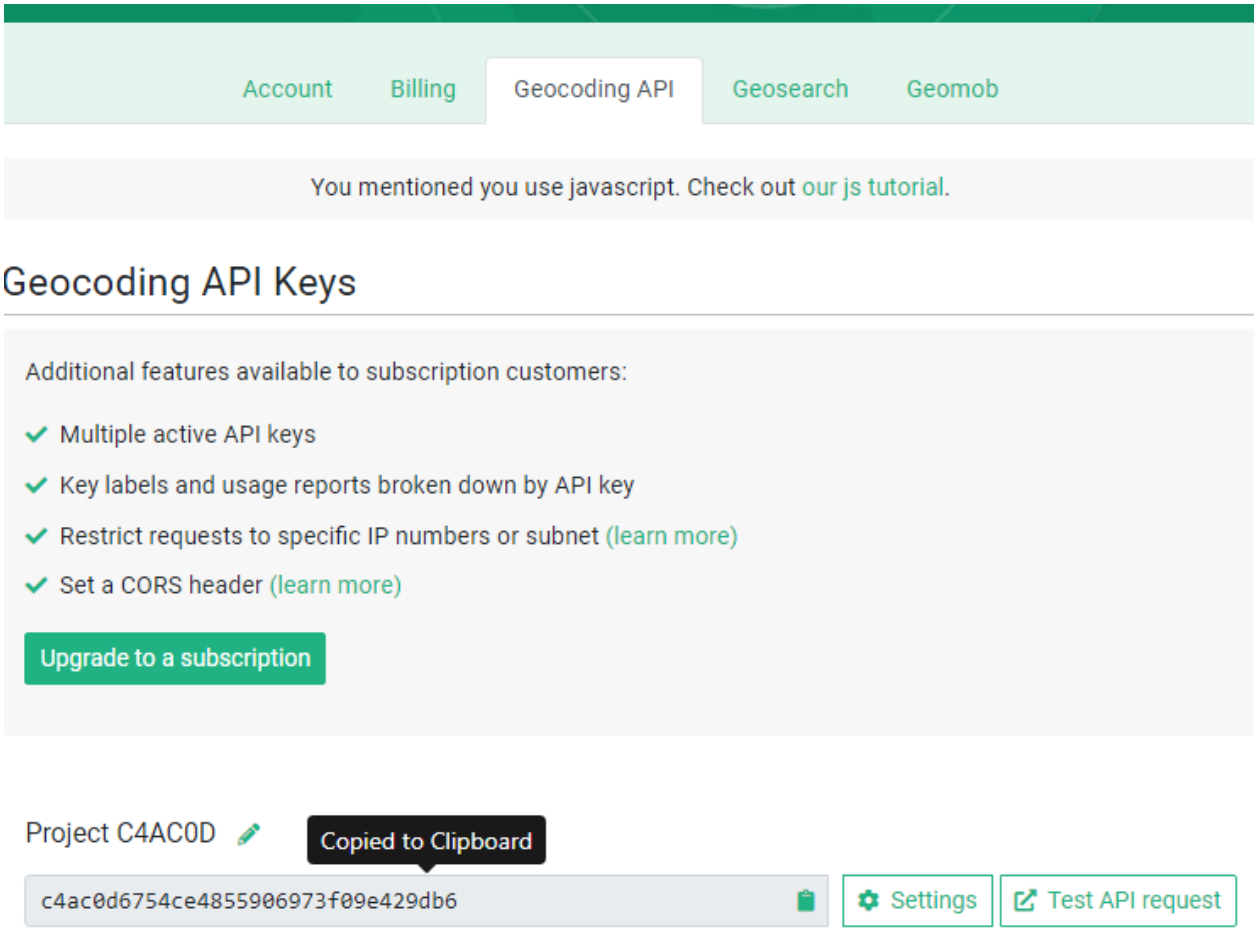

Рис. 3.4 Персональний кабінет пробного тарифного плану на платформі OpenCage

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

Одним із тривіальних варіантів використання саме MongoDB є дуже простий випадок, коли потрібно десь зберігати налаштування програми (які неефективно розміщувати в таблицях, оскільки практично всі вони є унікальними і не повторюються). В загальному випадку ця система використовується, коли у сховищі даних слід зберігати обширні дані, що не є структурованими, наприклад, являють собою окремі документи (і тому сама СУБД називається документоорієнтованою).

СУБД MongoDB слід завантажити з офіційного серверу mongodb.com, обираючи безкоштовну версію MongoDB Community Server – рис. 3.5.

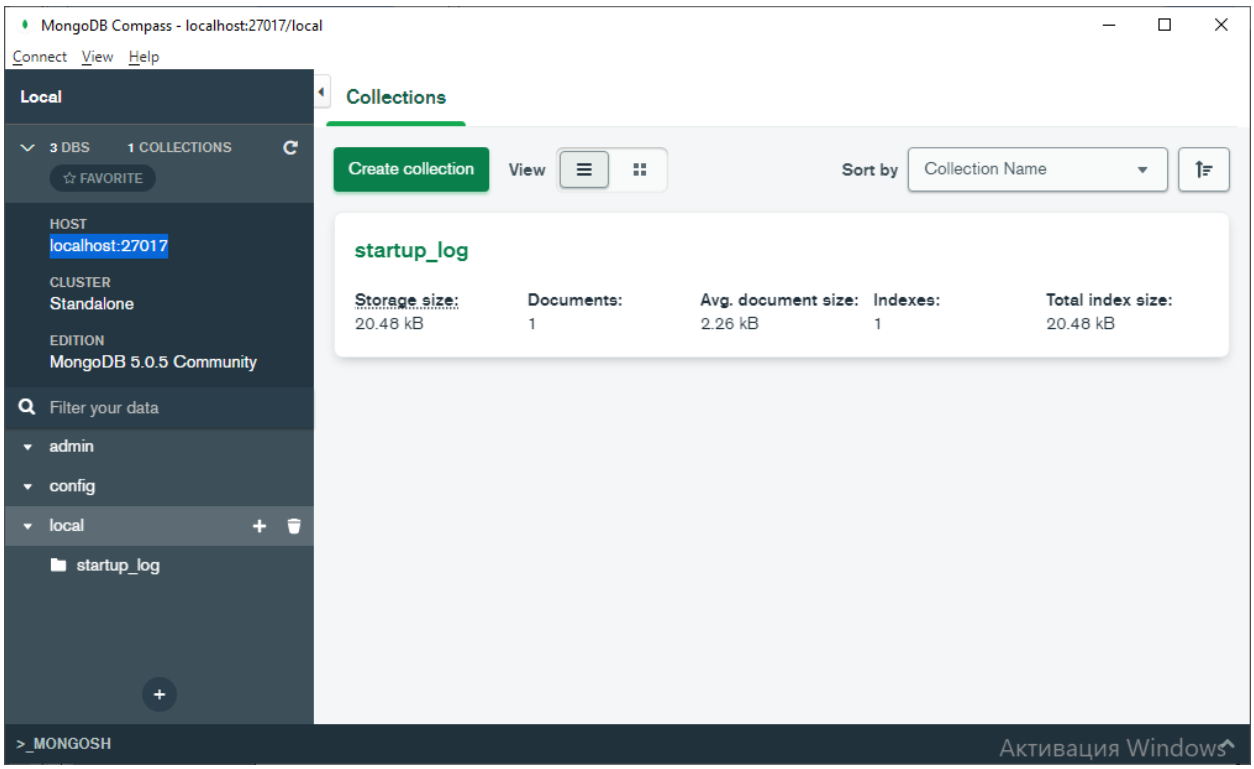

Рис. 3.5 Вікно засобу Compass для управління СУБД MongoDB

Також, як було встановлено вище, основою для розробки буде платформа Node.JS, яку також слід завантажити (з офіційного сайту https://nodejs.org) та встановити на цільовому комп'ютері.

Також для роботи з Node.JS проектами потрібний менеджер пакетів командного рядка. Один із них уже вбудований в систему Node.JS: npm розшифровується як Node Package Manager. За бажанням за допомогою npm можна встановити один із інших менеджерів пакетів, наприклад, yarn для того, щоби проводити встановлення та оновлення необхідних додаткових модулів системи Node.JS.

Отже, за допомогою менеджеру пакетів можна встановлювати нові пакети (бібліотеки, розширення), потрібні для конкретного програмного продукту, що розробляється. В першу чергу, слід створити окрему папку, де буде писатися весь проект і зайти у неї за допомогою командного рядку, використовуючи традиційні команди типу cd для зміни поточного каталогу. Далі слід виконати команду створення нового порожнього проекту Node.JS:

```
npm init
```

Для того, щоби вручну не відповідати на усі питання, а дозволити відповіді за умовчанням, команда має містити ключ yes:

```
npm init --yes
```

Після встановлення необхідних елементів нового проекту (яке може бути досить тривалим процесом, залежно від потужності цільового комп'ютера), слід вручну довстановлювати ті компоненти, які будуть специфічними для проекту чат-боту.

Для роботи з базою даних MongoDB зручно використати спеціальну бібліотеку для мови JavaScript, яка називається mongoose. Mongoose представляє собою спеціальну ODM-бібліотеку (Object Data Modelling) для роботи з MongoDB, яка дозволяє зіставляти об'єкти класів та документи колекцій із бази даних. Дана бібліотека встановлюється у командному рядку в каталозі проекту рядком виду:

```
npm install mongoose
```

Для надсилання запитів HTTP, за якими саме і працює чат-бот Telegram необхідно встановити бібліотеку Axios:

```
npm install axios
```

Також при розробці чат-боту Telegram доцільно завантажити пакет node-telegram-bot-api, що можна зробити, наприклад, командою:

```
npm install node-telegram-bot-api
```

На цьому підготовче налаштування програмного середовища можна вважати завершеним і переходити безпосередньо до реалізації чат-боту для відстежування метеорологічних умов за допомогою мови програмування Node.js.

3.2 Особливості реалізації алгоритмів роботи додатку

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

```
const TelegramToken =
"5292362960:AAEcNwiUlwvxxsi0Ct89IoBBIargZs6IuSU"
const OpenWeatherKey = "b672d75752e579bac45b3403d5aa7628"
```

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

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

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

В базі зберігаються ідентифікатор користувача в системі месенджера Telegram, обране ним місто та мова (за умовчанням – англійська). При додаванні нового користувача у БД створюється новий запис наступного виду – рис. 3.6.

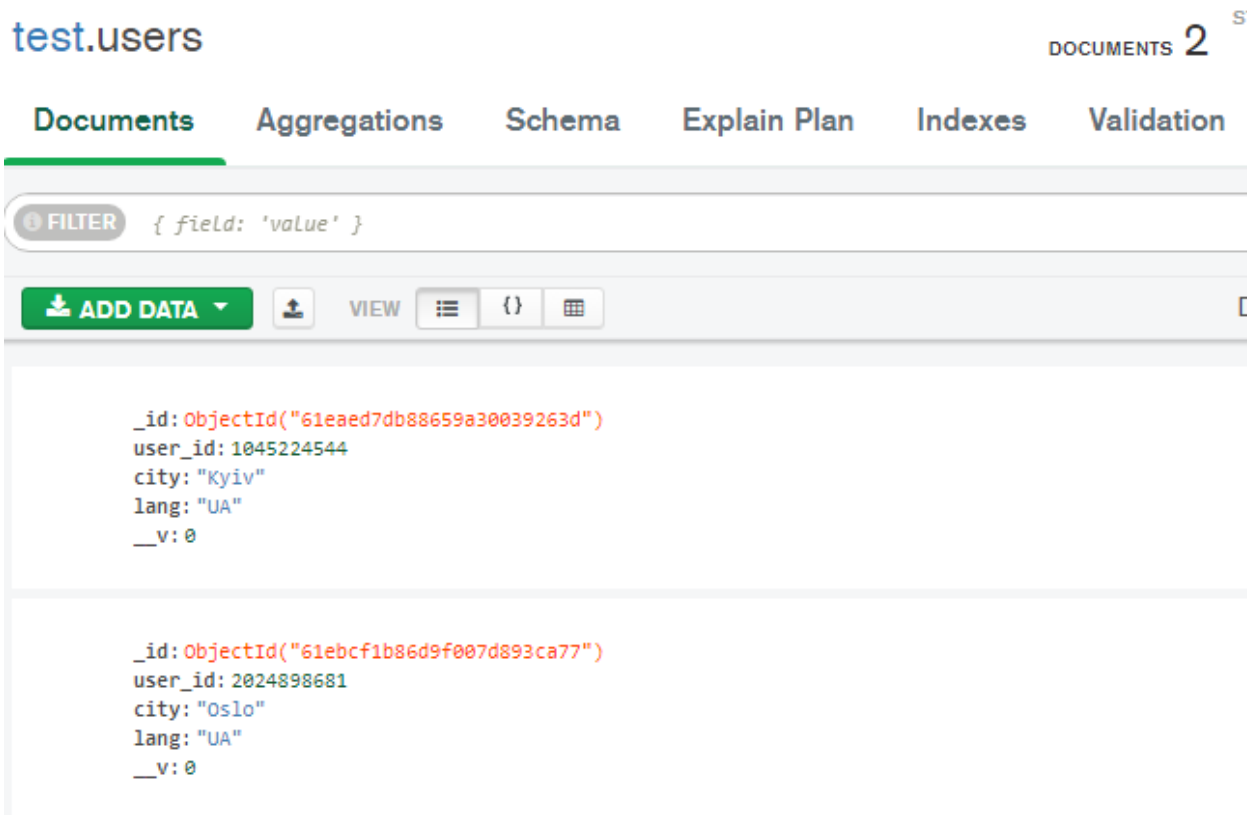

Рис. 3.6 Дані, що зберігаються у базі MongoDB

У тілі програми, яка по суті і являє собою чат-бота (файл index.js) запроваджено 7 окремих функцій та 12 методів для об’єкта MyBot, який створюється рядком:

```
const MyBot = new TelegramBot(TelegramToken, { polling: true });
```

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

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

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

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

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

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

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

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

Серед функцій, що є обробниками повідомлень користувача в чаті, можна назвати наступні:

- обробник запиту /start, що виводить перше повідомлення та видає перелік доступних команд;

- обробник запиту /set, що дозволяє задати базове місто даного користувача;

- обробник запиту /lang, що дозволяє встановити мову, якою буде видаватися пояснювальний текст про погоду;

- обробник запиту /week, що видає погоду на цілий тиждень у заданому місті;

- обробник запиту /now, що видає погоду зараз у заданому місті;

- обробник запиту /tomorrow, що видає погоду на завтра у заданому місті;

- обробник запиту /w, що на відміну від попередніх команд не потребує завдання міста, а видає погоду на три дні у базовому місті;

- обробник запиту /location, за допомогою якого можна автоматично вказати поточне місцезнаходження без необхідності введення назв чи координат (тобто за допомогою програмно-апаратного забезпечення смартфону);

- обробник запиту /help, що видає допоміжну інформацію про чат-бот (аналогічно команді /start).

Інші особливості програмного коду можна дізнатися із початкового тексту, наведеного у Додатку А.

3.3 Тестування створеного програмного продукту

Будь-який програмний продукт після проведення реалізації має бути підданий тестуванню. У даному випадку процедура тестування є наступною. У каталозі проекту у командному рядку задається команда завантаження проекту:

```
npm start
```

Після цієї команди проект завантажується на виконання і не закінчується, а блокує консоль і знаходиться у режимі виконання – рис. 3.7.

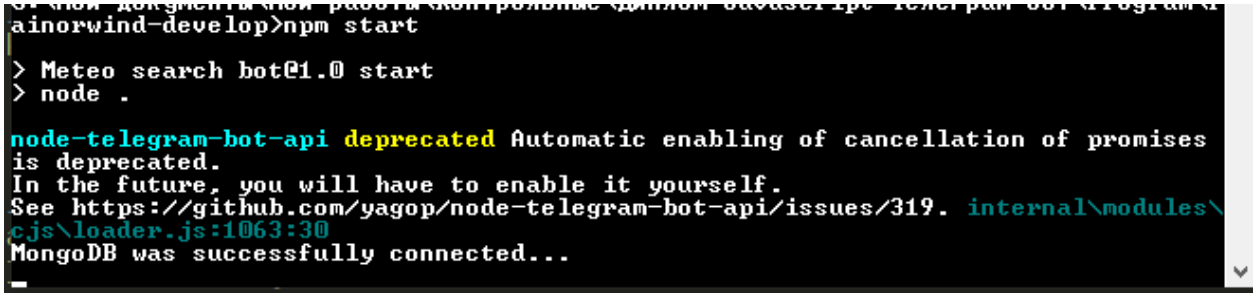

Рис. 3.7 Блокування консолі завантаженим проектом протягом всього часу його роботи

Для припинення роботи чат-боту потрібно натиснути Ctrl+C та підтвердити вихід із додатку натисканням кнопки «y» («yes»), або будь-яким способом закрити вікно консолі. Слід відмітити, що будь-яке спілкування з ботом у Telegram можливе тільки тоді, коли на якомусь комп’ютері, підключеному до мережі Інтернет, запущено консоль із відповідним (розробленим у даній роботі) додатком на платформі Node.JS. Ця умова виключає розміщення програмного забезпечення чат-боту на домашньому комп’ютері, який періодично вимикається (адже під час таких вимкнень бот у Telegram не буде працювати). Тому, для розміщення програми чат-боту у професійному випадку слід використовувати хостинг мережі Інтернет, який дозволяє розміщувати Node.JS+MongoDB проекти. Нажаль,

отримати безкоштовний доступ до сервера саме Node.JS не є простою задачею (на відміну від повністю вільних PHP хостингів, яких чимало): часто доступні сервери змінюють власну політику, а через їх малу кількість зміна навіть одного сервера є досить важким ударом по даній галузі. Таким чином, більш надійним рішенням є використання платного хостингу, або безкоштовних варіантів із веденням постійного контролю, чи не потрібно переходити на інший безкоштовний варіант.

Отже, тестування відбувалося у повністю локальному режимі, тобто і Node.JS сервер і MongoDB сервер запускалися на звичайному локальному комп'ютері, що має доступ до мережі Інтернет.

В першу чергу, слід сказати про стартовий екран – рис. 3.8.

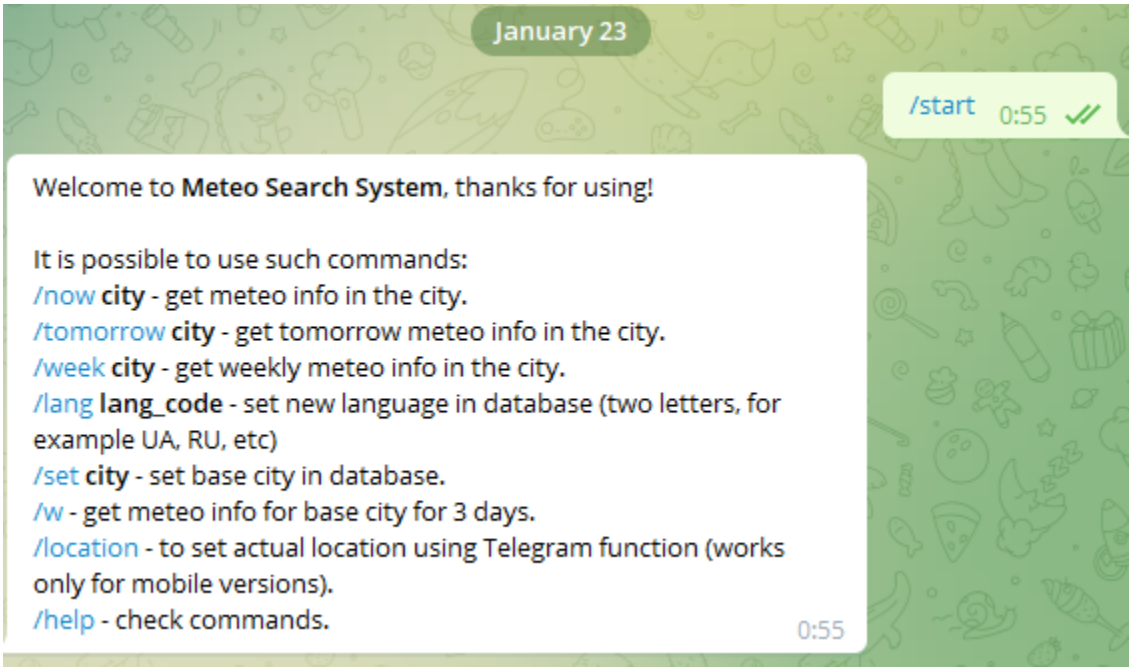

Рис. 3.8 Зовнішній вигляд стартового екрану створеного чат-боту

Екран, аналогічний до рис. 3.8 з'являється також при виборі команди /help.

При завданні команди /now із вказанням назви якогось міста у відповідь чат-бот видає правильну поточну погоду – рис. 3.9. Тут слід звернути увагу, що по-перше, видається картинка, що відповідає погоді, наприклад, на рис.3.9 абсолютно у відповідності до тексту "light rain" відображується картинка із невеликим

дощиком. – це означає, що функція wIcon працює вірно. Під картинкою видається гарно відформатоване повідомлення, яке у точності відповідає HTML-шаблону, що формується функцією wHTMLTemplate.

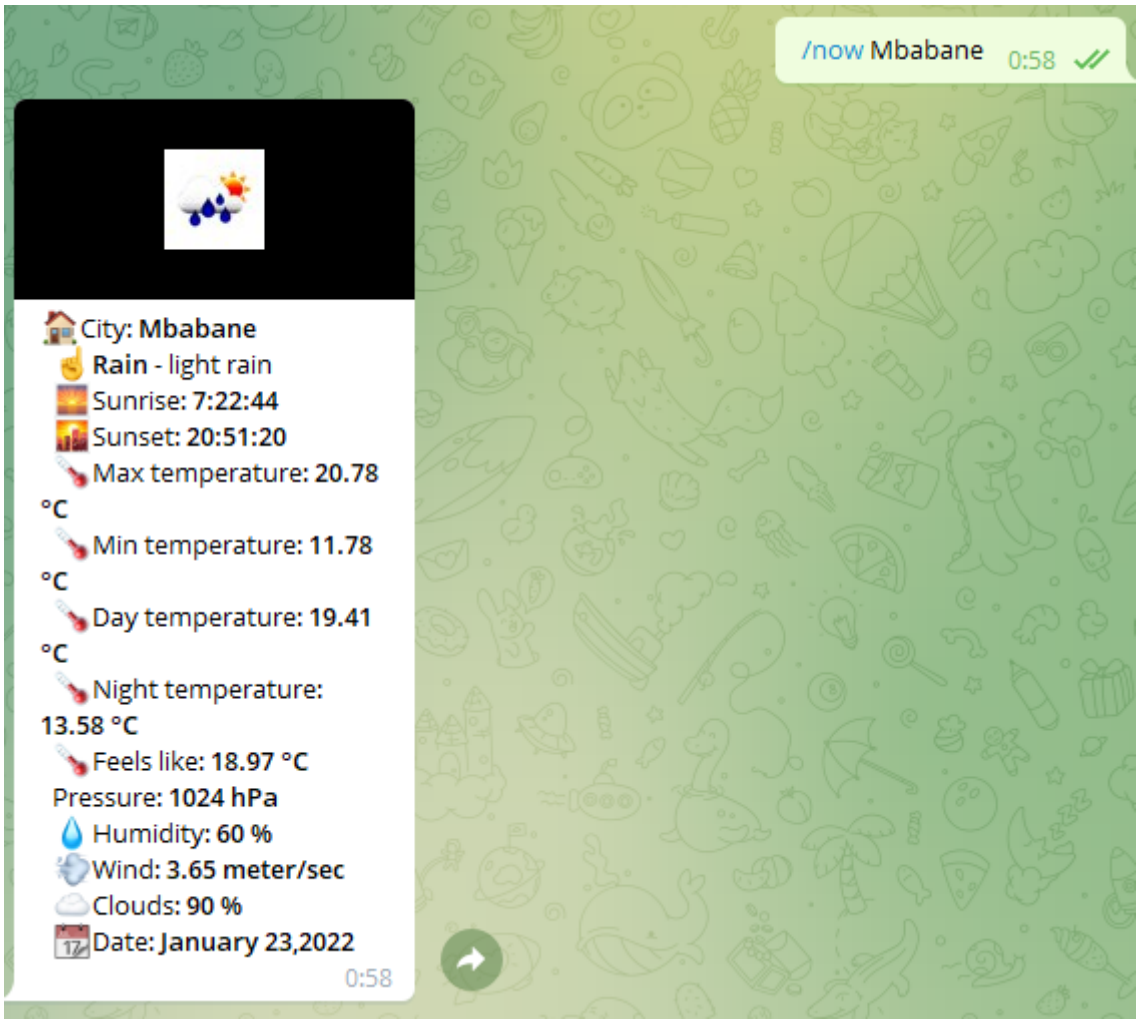

Рис. 3.9 Коректне відображення поточної погоди для заданого міста

Аналогічно до команди /now працюють і дві інших команди /tomorrow та /week. Усім трьом розглянутим командам обов’язково слід задавати назву міста, де користувач хоче дізнатися погоду.

Ще одна команда для визначення погоди /w навпаки не потребує завдання міста, а використовує те місто, яке користувач задав раніше командою /set. Наприклад, на рис. 3.10, спочатку задано місто London, а потім команда /w, що працює без вказування міста, видає інформацію про погоду саму у Лондоні.

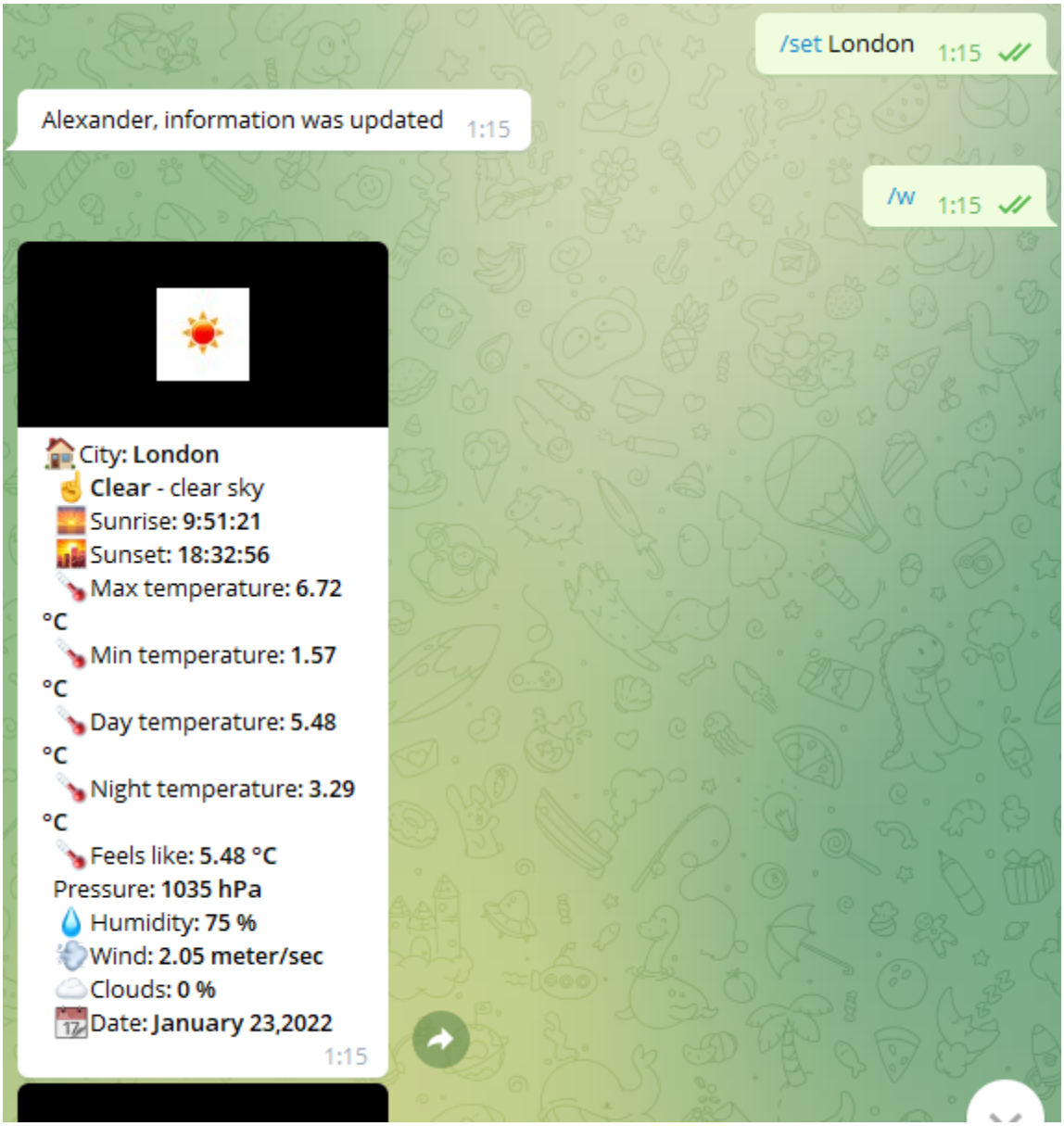

Рис. 3.10 Коректна робота команди /set для завдання міста у базі даних та /w для видачі погоди на три дні у цьому (базовому) місті

Нарешті, тестування команди /lang показало, що вона дійсно коректно працює (на рис. 3.11 спочатку задається українська мова, а потім видно, що видача використовує український текст, який можна порівняти з рис. 3.10), але частина всієї інформації, що підлягає переведенню є досить малою, тому особливої доцільності у використанні даної команди немає.

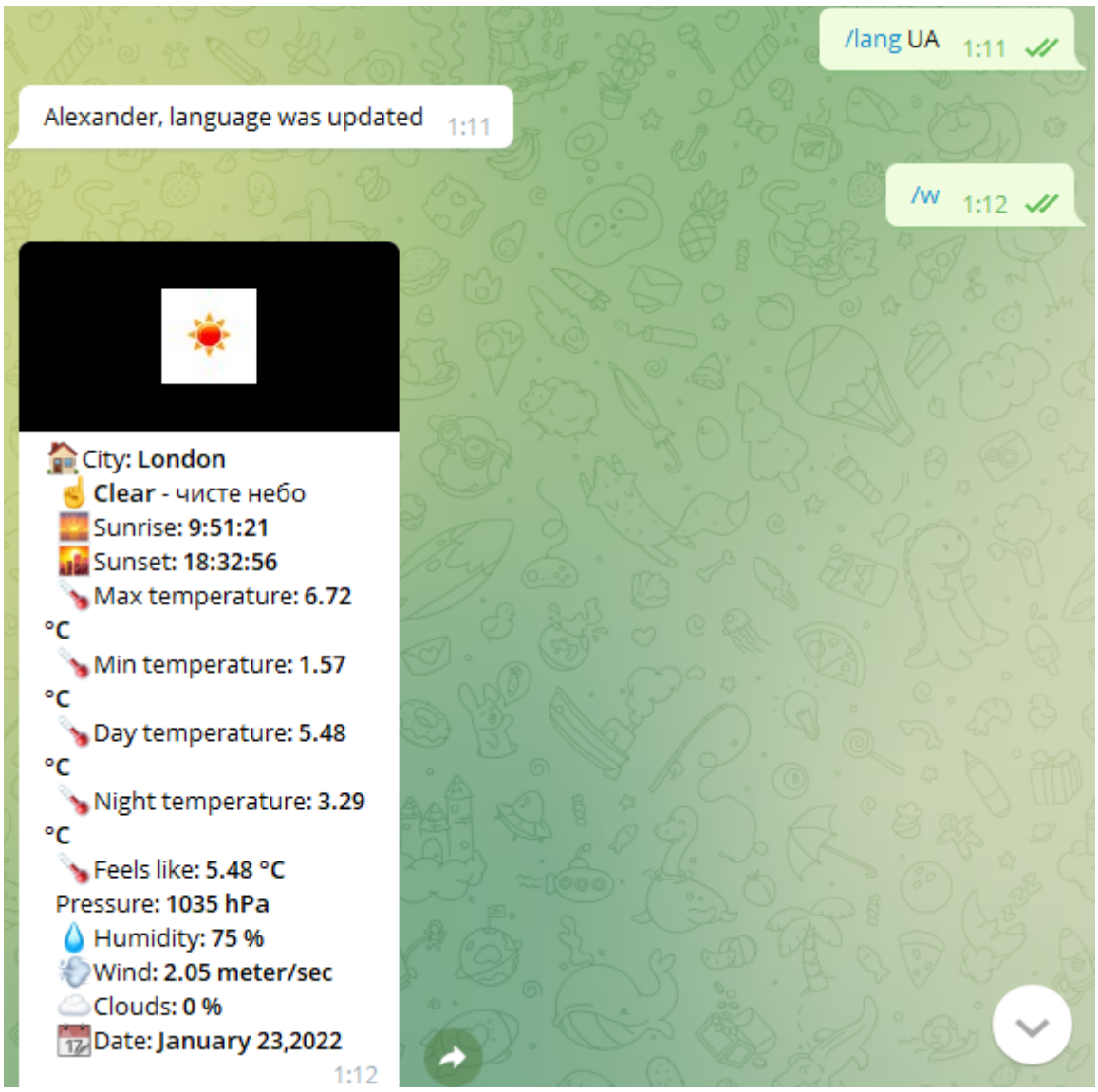

Рис. 3.11 Коректна робота функції завдання мови для видачі прогнозу погоди

Таким чином, проведене тестування показало, що чат-бот коректно виконує власну функціональність, що полягає у видачі користувачеві метеоумов поточного моменту, а також прогнозних значень на найближчі дні, причому на різних мовах та у географічно різних місцях.

3.4 Аналіз ефективності розробленого рішення

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

Для того, щоби дізнатися погоду за допомогою сайту потрібно завантажити браузер (на відміну від Telegram, браузер не є постійно наявним в оперативній пам'яті смартфону чи ПК), ввести адресу сайту погоди та у досить великому, насиченому інформацією вікні, знайти відомості про метеорологічні умови та прочитати їх. Умовно можемо вважати, що на ці дії користувач витрачає $\Delta t = 3$ хв.

Після запровадження чат-боту дізнатися погоду можна буде через Telegram, який постійно знаходиться у пам'яті смартфону а також надає інформацію у значно більш стисненому вигляді: тільки корисні текстові дані і жодних відволікаючих факторів (якими насичені веб-сайти для перегляду погоди). Через усі ці фактори можемо вважати, час проведення операції тепер складатиме $\Delta t' = 1$ хв.

Економія часу на однократному визначенні погоди складає:

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

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

$$N = 250 \text{ днів/рік.}$$

Загальна економія часу на рік складатиме:

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

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

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

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

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

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