НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет електроніки
(повна назва інституту/факультету)

Кафедра акустичних та мультимедійних електронних систем
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

С.А. НАЙДА
(ініціали, прізвище)

“ 10” червня 2022 р.

Дипломна робота
на здобуття ступеня бакалавра

зі спеціальності (спеціалізації) 171 Електроніка (Електронні системи мультимедія та засоби Інтернету речей)
(код та назва спеціальності)

на тему: «Створення інтернет-боту для інформаційного порталу»

Виконав (-ла): студент IV курсу, групи ДВ-81
(шифр групи)

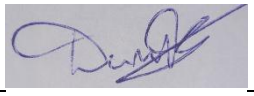

Димитров Артем
(прізвище, ім’я, по батькові) (підпис)


Керівник доцент, к.т.н., Філіпова Н. Ю.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)


Рецензент доцент каф. ЕПС, к.т.н., доц. Клен К. С. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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


Студент ___ ___

Київ – 2022 РОКУ

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»Факультет елетроніки
Кафедра Акустичних та мультимедійних електронних систем

Рівень вищої освіти — перший (бакалаврський) Спеціальність — 171 «Електроніка»
Освітньо-професійна програма «Електронні системи мультимедія та засоби Інтернету речей»

ЗАТВЕРДЖУЮ
Завідувач кафедри

Сергій НАЙДА
«02»травня 2022 р.

ЗАВДАННЯ

на дипломну роботу студенту

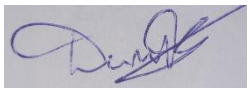
Димитрову Артему

1. Тема роботи:« Створення інтернет-боту для інформаційного порталу », керівник роботи: Філіпова Наталія Юріївна, кандидат технічних наук затверджені наказом по університету від «06» червня 2022 р. № 911-С.
2. Термін подання студентом роботи: «10» червня 2022 р.
3. Вихідні дані до роботи: Мова програмування Javascript, середовище програмування Visual Studio Code, цільова платформа Telegram, обрана для розробки технологія – Telegram bot, обранна технологія для спрощення праці GIT, HTTP запити для взаємодії з сервером.
4. Зміст роботи: порівняльний аналіз платформ чат-ботів, аналіз та вибір технологій для розробки системи керування чат-ботом, розробка клієнтської частини системи керування чат-ботом, висновки, додатки.
5. Перелік ілюстративного матеріалу: набір слайдів презентації з узагальнюючими матеріалами та описом створеного ефекту.

6. Дата видачі завдання 02.05. 2022р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Написання першого розділу	16.05.2022	виконано
2	Написання другого розділу	24.05.2022	виконано
3	Написання третього розділу	25.05.2022	виконано
4	Підготовка матеріалів до друку та оформлення пояснювальної записки	01.06.2022	виконано
5	Підготовка та оформлення презентації для доповіді	10.06.2022	виконано

Студент  Димитров А.

Керівник роботи  Філіпова Н.Ю.

УДК 681.3.07

РЕФЕРАТ

Димитров А. Розроблення telegram боту з візуальним контентом для пошуку інформації по регіонам: дипломна робота бакалавра : 171 Електроніка. - КПІ ім. Ігоря Сікорського, Київ, 2022. - 80 с.

Ключові слова: HTTP, JavaScript, node, Heroku, Telegram API інтернет-ресурс, web-ресурс, мова програмування.

Об’єктом дослідження telegram bot з візуальним контентом.

Метою роботи є дослідження та розроблення telegram боту з візуальним контентом для пошуку інформації по регіонам

Методом дослідження є теоретичне дослідження особливостей створення інформаційного ресурсу за допомогою бібліотеки node-telegram-bot-api.

UDK 681.3.07

ABSTRACT

Dymytrov Artem "Development of a telegram bot with visual content to search for information by region: Bachelor Thesis: 171 Electronics. - Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2022. - 80 p.

Keywords: HTTP, JavaScirt, node, Heroku, Telegram API Internet resource, web resource, programming language.

The object of study is a telegram bot with visual content.

The aim of the work is to research and develop a telegram bot with visual content to search for information by region

The research method is a theoretical study of the features of creating an information resource using the library node-telegram-bot-api.

ЗМІСТ

ВСТУП

З кожним роком все більше і більше люди впроваджують автоматизацію у всі сфери діяльності людини. Пару років тому зайшовши на сайт або в якусь групу ви могли побачити менеджера, який міг відповісти на всі ваші запитання, то тепер через те, що розширюється веб-ком'юніті і все більше і більше людей починають користуватися месенджерами лідерство починають займати Чат-боти.

Для початку потрібно визначити, що представляє собою чат-бот. Боти — спеціальні акаунти в Telegram та інших месенджерах, створені для автоматичного оброблення та надсилання повідомлень. Користувачі можуть взаємодіяти з ботами за допомогою повідомлень, що надсилаються через звичайні або групові чати. Логіка робота контролюється за допомогою HTTPS запитів до API Telegram для роботів [1].

Актуальність бакалаврської роботи зумовлена війною, високою популярністю месенджерів, і таких засобів автоматизації як чат-боти серед користувачів мережі Інтернет, складністю пошуку достовірної інформації та достовірних джерел. Чат-бот суттєво спрощує пошук інформації, таку як останні новин, достовірні джерела та іншого. Було поставлено мету: створити телеграм-бота, який зможе відповідно до обраного регіону користувача, надати йому необхідну інформацію про офіційні групи, карту бойових дій, номера телефонів гарячої лінії та інші корисні ресурси [1].

Виходячи з , були поставлені наступні цілі: а) аналіз та
вивчення ресурсів за обраною темою;
б) порівняння та вибір чат-ботів в
усіх месенджерах; в) вибір
IDE(інтегрованої середи розробки)
та технологій г) розробка чат-бота
на платформі Telegram.

РОЗДІЛ 1 ПОРІВНЯЛЬНИЙ АНАЛІЗ ПЛАТФОРМ ЧАТ-БОТІВ

1.1 Цикл роботи чат-боту

Чат-боти це по суті як інший користувач, в якого є обмежений функціонал заданий у певному алгоритмі. Стандартний цикл аналізу та виконання запиту користувача можна уявити у наступному ланцюгу:

- отримання запиту від користувача(певний набір команд який є прописаним);
- розбір запиту - (порівняння запиту та аналіз);
- виконання певної функції відповідно до запиту користувача;
- Генерація результату(відповіді) після виконання певної функції;
- відправка відповіді користувачеві.

Як і всі інтернет ресурси, програми та додатки, які зберігають та віддають якусь інформацію користувачу, чат-боти мають таку структуру:

a) Серверна частина (backend), набір засобів, за допомогою яких відбувається реалізація логіки веб-сайту, додатку тощо;

b) Клієнтська частина (frontend), розробка функціональності та інтерфейсу користувача, що працюють на клієнтській стороні програми або веб-сайту. Як правило, месенджер надає API та документацію для зв'язку платформи backend'а з чат-інтерфейсом бота через HTTP[2].

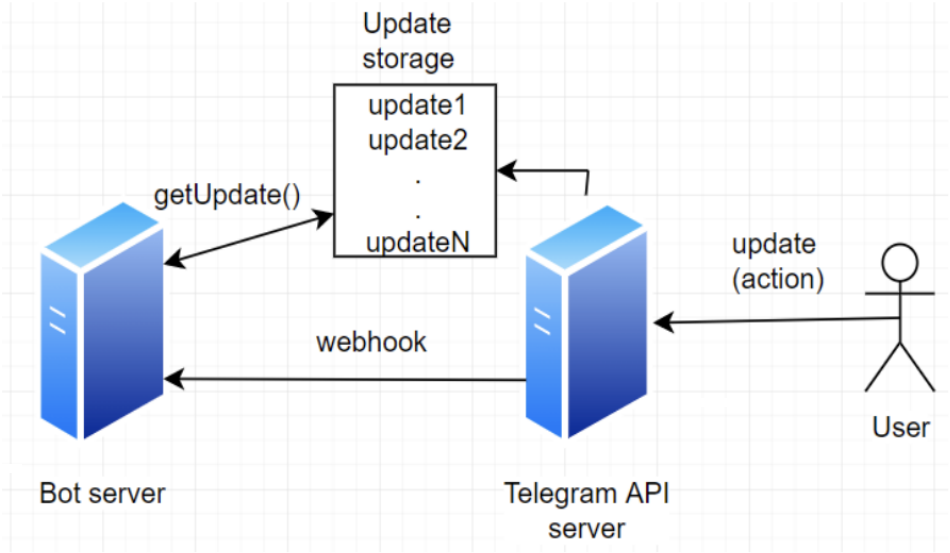


Рисунок 1.1 - Схема отримання запитів від Telegram API.

1.2 Класифікація чат-ботів

У чат-ботів є кілька класифікацій: за алгоритмом, видом та функціональністю.

За алгоритмом бувають примітивні чат-боти, та що саморозвиваються.

Примітивні

До примітивних (обмежених) відносяться найпростіші чат-боти. Вони мають невеликий функціонал і заздалегідь підготовлений скрипт, яким вони спілкуються з користувачами.

Саморозвиток

В основі чат-бота, що саморозвивається, лежить нейронна мережа. Завдяки цьому бот розуміє, що пише користувач і, аналізуючи інформацію, видає цілком реалістичні відповіді. Іноді може здатися, що ви спілкуєтесь зі справжньою людиною.

На вигляд чат-боти діляться на кнопкові та текстові.

Кнопковий

Спілкування із чат-ботом відбувається за допомогою спеціальних кнопок, на яких запропоновані заздалегідь підготовлені відповіді.

Текстовий

Діалог між чат-ботом та користувачем відбувається за допомогою тексту. Бот розпізнає ключові слова у запиті користувача та на основі отриманої інформації видає відповідь.

За функціоналом виділяють комунікаційних та функціональних чат-ботів.

Комунікаційні

Комунікаційні чат-боти беруть він обов'язки консультантів. Їхнє основне завдання - це спілкування з користувачем. У їх функціонал може входити відповіді на запитання користувачів, допомога у підборі товару або послуги, а також інформування про різні акції та знижки.

Функціональні

Завдяки функціональним чат-ботам ставати можливим здійснювати певні дії, такі як купівля товарів чи послуг, проведення оплати тощо[3].

Популярність чат-ботів - це не просто віяння якихось сучасних трендів, а дійсно дуже зручна та ефективна річ для вирішення типових завдань, наприклад:

- допомогти користувачеві зробити замовлення;
- прийняти заявку;
- зібрати зворотний зв'язок від користувачів;
- дати консультацію;
- зібрати контактні дані;
- реалізувати різні товари та послуги;
- допомогти користувачеві записатися в салон краси/ прийом до лікаря та ін;
- завантажити музику, фільм чи книгу.

Це далеко не весь список із можливостями чат-ботів. Є навіть боти для створення ботів.

Чат-бот зручний не лише для підприємців, а й для клієнтів. Одна з головних причин – це здатність чат-бота відповідати 24/7, що рідко зустрічається під час роботи з людським фактором. Причому при спілкуванні з чат-ботом відповідь надходить миттєво, не потрібно чекати до наступного дня, поки консультант прочитає повідомлення і вирішить, що вам відповісти [3].

- чи правильно працює випадаюче меню
- чи зручно натискати «Замовити»
- чи добре читається текст зі смартфона
- чи легко заповнювати форму
- вірне повідомлення видає сайт при дії

1.3 Порівняльний аналіз месенджерів з платформою чат-ботів

Месенджер (розсилач повідомлень) - це програма для комунікації в інтернеті, яка встановлюється на ваш смартфон або інший пристрій. Простими словами, це додаток, який підтримує можливість обміну миттєвими повідомленнями.

У контексті дипломної роботи необхідно проаналізувати та обрати один із сучасних месенджерів з підтримкою платформи чат-ботів. Для подальшої роботи та використання треба зробити аналіз месенджерів щоб побачити плюси та мінуси: Facebook, Telegram та Viber. Ми використаємо наступні параметри для аналізу які показані в таблиці.

В табл. 1.1 представлена порівняльна інформація популярних месенджерів з підтримкою платформи чат-ботів.

Таблиця 1.1 - Порівняльна характеристика платформ для чат-ботів популярних месенджерів.

	Telegram	**Facebook**	**Viber**
Форма використання	Безкоштовно	Умовно безкоштовно	Умовно безкоштовно
Доступи	Доданий до групи, за підпискою, вбудований в діалог	Доданий до групи, за підпискою, вбудований в діалог	За підпискою, вбудований в діалог
Інтерфейси Користувача	Текстовий, кнопковий, голосовий		
Протоколи передачі даних	HTTPS		
Формати передачі даних	JSON, URL query, multipart/formdata	JSON, multipart/form - data	JSON, multipart/form-data
Функціональні можливості	Листування, опитування, передача файлів, магазин покупок, ігри		Листування, передача файлів, магазин покупок, ігри

Відповідно до даних за таблиці робимо висновок, що для втілення поставлених завдань найкращій месенджер для використання - Telegram, оскільки, в порівнянні з іншими месенджерами, він має наступні переваги:

- Безкоштовне використання.
- Великий список функціональних можливостей.

1.4 Telegram Bot API

Для керування Telegram-ботом існує спеціальне API (протокол для взаємодії комп'ютерних програм), яке вимагає використання HTTPS-протоколу. За допомогою Telegram-боту можливо:

1. Отримання кастомізованих сповіщень та новин.

2. Взаємодія зі сторонніми сервісами.

3. Отримання платежів від користувачів.

4. Можливість створення функціоналу перекладання з різних мов, форматування тексту, нагадування тощо.

5. Розробка ігор, що базуються на технології HTML5.

Telegram-бот з точки зору месенджеру – це спеціальний аккаунт, з яким користувачі можуть взаємодіяти двома способами:

1. Безпосередньо вводити команди в приватному чаті з ботом, або добавити Telegram-бота в публічну групу/чат.

2. Відкрити будь-який чат та вписати “@ім’яБота” в поле вводу повідомлень.

При створенні бота треба зазначити, що:

1. Ім’я бота повинно мати приставку “bot” в кінці.

2. Почати взаємодію з ботом може тільки користувач, це зробленно для того щоб уникнути спаму та набежаної розсилки.

3. Telegram-бот має обмежене сховище повідомлень [4].

Кожен бот потребує авторизації з боку серверів Telegram. При створенні бота генерується спеціальний унікальний токен(призначиний для авторизації користувача, безпечного віддаленого доступу до інформаційних ресурсів авторизації).

Приклад <token>=123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11, який передається частиною URL при кожному запиті. Усі запити до Telegram Bot API повинні відбуватися з використанням HTTPS протоколу на наступну адресу: https://api.telegram.org/bot<token>/METHOD_NAME.

Підтримуються GET та POST HTTP запити.

- GET - Запитує дані із вказаного ресурсу
- POST - Надсилання даних для обробки у вказаний ресурс

Існують наступні методи передачі параметрів у запиті:

1. URL query string.

2. Application/x-www-form-urlencoded.

3. Application/json.

4. Multipart/form-data.

URL query string – частина URL, яка містить пари “ключ=значення”, де ключ – певний параметр.

Структура query string: URL?param1=1&m2=2&m3=3, де:

URL – унікальний ідентифікатор ресурсу,

? – символ, що відділяє URL та безпосередньо query string,

& –символ, що розділяє декілька параметрів.

Application/x-www-form-urlencoded – тип тіла HTTP запиту, при якому дані запиту передаються у форматі “ключ=значення”. Тип зазначається у заголовка Content-Type запиту. Зазначені вище методи можуть використовувати спеціальне кодування символів – URL encoding або ж інакше percent encoding. Це кодування використовуються для того, щоб вирішити проблему використання заборонених символів (приклад для URL – пробіл). Механізм кодування – заміна 8-ми бітових заборонених символів іншими, дозволеними символами [4].

':'	'/'	'?'	'#'	'['	']'	'@'	'!'	'$'	'&'	"'"	'('	')'	'*'	'+'	','	';'	'='	'%'	' '
%3A	%2F	%3F	%23	%5B	%5D	%40	%21	%24	%26	%27	%28	%29	%2A	%2B	%2C	%3B	%3D	%25	%20 or +

Рисунок 1.2 - Кодування символів.

1.5 Висновки до першого розділу

У даному розділі розкрито такі аспекти користувацьких інтерфейсів:

- Цикл роботи чат боту
- Класифікація чат-ботів
- Порівняльний аналіз месенджерів з платформою чат-ботів
- Документація Telegram боту
- Telegram Bot API

Переглянувши всі наявні месенджери на платформі чат-ботів визначено, що для виконання поставлених цілей найкращим буде месенджер Telegram, оскільки він має документоване API з широким набором функціоналу, має велике ком'юніті, надійно приховує особисту інформацію користувача та користується популярністю у воєнний час.

РОЗДІЛ 2 АНАЛІЗ ТА ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ КЕРУВАННЯ ЧАТ-БОТОМ

2.1 Опис інтегрованого середовища розробки

IDE (Інтегроване середовище розробки) — це програмне забезпечення для створення програм, яке поєднує звичайні інструменти розробника в єдиний графічний інтерфейс користувача (GUI). IDE зазвичай складається з:

- **Редактор вихідного коду** : текстовий редактор, який може допомогти в написанні програмного коду з такими функціями, як підсвічування синтаксису за допомогою візуальних підказок, забезпечення автоматичного заповнення для певної мови та перевірка помилок під час написання коду.
- **Локальна автоматизація збірки** : утиліти, які автоматизують прості, повторювані завдання в рамках створення локальної збірки програмного забезпечення для використання розробником, як-от компіляція вихідного коду комп'ютера в двійковий код, упаковка двійкового коду та запуск автоматизованих тестів.
- **Дебаггер**: програма для тестування інших програм, яка може графічно відображати розташування помилки в оригінальному коді[5].

IDE дозволяє розробникам швидко починати програмувати нові програми, оскільки не потрібно вручну конфігурувати та інтегрувати декілька утиліт як частину процесу налаштування. Розробникам також не потрібно витрачати години окремо, вивчаючи, як використовувати різні інструменти, коли кожна утиліта представлена на одному робочому місці. Це може бути особливо корисно для залучення нових розробників, які можуть покладатися на IDE, щоб освоювати стандартні інструменти та робочі процеси команди. Насправді більшість функцій IDE призначені для економії часу, як-от інтелектуальне завершення коду та автоматичне генерування коду, що усуває необхідність вводити повні послідовності символів.

Інші поширені функції IDE покликані допомогти розробникам організувати свій робочий процес і вирішувати проблеми. IDE аналізують

код так, як він написаний, тому помилки, спричинені людською помилкою, визначаються в режимі реального часу. Оскільки утиліти представлені одним графічним інтерфейсом користувача, розробники можуть виконувати дії без перемикання між додатками. Підсвічування синтаксису також поширене в більшості IDE, яке використовує візуальні підказки для розрізнення граматики в текстовому редакторі. Деякі IDE додатково включають браузери класів і об'єктів, а також діаграми ієрархії класів для певних мов[5].

Можна розробляти додатки без IDE або для кожного розробника по суті створювати власну IDE, вручну інтегруючи різні утиліти з легким текстовим редактором, таким як Vim або Emacs. Для деяких розробників перевагою цього підходу є ультраналаштування та контроль, які він пропонує. Однак у контексті підприємства економія часу, стандартизація середовища та можливості автоматизації сучасних IDE зазвичай переважають інші міркування.

Сьогодні більшість корпоративних команд розробників вибирають попередньо налаштовану IDE, яка найкраще підходить для їх конкретного випадку використання, тому питання полягає не в тому, чи використовувати IDE, а в тому, яку IDE вибрати[5].

2.2 Порівняння та вибір IDE

Серед усіх можливих інтегрованих систем розробки буде опис та розгляд двух самих популярних та функціанальніх IDE VS Code та WebStorm. Розглянемо плюси і мінуси VS Code і WebStorm

VS Code безкоштовний - текстовий редактор з відкритим кодом, створений для сучасних веб та хмарних додатків компанією Microsoft який працює на будь-якій платформі – Linux, OSX та Windows. Великі переваги: VS Code є настроюваним, багатомовним, швидким і легким, поєднуючи сучасне редагування та налагодження з підтримкою коду та навігацією.

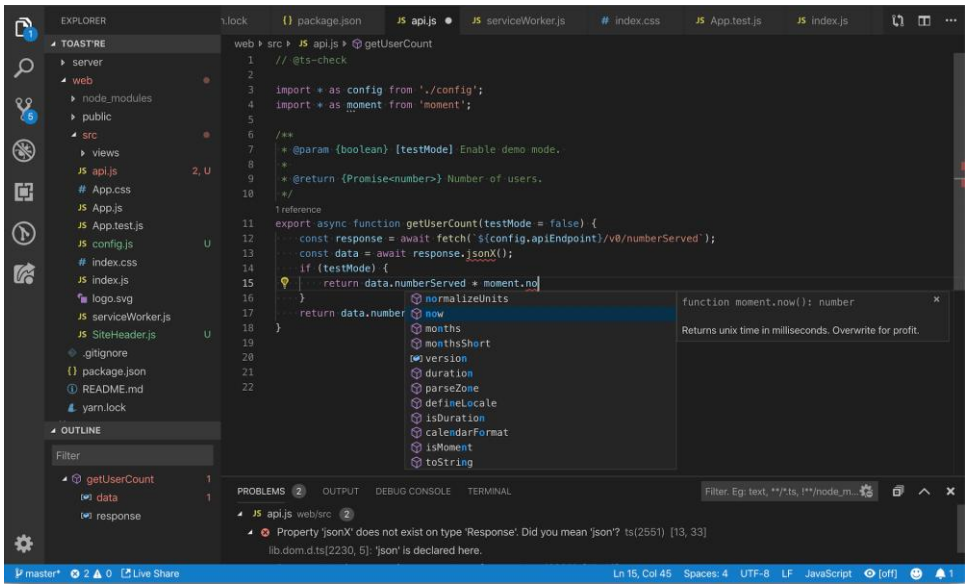


Рисунок 2.1 - Інтерфейс VS Code.

WebStorm — це інтуїтивно зрозуміла, платна й ліцензована IDE JavaScript, створена JetBrains для розробки інтерфейсу та серверного JavaScript. Основні моменти WebStorm: це дуже повна та інтелектуальна IDE для легкого запуску, налагодження та модульного тестування додатків Node.js, а також має хороший рефакторинг коду та автоматичний імпорт.

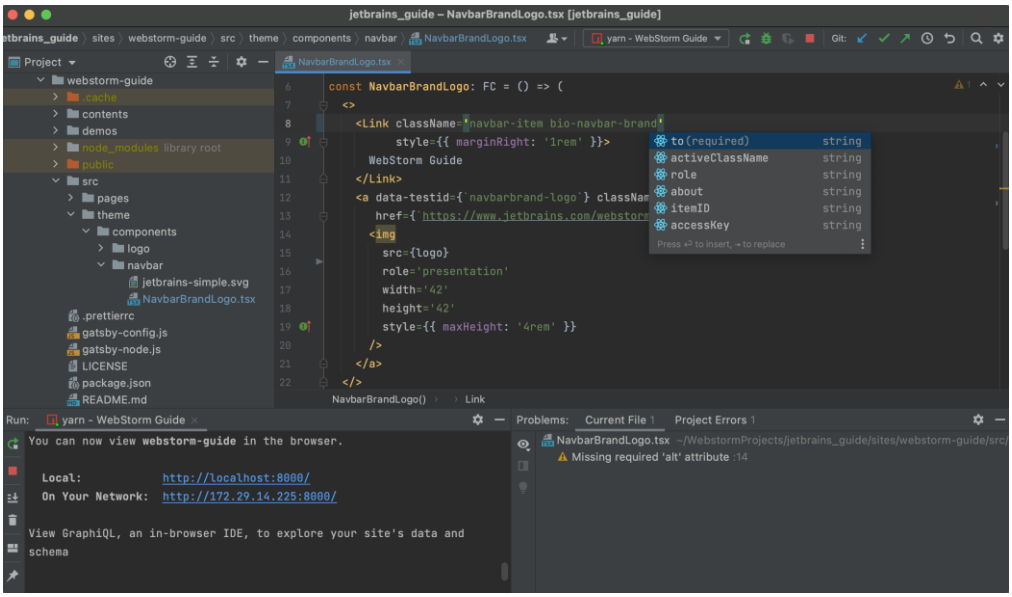


Рисунок 2.2 - Інтерфейс WebStorm

Ціноутворення може перешкодити деяким розробникам взагалі використовувати WebStorm, оскільки він вимагає щомісячної або річної передплатної плати після 30-денного пробного періоду [5].

Будучи у відкритому доступі, VS Code не потребує витрати коштів та придбання підписки.

- WebStorm має більшість функцій вже вбудованих, тоді як із VS Code вам, ймовірно, доведеться вручну встановити деякі розширення, щоб тестувати программи, отримати розширені функції, специфічні для мови тощо.
- WebStorm надає дивовижні інструменти злиття Git з чудовою візуалізацією змін Git. Це значно полегшує об'єднання складних змін і вирішення конфліктів.
- WebStorm автоматично відстежує всі зміни файлів із коробки, дозволяючи перевіряти історію файлів, каталоги та відкати.
- І VS Code, і WebStorm швидко форматують файли під час збереження, налаштовуються та добре працюють із автозаповненням, налагодженням та документацією.

Продуктивність є явною перевагою VS Code. Він відчуває себе швидко при загальному використанні, має швидкий час запуску та швидке редагування файлів. На основі JavaScript і HTML основою VS Code є Electron, тоді як WebStorm розроблено на Java. Поширеною скаргою є те, що WebStorm працює повільно.

Що стосується дизайну інтерфейсу користувача, VS Code дуже мінімалістичний, але користувачі можуть налаштувати його за допомогою готових до використання тем, щоб допомогти розробникам зробити його таким, яким вони хочуть. На відміну від цього, WebStorm має безліч функцій. Користувачі можуть налаштувати, щоб приховати функції, кнопки, меню та ярлики, але може знадобитися час, щоб зрозуміти та адаптуватися до всіх параметрів [5].

VS Code також дуже легкий, що робить його придатним для віддаленої розробки, швидкого створення прототипів і великих бізнес-додатків. Розширення VS Code додають підтримку покращень GIT, вбудованих помилок і кращих коментарів.

Виходячи з вищеперелічених плюсів і мінусів, для дипломного проекту найкращим буде VS Code, тому що у нього є весь необхідний функціонал, швидка робота і найголовніший фактор - безкоштовність.

2.3 Система контролю версій GIT

Git — це інструмент DevOps, який використовується для керування вихідним кодом. Це безкоштовна система контролю версій з відкритим вихідним кодом, яка використовується для ефективної обробки малих і дуже великих проектів. Git використовується для відстеження змін у вихідному коді, що дозволяє кільком розробникам працювати разом над нелінійною розробкою [6].

До появи системи контролю версій розробникам приходилось витрачати багату часу на підтримку актуальності коду:

- Раніше розробники надсилали свої коди на центральний сервер, не маючи власних копій
- Будь-які зміни, внесені до вихідного коду, були невідомі іншим розробникам
- Між розробниками не було жодного спілкування

Після того як була створена система контролю версій для розробників відкрились нові можливості:

- Кожен розробник має повну копію коду в своїх локальних системах
- Будь-які зміни, внесені до вихідного коду, можуть відстежуватися іншими
- Між розробниками відбувається регулярне спілкування

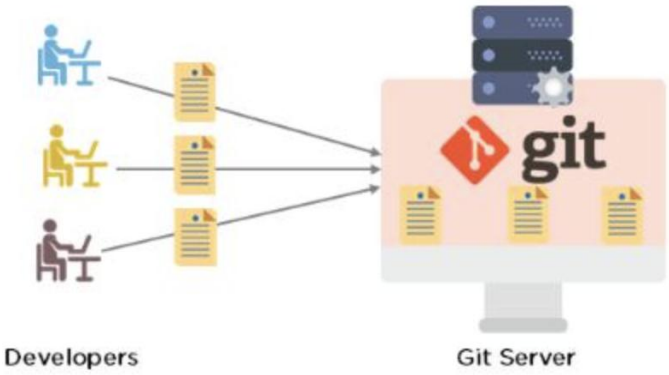


Рисунок 2.3 - Умовна робота GIT.

Git — це система контролю версій, яка використовується для відстеження змін у файлах комп'ютера. Зазвичай він використовується для управління вихідним кодом при розробці програмного забезпечення.

- Git використовується для відстеження змін у вихідному коді
- Для керування вихідним кодом використовується розподілений інструмент контролю версій
- Це дозволяє кільком розробникам працювати разом
- Він підтримує нелінійний розвиток через тисячі паралельних гілок

На рис 2.4 продемонстровано схему, як гілки GIT використовуються. Скажімо, вам потрібно попрацювати над новою функцією для веб-сайту. Ви створюєте нову гілку і починаєте працювати. Ви ще не завершили роботу над новою функцією, але отримуєте запит на внесення швидких змін, які мають опублікувати на сайті сьогодні. Ви повертаєтеся до головної гілки, вносите зміни та запускаєте її в реальному часі. Потім ви можете повернутися до нової гілки функцій і закінчити роботу. Коли ви закінчите, ви об'єднуєте нову гілку функцій з головною, і нова функція, і швидкі зміни зберігаються!

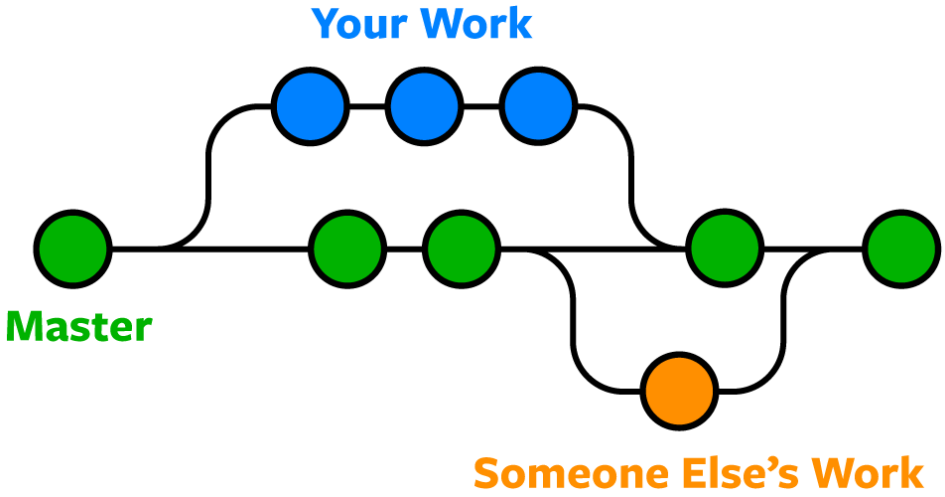


Рисунок 2.4 - Використання гілок GIT.

2.4 Огляд мови програмування JavaScript

JavaScript — це текстова мова програмування, яка використовується як на стороні клієнта, так і на стороні сервера, що дозволяє зробити веб-сторінки інтерактивними. Якщо HTML і CSS є мовами, які надають структуру та стиль веб-сторінкам, JavaScript надає веб-сторінкам

інтерактивні елементи, які залучають користувача. Поширені приклади JavaScript, які ви можете використовувати щодня, включають вікно пошуку на Amazon, відео з підсумком новин, вбудоване в The New York Times, або оновлення вашої стрічки Twitter.

Включення JavaScript покращує роботу веб-сторінки, перетворюючи її зі статичної сторінки в інтерактивну. Нагадаю, JavaScript додає поведінку веб-сторінкам [7].

JavaScript в основному використовується для веб-додатків і веб-браузерів. Але JavaScript також використовується за межами Інтернету в програмному забезпеченні, серверах та вбудованих апаратних елементах керування. Ось деякі основні речі, для яких використовується JavaScript:

- Додавання інтерактивної поведінки на веб-сторінки;
- Створення веб- та мобільних додатків;
- Створення веб-серверів і розробка серверних додатків;
- Розробка гри;

Окрім необмежених можливостей, для веб-розробників є багато причин використовувати JavaScript замість інших мов програмування:

- JavaScript є єдиною мовою програмування, яка є рідною для веб-браузера;
- JavaScript є найпопулярнішою мовою;
- Для початку існує низький поріг;
- Це цікава мова для вивчення;

2.5 HTTP запити

Інтернет складається з багатьох ресурсів, розміщених на різних серверах. Щоб отримати доступ до вмісту в Інтернеті, браузер повинен запитати у цих серверів потрібні ресурси. Цей протокол запитів і відповідей дає змогу переглядати цю сторінку у своєму браузері.

Передача ресурсів відбувається за допомогою протоколу керування передачею, або TCP. TCP використовується для керування багатьма типами

інтернет-з'єднань, у яких один комп'ютер або пристрій хоче щось надіслати іншому. HTTP (Hypertext Transfer Protocol) — це мова команд, якої повинні дотримуватися пристрої з обох сторін з’єднання, щоб спілкуватися.

Запит HTTP виконується клієнтом до іменованого хоста, який розташований на сервері. Метою запиту є доступ до ресурсу на сервері.

Щоб зробити запит, клієнт використовує компоненти URL-адреси (Uniform Resource Locator), яка містить інформацію, необхідну для доступу до ресурсу. Компоненти URL-адреси пояснюють URL-адреси [8].

Правильно складений HTTP-запит містить такі елементи:

- Рядок запиту.
- Серія заголовків HTTP або полів заголовків.
- Тіло повідомлення, якщо потрібно.

За кожним HTTP-заголовком йде передача рядка повернення каретки (CRLF). Після останнього із заголовків HTTP використовується додатковий CRLF (щоб надати порожній рядок), а потім починається будь-яке тіло повідомлення.

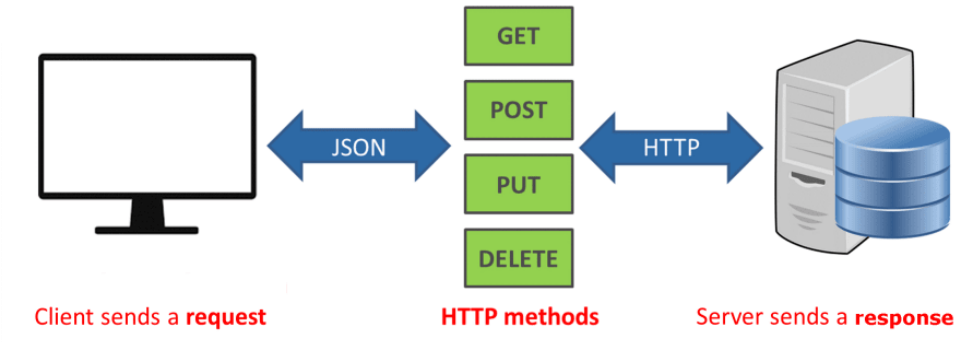


Рисунок 2.5 - HTTP запит.

Метод запиту HTTP — це спосіб вказати бажану дію, яку має виконати ресурс. Хоча деякі з них є іменниками, методи HTTP називаються дієсловами. У той же час вони чутливі до регістру та завжди пишуться у верхньому регістрі. Існують різні методи HTTP-запиту, кожен з яких має своє призначення:

GET

Мабуть, найпопулярніший метод запиту HTTP, GET, використовується для отримання даних з певного сервера.

HEAD

Подібно до методу GET, цей метод не матиме повідомлення в тілі. Зазвичай запит HEAD використовується під час спроби оцінити доступність кінцевої точки API.

POST

POST — ще один популярний метод запиту HTTP. Ви використовуєте POST, коли хочете надіслати інформацію на сервер для створення або оновлення ресурсу, використовуючи інформацію, що зберігається в тілі HTTP-запиту.

PUT

Подібно до того, як працює метод POST, PUT оновить або створить ресурс. Однак різниця полягає в тому, що запити PUT є однаковим, а це означає, що результат залишиться незмінним незалежно від того, скільки разів ви викликаєте запит PUT.

DELETE

Метод DELETE дозволяє видалити певний ресурс на сервері.

PATCH

Подібно до запитів PUT і POST, PATCH оновить інформацію про сервер, але на відміну від двох, PATCH застосує часткові зміни. Ви хочете використовувати PATCH, коли змінюєте лише назву запису продукту.

TRACE

Запит TRACE викликає зворотний тест уздовж шляху цільового ресурсу. Зазвичай він використовується для виконання налагоджувальних та діагностичних тестів на API.

CONNECT

Хоча він менш відомий, ніж інші методи запиту, CONNECT використовується для створення підключення до сервера через HTTP. Він створює тунельне підключення до сервера, зазначеного параметром URL [8].

Недоліком HTTP-запитів було те, що вони не забезпечували безпечне з’єднання. між клієнтами і серверами.

HTTPS — це розширення класичного запиту HTTP, яке захищає протоколи запитів за допомогою двонаправленого шифрування за допомогою цифрових сертифікатів на стороні сервера, які називаються SSL.
Ці сертифікати SSL видаються центром сертифікації (ЦС), який є надійною незалежною третьою стороною, яка автентифікує обидві сторони транзакції.

2.6 Висновки до другого розділу

Проаналізовано інтегральні середовища розробки, серед яких найбільш простим та доступним у використанні стало середовище VS Code тому що воно безкоштовне у користуванні, має безліч кастомних налаштувань та підтримує всі мови програмування.

Проаналізовано систему контролю версій, та виявлено переваги та необхідність у її використанні за для того щоб уникнути втрачання коду та для можливості завантаження бота на хостинг.
Досліджено функціонал, всі методи та роботу HTTP запитів для отримання та відправки інформації з/на сервер.

РОЗДІЛ 3 РОЗРОБКА TELEGRAM-БОТА

3.1 Створення та налагодження Telegram боту

Telegram - інтерактивна платформа, яка максимально взаємодіє з користувачем, навіть при створенні бота вона спрощує його написання та налаштування заьдопомогою автоматизації більшості процесів та документації. Для створення Telegram боту існує BotFather, в якому присутня велика кількість налаштувань бота та великий функціонал.

Для початку роботи з власним Telegram ботом, потрібно отримати Token - спеціальний набір символів для доступу до HTTP API Telegram Bot. При переході в BotFather з’явиться список команд (установки додаткових параметрів, таких як іконка чат-бота, вітальне повідомлення, опис чат-бота, а так само видалення наявних чатботи та ін.) [9]:

Створення бота

- /newbot - створити нового бота
- /mybots - список існуючих ботів

Редагування ботів

- /setname - змінити ім'я бота
- /setdescription - змінити опис бота
- /setabouttext - змінити інформацію про бота
- /setuserpic - змінити фотографію профілю бота
- /setcommands - змінити список команд
- /deletebot - видалити бота

Налаштування бота

- /token - генерувати маркер авторизації
- /revoke - анулювати маркер доступу бота
- /setinline - перемикати вбудований режим
- /setinlinegeo - перемикати вбудовані запити розташування
- /setinlinefeedback - змінити налаштування вбудованого зворотного зв'язку

- /setjoingroups - чи можна додати вашого бота до груп?
- /setprivacy - перемикати режим конфіденційності в групах

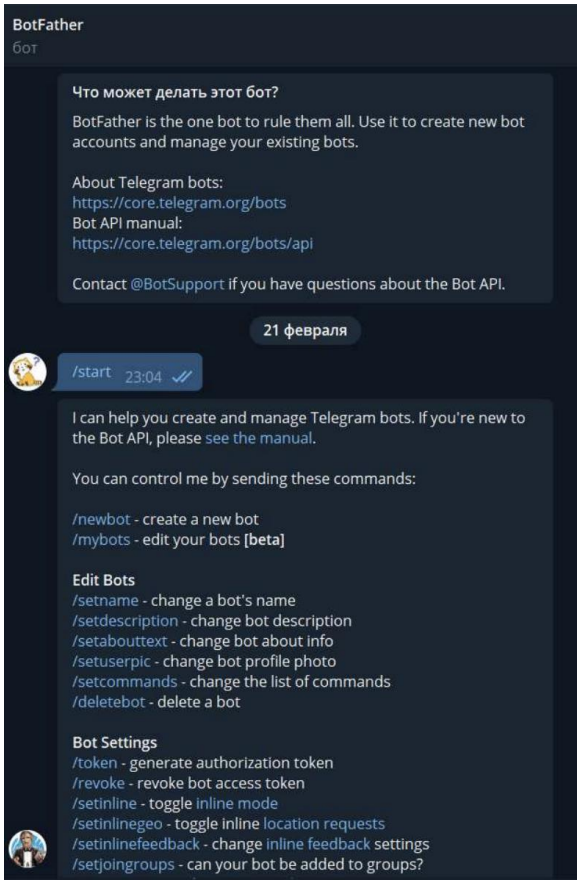


Рисунок 3.1 - Початок роботи з BotFather.

Створення боту починається з команди «/newbot». Після цього потрібно обрати назву боту та юзернейм з приставкою «Bot» або «_bot», за допомогою якої можна буде знайти бота, в нашому випадку це “Актуальна інформація” та “WarRegionInfo_bot”[9].

Після виконання всих дій BotFather видає Token і URL-адресу для доступу до чат-боту.

3.2 Налаштування проекту

Для внесення змін та роботи з ботом, потрібно встановити бібліотеку node-telegram-bot-api для того щоб мати доступ до bot-api, всі команди використовуються в терміналі:

npm install --save node-telegram-bot-api

Після встановки бібліотеки потрібно в репозиторії проекту створити файл index.js (або будь-яке інше ім'я) і всередині файлу потрібно підключити бібліотеку node-telegram-bot-api[10]:

```
const TelegramBot = require('node-telegram-bot-api');
```

Наступним кроком буде визначення токену, якого ми отримали з BotFather:

```
const token = 'YOUR_TELEGRAM_BOT_TOKEN';
```

Далі нам потрібно ініціалізувати нашого бота:

```
const bot = new TelegramBot(token, {polling: true});
```

Для отримання повідомлень якиі надсилає користувач потрібно використати наступний код:

```
bot.on('message', (msg) => {
  (будь-який код)
});
```

Для наглядного ознайомлення створимо стандартне привітання:

```
const TelegramBot = require('node-telegram-bot-api');
const token = 'YOUR_TELEGRAM_BOT_TOKEN';
const bot = new TelegramBot(token, {polling: true});
bot.on('message', (msg) => {
 (будь-який код)
});
bot.on('message', (msg) => {
var Hi = "Hi";
If (msg.text.toString().toLowerCase().indexOf(Hi) === 0) {
bot.sendMessage(msg.chat.id,"Hey there);
}});
```

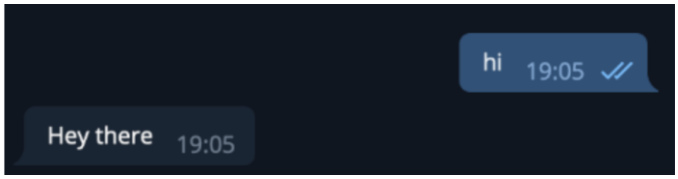


Рисунок 3.2 - Результат виконання програми.

3.3 Реалізація клієнтської частини бота

Користувачі Telegram можуть взаємодіяти з кількома чат-ботам способи: використовуючи команди з параметрами або вбудовані клавіатури. Для зручності користувачів було вирішено зробити інтерфейс із вбудованою клавіатури. Для реалізації поставленої мети необхідно реалізувати кілька меню зі своїми параметрами. На малюнку 3.3 зображено початок роботи з ботом, після того як йому тільки дали опис та встановили Зображення профілю.

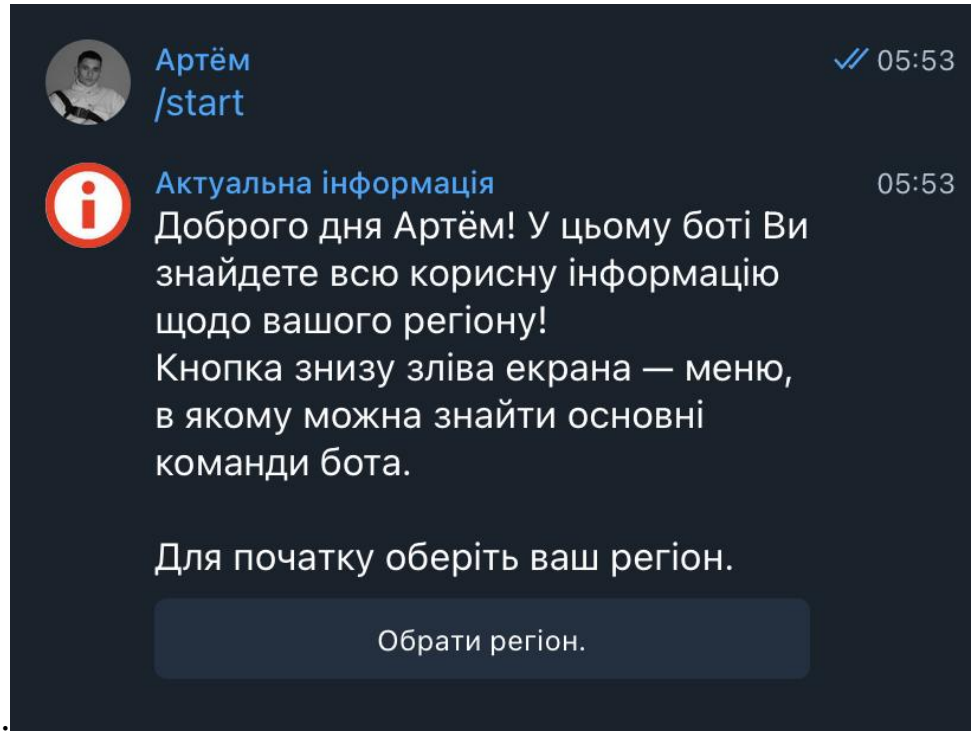


Рисунок 3.3 - Результат виконання програми.

Початок роботи з чат-ботом розпочинається з команди /start. Ця команда активує команду і бот починає діалог із користувачем. Код початку роботи представлений малюнку 3.4.

```
try {
    if (text === '/start') {
        return bot.sendMessage(chatId, `Доброго дня ${msg.from.first_name}!
        У цьому боті Ви знайдете всю корисну інформацію щодо вашого регіону!
        \nКнопка знизу зліва екрана – меню, в якому можна знайти основні команди бота.
        \n\nДля початку оберіть ваш регіон.`, againOptions);
    }
```

Рисунок 3.4 - Код привітання.

Далі слідує реалізація меню, через яке відбуватиметься взаємодія із створюваним чат-ботом, весь список команд, який складається з 7-ми позицій, можна знайти з лівого нижнього кутка, приклад на малюнку 3.5.

Після натискання кнопки “Обрати регіон” з’явиться список всіх регіонів на вибір малюнок 3.6.

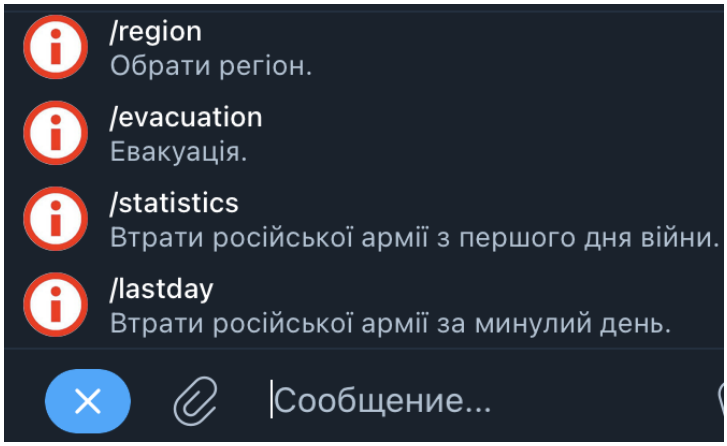


Рисунок 3.5 - Меню команд.

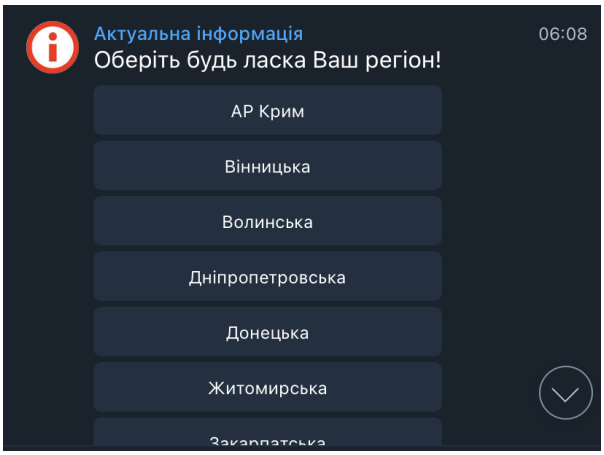


Рисунок 3.6 - Меню вибору регіону.

Після створення основних команд починається реалізація меню вибору регіону. У цьому меню і відбувається вибір параметрів, завдяки яким, чат бот буде надсилати інформацію по регіонам відносно вибору користувача малюнок 3.7.

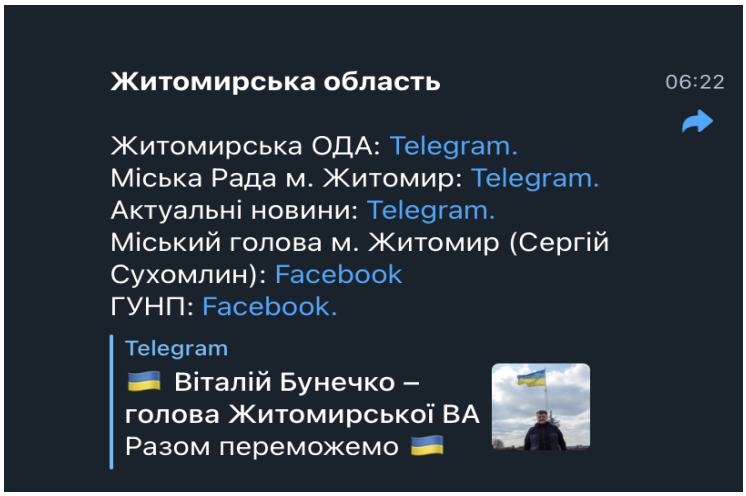


Рисунок 3.7 - Результат по обраному регіону.

Важлива частина бота це відображення корисних інтернет ресурсів для користувача. Саме в цьому меню користувач дізнається більшу частину необхідної інформації, що допоможе йому швидко знайти ресурс який йому потрібен. Для відображення всієї інформації було проаналізовано велику кількість інтернет ресурсів та сформовано інтерфейс, в якому записана вся інформація по регіонам в файлі filter.js (рис. 3.8). Після вибору регіону, бот видає список корисних посилань по заданому регіону. Зі списком можна взаємодіяти переходячи за посиланням. На рисунку 3.9 можна побачити код функції яка містить основну логіку відображення інформації, де за допомогою методів .filter() та .map() відбувається фільтрація в масиві об’єктів та обирається потрібне значення в потрібному об’єкті.

Рисунок 3.8 - Інформація по всім регіонам.

```
bot.on('callback_query', async msg => {
    const data = msg.data;
    const chatId = msg.message.chat.id;
    let filteredInfo = cityList.filter(region => region.city === data).map(region => region.info) + '';

    if (data === '/again') {
      return chooseRegion(chatId)
    } else if(data) {
      await bot.sendMessage(chatId, filteredInfo, {parse_mode : "HTML"}, againOptions);
    } else {
      await bot.sendMessage(chatId, `Обрати інший регіон`, againOptions);
    }
})
```

Рисунок 3.9 - Блок коду відповідальний за відображення інформації по регіону.

Не менш важлива секція боту - відображення інформації щодо евакуації, де зібрана акутальна інформація та ресурси, та посилання на карту бойових дій, оскільки через великий потік інформації важко знайти потрібні посилання та ресурси. На малюнках 3.10 та 3.11 показані реалізація та відображення інформації щодо евакуаційної секції яку можна отримати за допомогою команди “/evacuation”.

```
const EVACUATION_INFO = `В цьому розділі ми публікується інформація від місцевих та міжнародних волонтерських ініціатив, які хочуть допомогти людям евакуюватися з н
\n\n<a href=\"https://viyna.net/61c28261da6041b884fb305c8cc2234e\">Сервіс з підбіркою ботів для евакуації з різних міст України</a> \n \nСотня юристів запустили Бот
\n\nЯк вберегти себе від можливості ризику торгівлі людьми закордоном. <a href=\"https://docs.google.com/document/d/1wE024cFH5CUxcgfWlhLS7EvAgPp7TAIbqXv71Eqtgvw/edi
\n\n👉“Будь у безпеці! Евакуація” <a href=\"https://bit.ly/3pJZNE0\">Корисний канал про евакуацію.</a>
\n\n👉<a href=\"https://bit.ly/3MwZiXu\">UAhelpinfo.</a>
\nРесурс зі структурованою інформацією про допомогу українцям з тимчасовим житлом, транспортом, медичними та юридичними питаннями та іншим в Україні та за кордоном.
\n\n👉<a href=\"https://bit.ly/3MrC0lM\">Ресурс “Допомагай”</a>
\nБезкоштовне тимчасове житло у безпечних містах України або за кордоном.
\n\n📞 Гаряча лінія 0 (800) 332 238`;

module.exports = EVACUATION_INFO;
```

Рисунок 3.10 - Реалізація відображення даних.

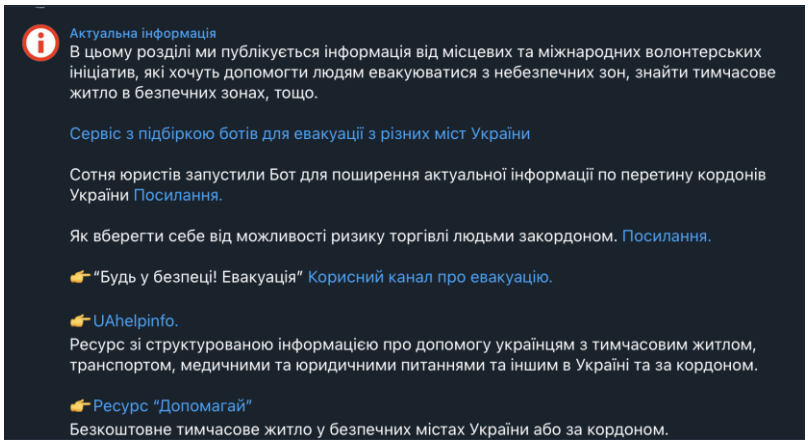


Рисунок 3.11 - Відображення інформації по евакуації.

Для реалізації динамічного підтягування інформації з бази даних було додано команду “/statistics” яка відображає кількість зареєстрованих випадків знищеної техніки з першого дня війни, данні оновляються кожного дня о 8-й вечора, до цього часу можна передивитись статистику за попередній день.

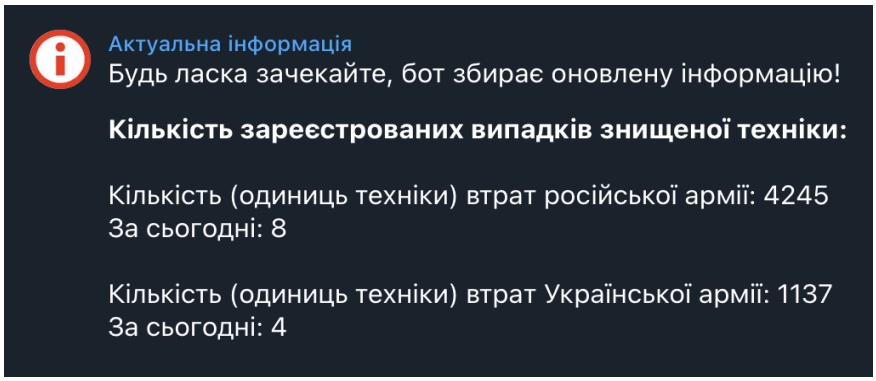


Рисунок 3.12 - Відображення статистики.

```
const URL_TO_FETCH = "https://script.google.com/macros/s/AKfycbyC0QPlbr0K8ngSwOskGMv4gbP6ydhi6qwLSy4MgfuNAmb9CIWFYAO_8ccm68kZ0YZE5A/exec";

async function getStatistics() {
  const response = await fetch(URL_TO_FETCH)
  const data = await response.json()
  let currentStatistics = data.result.find(el => el[0] == getDateFormat())
  const formatData = `<b>Кількість зареєстрованих випадків знищеної техніки:</b>\n\nКількість (одиниць техніки) втрат російської армії: ${cur
  return formatData
}
```

Рисунок 3.13 - Блок коду відображення статистики.

3.4 Розміщення чат-бота на віддаленому сервері Heroku

Для подальшого працювання та використання бота потрібно завантажити наш код на GitHub. Таким чином, ми зможемо редагувати наш код з будь-якого місця, а також миттєво розгорнути внесені зміни в хмарі.

Ддя цбого потрібно створити новий репозиторій на GitHub, натиснувши Новий репозиторій.[11]

GitHub створить репозиторій і дасть кілька команд, які використовуються для клонування своєї локальної папки за допомогою свого репозитарію GitHub.

У командному рядку треба виконати наведені нижче команди в такій послідовності.

1. Ініціалізування репозиторію Git на кореневому рівні:

   git init

2. Додавання всіх файлів до локального Git (проміжна версія).

git add .

3. Внесення зміни до свого локального Git:

git commit -m “first commit”

4. Посилання на свій репозиторій GitHub.

git remote add origin https://github.com/rramname/WarUkraineStatistics.git

5. Відправка змін на вітдалений репозиторій:

git push — set -upstream origin master

Тепер, після відкриття та оновлення репозиторію GitHub можна побачити побачити код. Для початку користування Heroku, потрібно зарєеструватися. Після реєстрації потрібно створити новий додаток. Цей крок перенесе нас на інформаційну панель додатка. У вкладці «Розгортання» в розділі «Метод розгортання» треба вибрати GitHub як метод.Після відображення «Підключитися до GitHub» треба надати доступ до GitHub репозиторію. Після вдалого підлкючення репозиторію GitHub з’явиться розділ «Розгортання», де треба вибрати автоматичне розгортання[11].

Тепер треба вказати Heroku, що наша програма є програмою NodeJs. Для цього потрібно натиснути «Додати buildpack» вибрати nodejs з параметрів і натиснути Зберегти зміни. Після цбого на вкладці «Розгортання» треба натиснути «Розгорнути філію» внизу. Пісял цього у вкладці активність можна побачиит, що сервер запущено[11].

3.5 Висновки до третього розділу

У третьому розділі розглянуто та обрано технології для розробки системи керування чат-ботом. Виявлено, що для створення системи керування чат-ботом потрібно мати великі знання в області розробки телеграм ботів, а важливічсть вибору технологій для розробки безпосередньо впливатиме на якість та легкість розробки. Виконано великий об’єм роботи, проаналізовано велику кількість ресурсів для знаходження достовірної

інформації, розроблено інтерфейс керування чат-ботом, інтегровано відображення статистики втрат техніки окупанта..

ВИСНОВОКИ

В даний час популярність чат-ботів продовжує зростати. Даних помічників використовують у різних цілях, від простого спілкування, до скоєння замовлень. Також варто зазначити, що завдяки месенджерам чат-боти отримали розвиток. З'явилися різні види для різних цілей.

В ході виконання дипломної роботи показано, що основне завдання чат-ботів - вести діалог з користувачем та відповідати на його запити, спираючись на базу знань. Не зважаючи на свої завдання чат-боти створюються для ведення діалогу з користувачем, імітуючи цим співрозмовника чи спеціаліста.

Зазначено, що чат-боти можуть виконувати різні завдання: консультувати клієнтів, допомагати знайти потрібний файл, повідомляти новини, замовляти їжу, бути співрозмовниками і все в рамках однієї програми, на платформі якої реалізований бот. Це зменшує необхідність встановлювати безліч додатків, через те, що багато необхідних користувачеві запитів можуть виконуватися в рамках однієї програми (месенджера).

Через переваги та користі чат-ботів компанії та фірми намагаються перекладати спілкування з клієнтами у формат надсилання повідомлень.

Продемонстровано, що багато ботів виконують таке завдання, як імітація віртуального співрозмовника. Боти не видають відповіді на складному для розуміння користувача мовою. Вони видають прості та зрозумілі для користувача відповіді на запит користувача. Користувач же у відповідь ставить нове запитання або припиняє роботу з чат-ботом. В рамках дипломної роботи було виконано поставлені завдання.

Під час виконання роботи визначено переваги та недоліки яких, та виявлено, що Telegram має багато різних зручних переваг для розробки бота-помічника. На основі цього виявлені вимоги для розробки чат-боту для інформаційного ресурсу під час війни. Обрані технології та середовище для розробки чат-бота.

Таким чином, результатом дипломної роботи є розроблений бот-помічник, який допомагає користувачеві швидко знайти корисну інформацію по регіонам. Чат-боти для інформаційного поля мають ряд переваг одні з основних: можливість отримання від клієнтів запиту, знаходження інформації, велика функціональність та доступність.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bots: information for developers. URL: https://tlgrm.ru/docs/bots (дата звернення: 18.05.2022).

2. Що таке backend та frontend. URL: https://otus.ru/nest/post/943/(дата звернення: 18.05.2022).

3. Чат-боты: виды, классификация, предназначение. URL: https://grandsoft.tech/ru/blog/CHat-boty-vidy-klassifikaciya-prednaznachenie/ (дата звернення: 18.05.2022).

4. Bot API Reference. URL: https://tlgrm.ru/docs/bots/api(дата звернення: 18.05.2022).

5. What is an IDE. URL https://www.redhat.com/en/topics/middleware/what-is-ide (дата звернення: 22.05.2022).

6. What is Git: Features, Command and Workflow in Git. URL: https://www.simplilearn.com/tutorials/git-tutorial/what-is-git (дата звернення: 25.05.2022).

7. What is JavaScript used for. URL: https://www.hackreactor.com/blog/what-is-javascript-used-for (дата звернення: 25.05.2022).

8. HTTP Requests. URL: https://sematext.com/glossary/http-requests/ (дата звернення: 27.05.2022).

9. Підключення Telegram bot. URL: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-channel-connect-telegram?view=azure-bot-service-4.0 (дата звернення: 01.06.2022).

10. Старт Telegram bot. URL: https://lavalite.org/blog/nodejs-telegram-bot-api (дата звернення: 01.06.2022).

11. Налаштування Heroku. URL: https://www.freecodecamp.org/news/how-to-deploy-a-nodejs-app-to-heroku-from-github-without-installing-heroku-on-your-machine-433bec770efe/ (дата звернення: 01.06.2022).

ДОДАТОК А

SUMMARY

SUMMARY

There is a fairly popular misconception that chatbots are a development of the 21st century. This is due to their current availability, which became possible after the advent of modern messaging platforms. But in fact, chatbots appeared long before the creation of instant messengers, social networks, mobile phones and even personal computers!

The famous English mathematician and cryptographer Alan Turing can be safely called the founder of chatbots. Alan Turing, back in 1941, began to study the theory of machine intelligence, which by 1947 had turned into "computer intelligence" - the prototype of modern artificial intelligence.

The first chatbot was developed by the MIT AI Lab by Joseph Weizenbaum in 1966 and named Eliza. It validated the keywords given as input and then triggered the output according to a specific set of rules. This output generation methodology is still used by a number of chatbots today.

The next was Parry, written by Stanford University psychiatrist Kenneth Colby in an attempt to mimic a person with paranoid schizophrenia. This was followed by ALICE, developed in 1995 by Richard Wallace. Although ALICE won the Loebner Prize three times, it did not pass the Turing test (the Turing test tests whether a machine is capable of thinking intelligently like humans).

After that, various virtual assistants were launched. Apple's Siri was the first to introduce conversation assistants. These concepts gained popularity, and soon after, Google launched its Assistant for Android. Following their example, Microsoft created Cortana. In the future, smart speakers were introduced that made possible voice conversation between humans and bots. Amazon Alexa and Google Home are another category of conversational interface.

Siri - smart and daring - this is the name of the famous Apple development, capable of answering questions and performing various tasks on the network. The program appeared 7 years ago and became a powerful catalyst for the development of chatbots with artificial intelligence.

Google Now came about two years after Siri. The developers created it for a mobile search engine. A bot from Google can answer questions, give good advice and process user requests.

In 2015, the famous Alexa from Amazon and Cortana from Microsoft appeared. And in principle, it has already become clear that we have entered the era of chatbots. Smart programs have learned to easily recognize human speech, learn, answer all kinds of questions, respond to voice commands, order goods at home, and much more.

Chatbots have been a great communication agent for businesses. For a long time, they have effectively replaced a customer service provider that works 24/7. However, as chatbots took on new roles and responsibilities, they became an alternative to mobile apps.

Chatbots can do just about anything, from simple conversations to complex transactions. They are now used by businesses, regardless of industry, to place orders, check order status, book flights, complete financial transactions, process complex customer/customer requests, improve marketing campaigns, and more. In short, chatbots are the new business applications.

Chatbots are classified according to several criteria: the algorithm of work, the format of interaction and the method of assignment.

Depending on the algorithm of work, there are bots limited and self-developing. Limited - respond to specific user requests according to a pre-developed script, have a limited number of responses. Self-developing - work on the basis of an artificial neural network, can understand the essence of the conversation and have realistic conversations with the buyer, learn over time and give more relevant answers to queries.

According to the format of user interaction, there are the following types of chatbots:

Button. Most often used in messengers. Communication with the client occurs through buttons with options for action. The bot reacts to them as to commands, and offers the user clarifying buttons or gives an answer to the question posed.

Text. The most functional type of virtual interlocutor. Communication with him is close to human, as the robot recognizes the request, analyzes the information and selects the most relevant answer for it from the prepared ones;

Embedded (inline). Appears inside the dialogue in the messenger after the call and offers options for action. The result can be shared with the interlocutor with whom the dialogue took place. Usually used to find suitable locations, order food and other similar services.Chatbots are also divided into communication and functional ones.

Communication bots live up to their name and provide communication between the company and customers. They can be very primitive. For example, answer frequently asked questions with template phrases, offer a callback, or redirect to a real manager. But it can also be marketing chatbots. By default, they perform the function of a two-way communication channel with clients. Provide them with information about services and special offers, answer counter questions on a pre-built architecture.

Functional chatbots are a replacement for full-fledged mobile applications. Already, most programs allow you to search, consult, book, buy, perform banking transactions in one window, as well as offer users interactive actions and personalized answers. Further, the range of operations can be expanded to infinity.

In wartime, it is very difficult to find reliable sources and the necessary information, because every day new groups and new news appear on social networks, as a result of which the necessary information is lost in the flow. The application increases the speed of customer interaction and service delivery through a chatbot.

To solve this problem, the market of competitors was analyzed to select the optimal platform for hosting our bot. The choice fell on the company Telegram with the same name messenger. This messenger won primarily for its openness to third-party developers and their comprehensive support, which is much less than competitors. Also, the statistics of new user growth played an important role. Unlike competitors who have already reached their peak and at best will support their audience and do not attract new ones. There may be different explanations for this. Some companies may not have had competition in their class and it was too easy for them to gain new users. And when new competitors came to the market, they had no experience and, above all, motivation to develop. Therefore, we saw a

prospect in a relatively young company, which is our winner for the role of the main platform.

Node.js was chosen as the main language of development, which is rapidly gaining momentum and has every chance to become very popular. Get statistics in real time from Google Spreadsheets, which are updated daily. The user can select the region he needs and find all the information and links to official publics and news. In the future, chatbots will replace almost all professions related to consulting, content managers and others.

As a result, we have an application that allows you to quickly find information on the region and get statistics on the losses of the Russian army and information on evacuations.

ДОДАТОК Б

Конфіругаційний файл package.json клієнтської частини

```
{
  "name": "warinfo-bot",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "dev": "nodemon index.js",
    "start": "node index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "node-fetch": "^2.6.5",
    "node-telegram-bot-api": "^0.57.0",
    "nodemon": "^2.0.16",
    "request": "^2.88.2"
  },
  "devDependencies": {},
  "description": ""
}
```

ДОДАТОК В

Основний код програми

```
const fetch = require('node-fetch')
const TelegramApi = require('node-telegram-bot-api')
const {cityOptions, againOptions} = require('./options')
const EVACUATION_INFO = require('./message')
const cityList = require('./filter')
const token = '5455523407:AAGwSNYgu5U7wncrr0nsJAUfrNiAo6NXfGQ'
const bot = new TelegramApi(token, {polling: true})
const URL_TO_FETCH =
"https://script.google.com/macros/s/AKfycbyCOQPlbr0K8ngSwOskGMv4gbP6ydhi6qwLSy4MgfuNAmb9CIWFYAO_8ccm68kZ0YZE5A/exec";

async function getStatistics() {
  const response = await fetch(URL_TO_FETCH)
  const data = await response.json()
  let currentStatistics = data.result.find(el => el[0] == getDateFormat())
  const formatData = `<b>Кількість зареєстрованих випадків знищеної техніки:</b>\n\nКількість (одиниць техніки) втрат російської армії: ${currentStatistics[1]}\nЗа сьогодні: ${currentStatistics[2]}\n\nКількість (одиниць техніки) втрат Української армії: ${currentStatistics[3]}\nЗа сьогодні: ${currentStatistics[4]}`
  return formatData
}

async function getLastDayStatistics() {
  const response = await fetch(URL_TO_FETCH)
  const data = await response.json()
  let currentStatistics = data.result.find(el => el[0] == lastDayInfo())
  const formatData = `<b>Кількість зареєстрованих випадків знищеної техніки:</b>\n\nКількість (одиниць техніки) втрат російської армії: ${currentStatistics[1]}\nЗа сьогодні: ${currentStatistics[2]}\n\nКількість (одиниць техніки) втрат Української армії: ${currentStatistics[3]}\nЗа сьогодні: ${currentStatistics[4]}`
```

```
  return formatData
}

function getDateFormat() {
  let D = new Date();
  return D.getFullYear() + '-' + ('0' + (D.getMonth() + 1)).slice(-2) + '-' + ('0' +
D.getDate()).slice(-2)
}

function lastDayInfo() {
  let D = new Date();
  return D.getFullYear() + '-' + ('0' + (D.getMonth() + 1)).slice(-2) + '-' + ('0' +
(D.getDate() - 1)).slice(-2)
}

const chooseRegion = async (chatId) => {
  await bot.sendMessage(chatId, 'Оберіть будь ласка Ваш регіон!', cityOptions);
}
const start = async () => {
  bot.setMyCommands([
    {command: '/start', description: 'Обрати регіон.'},
    {command: '/again', description: 'Обрати регіон знову.'},
    {command: '/map', description: 'Карта бойових дій.'},
    {command: '/region', description: 'Обрати регіон.'},
    {command: '/evacuation', description: 'Евакуація.'},
    {command: '/statistics', description: 'Втрати російської армії з першого дня
війни.'},
    {command: '/lastday', description: 'Втрати російської армії за минулий
день.'},
  ])
  bot.on('message', async msg => {
    const text = msg.text;
```

```
    const chatId = msg.chat.id;
    try {
        if (text === '/start') {
          return bot.sendMessage(chatId, `Доброго дня ${msg.from.first_name}! У цьому боті Ви знайдете всю корисну інформацію щодо вашого регіону!\nКнопка знизу зліва екрана — меню, в якому можна знайти основні команди бота.\n\nДля початку оберіть ваш регіон.`, againOptions);
        }
        if (text === '/again') {
          return chooseRegion(chatId)
        }
        if (text === '/map') {
          return bot.sendMessage(chatId, `<a href=\"https://liveuamap.com/">Відкрити мапу.</a>`, {parse_mode : "HTML"});
        }
        if (text === '/region') {
          return chooseRegion(chatId);
        }
        if (text === '/lastday') {
          await bot.sendMessage(chatId, 'Будь ласка зачекайте, бот збирає оновлену інформацію!');
          return bot.sendMessage(chatId, await getLastDayStatistics().then(el => el), {parse_mode : "HTML"});
        }
        if (text === '/evacuation') {
          return bot.sendMessage(chatId, EVACUATION_INFO, {parse_mode : "HTML"});
        }
        if (text === '/statistics') {
          await bot.sendMessage(chatId, 'Будь ласка зачекайте, бот збирає оновлену інформацію!');
```

```
        await bot.sendMessage(chatId, await getStatistics().then(el => el),
{parse_mode : "HTML"});
      }
      else {
        return bot.sendMessage(chatId, 'Я Вас не розумію, спробуйте ще!)');
      }
      } catch (e) {
        await bot.sendMessage(chatId, `Выбачте, але поки що немає оновленої
інформації за сьогоднішній день, спробуйте будь ласка пізніше, або ви
можете подивитись статистику за минулий день обравши команду: /lastday`);
      }
  })
   bot.on('callback_query', async msg => {
      const data = msg.data;
      const chatId = msg.message.chat.id;
      let filteredInfo = cityList.filter(region => region.city === data).map(region =>
region.info) + '';
      if (data === '/again') {
        return chooseRegion(chatId)
      } else if(data) {
        await bot.sendMessage(chatId, filteredInfo, {parse_mode : "HTML"},
againOptions);
      } else {
        await bot.sendMessage(chatId, `Обрати інший регіон`, againOptions);
      }
   })
}

start()
```

ДОДАТОК Г

Інтерфейс регіонів

```
module.exports = {
  cityOptions: {
    reply_markup: JSON.stringify({
        inline_keyboard: [
            [{text: 'АР Крим', callback_data: `1`}],
            [{text: 'Вінницька', callback_data: '2'}],
            [{text: 'Волинська', callback_data: '3'}],
            [{text: 'Дніпропетровська', callback_data: '4'}],
            [{text: 'Донецька', callback_data: '5'}],
            [{text: 'Житомирська', callback_data: '6'}],
            [{text: 'Закарпатська', callback_data: '7'}],
            [{text: 'Запорізька', callback_data: '8'}],
            [{text: 'Івано-Франківська', callback_data: '9'}],
            [{text: 'Київська', callback_data: '10'}],
            [{text: 'Кіровоградська', callback_data: '11'}],
            [{text: 'Луганська', callback_data: '12'}],
            [{text: 'Львівська', callback_data: '13'}],
            [{text: 'Миколаївська', callback_data: '14'}],
            [{text: 'Одеська', callback_data: '15'}],
            [{text: 'Полтавська', callback_data: '16'}],
            [{text: 'Рівненська', callback_data: '17'}],
            [{text: 'Сумська', callback_data: '18'}],
            [{text: 'Тернопільська', callback_data: '19'}],
            [{text: 'Харківська', callback_data: '20'}],
```

```
            [{text: 'Херсонська', callback_data: '21'}],
            [{text: 'Хмельницька', callback_data: '22'}],
            [{text: 'Черкаська', callback_data: '23'}],
            [{text: 'Чернівецька', callback_data: '24'}],
            [{text: 'Чернігівська', callback_data: '25'}],
        ]
    })
},

againOptions: {
    reply_markup: JSON.stringify({
        inline_keyboard: [
            [{text: 'Обрати регіон.', callback_data: '/again'}],
        ]
    })
}
}
```