МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
АВТОМАТИЗОВАНИХ СИСТЕМ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему Особливості застосування скінченного автомата для обробки інформації в online-обслуговуванні

Виконав: студент 2 курсу, групи 8.1210-2іпз
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
М. Г. Левченко
(ініціали та прізвище)
Керівник доцент, к.ф.-м.н. І. А. Скрипник
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П. О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ

Кафедра _____ програмного забезпечення автоматизованих систем
Рівень вищої освіти _____другий (магістерський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри В.Г. Вербицький
"_01_" __вересня_____2021_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

__Левченку Максиму Геннадійовичу__
(прізвище, ім'я, по батькові)
1.Тема роботи __Особливості застосування скінченного автомата для обробки інформації в online-обслуговуванні_____

керівник роботи ___Скрипник Ірина Анатоліївна, доцент, к.ф.-м.н._____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від ___"30" червня 2021 року № 974-с_____
2. Строк подання студентом кваліфікаційної роботи_____30.11.2020_____
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми обробки інформації в online-обслуговуванні та розробка методів її вирішення;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
___слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.09.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	01.09-04.09.2021	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	05.09-07.09.2021	виконано
3	Аналіз існуючих методів рішення	08.09-12.09.2021	виконано
4	Дослідження сервісів online-обслуговування	13.09-17.09.2021	виконано
5	Узгодження подальших дій з науковим керівником	18.09-19.09.2021	виконано
6	Аналіз теоретичних відомостей	20.09-30.09.2021	виконано
7	Проектування архітектури застосунку	01.10-07.10.2021	виконано
8	Узгодження архітектури застосунку	08.10-09.10.2021	виконано
9	Реалізація бази даних замовлень	10.10-15.10.2021	виконано
10	Представлення отриманих результатів науковому керівнику і узгодження плану подальшого дослідження	16.10-17.10.2021	виконано
11	Реалізація функціоналу застосунку	18.10-14.11.2021	виконано
12	Проведення аналізу можливостей розробленого програмного застосунку	15.11-20.11.2021	виконано
13	Оформлення звіту	21.11-30.11.2021	виконано

Студент _____ М.Г. Левченко
(підпис) (прізвище та ініціали)

Керівник роботи _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 115

Рисунків: 49

Таблиць: 4

Джерел: 34

Левченко Максим Геннадійович. Особливості застосування скінченного автомата для обробки інформації в online-обслуговуванні.

Кваліфікаційна робота для здобуття ступеня вищої освіти магістра за спеціальністю 121 — Інженерія програмного забезпечення, науковий керівник І. А. Скрипник. Інженерний навчально-науковий інститут Запорізького національного університету.

Мета кваліфікаційної роботи полягає у дослідженні області online-обслуговування, вивченні інтелектуальних систем типу чат-бот, реалізації чат-бота для обробки інформації на основі автоматного програмування та демонстрації переваг використання автоматного підходу і можливих наслідків його відсутності.

Під час дослідження були розглянуті методи створення інтелектуальних систем, методи розробки застосунків по обробці запитів в online-обслуговуванні. В результаті створена програмна система чат-бот online-обслуговування, для керування якої використано автоматний підхід. Перевагою моделі цього типу є достатній рівень абстрактності та можливість застосування до об’єктів будь-якої природи [6]. Розробка виконувалась за допомогою: Microfost Visual Studio 2017 (середовище розробки), Telegram Bot Api (інтерфейс на основі HTTP), ASP.NET Core (платформа), Telegram (багатоплатформний месенджер), патерну MVC, мови програмування C#

Ключові слова: *месенджер, telegam, обробник інформації, mvc, чат-бот, користувач, інтелектуальна система, скінченний автомат.*

SUMMARY

Pages: 115

Figures: 49

Tables: 4

Sources: 34

Levchenko Maksym. Features of the use of a finite machine for information processing in online service.

Qualification work for higher master's degree in specialty 121 — Software Engineering, supervisor Iryna Skrypnyk. Engineering Educational Scientific Institute of Zaporizhia National University.

Qualification work goal is in research of online serving subject area, of the intelligent system like a chatbot, implementation of the chatbot for processing information using a finite state machine, demonstrate benefits of using a finite state machine and possible consequences of its absence.

During the research, methods of creating intelligent systems were investigated, methods of implementing web applications to process requests were considered. As a result, the chatbot application for online serving has been created. The application control layer is the finite state machine. The benefit of the current type of model is in sufficient abstraction layer and the possibility to apply it to objects of any nature [6]. Development was done by using: Microsoft Visual Studio 2017 (IDLE), Telegram Bot Api (HTTP interface), ASP.NET Core (web framework), Telegram (cross platform messenger), MVC pattern, C# as programming language.

Keywords: *messenger, telegram, data processing, mvc, chatbot, customer, intelligent system, finite state machine.*

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 ОГЛЯД ОБЛАСТЕЙ ЗАСТОСУВАНЬ ЧАТ-БОТІВ 13
1.1 Що є ChatBot ? 13
1.2 Опис проблеми 14
1.3 Приклади використання чат-ботів 17
1.4 Типи чат-ботів 21
1.5 Переваги чат-ботів 23
1.6 Історія розвитку чат-ботів 24
1.7 Постановка задачі 27
1.8 Огляд існуючих методів рішення і опис предметної області 27
1.9 Середовище розгортання 32
1.10 Приклад створення ChatBot 35
1.10.1 Перший спосіб 37
1.10.2 Другий спосіб 39
РОЗДІЛ 2 АНАЛІЗ ТЕХНОЛОГІЇ АВТОМАТНОГО ПРОГРАМУВАННЯ . 40
2.1 Області застосування автоматного підходу 40
2.2 Основні поняття 43
2.3 Парадигма автоматного програмування 44
2.4 Особливості імперативного програмування 47
2.5 Зв'язок з об'єктно-орієнтованим програмуванням 48
РОЗДІЛ 3 РОЗРОБКА ОБРОБНИКА ІНФОРМАЦІЇ 50
3.1 Аналіз існуючих рішень 50
3.2 Архітектура системи 52
3.3 Засоби реалізації 53
3.3.1 Visual Studio 53
3.3.2 Web API 54
3.3.3 ASP.NET Core 54
3.3.4 MVC 55

3.3.5 Bot-API 56
3.3.6 Ngrok 57
3.3.7 SQLite 57
3.4 Функціональні можливості 58
3.5 Методи і алгоритми 61
3.5.1 Основні принципи 61
3.5.2 Автомат Мура 90
3.6 Діаграма класів 92
3.7 База даних 100
3.8 Інтерфейс 103
РОЗДІЛ 4 ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ СИСТЕМИ ONLINE-ОБСЛУГОВУВАННЯ 108
4.1 Аналіз результатів 108
4.1.1 Повноцінне формування замовлення 109
4.1.2 Формування замовлення з корекцією раніше введених даних 109
4.1.3 Формування замовлення з обробкою помилково введених даних 110
ВИСНОВКИ 111
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 112

ВСТУП

Актуальність теми

З кожним днем кількість існуючих послуг зростає, відповідно зростає їх попит, але далеко не кожен може відразу отримати доступ до них. Саме тому утворюються величезні черги, через що споживачі втрачають власний час, а представники послуг гроші.

Рішень цієї проблеми існує дуже велика різноманітність, деякі з них менш ефективні, а деякі більш. Наприклад, можна збільшити кількість філіалів, котрі постачають послуги, а можна автоматизувати систему, що суттєво зменшить затрати.

Сучасний світ переходить на системи, котрі не потребують втручання людини. Яскравим та доречним прикладом є інтелектуальна система у вигляді чат-боту на основі скінченного автомату. Скінчений автомат це система, котра чітко займається врегулюванням послідовності дій, в нашому випадку послідовності надання послуг. Автомат має так звані стани, він переходить з одного в інший, тим самим забезпечують дотримання необхідної послідовності дій.

Автомати використовуються досить давно, в різноманітних сферах, але зазвичай людина навіть не здогадується, що саме стоїть за цим всім.

Головна перевага використання автоматів в тому, що вони мають достатній рівень абстрактності та можливість застосування до об’єктів будь-якої природи [6]. Також забезпечується чітке виконання необхідних дій, вони задаються заздалегідь, тому їх самочинне виникнення, або відсутність наближена до нуля.

Використання саме цієї технології забезпечить відмовостійкість, надійність, масштабованість, а також простоту всієї системи загалом. Завдяки цьому отримаємо полегшену програму, котра не потребує великих зусиль з її підтримки, а саме головне розмір самої системи не має значення. Автомат впорається як з невеличкою структурою так і з великими складними гілками.

Мета і завдання дослідження

Головною метою цієї роботи є дослідження доцільності використання інтелектуальної системи чат-боту на основі автоматного підходу у найпоширеніших сферах надання послуг і отримання зворотного зв’язку з даними про переваги або недоліки в застосуванні, а також створення застосунку по обробці вхідних запитів. Отримані знання повинні стимулювати громадськість в зацікавленні цього типу автоматизованості, його поширенню та розвитку застосування у нашому світі.

- Ознайомитись з принципами побудови (інтеграції) автоматизованих систем.
- Ознайомитись з інтерфейсом розробки The Bot Api.
- Познати особливості створення власних ботів.
- Розібратись з впровадженням власних ботів у систему Telegram.
- Побудова додатку, що представляє собою обробку вхідної інформації за допомогою власного бота.

Об’єкт дослідження

Об’єктом дослідження є процес обробки вхідних запитів зі сфери online-обслуговування.

Предмет дослідження

Предметом дослідження є інтелектуальна система чат-бот, що використовує автоматний підхід.

Методи дослідження

Для отримання необхідних результатів дослідження використовувалися певні методи:

- Знаходження та обробка літератури з побудови та управління додатками, що займаються автоматними систем.
- Пошук незвичайних та непростих аспектів.
- Аналіз існуючих рішень.

Наукова новизна одержаних результатів

Новизною є застосування скінченого автомату як головного механізму керування в застосунках з обробкою ніскінченної кількості вхідних запитів. Різноманітними прикладами використання цього типу автоматизації є забезпечення конфіденційності та цілісності даних в криптографії [7]. При керуванні проектами, розробники часто зустрічаються для обговорення певних аспектів їх роботи, одним з головних інструментів на цих зустрічах є візуалізатори алгоритмів [8]. Робототехніка є невід'ємною частиною при роботі, де потребується високоточні дії [9]. При виробництві програм, тестування є запорукою гарантування гідної якості продукту [10]. Сучасні навчальні організації використовують інтелектуальні системи, для забезпечення надання необхідних знань учням [11]. Все це стає можливим завдяки використанню скінчених автоматів.

Існує велика кількість типів скінченних автоматів та всі вони підходять під певний тип роботи. Через небажання розробників розбиратись у всіх особливостях виникає істотна проблема. А саме поява коду дуже поганої якості з великими новими безглуздими системами, котрі відтворюють вже існуючі алгоритми, але з дуже поганою інтерпретацією. Відбувається побудова так званих «велосипедів» з трьома або більше колесами. Звідси отримуємо майже неможливість точної підтримки та обслуговуванні такої системи, а масштабування потребує переробки всієї системи.

Через вищезазначені достоїнства приймається рішення використовувати саме цей тип автоматизованої роботи, а саме скінчений автомат при розробці додатку, що займається обробкою вхідної інформації.

Практичне значення одержаних результатів

В результаті створено застосунок, котрий представляє собою обробник інформації в online-обслуговуванні, на базі скінченного автомату, виконаний мовою C#. Клієнтська частина застосунку представлена у виді додатку Telegram, використовується для відправки вхідної інформації. Програмно-

апаратна частина сервісу виконана за допомогою інтерфейсу на основі HTTP, Telegram Bot Api, а також платформи ASP.NET Core.

Створена система є гарним прикладом, як можна зменшити навантаження з підприємств, що займаються обробкою замовлень та надають консультативну інформацію. Побудова та розробка саме таким чином покаже позитивні та негативні моменти даного підходу, котрі можна врахувати в майбутньому при створенні систем з подібним напрямком.

Апробація одержаних результатів

Підсумки дослідження були представлені на XIV університетській науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених Запорізького національного університету «Молода наука-2021» [28], а також на I Всеукраїнській науково-практичній конференції здобувачів вищої освіти, аспірантів та молодих вчених Інженерного навчально-наукового інституту Запорізького національного університету [29].

Глосарій

Bot Api — це інтерфейс на основі HTTP, створений для розробників, зацікавлених у створенні ботів для Telegram.

ASP.NET Core — вільно-розповсюджуваний крос-платформний фреймворк для створення веб-додатків з відкритим вихідним кодом. Дана платформа розробляється компанією Майкрософт спільно з спільнотою і має велику продуктивність в порівнянні з ASP.NET. Має модульну структуру і сумісна з такими операційними системами як Windows, Linux і macOS.

MVC — схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, представлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Модель (Model) — надає дані і реагує на команди контролера, змінюючи свій стан.

Представлення (View) — відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.

Контролер (Controller) — інтерпретує дії користувача, сповіщаючи модель про необхідність змін.

C# — об'єктно-орієнтована мова програмування. C # відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, змінні, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі в форматі XML. C #, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C # на відміну від C ++ не підтримує множинне успадкування класів (між тим допускається множинна реалізація інтерфейсів).

Telegram — багатоплатформовий месенджер з функціями VoIP, що дозволяє обмінюватися текстовими, голосовими та відеоповідомленнями, стікерами та фотографіями, файлами багатьох форматів. Також можна здійснювати відео- і аудіозвонкі, організовувати конференції, розраховані на багато користувачів групи і канали. Клієнтські програми Telegram доступні для Android, iOS, Windows Phone, Windows, macOS і GNU / Linux.

Microsoft Visual Studio — лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментів.

NuGet — система управління пакетами для платформ розробки Microsoft, в першу чергу бібліотек .NET Framework.

Скінченний автомат (СА) — математична абстракція, модель дискретного пристрою, що має один вхід, один вихід і в кожен момент часу знаходиться в одному стані з множини можливих та може нескінченно обробляти вхідні дані.

РОЗДІЛ 1 ОГЛЯД ОБЛАСТЕЙ ЗАСТОСУВАНЬ ЧАТ-БОТІВ

1.1 Що є ChatBot ?

ChatBot — програмне забезпечення, головна мета якого формування відповіді згідно отриманому запиту, котре було надіслано користувачем. Завдяки цьому надається можливість підтримки зв’язку 24 години на добу, що дозволяє завжди бути у доступі для будь-якого бажаючого отримати допомогу. В методику роботи закладено алгоритм листування з усіма можливими варіантами відповідей та питань. Сфера застосування такої програми дуже широка: месенджери, сайти, соціальні мережі. Дуже часто різноманітні підприємства заощаджують на розробці власного додатку використовуючи чат-бот.

Існують боти різної складності, легкі та продвинуті. Перший варіант асистенту здатен відповідати лише на питання, що були заздалегідь прописані у його базі даних. В разі відсутності відповідного шаблону питання утворюються складнощі в роботі, через що потребується вже допомога реальної людини. Видача відповідей виконується згідно прописаному протоколу.

Сучасні боти мають на озброєнні великий склад технологій, однією з них є штучний інтелект та підтримка мов усього світу. Програми такого типу з плином часу самі себе удосконалюють завдяки постійному аналізу та обробці інформації, що поступає. Завдяки чому при звернені до бота не має потреби точно формувати питання. Головним секретом такого програмного забезпечення є використання сучасних обробників мов та останніх версій машинного навчання.

Використання онлайн асистента — це не тільки підтримка з приводу технічних справ, а ще й залучення нових користувачів. Також потрібно не забувати про підтримку інтересу у створеній базі клієнтів. Через широкий спектр застосування та можливостей, впровадження таких програм набирає все більше і більше обертів.

Функціонал онлайн асистента:

- звичайне спілкування;
- консультація з різних сфер діяльності;
- купівля пропусків до різних закладів розваг;
- аналіз даних;
- грошовий обіг;
- реєстрація походу до косметичного салону;
- резервація місць в закладах харчування.

В 1966 році Джозеф Вейзенбаум створив першого чат-бота під назвою Еліза. Програма імітувала роботу психотерапевта таким чином, щоб користувач мав бажання розповісти якомога більше, що його бентежить.

Програми подібного роду зобов'язані проходити тест Т'юрінга, щоб мати враження, чи в правильному напрямку виконується розробка.

1.2 Опис проблеми

Процедурна ефективність та достатній рівень економії підприємств забезпечується чат-ботами. Також підвищується комфортабельність та рівень послуг, що постачаються до клієнтів. Підприємствам набагато легше, швидше формуються відповіді та вирішуються проблеми їх клієнтів, завдяки чому зменшується кількість людино-годин та витрати на зарплатню робочому персоналу.

Чат-боти підвищують конкурентну спроможність компаній. Вони налагоджують як достатній рівень бізнес-охоплення, так і рівень персональних послуг, а це дуже важливо в сфері конкуренції. Завжди існує обмеження в одночасному наданні послуг через кількість співробітників. Існування вірогідності нестачі робочої сили ніколи не можна виключати. Конкурентна спроможність підприємства знаходиться під загрозою у разі покладання тільки на людські можливості, тому що в цьому випадку вдаються використанню певним зразковим моделям, через що рівень персонального підходу знижується.

Використання чат-ботів забезпечує взаємодію підприємства з необмеженою кількістю клієнтів. Рівень їх застосування можна коригувати в залежності від потреб. Підприємства, застосовуючи подібні програмні системи, мають змогу забезпечити майже нескінченну кількість індивідуальних послуг в одну мить та передчувати майбутні потреби ідентично, якби це робила жива людина.

Дослідження в сфері обслуговування показали, що застосунки типу листувань є найліпшим шляхом зв'язку з підприємствами з приводу рішення необхідних питань. Чат-боти забезпечують в рази кращий рівень обміну листами та комфорт в порівнянні з можливостями людини. Якщо взяти за приклад банківську сферу обслуговування, то чат-боти будуть в 4 рази швидше при порівнянні їх з реальними людьми з відповідного відділу. Все, що компанії роблять для підвищення власного благополуччя, водночас позитивно відображується на задоволеності клієнтів. Тобто в компаніях діє методика взаємовиручки.

Цілодобова підтримка. Компанії зменшують чисельність робітників заміняючи їх відповідним програмним забезпеченням. Користувач може скласти замовлення у будь-який зручний для нього момент часу, та воно буде відразу оброблене, що в свою чергу збільшує кількість позитивних відгуків в сторону компаній.

Покращення запасів. Чат-боти виводять економіку підприємства на достатній рівень. Впровадження автоматизації приносить гарні показники. Якщо виконати загальний аналіз отриманих ботом питань, можна сформувати шаблони відповідей та впровадити їх до системи. У цьому випадку жива робоча сила розподіляється на інші ділянки, тим самим вдосконалюючи інші частини бізнесу, а користувачі отримають задовільну швидкість обслуговування.

Покращення клієнтської бази даних. Підприємства за допомогою чат-ботів рішучіше видає довідку з приводу товарів власного виробництва. Гарна швидкість обслуговування гарантує підвищення притоку клієнтів. То-

му систему листування потрібно спроектувати так, щоб не було ситуацій, з котрих немає виходу. Навіть якщо проблемна ситуація стосовно обслуговування сформувалася, то програмне забезпечення повинно знайти обхідний шлях, котрий приведе до необхідної відповіді, що потребує користувач. Також необхідно враховувати спосіб та манеру піднесення інформації до клієнта. Глобальна мережа Інтернет завжди налаштовує співрозмовника на ненапружену атмосферу, саме це повинно забезпечуватись програмною системою, щоб користувач був в безпеці від негативних думок.

Забезпечення підвищеного інтересу. Завдяки відмінному проектуванню та архітектурі чат-боту гарантується високий шанс отримання нового замовлення навіть від легкої зацікавленості людини. Використовуючи систему листування, можна організувати отримання відгуків від користувачів. На основі нової інформації з’являється можливість враховувати в майбутньому людські вподобання та тим самим покращувати показники продажів. Чат-бот є гарним відображенням показників заінтересованості користувачів в продукції.

В сучасному світі одним з головних критеріїв людського блага є мобільність. Тому застосування та розробка чат-ботів в нинішній час є дуже актуальною темою. Програмні системи автоматизованого листування отримують все більше і більше поширення та стають на заміну старим технологіям.

Чат-боти забезпечують клієнтам підприємств персональний догляд та конфіденційність без участі можливих посередників, а саме живих співробітників. Основна кількість питань, що надходить від клієнтів, має легку сутність, рішення яких знаходиться за лічені секунди. Досягнення таких результатів забезпечується постійним оновленням довідкової інформації. Чат-боти відмінно автоматизують це питання, не потребуючи зайвої допомоги та зайвих витрат. При виникненні надзвичайної ситуації, а саме не оброблення запиту, вся необхідна інформація передається іншому відповідному відділу підприємства для її вирішення. Подібна технологія автоматизованого листу-

вання заощаджує час, гроші компаній та виводить спілкування з замовником на зовсім новий рівень комфорту.

1.3 Приклади використання чат-ботів

Велика кількість ІТ компаній заохочує використання чат-ботів у власному середовищі для обслуговування персоналу, таким чином переходячи на самообслуговування та автоматизацію. Завдяки інтелекту, яким наділений чат-бот, гарантується автоматичне вирішення питань з приводу своєчасної зміни у системі статусу, редагування користувальницьких пін-кодів, відображення попереджувального повідомлення про відключення на деякий час, а також керування базою даних та видача доступу до неї у будь-який необхідний момент часу.

З приводу підприємств, що займаються бізнес-діяльністю, чат-боти запроваджуються в централізованих відділах з допомоги замовникам для керування зверненнями, що надходять, та спрямування клієнтів до необхідних фахівців. Боти знаходять застосування у підготовці робітників, що тільки-тільки поступили до складу працівників. Ще подібні програмні системи забезпечують допомогу усьому робочому складу у вирішенні звичайних робочих питань, сюди входить складання плану відгулів, здобуття нових знань, формування кошторису по закупівлі нового обладнання та багато іншого, де потребується вторгнення людське. Онлайн-магазини, використовуючи ботів, надають користувачам можливість знайти необхідну продукцію, а також сповіщають про акції, що наближуються. Сучасні заклади медичного призначення спрощують процедуру запису до необхідного лікаря завдяки ботам. В закладах освітнього характеру чат-боти вирішують питання стосовно технічних моментів, сплачування навчального семестру, донесення розкладу до студентів, що тільки почали свій навчальний шлях.

Компанії, зв’язані з грошовим обігом, що страхують майно та банки використовують ботів, адже їм потрібен постійний зв’язок з абонентом. Аге-

нтства з організації мандрівок та подорожей встановлюють чат-ботів для своєчасного обслуговування клієнтів, надаючи змогу без проблем резервувати квитки на цікаві для них події, формувати доставку продуктів харчування.

Сфера застосування чат-ботів дуже широка — це відкриття доступу до певних центрів послуг, бронювання квитків до якихось виступів, резервування місць в хостелах. Чат-боти приходять на допомогу при виконанні дій, що постійно виконуються одна за одною, та вони дуже схожі між собою, наприклад банківська справа, торгівельні справи. Подібне програмне забезпечення використовується державою для ведення обліку громадян та їх споживчих потреб, побажань. Комунальні служби спрощують собі життя ботами в обробці великих обсягів інформації та формування грошових підсумків.

Впровадження чат-ботів є й в засобах спілкування (Facebook,WhatsApp, Telegram, Вконтакте, Viber), на сторінках мережі Інтернет, в застосунках мобільного напряму, в помічниках з голосовим керуванням.

У підприємства під назвою «Pizza Hut», діяльність якої є виготовлення піцци, мається власний чат-бот, що надає можливість користувачам дрібнити замовлення між собою та отримувати сповіщення з промо-кодами (див. Рис.1) [33].

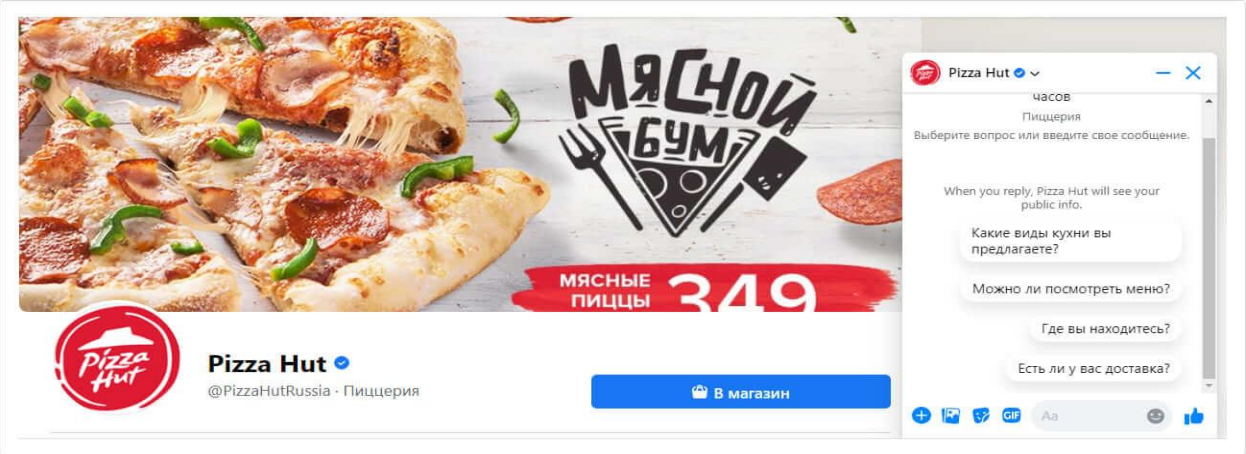

Рис.*1 Чат-бот від Pizza Hut*

«Aviasales», сервіс з авіа перельотів, налаштував бота з відстеженням квитків з привабливими цінами [33]. Забезпечується контроль квитків згідно

параметрам, що були введені користувачем. При зниженні цін на зацікавлені перельоти відбувається видача повідомлення з даної події (див. Рис. 2).

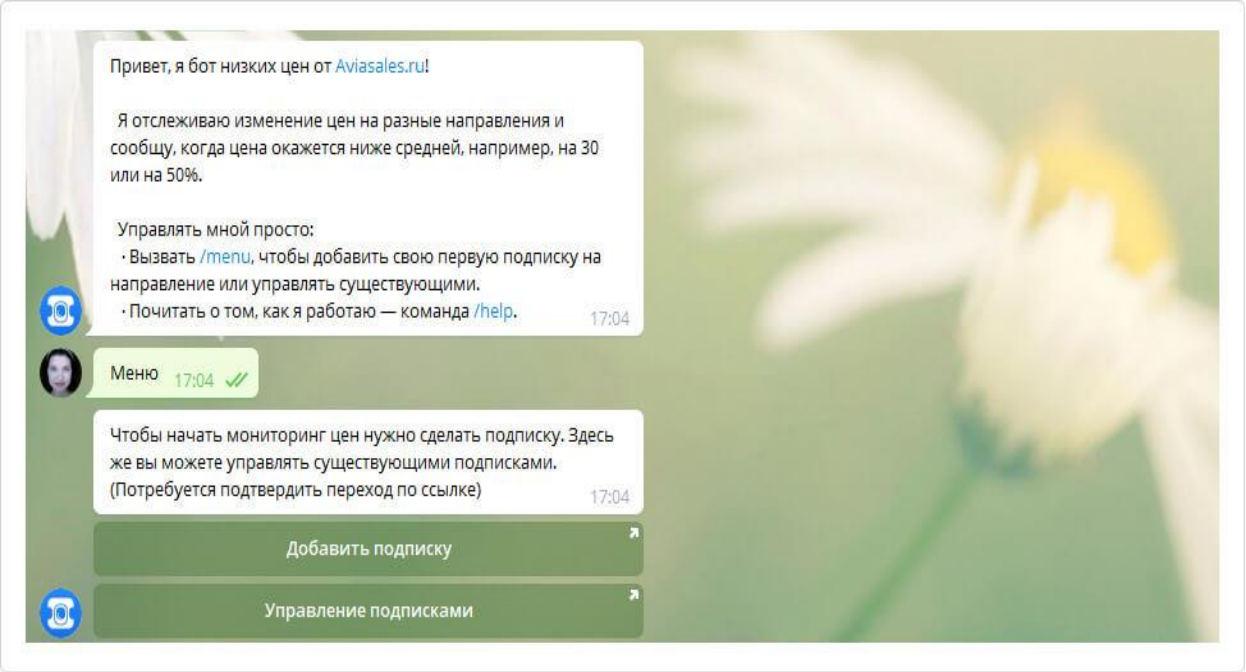

Рис.*2 Бот від Aviasales*

«Skoda», компанія по виробництву автомобільних продуктів, винайшла бота під назвою «Люсі» [33]. Він сприймає команди, котрі стосуються автомобіля, що потребується підібрати, наприклад намір та підсумкова вартість. Згідно отриманих побажань формується перелік автомобілів, що можуть підійти. Є можливість уточнити подробиці угоди та спланувати тестування машини, що сподобалась користувачеві (див. Рис.3).

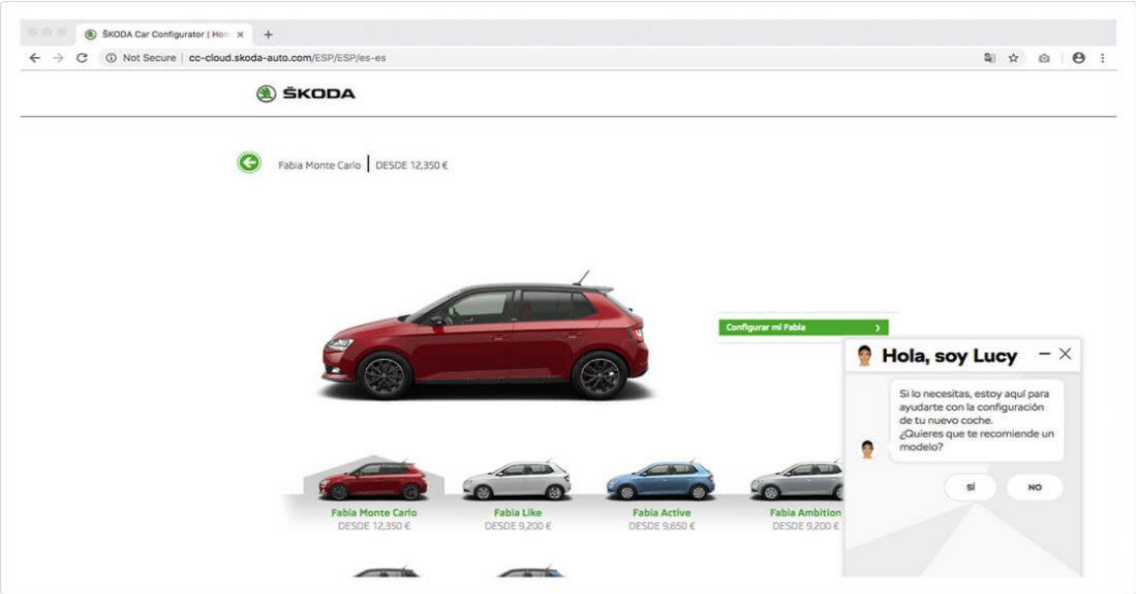

Рис.*3 Бот «Люсі» від Skoda*

Підприємство «Альфа-Банк», спрямоване на грошовий обіг, налаштувало бота з підтримкою різноманітних мов [33]. Його база знань утворена на основі аналізу операцій, що відбуваються в центрі обробки запитів. Чат-бот навчився сприймати інформацію щодо кредитних карт, акцій, та є підтримка варіативності питань, що надходять. Він може розпізнати такі запити: «поточний грошовий баланс» та «скільки в мене залишилось грошей» , адже це питання з однаковим сенсом.

Області застосування чат-ботів [31]:

- банківська справа — грошові перекази та погашення кредиту;
- диспетчер зв’язку — вирішення питань з технічних проблем;
- страхування майна — роз’яснення пунктів бланку та отримання порад;
- торгівельна діяльність в мережі Інтернет — отримання порад відносно подробиць грошового переказу, надходження товару та місця отримання товару;
- медицина — запис до лікаря, отримання мінімальної кількості порад з приводу здоров’я;

- подорожі — повідомлення про появу нових квитків та місць, куди можна відправитись;
- навчання — обговорення проходження навчального семестру та успішності студентів;
- державний напрям — ведення обліку побажань громадян та насущних проблем;
- відділ кадрів — аналіз портфоліо, видача робочих подробиць.

1.4 Типи чат-ботів

Існує велика кількість різновидів чат-ботів [31]. Вони характеризуються на основі використаних алгоритмів:

- Стиснутий — методика роботи заснована лише на певних питаннях, формування запитів потребує жорсткого зазначення.
- З модулем самовдосконалення — користувач поряд з подібною програмною системою почуває себе максимально комфортно, начебто розмовляє з живою людиною. Програма постійно виконує аналіз даних, що поступають, та робить собі замітки, для вдосконалення своєї системи обслуговування.

Боти відрізняються способами зв’язку з користувачем:

- З використанням ключів інтерфейсу — за допомогою фіксованих елементів інтерфейсу відбувається подання заздалегідь визначеного функціоналу клієнту. Подібні програмні системи є найпоширенішими. Такий чатбот є одним з найпростіших, налаштовується для отримання власної інформації користувача та складання замовлень.
- Рядковий — виконується аналіз даних, отриманих у вигляді множини рядків, на цій основі підбирає підходящий шаблон відповіді.

Поділяють за цільовим призначенням:

- Інформаційний — зазвичай знаходиться на озброєнні підприємств з метою видачі звичайної відповіді, у вигляді деяких даних. Видає за-

здалегідь проробленні вирази, що працюють за певною методикою, також повідомляє про наявність організації прямого зв’язку з живим фахівцем.

- Різнобічний — є аналогом застосунків, що використовуються у сучасних смартфонах. Надає функції з пошуку необхідного продукту, сервісу, виконання закупівлі, отримання довідкової інформації.

Чат-боти поділяються за різновидом надаваємих послуг:

- Боти з сфери торгівлі — заохочує нових клієнтів за допомогою спеціальних пропозицій та промо-кодів. Є максимально продуктивним з приводу своїх товарів, завжди допоможе підібрати альтернативу відібраній продукції.
- Бот, спрямований на збільшення клієнтської бази даних, — має у своєму розпорядженні великий обсяг даних, отриманих від користувачів, та відмінно рекламує виробничий продукт й заохочує до його купівлі.
- Бот з формування угод — підготовлює усе необхідне для формування угоди, бронювання чогось або завідує грошовим обігом.
- Бот з сфери комунікацій — виступає у ролі довідкової, показує поточні можливі виходи з певної ситуації.
- Бот з технічних питань — роз’яснює інструкційні вказівки та вирішує виникаючі надзвичайні ситуації.
- Бот-помічник — зазвичай працює пліч о пліч з іншими частинами системи. Має технологію багатопоточності, що позитивно відображується на його продуктивності. Має схожості з пошуковими системами.

За формою представлення:

- Засоби спілкування.
- Асистенти в глобальній мережі Інтернет.
- Застосунки в смартфонах.
- Повідомлення.
- Програми контролю дій.
- Програми з досліджень.

1.5 Переваги чат-ботів

Продуктивний зв’язок з користувачем. Системи, засновані на ботах, займаються залученням нових заінтересованих осіб, а також зміцненню бази постійних клієнтів, що в свою чергу спрямовано на розквіт бізнесу [31, 32]. Якщо зробити порівняння з підприємствами, котрі використовують стандартні методики зв’язку з клієнтом, то боти мають переваги, тому що вони видають лише потрібні та цікаві дані, згідно отриманому запиту строго по суті діла. Завдяки чому користувачі отримають лише актуальну інформацію. Політика підприємств, що використовує автоматизоване програмне забезпечення, спрямована на гарантоване зміцнення свої позицій на ринку.

Бережливість. Підприємства витрачають немалі кошти на сплачування заробітної плати людському складу свого штату [31, 32]. Відповідно при зростi рівня підприємства, зростає чисельність співробітників, що веде до ще більш масштабних втрат грошей. Боти є ідеальними мовчазними, невибагливими співробітниками, на котрих не потребується видача заробітної плати. Підприємства без складнощів впроваджують подібні програмні системи, що призводе до зменшенню живої робочої сили та виключенню людського фактора (помилок) при обслуговуванні клієнтів. При необхідності запити передаються більш вповноваженим особам.

Збір відгуків та переваг клієнтів. Боти організують зворотній зв’язок з користувачами з ціллю покращити рівень сервісу та визначити недоліки у системі [31, 32]. На основі отриманих даних можна розробити власну класифікацію користувачів, щоб ще краще розуміти та обслуговувати їх.

Опрацювання інтересів. Боти організують аналіз вхідних даних, якіпоступають від клієнтів, та реалізують індивідуальний підхід для кожного протягом всього робочого процесу [31, 32]. Чат-боти володіють актуальними даними та даром переконання. Подібні системи також спрямовані на знаходження недостатньо опрацьованих ділянок у підприємстві, шляхом врахування власно створених атрибутів.

Простота використання. Боти гарантують гідну якість виконання роботи, найчастіше незважаючи при цьому на складність та спосіб надання запитів [31, 32]. Що в свою чергу забезпечує відмінну конкурентну спроможність.

1.6 Історія розвитку чат-ботів

Алан Тьюринг — початок всього.

В 1950 році молодий комп'ютерний геній, Алан Т'юринг, випустив з під пера власну публікацію, котра мала назву «Обчислювальні машини і інтелект». Свою роботою науковець демонстрував, що програмне забезпечення може мати інтелект. Доведення цьому було виконано за допомогою випробування, котре мало назву «Імітаційна гра», або ще як відоме у світі тест Т'юринга [30].

Випробування Т'юринга складалося з добровольця, котрий повинен був вести бесіду одночасно з написаною програмою та реально живою людино, за допомогою комп'ютера, при умові, що учасники бесіди не бачили один одного. Головною метою добровольця було визначити де людина, а де штучна програма. Науковець робив ставку на те, що доброволець не зможе класифікувати відповіді відносно їх власників, а це б доводило існування можливості наслідування людині. Думка науковця була така, що з 2020 року будь-яка машина майбутнього часу зможе без складнощів складати його тест.

Еліза — штучний асистент.

Джозеф Вейценбаум, науковий діяч, виходець інституту технічного напряму зі штату Массачусетс, спроектував програмну систему з назвою Еліза, в 1966 році [30]. Згідно мировій думці та даним, ця розробка є першим чат-ботом. Еліза є легким та звичайним ботом, що використовував слова у вигляді ключів з ціллю наслідування живого фахівця, з психічних розладів. Програмне забезпечення вело бесіду з людиною, використовуючи при цьому заздалегідь розроблені шаблони, щоб відповідь здавалась досконалою.

Коли клієнт звертався до системи зі словами: «Я люблю співати», то бот видавав наступну відповідь: «Повідай про себе ще щось». Це надавало змогу імітувати зацікавленого співрозмовника. У користувачів складувалося таке відчуття, що програма начебто свідомо формує відповіді та розуміє людину ніхто як інший. Звісно це було лише імітування. Винахід Джозефа Вейценбаума став могучим поштовхом в розробці вдосконалених схожих систем з розумінням людських мов

Перрі — імітація захворювання з назвою шизофренія.

Один вчений з психіатричного напряму, що працював лаборантом в Стенфорді над тематикою імітації інтелекту, розмірковував про можливості програмної системи, а саме про свідоме зрозуміння роботи мозку, це був Кеннет Колбі в 1971 році [30]. Науковець вірив, що обчислювальна машина зможе лікувати людей з захворюваннями психологічного характеру. Таким чином Кеннетом було винайдено Перрі, систему, що грає роль персони з захворювання шизофренія.

Перрі був розроблений з ідеєю поліпшення в навчанні молодих студентів медичного закладу, щоб вони отримали деякий досвід з практики лікування. Цей проект є одним з перших хто пройшов відомий тест Т'юринга. Розробка в ті часи дуже сильно обурювала навколишнє суспільство, тому що це вважалося неможливим.

Jabberwacky

Одна особистість в 1988 році на основі власної самозайнятості спроектував Jabberwacky, це був Ролло Карпентер [30]. Програмне забезпечення було націлене на створювання живої бесіди з розважальним характером. Однією з особливостей системи була функція власного вдосконалення за рахунок аналізу минулих даних.

Dr. Sbaitso

Підприємство Creative Labs, місцезнаходження якого в Сінгапурі та діяльність технічні розробки, винайшла Dr.Sbaitso, в 1992 році [30]. Обчислювальна машина мала на озброєнні штучний інтелект за допомогою якого

створювала вигляд справжнього психолога. Система працювала з звуковим супроводженням. Кожний хто придбав ліцензійну копію подібного програмного забезпечення міг почути цифровий голос.

A.L.I.C.E.

Річард Уоллес повідав світу свою розробку в 1995 році, це був бот, що вживався в роль жінки [30]. Ім'я системи, це абревіатура повна назва якої Artificial Linguistic Internet Computer Entity (штучний лінгвістичний об'єкт Інтернет комп'ютеру). Розробник не вам мети розбагатіти, мета була світового характеру, тобто всі вихідні дані проекту були в відкритому широкому доступі, що повинно було спонукати інших на нові винаходи.

SmarterChild

У 2001 році підприємство, що спеціалізувалося на розробці інтерфейсів діалогу, винайшло бота з штучним інтелектом під назвою Smarter Child, це був результат роботи ActiveBuddy [30]. Бота створили з метою імітації живої, справжньої, натуральної бесіди. Є плітки, котрі вказують, що це була платформа на якій заснували голосового помічника від Apple.

Siri

Група людей: Даг Кітлаус, Адам Чейер, Том Ґрюбер, Норман Вінарський — спроектувала в 2007 році продукт під назвою Siri [30]. Ця розробка на даний момент часу входить в склад усіх розробок від Apple. Проект представляє собою індивідуального помічника, з розподіленим зберіганням на серверах мережі Інтернет, на озброєнні якого система відповідей та питань. Застосунок займається аналізом реальної мови з метою формування відповіді на поставлене питання. Система знаходить персональний підхід до кожного клієнта враховуючи його вподобання.

Cortana

В 2014 році на заході під назвою Build, що проходив в Сан-Франциско, було представлено онлайн асистента з голосовим керуванням та елементами власного інтелекту, це був Cortana [30]. Проект на даний момент часу використовується більшістю розробками компанії Microsoft та іншими розробни-

ками, що спеціалізуються на мобільних технологіях. Назва системи була взята з ім’я однієї героїні комп’ютерної гри.

1.7 Постановка задачі

Метої роботи є дослідження інтелектуальних систем чат-ботів, можливості застосування автоматного підходу та побудова обробника вхідної інформації в online-обслуговуванні у вигляді чат-боту за допомогою використання сучасних засобів розробки та сервісів.

Ціль роботи складається з аналізу методів створення обробників інформації.

Готова програмна система повинна:

- Система повинна мати змогу отримувати вхідні дані.
- Система повинна обробляти вхідні дані .
- Система повинна надавати точну відповідь, згідно типу отриманих даних.
- Зберігання вхідних даних у базі даних.
- Оновлення даних у базі даних.
- Надання клієнту списку існуючих команд, або можливих відповідей.

Майбутня доля додатку складається з підтримки, оновлень та можливості розповсюдження.

1.8 Огляд існуючих методів рішення і опис предметної області

Оскільки кожен розроблений продукт є чиєюсь власністю, а саме він коштує грошей та є системою з закритим вихідним кодом, то загальнодоступних схожих систем немає, отже, можна вважати вироблений програмний продукт оригінальним.

Взявши до уваги усі аспекти предметної області, було вирішено, що вигідною є розробка у вигляді чат-боту.

Чат-бот — програмне забезпечення, мета якого дізнатись потреби, клієнта, на основі яких буде сформовано відповіді для того, щоб догодити споживачу. Програмна система, заснована на автоматизації, використовує в основному голосові записи та рядки для бесіди з клієнтом. Чат-бот є представницьким обличчям тієї компанії, на яку він працює, з ціллю надання допомоги або релевантної інформації. Ця технологія заміщує спілкування з реальною людиною, а, значить, економить час та нерви.

Принцип дії

Штучним учасникам бесіди доводиться використовувати природні мови спілкування. Головною проблемою програм з імітацією інтелектуальних здібностей є аналіз повсякденного стилю спілкування людей.

Програмне забезпечення, котре має на озброєнні штучний інтелект, засновується на великій кількості інформації. Мається на увазі заздалегідь заготовлені шаблони, згідно яким виконується аналіз запитів, що поступають та згідно яким видається потрібні відомості. Відомі методології видачі відомостей мають наступний вигляд.

Реагування за певними словами. Спосіб такого характеру застосовувався Елізою. Клієнт звертався до програми з наступними виразами: «жовтий», «синій», «рожевий» — бот надавав відповідну фразу: «Повідайте мені більше про ваші улюблені кольори».

Збіг виразу. Розповідається про відповідність запиту, що надійшов до шаблону, котрий зберігається в базі даних. Зауваження робиться на послідовності словосполучень та інших елементах.

Збіг фрагменту. Досвідчені розробники чат-ботів виносять попередження користувачам, що потрібно уникати займенників в повідомленні, що бажають надіслати. Відмінна робота програмного забезпечення полягає у формуванні ідеальної відповіді на основі аналізу минулих даних.

Однією з перешкод в роботі легких програм-співрозмовників є розпізнавання формулювань слів та їм подібних.

Також є чат-боти, що мають змогу розумнішати, а саме займатись дослідженням. Це забезпечується завдяки технологіям, заснованим на використанні рішень з минулих завдань. Подібні програмні системи є на порядок розумнішими більшості тих, що згадувались до цього. Програмне забезпечення такого типу поділяється на пошукове та генеративне.

Пошукові системи застосовують евристичні методики з метою підбору необхідної відповіді на основі заздалегідь створеної бази фраз. Програми даного типу враховують вміст запиту та суть бесіди для видачі відповіді з бази знань. Суть складається з місця знаходження на даний момент часу в бесіді, також вся минула історія переписки і деякі проміжні дані. Евристичний метод, завдяки якому формується відповідь, буває різних типів: з відносною закономірністю або машинними систематизаторами.

В генеративних системах є можливість формулювання відповіді на власний розсуд програми без використання шаблонів бази даних. Подібній функціонал свідчить про достатній рівень технологічності програмного забезпечення.

Декларативні чат-боти, спрямовані на завдання — програмне забезпечення, ціллю якого є реалізація зазначеного функціоналу [34]. Завдяки аналізу природньої мови, а також методології штучного інтелекту, процес формування відповідей програмною системою автоматизований, це відбувається у вигляді бесіди. Вести бесіду з подібним ботом потрібно згідно певним правилам, не забуваючи про його власні особливості, через що він зазвичай працює у відділах з технічних питань та надання послуг, у ролі сервісу, що приймає інформацію та видає потрібну у відповідь. Боти, спрямовані на виконання певного завдання, мають можливість формулювати відповіді на стандартизовані питання — це може бути вимірювання проміжку часу від моменту початку роботи до даного моменту часу, або здійснення легких операцій без складнощів. При всьому тому, що в системі використовується технологія аналізу природньої мови, бесіда з подібною програмою має достатню кількість обмежень.

Предикативні інформаційні чат-боти, що функціонують у вигляді бесіди, що постійно кличуть штучними, або сучасними асистентами [34]. Такі програми мають більш широкий спектр застосування та можливостей, а також вони розумніші в порівнянні з тими ботами, що спрямовані на завдання. Подібні програмні системи приймають до уваги суть бесіди, вирішують питання з неоднозначності, здатні займатись аналізом природної мови та мають деякий власний інтелект, все це потрібно для власного вдосконалення програми, самої себе, на основі виконаної раніше роботи. Такі програмні забезпечення володіють технологією розрахунку виникнення майбутніх ситуацій, а також виконують класифікацію всіх клієнтів на основі минулих запитів. Сучасні асистенти враховують важливі моменти для користувача та мають змогу видавати актуальну інформацію, а також розрахувати майбутні інтереси клієнта. Головна особливість таких ботів — це виконання першого кроку для створення бесіди зі сторони програми, тобто виконується зацікавлення користувача, а не навпаки, як це зазвичай буває.

Text Analytics API

За допомогою компанії Microsoft та її найкращих методологій штучного інтелекту була створена група Інтернет сервісів, спрямована на аналіз рядкових даних. Цю розробку в основному застосовують для вивчення хаотичного тексту з метою отримання інформації про рівень емоційної забарвленості, виявлення головних тез та розпізнання мови, що використовувалась. Запуск цієї програмної системи не потребує ані бази даних, ані інформації з минулих запитів. Потрібно лише надати вхідну інформацію програмному забезпеченню, та воно зробить своє діло.

Цю технологію можна побачити в Azure Cognitive Services, будь-який бажаючий може випробувати програмну систему, але це виходить на платній основі.

Аналіз емоційного стану

Виходячи з досліджень емоційних станів людини з’являється можливість дізнатись їх міркування з приводу якоїсь ідеї, або компанії. Такі дослі-

дження потрібні для збору інформації, щоб мати базу даних з поганого та сприятливого досвіду користувачів.

Подібний інструмент робить замітки, котрі описують емоційну забарвленість тексту, ця процедура виконується на основі глибокого аналізу вхідної інформації. Мається шкала вимірювання емоційного стану, вона застосовується до всіх запитів, що поступають. Шкала є числом з плаваючою точкою, границями якої є нуль та одиниця. Розгортка подібної програмної системи не потребує зайвих підготувань, встановлюється на будь-яку обчислювальну машину.

Новітні видання цього програмного забезпечення повідомляють, що при наявності бажання виконати дослідження міркувань людини реалізується через інструмент дослідження емоційних станів. Даний інструментарій базується на технологіях аналізу реальних мов.

Розпізнавання мови

Функціонал з розпізнавання мови надає можливість зафіксувати яку саме мову було використано в інформації, що була надіслана користувачем. Також є підтримка відповіді на необхідній мові до користувача, завдяки великій бібліотеці мов. Процес розпізнавання мови напряму пов’язаний з аналізом емоційного стану.

Отримання важливих виразів

За допомогою Extraction API виконується аналіз інформації хаотичного вигляду та повертає файл розширення JSON, в якому знаходяться найважливіші елементи тексту, їх ще називають ключові фрази.

Такий інструмент застосовується щоб за дуже короткий проміжок часу зрозуміти суть з всієї отриманої інформації. Якщо користувач звернеться до програми наступним чином: «День був чудовий, а також була неймовірна погода», — то система виявить найважливіші елементи, а саме «день» та «неймовірна погода».

Відсоток продуктивності отримання важливих виразів зростає з об’ємом інформації, що надсилається. Тут відрізняється принцип роботи в

порівнянні з методикою аналізу емоційного забарвлення, де процес проходить гладкіше, коли поступає мало інформації.

Файл типу JSON має вигляд ключ-значення.

В ідеальному вигляді, інформація, що бажають аналізувати, повинна мати 5200 знаків в одному файлі. На виході отримується близько тисячі ключових елементів.

Нинішній час вимагає від людей мати при собі смартфон, зараз увесь світ знаходиться в одній маленькій коробочці. Кожний із нас вже має власні улюблені сервіси та додатки. На даний момент часу розробка власного застосунку коштує великих грошей, тому підприємства підлаштовуються та інтегруються власні розробки в інші готові продукти. Саме через це популярність ботів набирає оберти.

На даний момент, найпоширенішими продуктами з листування є:

- Viber.
- WhatsApp.
- Telegram.

1.9 Середовище розгортання

Моїм вибором став Telegram, це найсучасніший та найкомфортніший месенджер із усіх існуючих. ChatBot реалізується через Telegram BotAPI. Саме цікаве, що цей інтерфейс дуже зручний та простий у використанні, треба звернутися на деякий URL та Telegram надасть JSON об’єт з усім необхідним.

Відмінності Telegram від WhatsApp [12, 13, 14, 15]:

1. Присутність безпеки максимального рівня, з сто відсотковою гарантією приватності. Це забезпечується завдяки високому рівню шифрування, або ще як полюбляють казати: «за останнім словом техніки». Технології безпеки розроблені таким чином, що отримати ключі шифрування неможливо.

2. Присутній функціонал, який дозволяє створити канал по розсилці інформації, з певним налаштуванням, котре вимикає здібність відправляти коментарі до постів. Гарантується анонімність власника каналу, до тих пір, поки сам автор не захоче себе продемонструвати.

3. Підтримка великої кількості платформ: мобільний телефон, персональний комп'ютер, мобільні версії комп'ютерів та навіть наручні годинники.

4. Забезпечення максимальної швидкості відправки повідомлень та файлів (все залежить від провайдера, що надає вам швидкість передачі даних).

5. Відсутність терміну придатності у повідомлень, в бесідах.

6. Незалежність від формату файлу, що хочуть відіслати.

7. Наявність класифікації бесід по папкам (новітній функціонал).

8. Присутнє обмеження, котре не надає можливості формування запрошення до бесіди ненадійним користувачам.

9. Найкращий рівень отримання актуальних оновлень.

10. Присутня можливість вимкнення сповіщення на необмежений час.

11. Наявність інструменту «двостороння відмова».

12. Присутня можливість відправки повідомлень з зазначеним терміном самознищення.

13. Підтримка 58 мов світу в порівнянні з WhatsApp, де лише 33 мови.

14. Наявність перегляду попереднього перегляду зображення.

15. Наявність інструментарію анонімності.

16. Створювання хештегів та можливість виконання пошуку за ними (див.Рис.4).

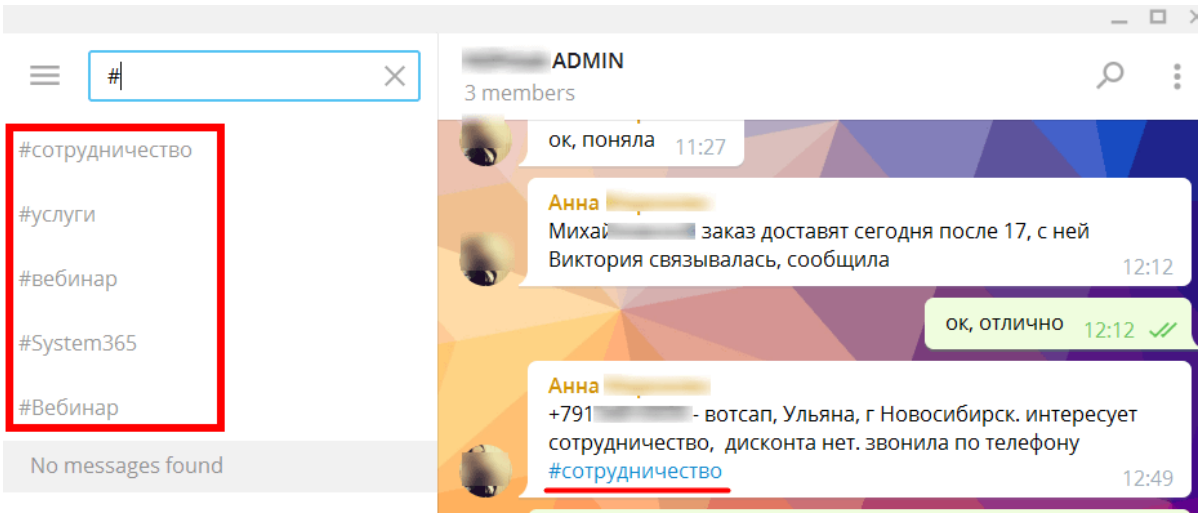

Рис.*4 Додавання хештегів*

17. Наявність прямого звернення до людини за його нікнеймом (див.Рис. 5).

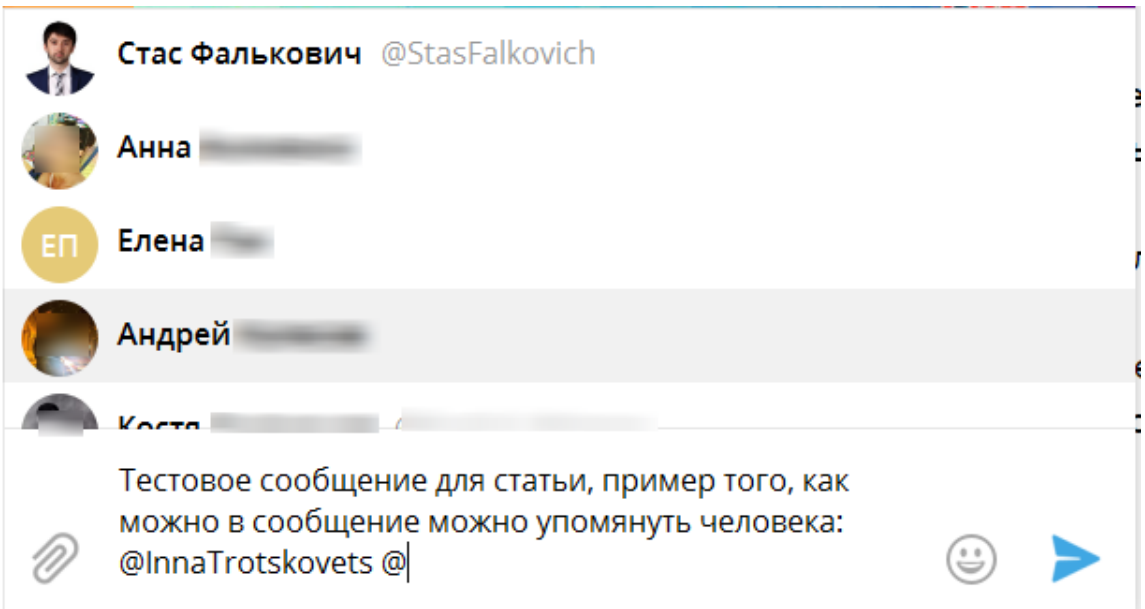

Рис.*5 Згадування учасника за нікнеймом (ім'ям)*

18. Надається змога відповісти на певне повідомлення (див.Рис.6).

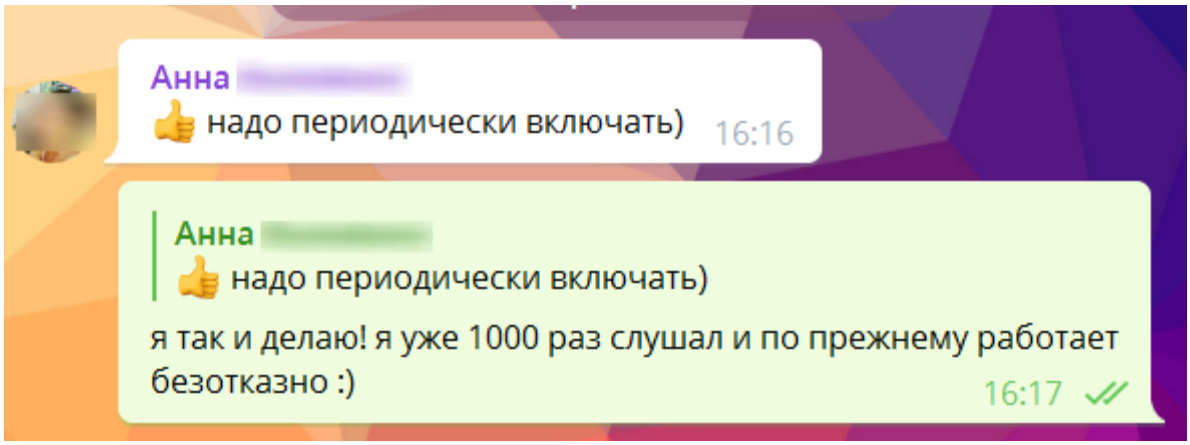

Рис.*6 Формування відповіді за певним повідомлення.*

19. Є можливість сформувати пряме посилання на певного користувача.

20. Наявність чату під назвою «Обране».

Детальне порівняння представлено в таблиці 1.

1.10 Приклад створення ChatBot

Для реалізації обов’язкова наявність бібліотеки Telegram.bot.

В результаті використання необхідної бібліотеки ви отримаєте JSON-серіалізовані об'єкти поновлення, котрі мають необхідну інформацію. Наприклад об’єкт Chat має такі елементи, дивись таблицю 2.

Таблиця *1*

Порівняльний аналіз популярних месенджерів

	WthasApp	Telegram
Підтримка платформи Android	Так	Так
Підтримка платформи IOS	Так	Так
Підтримка платформи Windows(PC)	Так	Так
Підтримка платформи Mac OS	Так	Так
Підтримка платформи Web–версія	Так	Так
Підтримка платформи Windows Phone	Так	Так
Підтримка платформи Blackberry	Так	Ні
Підтримка платформи Symbian	Так	Ні
Підтримка мови Російської Федерації	Так	Так
Підтримка версії додатку, що працює на персональному комп'ютері, при цьому без необхідності попередніх налаштувань на мобільному телефоні	Ні	Так
Підтримка функції створення власного пін-коду для доступу до застосунку	Ні	Так
Аутентифікація по двом факторам	Так	Так
Збереження файлів та повідомлень на серверах додатку	Ні	Так
Рівень швидкості передачі інформації	Середній	Великий
Наявність пам'яті, що відповідає за останній отриманий чи відправлений запит.	Ні	Так
Автоматичне збереження повідомлень, що за якимись причинами не були відправлені	Ні	Так
Присутність функції редагувань повідомлень	Ні	Так
Стікери	Ні	Так
Emoji	Так	Так
Чати за рівнем «секретний»	Так	Так
Хештеги	Ні	Так
Відповідь на пост	Так	Так
Дзвінки	Так	Так
Голосовий запит	Так	Так
Максимальна протяжність голосового запиту	15 хв.	60хв
Відправка одного або множини повідомлень	Одн/Гр	Одн/Гр
Показ інформації зі сторонніх додатків	Ні	Так

Таблиця *2*

Перелік вмісту об’єкта Chat

Поле	Тип	Опис
Id	Integer	Унікальний ідентифікатор чату. Це число може мати більше 32 значущих бітів, і деякі мови програмування можуть мати труднощі / мовчазні дефекти при його інтерпретації. Але він має не більше 52 значущих бітів, тому 64-бітове ціле число зі знаком або тип з плаваючою комою подвійної точності безпечні для зберігання цього ідентифікатора.
Type	String	Тип чату може бути «приватним», «груповим», «супергрупповим» або «канальним».
Title	String	Не обов'язково. Тема, для супергруп, каналів і групових чатів
Username	String	Не обов'язково . Ім'я користувача, для приватних чатів, супергруп і каналів, якщо вони доступні
First_name	String	Не обов'язково. Ім'я співрозмовника в приватному чаті
Last_name	String	Не обов'язково. Прізвище співрозмовника в приватному чаті
Photo	ChatPhoto	Не обов'язково. Фото чату.
Bio	String	Не обов'язково. Біографія співрозмовника в приватному чаті.

1.10.1 Перший спосіб

Даний спосіб душе легкий, Лістинг 1. Будуємо звичайний додатку типу консоль. Створюємо об’єкт класу TelegramBotClient та вказуємо токен, який належить ChatBot. Після чого треба підписатися на подію OnMessage. До цієї події ми підписуємо власний метод, котрий виконується при спрацьовуванні події, на котру ми підписалися. Цей спосіб роботи з TelegramBotClient є небе-

зпечним, тому що відбувається нескінченний потік запиту на сервера Telegram, через що створюється велике навантаження на сервіс. Дане рішення підходить лише для невеличких тестів, тому що при постійній роботі такої програми сервіси Telegram можуть заблокувати доступ.

Лістинг *1 Небезпечний приклад зв'язку з Telegram сервісом*

```
class Program
    {
        private static string token { get; set; } =
"1708898419:AAHptyxEVHS62rLwOZISIH7DKLa2dxMiJYI";
        private static TelegramBotClient client;
        static void Main(string[] args)
        {
            client = new TelegramBotClient(token);
            client.StartReceiving();
            client.OnMessage += OnMessageHandler;
            Console.ReadLine();
            client.StartReceiving();
        }
        private static async void
OnMessageHandler(object sender, MessageEventArgs e)
        {
         var msg = e.Message;
         if(msg.Text!=null)
         {
         Console.WriteLine($"Пришло сообщение с текс-
том: {msg.Text} от {msg.Chat.FirstName}");
         await client.SendTextMessageAsync(msg.Chat.Id,
msg.Text, replyToMessageId: msg.MessageId); } }}
```

1.10.2 Другий спосіб

Будуємо додаток типу WebApi, Лістинг 2. Створюємо об’єкт класу TelegramBotClient вказуємо токен та Url адресу знаходження нашого додатку в Інтернеті, при налаштуванні Webhook. Ми будемо отримувати вхідні поновлення через вихідний Webhook. При кожному оновленні для бота буде відправлятися HTTPS-запит POST на вказаний URL-адресу від сервісів Telegram, що містить серіалізовані оновлення JSON.

Webhook — метод розширення або зміни поведінки веб-сторінки або веб-додатки за допомогою зворотних викликів.

Лістинг *2 Приклад якісного зв’язку з Telegram сервісом*

```
public class Bot
    {
        private static TelegramBotClient botClient; //
об'єкт клієнта сервісу Telegram
        public static async Task<TelegramBotClient> Get-
BotClientAsync() // Налаштування зв'язку з сервісом Tele-
gram, Webhook
        {
            if (botClient != null)
            {
                return botClient;
            }
            botClient = new TelegramBotCli-
ent(AppSettings.Key);
            string hook = string.Format(AppSettings.Url,
"api/message/update");
            await botClient.SetWebhookAsync(hook);
            return botClient;
        }
    }
```

РОЗДІЛ 2 АНАЛІЗ ТЕХНОЛОГІЇ АВТОМАТНОГО ПРОГРАМУВАННЯ

2.1 Області застосування автоматного підходу

Згідно типізації, що була сформована Д.Харелом [16], кожне програмне забезпечення може представляти собою один з відомих видів [17].

- **Програми, що модифікують,** виконують певні зміни в отриманій вхідній інформації, це їх єдині існуючі обов'язки. В подібних програмах вхідна інформація завжди визначена та стан доступу до неї є відкритим відразу, як тільки програма була ввімкнена. Вихідна інформація доступна тільки після виконання закладеного алгоритму обробки. Архіватори та компілятори є програмами подібного роду.
- **Діалогові програми** підтримують зв'язок з оточуючою середою у вигляді листування (прикладом є редактор символів). Головна відзнака подібних систем це наявність можливості керувати темпом передачі даних з середовищем, що оточує, виконується принципом подачі сигналу «перерва».
- **Швидкодіючі програми** підтримують зв'язок з середою, що оточує, у режимі листування, швидкість якого визначається самою середою. Представниками даного виду є технічні засоби, що займаються транспортуванням великої кількості інформації через лінії зв'язку, типу оптоволокно та програми з керування тілесно-матеріальними механізмами.

Зазвичай при проектуванні компіляторів [18] використовують скінченні автомати та треба пам'ятати, що їх відносять до програмам, що модифікують. Автоматом вважають механізм, котрий займається обробкою деякої інформації, або розрахунками за допомогою вхідної та вихідної стрічки даних. Спершу вхідна стрічка повинна володіти певною інформацією, котру автомат потім покроково аналізує та виконує над нею необхідні дії. В кінці роботи скінченного автомата на виході маємо множину символів, котра формувалась теж покроково.

Іншою сферою застосування скінченних автоматів є керування бізнес-логікою [19], представляє собою підвид швидкодіючих систем. За першим враженням, в даній ситуації, автомат має зовсім іншу суть. Він обробляє відразу декілька вхідних стрічок, котрі володіють інформацією з оточуючого середовища, цей процес проходить в режимі онлайн. Відповідно у результаті створюється відразу деяка кількість вихідних стрічок.

Треба зауважити, що звичайні області використання скінченних автоматів включають різноманітні види програмного забезпечення. Адже обов'язки, при виконанні котрих раціонально залучати використання автоматів, та їх охоплення незрівнянно більше ніж може здатись на перший погляд, також вони вимагають проектування програмного забезпечення всіх трьох видів, що зазначались вище. Автоматний підхід, що використовується при розробці тих самих видів програм, залежачи від обставин має змогу мати відмінності в порівнянні з іншими підходами.

Визначення «складна поведінка» є вагомим чинником при залучанні скінченних автоматів. Вільно кажучи існує присутність складної поведінки при наявності варіації множини виходів, на один вхідний запит. Потрібно розуміти, що вихід залежить не тільки від входу, бо додатковим фактором може бути сукупність минулих дій. Відповідь, в простій сутності, для вхідного запиту буде залежати тільки від цього самого запиту (див. Рис. 7).

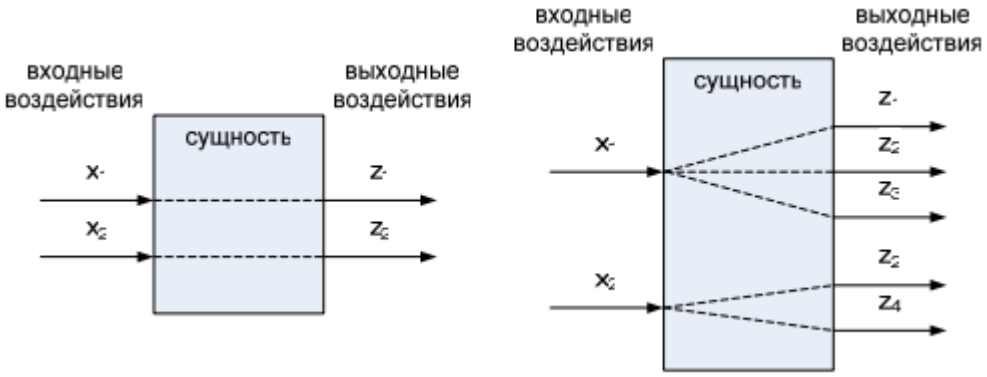

Рис.*7 Сутність з простою поведінкою та складною поведінкою*

Сутності зі складною поведінкою досить розповсюджені серед програмних забезпечень, що займаються аналізом та обчисленням інформації. «Складна поведінка» використовується в мережевих, комп'ютерно-ігрових, інтерактивних та інших сферах.

Вихідний код має характерні ознаки наявності сутності зі складною поведінкою: застосування булевих змінних, котрі ще називають мітками, та складних, з багаторівневим вкладенням, операторів розгалуження, в умови істинності котрих входять варіації груп міток. Організація бізнес-логіки програми подібним чином має неякісну структуру, котра призводить до виникнення помилок та проблем з внесенням змін й розумінням системи.

Головна істина створювання програмного забезпечення з використанням автоматів складається з проектування окремої частини умов, що вирішують виконання тих чи інших команд та окремої частини, котра володіє сенсом команд. При подібному автоматному програмуванні бізнес-логіка є жорстко структурованою, що робить її виразною, переконливою та раціональною.

Існує гарна порада при залученні використання автоматів в програмі: наявність сутності зі складною поведінкою свідчить про необхідність застосовувати автомати при розробці програмного забезпечення. Згідно практиці, кожна самодостатня програма має таку властивість. Хоча завдяки тому ж досвіду відомо, що не кожний елемент програми володіє особливістю складної поведінки. Через це є ще одна порада: слід застосовувати автоматне програмування лише до тих елементів програми, що відносяться до сутностей зі складною поведінкою. Ця порада застерігає від перевантаження коду відображенням бізнес-логіки саме в тому місці, де не потребується цього.

Завдяки саме цим порадам з'являється можливість проектувати та реалізовувати дійсно гідне програмне забезпечення, що має на озброєнні складну поведінку. Дуже великою та ресурсно-затратною роботою є, на моменті конструювання, процес розпізнавання сутностей зі складною поведінкою та

продумування їх бізнес-логіки. Людина, що займається цією справою опрацьовує кожну сутність окремо.

2.2 Основні поняття

Термін «стан» є однією з основ програмування з використанням автоматів [17]. Цей термін був створений А.Т'юрингом для парадигми, котра представляється, а також має широке розповсюдження в різноманітних напрямах науки, за приклад можна взяти теорію керування.

Головною особливістю поточного стану програми є те що, він зберігає минулу інформацію програми, котра потрібна для вирішення відповіді згідно вхідному запиту, що створюється в нинішній період часу.

Стан може сприйматись, як незвичайна особливість, котра умовно містить в собі вхідні запити, що надходили до цього, котрі мають вагу перед відповіддю в поточний період часу. Відповідь буде формуватись відповідно вхідному запиту та поточному стану.

Визначення «вхідний запит» теж можна віднести до основ створювання програмного забезпечення з використанням автоматів. Дуже часто під вхідним запитом розуміють множину, елементами якої може бути вхідні аргументи та прецеденти дивлячись як минув процес їх створення та яке значення вони мають. Комбінація скінченного ряду вхідних запитів та скінченного ряду станів створює скінченний автомат без виходів. Автомат даного типу трансформує поточний стан, у вигляді відповіді на вхідний запит. Трансформування станів виконується за певним принципом, котрий має назву «функція переходів автомата».

Програмне забезпечення, може вважатись що має скінченний автомат (див.Рис.8), утворення якого відбувається за рахунок комбінації визначення автомата без виходів з визначенням «вихідний запит». Подібний скінченний автомат створює вихідні елементи у вигляді відповіді на вхідний запит, а не

тільки займається трансформацією стану. Вихідні запити створюються на основі принципу, котрий має назву «функція виходів автомату».

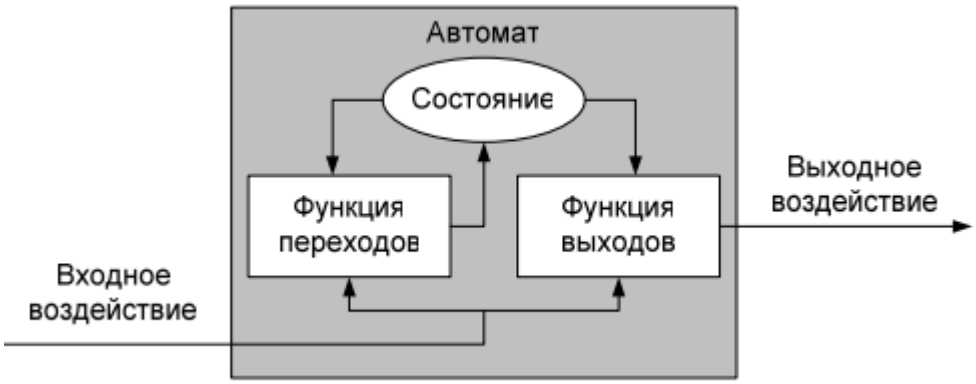

Рис.*8 Скінченний автомат*

2.3 Парадигма автоматного програмування

Спершу потрібно познайомитись з винаходом відомого науковця Т'юринга [17, 20, 21]. Винахід представляє собою обчислювальну машину, котра дуже розповсюджена в моделі під назвою «теорія формальних мов». В 1936 році А. Т'юринг представив світу свою обчислювальну машину, як формальний термін «алгоритм». Пристрій Т'юринга забезпечує всі потрібності, коли необхідно щось «обчислити», «ідентифікувати» та «запрограмувати» це стверджує «теза Черча-Т'юринга» [21].

Певні дві деталі відносяться до винаходу Т'юринга: механізм, що носить обов'язки керування та механізм, котрий зберігає деяку інформацію, найчастіше у вигляді стрічок або рядків (див.Рис.9). Деяка кількість ланків входить до складу стрічки, але тій самій кількості немає меж, там знаходяться знаки певного скінченного алфавіту. Існують можливості механізму керування, а саме виконувати запис та читання знаків, що повинні зберігатись в ланках, це забезпечується за допомогою головки «читання-запис».

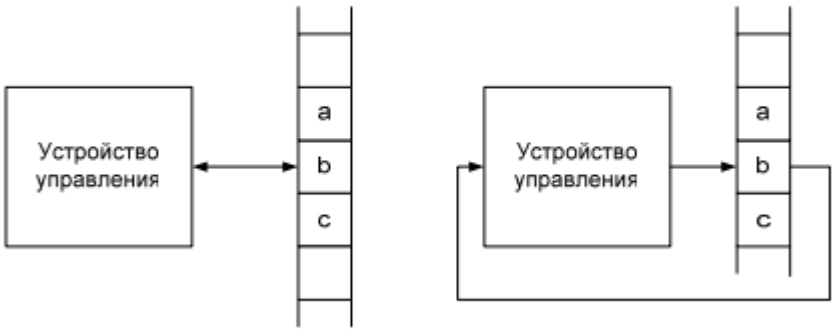

Рис.*9 Стрічка інформації у вигляді знаків*

Механізм з обов'язками керування є скінченним автоматом. Він має знак, що був прочитаний з стрічки та що представляю собою вхідний запит. Також до складу автомата входить знак, котрий необхідно записати на стрічку, а він вже є вихідним запитом, котрий доповнюється змішенням позиції, у вигляді 1 кроку, головки «читання-запис» в необхідному напрямку, також зрушення з місця може не виконуватись.

Будь-яку програму, що була написана той чи іншою мовою програмування, є можливість написати у вигляді, котрий пропонується пристроєм Т'юринга, про це свідчить «теза Черча-Т'юринга».

Є потрібність вбудувати цей процес до механізму, що виконує обов'язки керування та котрий входить до складу пристрою Т'юринга. Тобто потрібно встановити елементи автомату, що представляє керування, а саме: функцію переходів, функцію виходів й множину станів. Це відбувається під керівництвом графа переходів скінченного автомата. Граф представляє собою перелік вершин (станів) та дуг (шляхи між станами, переходи). Дуга свідчить про подію на переході, а саме видачу вихідного запиту, і також про наявність умови, що є вхідним запитом, котра є причиною початку самого переходу.

До функції, котра займається збільшенням аргумента на одиницю, можна представити граф переходів автомата (див.Рис.10), з обов'язками керування, пристрою Т'юринга. До нього маємо позначення: «*» — виконати запис прочитаного символу (знаходиться там, де потрібно виконати запис), а

також «b» — умовне позначення від слову blank. Стрілки свідчать про вказівки голівки «читання-запис», наприклад вказівка «залишитись на місці» буде у вигляді стрілки, що зображено до низу. Зверху риски умова, а знизу дія все це замітка переходу.

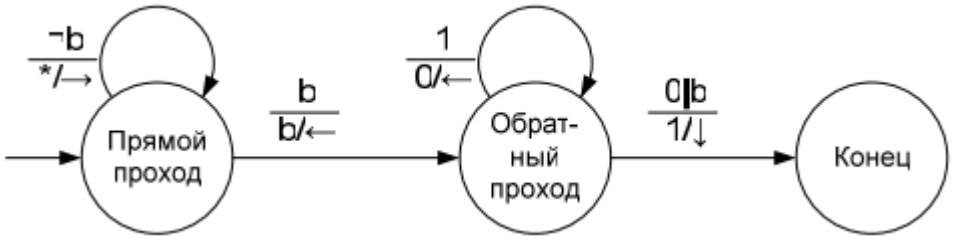

Рис.*10 Граф переходів функції «інкремент»*

Треба зауважити, що є імена, котрі видаються кожному стану автомата у графі переходів. Наданні імена показують саме сенс стану, котрий передає образ дії обчислювального пристрою саме в тому стані, в якому він знаходиться.

В обчислювальній машині Т'юринга бізнес-логіка знаходиться в індивідуальному механізмі, котрий має керуючі обов'язки, тобто в автоматі, тому процес з акцентуванням станів, що керують, є примітивним. Захід такого характеру зазвичай залучають при створенні автоматизованих програм, в котрих приділення уваги до елементів, якими можна керувати та керуючих механізмів є звичайною справою. Тому сутність зі складною поведінкою розбивається на наступні елементи, згідно згаданої теорії зверху:

- Елемент, котрий займається керуванням, вирішенням яким буде наступний стан та несе в собі бізнес-логіку, а саме обрання маніпуляцій, котрі повинні бути здійснені відносно поточного стану та запиту, що поступив.
- Елемент, котрим керують, він слідкує за здійсненням маніпуляцій вибраних елементом керуванням та за створенням деталей вхідного запиту, котрі потрібні для налаштування відповіді.

Елемент, котрим керують ще має назву «об’єкт управління», а елемент, котрий керує має назву «система управління» про це свідчить «теорія управління».

Ще автоматом називають елемент, котрий займається керуванням, тому що при його створенні застосовують автомати.

Завдяки подібному розбиттю сутності, котра має складну поведінку, її легко створювати, а також досить важливо, що присутня можливість без зусиль вносити зміни. Автомат, в обов’язки якого входить керування, володіє бізнес-логікою. «Об’єкт управління» має звичайний образ дій. В його обов’язки не входить аналіз вхідних запитів, що надійшли з оточуючого середовища, він лише повинен виконувати накази автомата по здійсненню необхідних маніпуляцій. Головною особливістю цих наказів є те, що вони не відрізняються один від одного — звичайний образ дій.

«Об’єкт управління» з рисами автоматизації повинен відображати сутність зі складною поведінкою, при використанні автоматів — це все є одним механізмом, котрий має на борту «об’єкт управління» з вбудованою системою управління, про це свідчить «теорія управління».

Парадигма по створенню програмного забезпечення з автоматами вказує, що «об’єкт управління» з рисами автоматизації має представляти сутність зі складною поведінкою.

2.4 Особливості імперативного програмування

Термін стан системи є не тільки при створенні програмного забезпечення з використанням автоматів. Тому що сам процес роботи якоїсь системи, котра має великий обсяг даних і при цьому мається змога вносити до них зміни говорить про існування стану [3]. Стан системи, що застосовує імперативну форму, вміщує в собі [17]:

- Набори аргументів, рівень доступу до яких дозволяє звертатись з будь-якої частини програми та те, що зберігається в оперативній пам’яті.

- Набори даних регістрів, доступ до яких має будь-яка програма.
- Набори даних зі стека.
- Покажчик розташування, на даний момент часу, в коді системи.

Ще елементи стану є можливість описати за допомогою таких понять, як неявних (набори даних зі стеку та покажчик розташування) та явних (вміст аргументів).

Згідно суті обговорених понять програмне забезпечення, що використовує скінченні автомати, відноситься до розробок з імперативними методами. В такій системі елементи з неявним описом мають мізерну вагу. У випадку розгляду системи в періоди, коли виконується наступний рух автомата видно, що стани мають відмінності лише за явними елементами. Цей факт допомагає знаходити особливості системи.

2.5 Зв'язок з об'єктно-орієнтованим програмуванням

Одним з елементів моделі об’єктно-орієнтованого програмування є об’єкт з власним станом, котрий має можливість створювати зв’язок з іншими об’єктами, тобто спілкуватись, а це говорить про здатність приймати сигнали, їх відсилати та під час аналізу модифікувати власний стан [17]. Здатність відсилання сигналу об’єкту можна вважати аналогом здатності подання сигналу на використання методу об’єкта.

Виходячи з цього є можливість сприймати об’єкт, як скінченний автомат, стан якого представляється у вигляді набору власних властивостей, так і рух автомата сприймається у ролі методу об’єкта, може бути декілька, але при цьому повинно виконуватись одне обмеження, потрібно дотримуватись відсутності можливості методами подавати сигнали на використання самих себе, або один одного.

Таким чином за допомогою об’єкта можна якісно створити скінченний автомат. Коли використовується парадигма про побудову програмного забезпечення з скінченними автоматами, мови [1, 2], що підтримують об’єктну

орієнтовану концепцію, подають власні класи у ролі варіантів автомата, власні приватні властивості у ролі стану автомата, а метод класу у ролі руху автомата, треба зауважити, що цей метод є всього одним з неприватним рівнем доступу, в обов’язок якого входить модифікація стану автомата. А решта методів з неприватним рівнем доступу займається прийняттям даних, котрі свідчать о стані автомата, при цьому його не модифікуючи. Весь інший асортимент методів є з закритим рівнем доступу.

РОЗДІЛ 3 РОЗРОБКА ОБРОБНИКА ІНФОРМАЦІЇ

3.1 Аналіз існуючих рішень

В сучасному світі існує дуже велика кількість додатків, котрі займаються обробкою інформації у сфері онлайн-обслуговування людей. Кількість та їх різноманітність дуже вражає, в гарному сенсі цього слова. Наявність альтернативи дає змогу задовольнити кожного користучава програмного забезпечення. Кожна розробка є приватною власністю, плід розумової діяльності, це говорить про те, що майже неможливо дізнатись докладні особливості розробки, а саме використані технології та рішення.

В якості прикладу з існуючих рішень візьмемо службу доставки «Ракета». Це звичайний додаток, головне його призначення отримати складене замовлення та доставити його до замовника. Якщо ми хочемо створити щось подібне, то необхідно проаналізувати весь технологічний склад додатку конкурента, дізнатись як користуватись всім функціоналом та виконати сортування необхідного й зайвого функціоналу. Це все займає дуже велику частину часу та ресурсів.

Формула рішення цієї проблеми є дуже легка, це використання Telegram Bot Api, інтерфейсу на основі HTTP, створеного для розробників, котрі зацікавлені в побудові ботів в месенджері Telegram.

В таблиці 3 представлено критерії ресурсних затрат для розробки додатку з обслуговування людей.

Таблиця *3*

Критерії ресурсних затрат для розробки додатку

	«Ракета»	Telegram-bot
Необхідність великої команди back-end розробників	+	-
Необхідність великої команді front-end розробників.	+	-
Необхідність великої команди тестувальників.	+	-
Необхідність наявності великої Support команди.	+	-
Необхідність наявності стійкого алгоритму шифрування даних.	+	-
Наявність потужних апаратних засобів.	+	-

Завдяки Telegram Bot Api можна заощадити велику кількість ресурсів та часу. Немає потреби в розробці власної back-end частини, що відповідає за отримання та доставку повідомлень. Не потрібно наймати команду front-end частини для розробки інтерфейсу, не витрачається час на коригування ідей з приводу зовнішнього вигляду програми. Відсутня необхідність в піклуванні маскування даних, що передаються. Все це надається зі сторони месенджеру Telegram. Там власна системи потоку даних, котрі мають на озброєнні алгоритми RSA-2048, DH-2048 для авторизації та аутентифікації, для передачі повідомлень використовується шифрування з AES-ключем. Сам месенджер має на борту власний протокол MTProto 2.0, з криптографічним хеш-алгоритмом SHA-256. Все що має Telegram, це має й власний розроблений Telegram-bot.

Завдяки інтерфейсу Telegram Bot Api заощаджується час, кошти, а це найголовніші критерії у світі. З’являється можливість швидко, легко та без проблем організувати власну систему, використовуючи безкоштовний open-source стек технологій, котрий не потребує підтримки з нашої сторони. Достатньо мати обчислювальну машину з наступними характеристиками:

- Процесор з частотою не нижче 1,8 ГГц.
- 4 ГБ оперативної пам’яті.

- Вільного місця на жорсткому диску від 1 ГБ до 40 ГБ.
- Наявність відеоадаптеру з підтримкою 1 ГБ відео-пам’яті.

3.2 Архітектура системи

Архітектура застосунку має вигляд, зображений на рисунку 11.

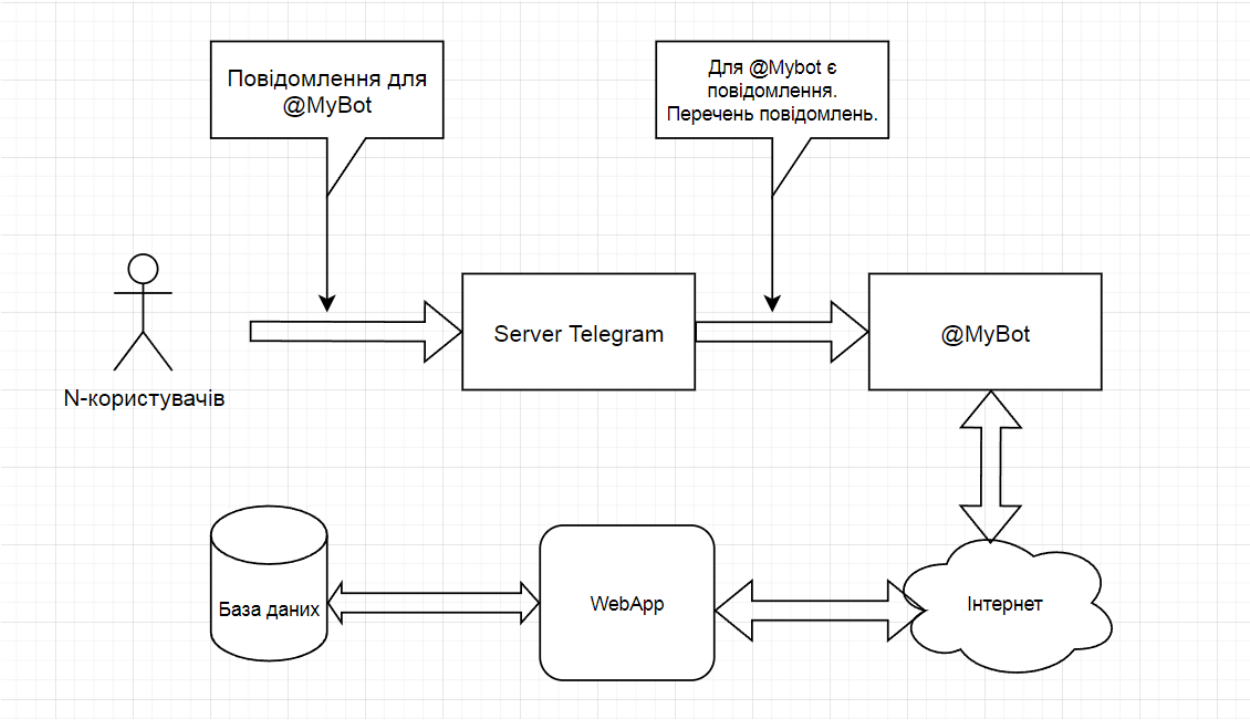

Рис.*11 Архітектура застосунку*

Необмежена кількість користувачів утворює повідомлення для нашого бота, @MyBot. Повідомлення передається через сервер месенджеру Telegram. Сам месенджер повідомляє нас про наявність новоутворених повідомлень та передає їх нашому боту. Через мережу Інтернет ми отримуємо необхідні повідомлення в наш додаток, де відбувається їх аналіз та зберігання потрібних даних в базі даних. Після чого формується відповідь з участю бази даних, котра відправляється назад відправнику повідомлення.

3.3 Засоби реалізації

Програма виконана за допомогою мови програмування C#. Середовищем розробки є Visual Studio 2017. Програма була написана даною мовою програмування через простоту її використання та великий спектр можливостей. Система має вигляд Web-API-застосунку, реалізованого за допомогою ASP.NET Core 2.1. Було застосовано патерн MVC. Додаток має доступ до мережі Інтернет завдяки програмі Ngrok. Базою даних виступає легкий у користуванні SQLite. Для побудови боту для Telegram використовувався інтерфейс на основа HTTP — Bot API.

3.3.1 Visual Studio

Visual Studio 2017 — це продукт, розроблений компанією Microsoft [22,23]. Представляє собою інтегроване середовище розробки програмного забезпечення з великою кількістю інструментів. Дане програмне забезпечення надає змогу створювати консольні застосунки, веб-застосунки, веб-служби, веб-сайти, виконувати розробки з графічним інтерфейсом, Windows Forms, а також ігри. Має підтримку багатьох платформ: Windows CE, Windows Phone, Windows, Silverlight, Windows Mobile, .NET Compact Framework, Xbox, .NET Framework.

Розробка від компанії Microsoft має на озброєнні редактор початкового коду, котрий підтримує IntelliSense та забезпечує рефакторінг коду. Відладник, котрий вбудовано, працює на рівні початкового коду та на рівні машинного коду. Велика кількість інструментів складається з редактора форм, котрий полегшує роботу по створенню графічного інтерфейсу додатку, проектувальника класів, проектувальника схем баз даних та веб-редактора. Мається можливість використовувати сторонні бібліотеки, що збільшують стандартний арсенал функцій та інструментів.

3.3.2 Web API

Web Api представляє собою інтерфейс програмування прикладного рівня, призначений для веб-сервера та веб-браузера [24]. Система обмежена частиною веб-додатку, а саме стороною клієнта, через це подробиці реалізації веб-серверу не входять до складу.

Web API зі серверної сторони є інтерфейсом, до складу якого входить декілька кінцевих точок або одна, котра відчуває на собі запит-відповідь системи, що займається транслюванням повідомлень, має вигляд XML-, JSON-файлів. HTTP веб-серверів впливають на ці самі файли. Представниками серверних Web API є мешапи та веб-хуки. Коли поєднуються в одному місці декілька серверних Web API, а саме в веб-додатку, утворюються так звані мешапи. Коли за допомогою зворотних викликів виконуються зміни в поведінці веб-додатку — утворюються веб-хуки.

Кінцеві точки відповідають за знаходження ресурсів, до яких стороннє програмне забезпечення отримує доступ.

Web API зі сторони клієнта є інтерфейсом, що збільшує функціональність веб-браузера чи клієнта HTTP.

3.3.3 ASP.NET Core

ASP.NET Core — фреймворк кроссплатформеного open-source характеру, ціль котрого створення веб-додатків, на базі .Net [4]. Компанія Microsoft займається розробкою даної платформи. На відміну з ASP.NET мається збільшена продуктивність. Структура складається з модулів, все необхідне можна завантажити через Nuget. Мається можливість працювати на MacOs, Windows, Linux.

В самому додатку є можливість розгорнути сайт на ASP.NET Core. Ця можливість надається за допомогою веб-серверу Kestrel, але не потрібно забувати про вбудований IIS. Завдяки цьому можна виконувати налаштування з гарною гнучкістю.

Вся функціональність MVC, Web API, Web Pages знаходиться у фреймворку MVC, що в свою чергу підтримується ASP.NET Core.

3.3.4 MVC

MVC (Model-View-Controller) — патерн, котрий став потужним поштовхом в розвитку новітніх технологій, з різноманітними природними походженнями [25]. Цей патерн був створений на основі необхідності пошуку рішення, яке б відділило графічну частину додатку від бізнес логіки, а базу даних від тієї ж бізнес логіки.

MVC складається з 3 частин: модель, представлення та контролер (див.Рис.12).

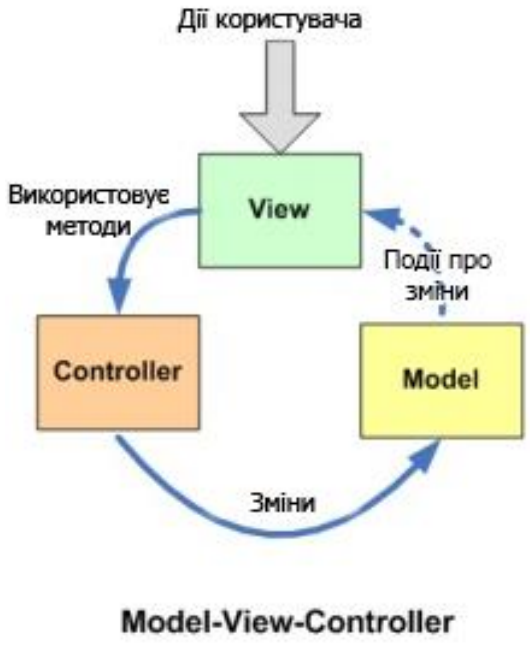

Рис.*12 Модель MVC*

Модель відповідальна за бізнес-логіку застосунку, за те, як мислить додаток, ще її можна назвати протоколом функціональності. Модель повинна мати повну незалежність від інших елементів програми. Моделі немає ніякого діла до графічної частини продукту, до того, як там все відображується. Відбувається лише показ даних, в той же час не торкаючись самої моделі.

Мається наступна характеристика моделі:

- Центр бізнес-логіки.
- Знає лише про власне існування.
- Інколи виступає лише у ролі бази даних.
- Може бути керуючим базою даних, з деяким переліком об’єктів.

На представлення лягає відповідальність за відображення даних, котрі поступають від моделі. Найголовніше, треба пам’ятати, що представлення ніяк не в змозі вплинути на модель безпосередньо.

Характеристичні риси представлення:

- Отримання даних від моделі та їх відображення.
- Прийняття запитів від користувача.

Контролер, він є постачальною системою, з можливістю аналізу, адже несе відповідальність за зв’язок користувача з системою, реалізується за допомогою використання моделі та представлення.

Риси контролера:

- Керує відображенням різноманітності представлення в необхідний час.
- Отримує сигнали від представлення, впливає на модель, вирішує, яке саме представлення відображати.
- Немає залежності від кількості представлень.

Головна ідея цього патерну — це розбити повноваження системи на частини, котрі разом утворюють єдиний організм.

Користувач взаємодіє з інтерфейсом додатку (представленням), дія користувача формує подію, котра передається до контролера, він, у свою чергу, аналізує подію та вирішує наступні дії системи за допомогою моделі, котра викликає необхідний метод, згідно закладеної в неї бізнес-логіки.

3.3.5 Bot-API

Bot-API — інтерфейс для розробників, бажаючих створювати ботів в месенджері Telegram, на базі HTTP [5]. Дана технологія надає змогу отрима-

ти доступ до системи Telegram. Боти в застосунку Telegram мають специфічний стан положення, котрим не потрібен номер телефону. Ці облікові записи мають вигляд інтерфейсу для коду, що виконується на власному сервері.

При користуванні даним засобом не потрібно піклуватись та розбиратись, як саме працюють їх алгоритми шифрування, протокол MTProto. Сервер-буфер від Telegram бере на себе всі обов'язки з цього приводу. Спілкування відбувається по захищеному протоколу HTTPS.

3.3.6 Ngrok

Ngrok — додаток на безкоштовній основі [26]. Надається змога утворювати тунелі на локальній обчислювальній машині користувача. Відбувається виділення публічної адреси, до якої прив'язується локальний порт. Завдяки цьому відкривається доступ до локального додатку через публічну адресу. Це дуже легкий та зручний спосіб тестування власної розробки. Тепер не потрібно витрачати час на пошук хосту та кошти на його аренду.

Тунель — протокол, відповідальний за зв'язок, в обов'язок якого входить передача даних з однієї мережі в іншу. Це забезпечується передачею пакетів повідомлення з локальної мережі до місця призначення через глобальну мережу Інтернет. Сам процес передачі пакетів включає в себе переробку цих пакетів з метою їх шифрування, що забезпечує захист даних від змін, або зчитування. Тунель використовує модель OSI та TCP / IP, вибір залежить від навантаження на систему.

3.3.7 SQLite

SQLite — представляє собою бібліотеку, що створює ядро бази даних, котре є самостійним, без необхідності сервера SQL [27]. Сама розробка є вільним програмним забезпеченням, з відкритим кодом, завдяки чому її можливо використовувати абсолютно безкоштовно з метою реалізації будь-якої системи. На базі SQLite було виконано безліч відомих проєктів, наприклад: Adope, Airbus, Apple, Bosch, Bentley Systems. Facebook, Dropbox та інші.

SQLite використовує конструкцію бази даних SQL, що вбудовано в сам додаток. SQLite не потребує власного серверу, на відміну від інших баз даних. Додаток працює лише з одним файлом на жорсткому диску, де й знаходиться вся база даних, куди входять таблиці, функції, процедури, тригери та інше. Формат файлу підтримує 32-бітні та 64-бітні системи, тобто забезпечується кроссплатформеність. SQLite є настільки компактною, що при використанні всього функціоналу буде займати менше 600 КБ.

Переваги SQLite:

- Простота.
- Відсутня необхідність в налаштуванні серверу СУБД.
- Ідеально комфортно використовувати у власному продукті.
- Безкоштовна ліцензія.
- Підтримка безлічі платформ.
- Гарна швидкість роботи.
- Підтримка великого функціоналу.
- Підтримка великої кількості мов програмування.
- Централізованість.
- Безпечність.

3.4 Функціональні можливості

Забезпечення можливостей є невід’ємною частиною при розробці програмного забезпечення.

До складу функціональних можливостей користувача входить (див.Рис.13):

1) звернення до боту;
2) створення замовлення;
3) перегляд існуючих замовлень;
4) видалення замовлення.

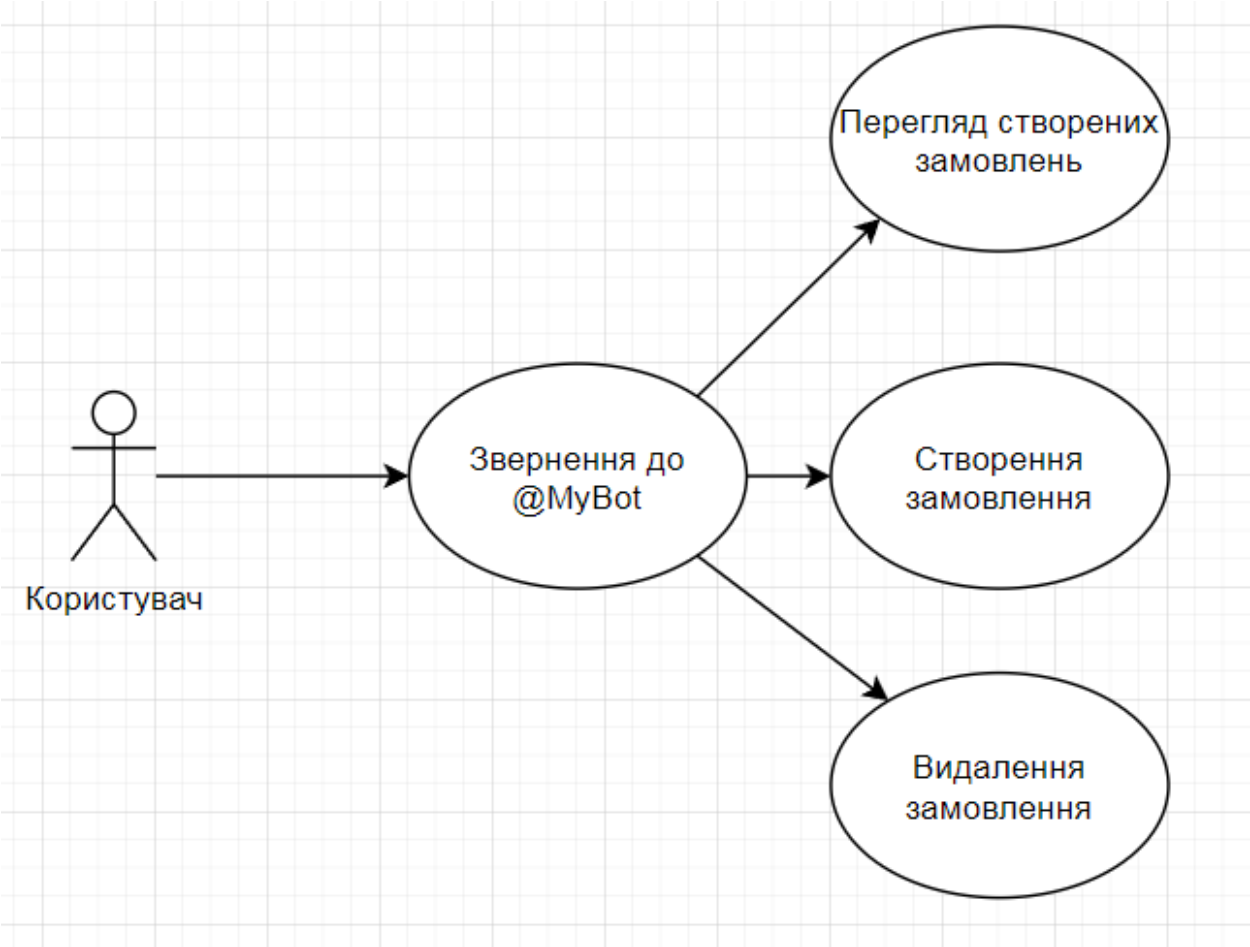

Рис.*13 Загалний перелік можливостей*

Перший варіант використання боту надає можливість звернутись до боту за необхідністю створення замовлення (див.Рис.14). Початок створення замовлення починається з вибору типу товару, а саме піцци. Існує на даний момент часу 3 типи продукту: мексиканська, гавайська та маргарита. Після визначення з типом піцци слідує вибір розміру. До розміру входить : 25 см, 35см, 45см. Коли, нарешті, користувач визначився з розміром, він відправляє підтвердження з приводу складеного замовлення. Під час будь-якого кроку формування замовлення є можливість подати команду на відмову від утворення замовлення.

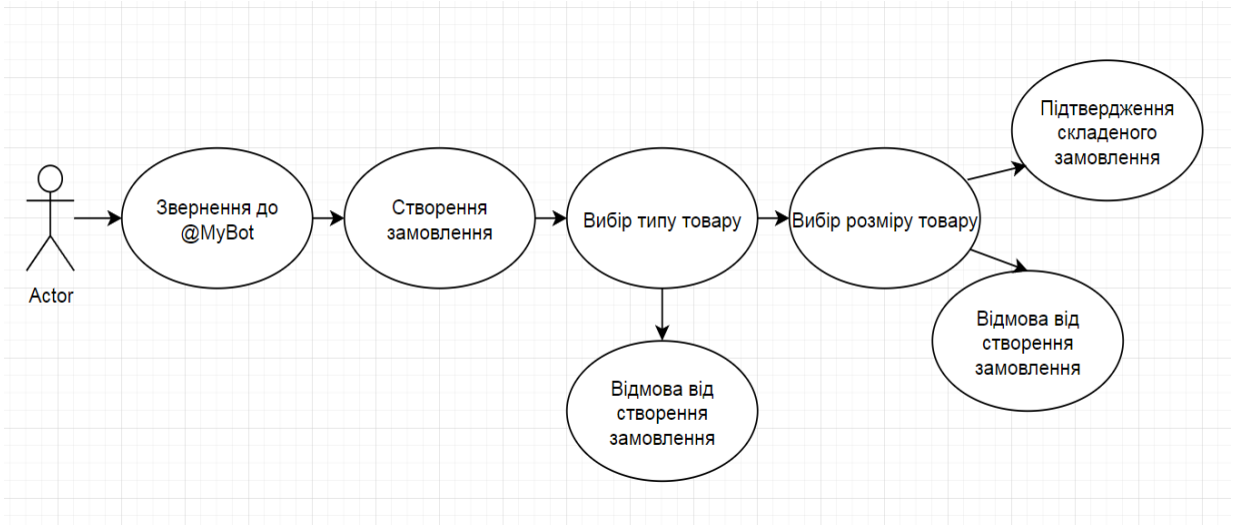

Рис.*14 Створення замовлення користувачем*

Другим варіантом використання боту є перегляд створених замовлень відповідним користувачем (див.Рис.15). Користувачеві потрібно звернутись до боту та подати команду перевірки створених замовлень. На отриману команду бот відправляє перелік існуючих замовлень.

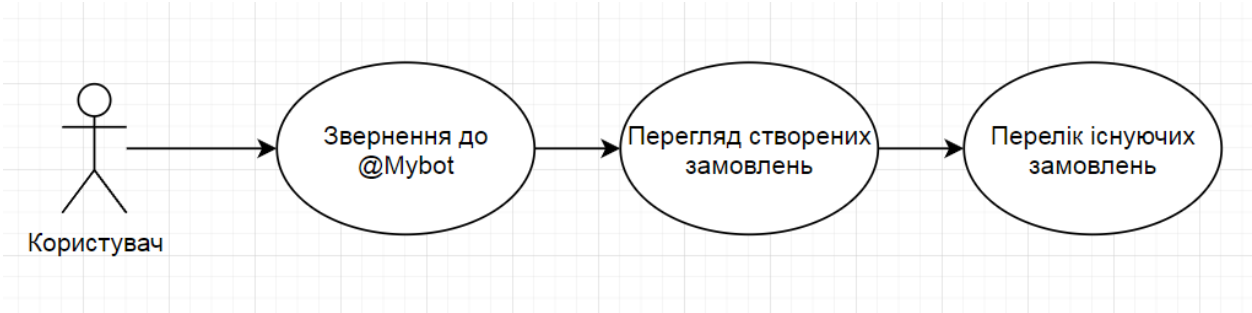

Рис.*15 Перегляд існуючих замовлень*

Також існує можливість видалення створеного замовлення (див. Рис. 16). Спочатку користувачеві потрібно дізнатись номер існуючого замовлення, попередньо викликавши перегляд замовлень. Після чого звернутись до бота з командою видалення, та передати номер замовлення, що потребує видалення.

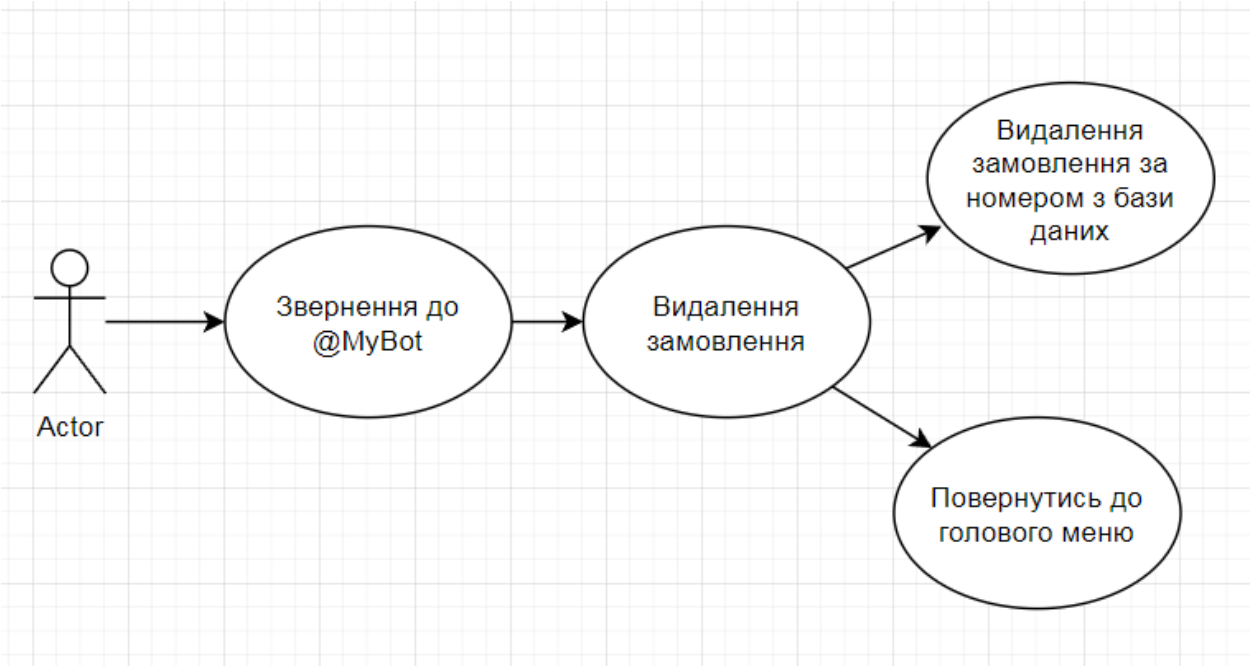

Рис.*16 Видалення замовлення зі списку*

3.5 Методи і алгоритми

3.5.1 Основні принципи

Система побудована на базі патерну MVC (Model-View-Controller).

Модель відповідальна за бізнес-логіку програми. Вона повністю незалежна від інших частин застосунку. Модель навіть не усвідомлює наявності існування інших елементів додатку.

Представлення займається відображенням даних, отриманих від моделі, а також прийняттям подій, виконаних користувачем, та передачею їх контролеру.

Контролер займається налаштуванням зв’язку між системою та користувачем. Він доставляє дані до користувача від системи та навпаки. Контролер приймає дані від користувача, аналізує їх, після чого вирішує якими будуть наступні дії за рахунок передачі запиту моделі, де відбуваються необхідні дії згідно прописаній логіці, та як результат після обробки модель утворює подію про свою зміну, представлення виконує запит на нові дані до моделі та видає їх користувачеві.

Кожна частина додатка має свої повноваження та обов’язки.

Система складається з 19 класів.

Першим кроком в роботі програми є реєстрація усіх необхідних служб, за це відповідає клас «Startup» (Лістинг 3).

Лістинг *3 Вміст класу «Startup»*

```
public class Startup
{
public Startup(IConfiguration configuration)
{
Configuration = configuration;
}
public IConfiguration Configuration { get; }
// This method gets called by the runtime. Use this
method to add services to the container.
public void ConfigureServices(IServiceCollection
services)
{services.AddMvc().SetCompatibilityVersion(Compatibilit
yVersion.Version_2_1);}
// This method gets called by the runtime. Use this
method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app,
IHostingEnvironment env)
{
if (env.IsDevelopment())
{
app.UseDeveloperExceptionPage();
}
else
{
app.UseHsts();
}
```

```
app.UseHttpsRedirection();
app.UseMvc();
Bot.GetBotClientAsync().Wait();}}
```

Наступним кроком є реєстрація команд бота у системі, ініціалізація даних та налаштування зв'язку з сервісом Telegram (Лістинг 4). З'єднання виконується за рахунок вказування токену, який належить нашому боту зареєстрованому в сервісі Telegram. Також необхідно провести налаштування Webhook, де потрібно вказати Url адресу розміщення нашого додатку в глобальній мережі Інтернет. Url адреса розміщення, а також токен знаходяться в класі «AppSettings» (Лістинг 5).

Лістинг *4 Вміст класу «Bot»*

```
public class Bot
    {
        private static TelegramBotClient botClient;
        private static List<Command> commandsList;
        public static IReadOnlyList<Command> Commands
=> commandsList.AsReadOnly();
        public static async Task<TelegramBotClient>
GetBotClientAsync()
        {
            if (botClient != null)
            {
                return botClient;
            }
            WorkWithFiles.InitializeAll();
            DB.GetAllTypePizza();
            DB.GetAllSizePizza();
            commandsList = new List<Command>();
            commandsList.Add(new StartCommand());
            commandsList.Add(new CheckCommand());
```

```
                commandsList.Add(new CancelCommand());
                commandsList.Add(new FinishCommand());
                commandsList.Add(new CreateCommand());
                commandsList.Add(new TypeCommand());
                commandsList.Add(new SizeCommand());
                commandsList.Add(new ContinueCommand());
                commandsList.Add(new AnotherCommand());
                //TODO: Add more commands
                botClient = new
TelegramBotClient(AppSettings.Key);
                string hook =
string.Format(AppSettings.Url, "api/message/update");
                await botClient.SetWebhookAsync(hook);
                return botClient;
            }
        }
```

Лістинг 5 *Вміст класу «AppSettings»*

```
    public class AppSettings
    {
    public static string Url { get; set; } = "https://4789-
77-222-156-46.ngrok.io/{0}";
    public static string Name { get; set; } =
"FoodEasybot";
    public static string Key { get; set; } =
"1708898419:AAHptyxEVHS62rLwOZISIH7DKLa2dxMiJYI";
    public static string ConnectionInf { get; set; } =
"Server=tcp:pizzamax.database.windows.net,1433;Initial
Catalog=PizzaDB;Persist Security Info=False;User
ID=myadmin;Password=Best_Pizza01;MultipleActiveResultSets=F
alse;Encrypt=True;TrustServerCertificate=False;Connection
Timeout=30;";
    }
```

Веб-перехоплювач (Webhook) — це «визначені користувачем зворотні виклики HTTP». Зазвичай вони запускаються якою-небудь подією, наприклад, репозиторій отримав код, або блог був підкріплений коментарем, в нашому випадку це відбувається коли користувач відправляє повідомлення нашому Telegram боту.

Глобальна Url адреса розміщення нашого додатку формується за допомогою програми Ngrok. Утворюється тунель на локальній машині обчислювання. Потребується тунель між локальним сервером та Інтернетом, щоб сервера Telegram мали змогу відправляти нам повідомлення. В Ngrok передається адреса локального застосунку, він у свою чергу видає потрібний глобальний URL (див.Рис.17,18).

Новоутворену URL потрібно занести до класу «AppSettings», після чого бот становиться повністю функціональним.

Коли на вхід поступає повідомлення, то представлення від месенджеру Telegram формує подію, класс «MessageController» перехоплює цю подію, та починає аналіз отриманих даних (Лістинг 6).

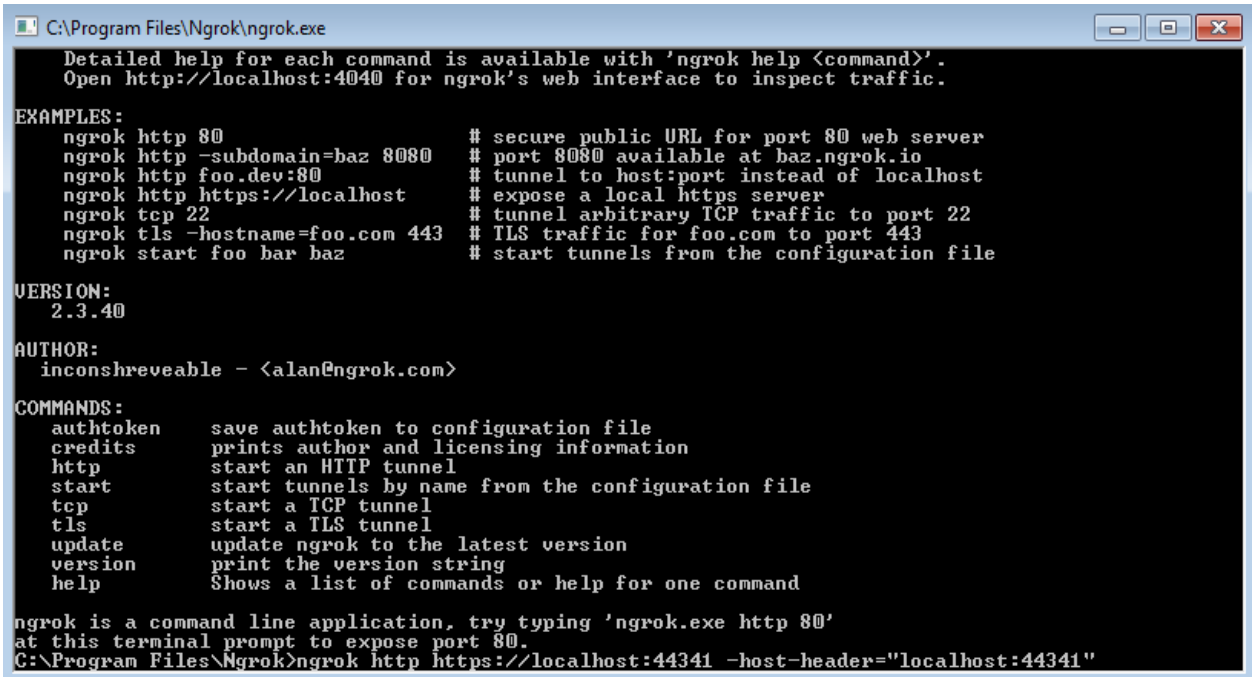

Рис.*17 Команда резервація глобальної адреси*

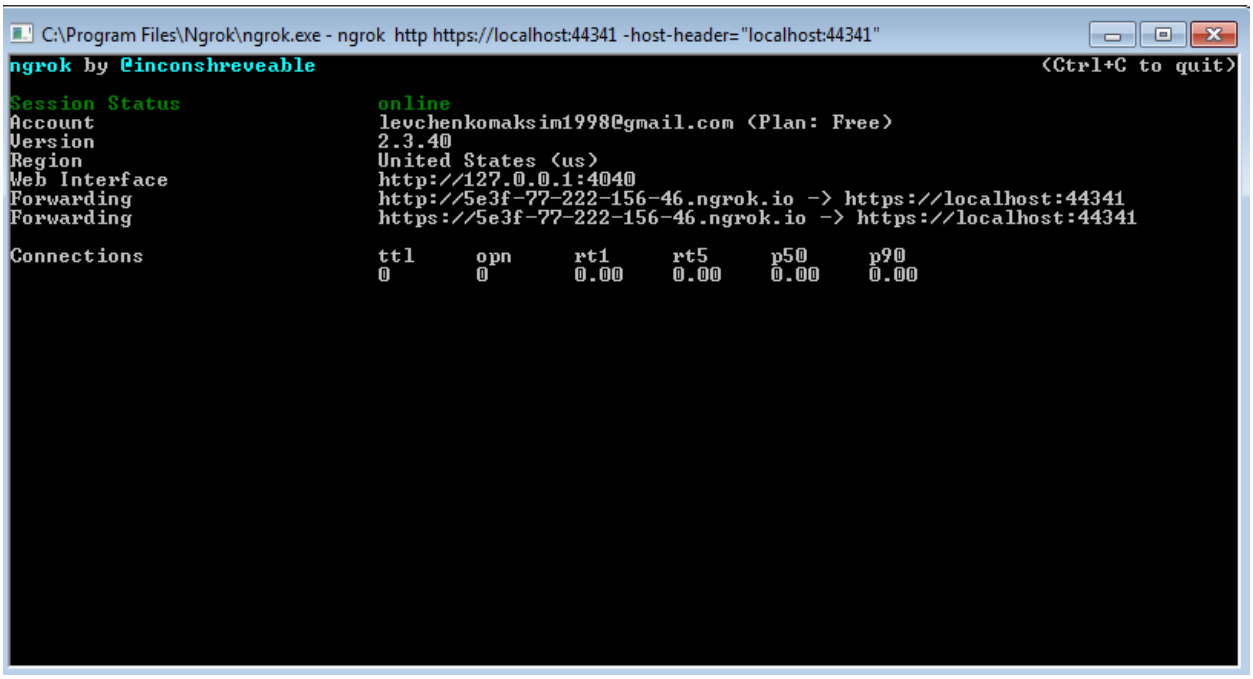

Рис.*18 Отримана глобальна адреса*

Лістинг *6 Вміст класу «MessageController»*

```
public class MessageController : ControllerBase
{
// GET api/values
[HttpGet]
public string Get()
{
return "Method GET unuvalable";
}
[HttpPost]
public async Task<OkResult> Post([FromBody]Update
update)
{
if (update == null) return Ok();
var commands = Bot.Commands;
var message = update.Message;
var botClient = await Bot.GetBotClientAsync();
string line = message.Text;
string com = Avtomat.Type_vhod(line);
```

```
foreach (var command in commands)
{
if (command.Contains(com))
{
await command.Execute(message, botClient);
break;}}return Ok();}}
```

Аналіз отриманих даних відбувається за допомогою класу «Avtomat». Клас «Avtomat» є одніє з вагомих частин бізнес-логіки програми. Аналіз відбувається за допомогою транслітератора (Лістинг 7), відбувається перетворення вхідного повідомлення, щоб скоротити його та зробити зрозумілим для скінченного автомата.

Лістинг *7 Вміст класу «Avtomat»*

```
public static class Avtomat
{
public static string[] vhod_alf;
public static string[] sost_alf;
public static string[,] matrix_sost;
public static string Check_sost(string text, string
current_sost)
{
string next_sost = "";
string current_vhod = Type_vhod(text);
int indexx_sost = Index_sost(current_sost);
int indexx_vhod = Index_vhod(current_vhod);
next_sost = matrix_sost[indexx_sost, indexx_vhod];
return next_sost;
}
public static string Type_vhod(string text)
{
string line = "";
```

```
if(text=="/create")
{
line = text;
return line;
}
else
{
if(Info.list_type_pizza.Contains(text))
{
line = "/type";
return line;
}
else
{
if(Info.list_size_pizza.Contains(text))
{
line = "/size";
return line;
}
else
{
if(text== "/continue")
{
line = "/continue";
return line;
}
else
{
if(text== "/cancel")
{
line = "/cancel";
return line;
}
```

```
else
{
if(text == "/finish")
{
line = "/finish";
return line;
}
else
{
if(text == "/start")
{
line = "/start";
return line;
}
else
{
if(text == "/check")
{
line= /check";
return line;
}
else
{
line= "/another";
return line;}}}}}}}}}
private static int Index_sost(string current_sost)
{
int index = 0;
for(int i=0;i<sost_alf.Length;i++)
{
if(sost_alf[i]==current_sost)
{
index = i;
```

```
return index;
}}
return index;
}
private static int Index_vhod(string current_vhod)
{
int index = 0;
for(int i = 0;i<vhod_alf.Length;i++)
{
if(vhod_alf[i]==current_vhod)
{
index = i;
return index;
}}return index;}
public static void Initialize_vhod_alf(int n)
{
vhod_alf = new string[n];
}
public static void Initialize_sost_alf(int n)
{
sost_alf = new string[n];
}
public static void Initialize_matrix_sost(int n, int m)
{
matrix_sost = new string[n, m];}}
```

У ролі скінченного автомата виступає автомат Мура. Він вирішує правильність дій користувача, якщо було допущено помилку, про це повідомляється та вказується який правильний наступний крок роботи бота.

Після аналізу вхідних даних контролер використовує один з методів моделі для зміни даних та визначення правильного наступного представлення. Контролер викликає методи з похідних класів від абстрактного класу

«Command». Цей клас спадкується наступними класами: «AnotherCommand», «CancelCommand», «CheckCommand», «ContinueCommand», «CreateCommand», «FinishCommand», «SizeCommand», «StartCommand», «TypeCommand», всі вони знаходяться в розпорядженні контролера. Викликається відповідний метод отриманої команди.

На першому кроці користувач повинен викликати команду «/start». Спрацює відповідний метод «Exucute» класу «StartCommand» (Лістинг 8).

Лістинг *8 Метод реакції на команду «/start»*

```
    public override async Task Execute(Message message,
TelegramBotClient botClient)
    {
    int chatId = Convert.ToInt32(message.Chat.Id);
    string LName = Convert.ToString(message.Chat.LastName);
    string FName =
Convert.ToString(message.Chat.FirstName);
    string text = Convert.ToString(message.Text);
    string current_order = "";
    int Customer_idd = DB.createOrReturnCustomer(chatId,
LName, FName);
    current_order = Info.CurrentSost(chatId,Customer_idd);
    string next_order = Avtomat.Check_sost(text,
current_order);
    if(next_order == "s1" && current_order == "s0")
    {
    int id_new_order = Info.GetLastIdAtlibrary();
    id_new_order++;
    Info.CreateAtLibrary(id_new_order, chatId,
Customer_idd, "start", next_order, "", "");
    string line = "Вижу ты хочешь заказать пиццу. \r\n
Держи инструкцию: \r\n /create — начать построение заказа
```

```
\r\n /check — проверить твои текущие заказы \r\n /cancel — отменить выбранный заказ";
    await botClient.SendTextMessageAsync(chatId, line, replyMarkup: PersonalButtons.GetButtonsFirstStep());
    }
    else
    {
    if(next_order == "er" && current_order == "s1")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Вижу ты хочешь заказать пиццу. \r\n Держи инструкцию: \r\n /create — начать построение заказа \r\n /check — проверить твои текущие заказы \r\n /cancel — отменить выбранный заказ";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine, replyMarkup: PersonalButtons.GetButtonsFirstStep());
    }
    else
    {
    if(next_order == "er" && current_order == "s2")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Какого типа пиццу вы желаете ? \r\nМексиканская \r\nГавайская \r\nМаргарита";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine, replyMarkup: PersonalButtons.GetButtonsCreate());
    }
    else
    {
    if(next_order == "er" && current_order == "s3")
    {
```

```
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Какого размера пиццу желаете ?
\r\n25см. \r\n35см. \r\n45см.";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsSize());
    }
    else
    {
    if(next_order == "er" && current_order == "s4")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Желаете еще что-то добавить в заказ
?\r\n /cancel — отмена всего заказа \r\n /continue —
добавить в заказ еще что-то \r\n /finish — подвердить ";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsFinish());
    }
    else
    {
    if(next_order == "er" && current_order == "s5")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Отправте номер заказа, который желаете
удалить\r\n";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsCancel());}}}}}}}}
```

Після чого виконається запис користувача до бази даних (БД). База даних складається з 4 таблиць: «Customer», «Order», «Pizza», «OrdersPizzas».

Виконується запис нового користувача до таблиці «Customer» використовуючи метод «createOrReturnCustomer» класу «DB» (Лістинг 9).

Лістинг *9 Метод внесення особистих даних користувача до БД*

```
public static int createOrReturnCustomer(int Chat_id,
string LastName, string FirstName)
{
int number = 0;
using (SQLiteConnection conn = new
SQLiteConnection(@"Data
Source=F:\TelegramBot2017\TelegrammAspMvcDotNetCoreBot\Tele
grammAspMvcDotNetCoreBot\PizzaDB.db;Version=3;"))
{
conn.Open();
SQLiteCommand command = new SQLiteCommand();
command.Connection = conn;
command.CommandText = @"
SELECT Customer_id FROM Customer
WHERE Chat_id=@chatidd AND Last_name=@lastnamme AND
First_name=@firtsnamme;";
command.Parameters.AddWithValue("@chatidd", Chat_id);
command.Parameters.AddWithValue("@lastnamme",
LastName);
command.Parameters.AddWithValue("@firtsnamme",
FirstName);
using (SQLiteDataReader reader =
command.ExecuteReader())
{
if (reader.HasRows)
{
while (reader.Read())
{
```

```
    number = reader.GetInt32(0);
    break;
    }}
    else
    {
    SQLiteCommand command2 = new SQLiteCommand();
    command2.Connection = conn;
    command2.CommandText = @"
    INSERT INTO Customer (Chat_id, Last_name, First_name)
    VALUES (@Chat, @LName, @FName);";
command2.Parameters.AddWithValue("@Chat", Chat_id);
command2.Parameters.AddWithValue("@LName", LastName);
command2.Parameters.AddWithValue("@FName", FirstName);
    command2.ExecuteNonQuery();
    }}
    using (SQLiteDataReader reader2 =
command.ExecuteReader())
    {
    while (reader2.Read())
    {
    number = reader2.GetInt32(0);
    break;
    }}conn.Close(); }return number;}
```

Виконавши запис до бази даних наступним необхідно зберегти у буфері проміжні значення в класі «Info» «CreateAtLibrary» (Лістинг 10). Проміжні значення поступово змінюються під час формування замовлення (під час дій користувача).

Лістинг *10 Метод по створенню проміжних значень користувача*

```
    public static void CreateAtLibrary(int Id, int
Chat_idd, int Customer_idd, string Status_order, string
Step_order, string Type_pizza, string Size_pizza)
    {
    Tuple<int, int, string, string, string, string> buff =
new Tuple<int, int, string, string, string,
string>(Chat_idd, Customer_idd, Status_order, Step_order,
Type_pizza, Size_pizza);
    library.Add(Id, buff);}
```

Далі від користувача повинна поступити команда «/create» пов'язана з початком формування замовлення. Користувачеві видається список з пропозицією вибрати тип піцци. Потім відповідно команді викликається метод «Execute» класу «CreateCommand» (Лістинг 11).

Лістинг *11 Метод реакції на команду «/create»*

```
    public override async Task Execute(Message message,
TelegramBotClient botClient)
    {
    int chatId = Convert.ToInt32(message.Chat.Id);
    string LName = Convert.ToString(message.Chat.LastName);
    string FName =
Convert.ToString(message.Chat.FirstName);
    string text = Convert.ToString(message.Text);
    string current_order = "";
    int Customer_idd =
DB.createOrReturnCustomer(chatId,LName,FName);
    current_order = Info.CurrentSost(chatId,Customer_idd);
    string next_order =
Avtomat.Check_sost(text,current_order);
    if (next_order == "s2" && current_order == "s1")
```

```
    {
    int Idd = Info.GetIdCurrentOrder(chatId, Customer_idd);
    Info.ChangeAtLibrary(Idd,chatId, Customer_idd,
"create", next_order, "", "");
    string line = "Какого типа пиццу вы желаете ?
\r\nМексиканская \r\nГавайская \r\nМаргарита";
    await botClient.SendTextMessageAsync(chatId, line,
replyMarkup: PersonalButtons.GetButtonsCreate());
    }
    else
    {
    if(next_order=="er" && current_order == "s2")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Какого типа пиццу вы желаете ?
\r\nМексиканская \r\nГавайская \r\nМаргарита";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsCreate());
    }
    else
    {
    if(next_order=="er" && current_order == "s3")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Какого размера пиццу желаете ?
\r\n25см. \r\n35см. \r\n45см.";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsSize());
    }
    else
    {
```

```
    if(next_order =="er" && current_order == "s4")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Желаете еще что-то добавить в заказ
?\r\n /cancel — отмена всего заказа \r\n /continue —
добавить в заказ еще что-то \r\n /finish — подвердить ";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsFinish());
    }
    else
    {
    if(next_order == "er" && current_order == "s0")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Добро пожаловать, " + FName + " " +
LName +"\r\nЧтобы приступить, жмякай /start";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsStart());
    }
    else
    {
    if(next_order == "er" && current_order == "s5")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Отправте номер заказа, который желаете
удалить\r\n";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsCancel());}}}}}}
```

Слідом йде оновлення проміжних даних замовлення, поточного користувача (Лістинг 12).

Лістинг *12 Метод по оновленню проміжних даних замовлення*

```
public static void ChangeAtLibrary(int Id, int
Chat_idd,int Customer_idd,string Status_order,string
Step_order,string Type_pizza,string Size_pizza)
{
if(library.ContainsKey(Id))
{
Tuple<int, int, string, string, string, string> buff =
new Tuple<int, int, string, string, string,
string>(Chat_idd, Customer_idd, Status_order, Step_order,
Type_pizza, Size_pizza);
library[Id] = buff;}}
```

Згідно вибраним типом піцци користувач надсилає команду «/type», котра відноситься до класу «TypeCommand» та викликає метод «Execute»(Лістинг 13). Згідно нової команди оновлюються проміжні дані та користувачеві потрібно визначитись с запропонованим розміром пицци.

Лістинг *13 Метод реакції на команду «/type»*

```
public override async Task Execute(Message message,
TelegramBotClient botClient)
{
int chatId = Convert.ToInt32(message.Chat.Id);
string LName = Convert.ToString(message.Chat.LastName);
string FName =
Convert.ToString(message.Chat.FirstName);
string text = Convert.ToString(message.Text);
string current_order = "";
```

```
    int Customer_idd = DB.createOrReturnCustomer(chatId,
LName, FName);
    current_order = Info.CurrentSost(chatId, Customer_idd);
    string next_order = Avtomat.Check_sost(text,
current_order);
    if(next_order == "s3" && current_order=="s2")
    {
    int Idd = Info.GetIdCurrentOrder(chatId, Customer_idd);
    Info.ChangeAtLibrary(Idd, chatId, Customer_idd,
"create", next_order, text, "");
    string line = "Какого размера пиццу желаете ? \r\n25см.
\r\n35см. \r\n45см.";
    await botClient.SendTextMessageAsync(chatId, line,
replyMarkup: PersonalButtons.GetButtonsSize());
    }
    else
    {
    if(next_order=="er" && current_order=="s3")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Какого размера пиццу желаете ?
\r\n25см. \r\n35см. \r\n45см.";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsSize());
    }
    else
    {
    if(next_order=="er" && current_order=="s4")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
```

```
     string line2 = "Желаете еще что-то добавить в заказ
?\r\n /cancel — отмена всего заказа \r\n /continue —
добавить в заказ еще что-то \r\n /finish — подвердить ";
     string allLine = line1 + "\r\n" + line2;
     await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsFinish());
    }
    else
    {
    if(next_order=="er" && current_order=="s5")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Отправте номер заказа, который желаете
удалить\r\n";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsCancel());
    }
    else
    {
    if(next_order== "er" && current_order=="s0")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Добро пожаловать, " + FName + " " +
LName + "\r\nЧтобы приступить, жмякай /start";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsStart());
    }
    else
    {
    if (next_order == "er" && current_order == "s1")
                                        {
```

```
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Вижу ты хочешь заказать пиццу. \r\n
Держи инструкцию: \r\n /create — начать построение заказа
\r\n /check — проверить твои текущие заказы \r\n /cancel —
отменить выбранный заказ";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsFirstStep());}}}}}
}}
```

Після визначення з розміром піцци від користувача поступає команда «/size» класу «SizeCommand», через що на виконання подається метод «Execute» (Лістинг 14) та користувачеві потрібно підтвердити створене замовлення.

Лістинг *14 Метод реакції на команду «/size»*

```
    public override async Task Execute(Message message,
TelegramBotClient botClient)
    {
    int chatId = Convert.ToInt32(message.Chat.Id);
    string LName = Convert.ToString(message.Chat.LastName);
    string FName =
Convert.ToString(message.Chat.FirstName);
    string text = Convert.ToString(message.Text);
    string current_order = "";
    int Customer_idd = DB.createOrReturnCustomer(chatId,
LName, FName);
    current_order = Info.CurrentSost(chatId, Customer_idd);
    string next_order = Avtomat.Check_sost(text,
current_order);
    if(next_order == "s4" && current_order =="s3")
    {
```

```
    int Idd = Info.GetIdCurrentOrder(chatId, Customer_idd);
    int numberrrr =
DB.ReturnPizzaId(Info.library[Idd].Item5, text);
    if(numberrrr == 0)
    {
    string line = "Приносим наши извенения. \r\n Выбранный
вами размер недоступен \r\n на данный момент времени. \r\n
Какого размера пиццу вы желаете ? \r\n25см. \r\n35см.
\r\n45см. ";
    await botClient.SendTextMessageAsync(chatId, line,
replyMarkup: PersonalButtons.GetButtonsSize());
    }
    else
    {
    Info.ChangeAtLibrary(Idd, chatId, Customer_idd,
"create", next_order, Info.library[Idd].Item5, text);
    string line = "Желаете еще что-то добавить в заказ
?\r\n /cancel - отмена всего заказа \r\n /continue -
добавить в заказ еще что-то \r\n /finish - подтвердить ";
    await botClient.SendTextMessageAsync(chatId, line,
replyMarkup: PersonalButtons.GetButtonsFinish());
    }
    }
    else
    {
    if(next_order=="er" && current_order == "s4")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Желаете еще что-то добавить в заказ
?\r\n /cancel — отмена всего заказа \r\n /continue —
добавить в заказ еще что-то \r\n /finish — подвердить ";
    string allLine = line1 + line2;
```

```
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsFinish());
    }
    else
    {
    if(next_order=="er" && current_order == "s0")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Добро пожаловать, " + FName + " " +
LName + "\r\nЧтобы приступить, жмякай /start";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsStart());
    }
    else
    {
    if(next_order=="er" && current_order == "s1")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Вижу ты хочешь заказать пиццу. \r\n
Держи инструкцию: \r\n /create — начать построение заказа
\r\n /check — проверить твои текущие заказы \r\n /cancel —
отменить выбранный заказ";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsFirstStep());
    }
    else
    {
    if(next_order=="er" && current_order=="s2")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
```

```
    string line2 = "Какого типа пиццу вы желаете ?
\r\nМексиканская \r\nГавайская \r\nМаргарита";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsCreate());
    }
    else
    {
    if(next_order=="er"&& current_order=="s5")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Отправте номер заказа, который желаете
удалить\r\n";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsCancel());}}}}}}}
```

Підтвердження замовлення відбувається після отриманої команди «/finish», класу «FinishCommand» виконанням методу «Execute» (Лістинг 15).

Лістинг *15 Метод реакції на команду «/finish»*

```
    public override async Task Execute(Message message,
TelegramBotClient botClient){
    int chatId = Convert.ToInt32(message.Chat.Id);
    string LName = Convert.ToString(message.Chat.LastName);
    string FName =
Convert.ToString(message.Chat.FirstName);
    string text = Convert.ToString(message.Text);
    string current_order = "";
    int Customer_idd = DB.createOrReturnCustomer(chatId,
LName, FName);
    current_order = Info.CurrentSost(chatId, Customer_idd);
```

```
    string next_order = Avtomat.Check_sost(text,
current_order);
    if(next_order == "s0" && current_order=="s4")
    {
    int Order_id = DB.CreateAndReturnOrder(Customer_idd,
"finish");
    List<int> spisok_order = new List<int>();
    Info.GetAllCurrentOrderId(spisok_order, chatId,
Customer_idd);
    for(int i=0;i<spisok_order.Count;i++)
    {
    string name_pizza =
Info.library[spisok_order[i]].Item5;
    string size_pizza =
Info.library[spisok_order[i]].Item6;
    int Pizza_Id = DB.ReturnPizzaId(name_pizza,
size_pizza);
    DB.InsertAtOrdersPizzas(Order_id, Pizza_Id);
    }
    for(int j=0;j<spisok_order.Count;j++)
    {
    Info.library.Remove(spisok_order[j]);
    }
    string line1 = "Спасибо за заказ ! \r\n";
    string line2 = "Добро пожаловать, " + FName + " " +
LName + "\r\nЧтобы приступить, жмякай /start";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsStart());
    }
    else
    {
    if(next_order == "er" && current_order =="s1")
```

```
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Вижу ты хочешь заказать пиццу. \r\n
Держи инструкцию: \r\n /create — начать построение заказа
\r\n /check — проверить твои текущие заказы \r\n /cancel —
отменить выбранный заказ";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsFirstStep());
    }
    else
    {
    if(next_order == "er" && current_order =="s2")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Какого типа пиццу вы желаете ?
\r\nМексиканская \r\nГавайская \r\nМаргарита";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsCreate());
    }
    else
    {
    if(next_order=="er" && current_order =="s3")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Какого размера пиццу желаете ?
\r\n25см. \r\n35см. \r\n45см.";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsSize());
    }
    else
```

```
    {
    if(next_order == "er" && current_order=="s5")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Отправте номер заказа, который желаете
удалить\r\n";
    string allLine = line1 + "\r\n" + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsCancel());
    }
    else
    {
    if(next_order == "er" && current_order =="s0")
    {
    string line1 = "Предыдущая команда — ошибочна ! \r\n";
    string line2 = "Добро пожаловать, " + FName + " " +
LName + "\r\nЧтобы приступить, жмякай /start";
    string allLine = line1 + line2;
    await botClient.SendTextMessageAsync(chatId, allLine,
replyMarkup: PersonalButtons.GetButtonsStart());}}}}}}}
```

Команда підтвердження формує кінцевий варіант замовлення та заносить дані у базу даних, таблиці «Orders», за допомогою метода «CreateAndReturnOrder» класу «DB» (Лістинг 16).

Лістинг *16 Метод внесення замовлення до бази даних*

```
    public static int CreateAndReturnOrder(int
Customer_idd, string Status_orderr)
    {
    int number = 0;
    using (SQLiteConnection connn = new
SQLiteConnection(@"DataSource=F:\TelegramBot2017\TelegrammA
```

```
spMvcDotNetCoreBot\TelegrammAspMvcDotNetCoreBot\PizzaDB.db;
Version=3;"))
     {
     connn.Open();
     SQLiteCommand command2 = new SQLiteCommand();
     command2.Connection = connn;
     command2.CommandText = @"
     INSERT INTO Orders (Customer_id, Status_order)
     VALUES (@cust_id, @stat);";
     command2.Parameters.AddWithValue("@cust_id",
Customer_idd);
     command2.Parameters.AddWithValue("@stat",
Status_orderr);
     command2.ExecuteNonQuery();
     SQLiteCommand command = new SQLiteCommand();
     command.Connection = connn;
     command.CommandText = "SELECT Order_id FROM Orders
WHERE Customer_id = @custt_idd;";
     command.Parameters.AddWithValue("custt_idd",
Customer_idd);
     using (SQLiteDataReader reader =
command.ExecuteReader())
     {
     if (reader.HasRows)
     {
     while (reader.Read())
     {
     number = reader.GetInt32(0);}}}
     connn.Close();}return number;}
```

3.5.2 Автомат Мура

Основна частина, бізнес-модель, програми виконується завдяки скінченному автомату Мура. Перш ніж буде задіян автомат Мура, вхідна стрічка повинна пройти крізь транслітератор. Головна мета транслітератора — перетворення або скорочення вхідної множини до необхідного вигляду, щоб автомат Мура міг обробити вхідне повідомлення, отримане від користувача.

Транслітерація відбувається завдяки співвідношенню вхідних символів повідомлення до таблиці лексем, котра складається з токену та значення (див. Табл. 4), деякі лексеми не мають значення, воно унікальне. Вхідне повідомлення перетворюється на нову стрічку, що скаладається з токенів. Після чого нова стрічка подається на вхід, до обробки автоматом Мура.

Таблиця *4*

Перелік токенів та їх значень

Токен	Значення		
/start	унікальне		
/create	унікальне		
/type	Мексиканська	Гавайська	Маргарита
/size	25	35	45
/continue	унікальне		
/cancel	унікальне		
/finish	унікальне		
/another	унікальне		
/check	унікальне		

Скінченним автоматом Мура є набір з п’яти елементів M =[A,S, Z, ν, ζ].

1) A = {/start,/create,/type,/size,/continue,/cancel,/finish,/another,/check} — вхідні елементи (вхідний алфавіт), множина скінченного характеру;
2) Z = {ERROR,exit} — вихідні елементи (вихідний алфавіт), множина скінченного характеру;
3) S = {s0,s1,s2,s3,s4,s5} — множина, що складається з внутрішніх станів (кожний стан, це виклик функції, що забезпечує створення замовлення).

4) ν: S×A→S — функція переходу в наступний стан.

5) ζ: S→Z — функція виходів.

Робота автомата Мура починається з початкового стану s0. Головна особливість автомата Мура, що він працює в два такти: спочатку новий стан, потім вихід. Даний автомат являє собою скінченний автомат-препроцесор, кожному його стану S_i відповідає виконання набору функцій, перелік яких наводиться на рисунку 19, а виходи з автомата відповідають функціям:

«er» — виклик функції обробки помилки, вхідних даних.

«ERROR» — виклик функції обробки неправильного замовлення.

«exit» — виклик функції обробки замовлення.

Загальний вигляд автомату Мура зображено на рисунку 20.

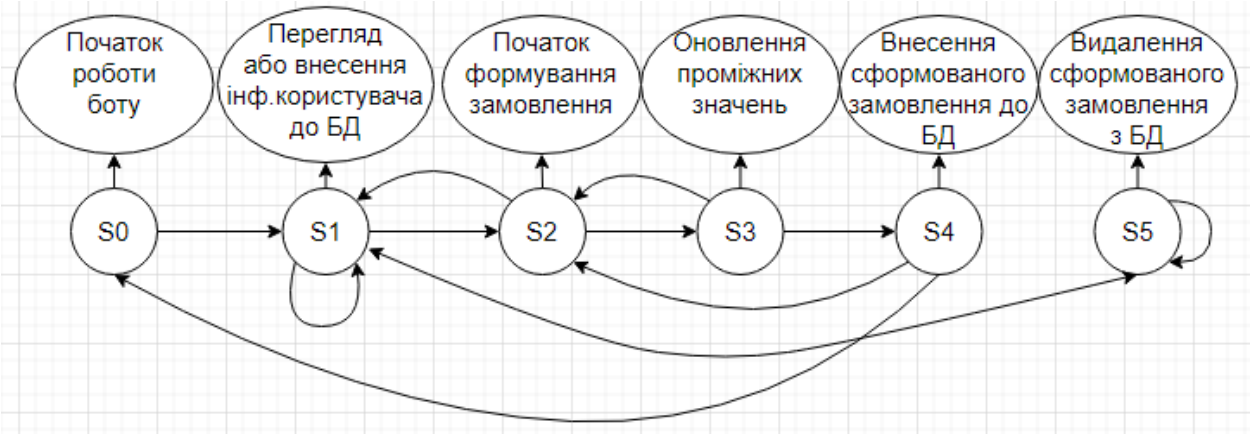

Рис.*19 Діаграма станів*

	/start	/create	/type	/size	/continue	/cancel	/finish	/another	/check	Виходи
s0	s1	er	er	er	er	er	er	er	er	ERROR
s1	er	s2	er	er	er	s5	er	er	s1	ERROR
s2	er	er	s3	er	er	s1	er	er	er	ERROR
s3	er	er	er	s4	er	s2	er	er	er	ERROR
s4	er	er	er	er	s2	s2	s0	er	er	exit
s5	er	er	er	er	er	s1	er	s5	er	ERROR

Рис.*20 Таблиця станів*

Розглянемо приклад роботи автомату. Спочатку отримується вхідна стрічка від користувача, що спрямована на формування замовлення та по ній виконується транслітерація.

Вхідна стрічка: «/start /create Мексиканська 35 /finish»

Результат транслітерації стрічки: «/start /create / type /size /finish»

Отримана нова стрічка подається на вхід скінченному автомату Мура.

ν: $(s_0, /start) \rightarrow s_1$

$(s_1, /create) \rightarrow s_2$

$(s_2, /type) \rightarrow s_3$

$(s_3, /size) \rightarrow s_4$

$(s_4, /finish) \rightarrow s_0$

ζ: (s4, /finish) → exit

В результаті роботи автомата ми маємо сформоване замовлення.

3.6 Діаграма класів

Загальна кількість класів — 19 класів (див.Рис.21).

Система виконана за патерном MVC (див.Рис.22).

Контролером виступає клас «MessageController».

Інтерфейс месенджера Telegram — представлення.

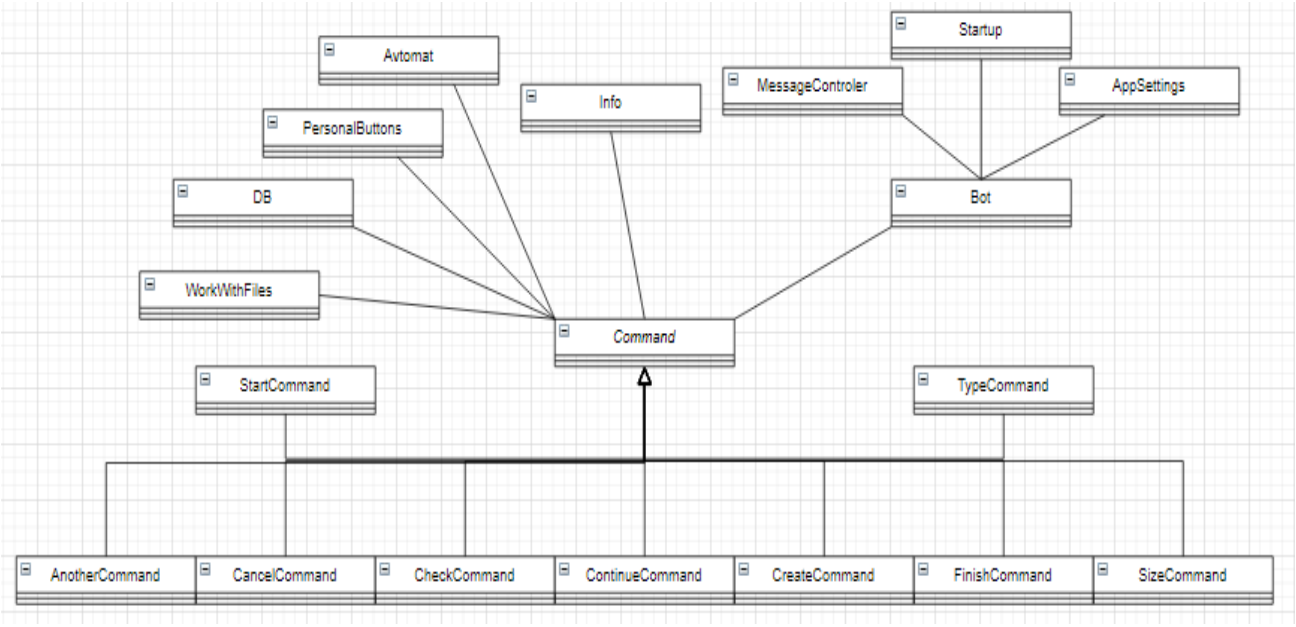

Рис.*21 Діаграма класів*

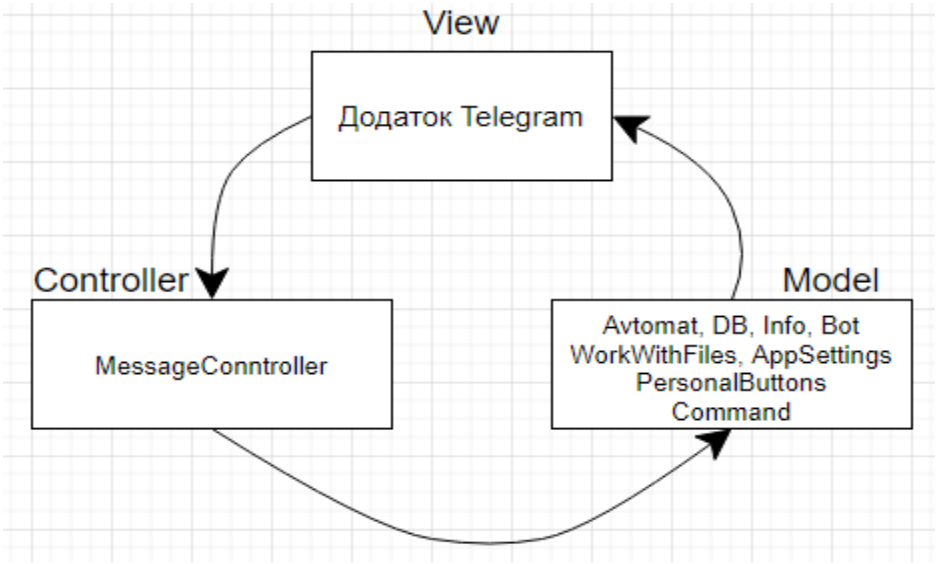

Рис.*22 Погляд на MVC зі сторони класів*

Startup — клас, що виконує реєстрацію усіх необхідних служб (див.Рис.23).

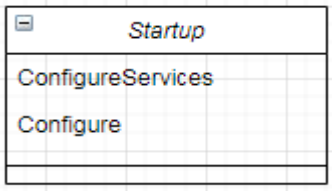

Рис.*23 Клас «Startup»*

Методи:

ConfigureServices (IServiceCollection services) — додає сервіси до контейнеру.

Configure (IApplicationBuilder app, IHostingEnvironment env) — налаштування HTTP запитів тунелювання.

MessageController — клас, який є контролером системи (див.Рис.24).

Рис.*24 Клас «MessageController»*

Методи:

Task<OkResult> Post ([FromBody]Update update) — реагує на дії користувача, отримує повідомлення, аналізує дані, утворює наступне представлення.

AppSettings — клас, зберігаючий поля налаштувань (див.Рис.25).

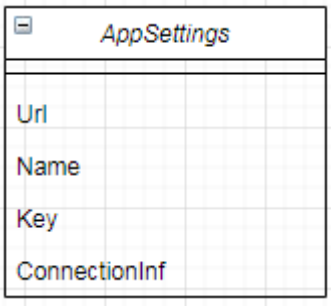

Рис.*25 Клас «AppSettings»*

Змінні:

Url — глобальна адреса для підключення Webhook.

Name — ім’я боту у системі Telegram.

Key — токен зв’язку бота у системі Telegram.

ConnectionInf — адреса для підключення до бази даних.

Avtomat — клас, реалізує скінченний автомат Мура (див.Рис.26).

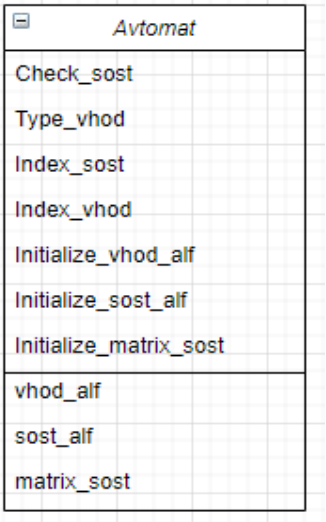

Рис.*26 Клас «Avtomat»*

Методи:

Check_sost (string text, string current_sost) — визначення наступного стану в роботі автомата Мура.

Type_vhod (string text) — транслітерація.

Index_sost (string current_sost) — визначення індексу стану серед множини станів.

Index_vhod (string current_vhod) — визначення індексу входу серед множини входів.

Initialize_vhod_alf (int n) — ініціалізація множини вхідного алфавіту.

Initialize_sost_alf (int) — ініціалізація множники алфавіту станів.

Initialize_matrix_sost (int n, int m) — ініціалізація матриці станів.

Змінні:

vhod_alf — множна входів, алфавіт

sost_alf — множина станів, алфавіт

matrix_sost — матриця станів

DB — клас, відповідає за доступ до бази даних (див.Рис.27).

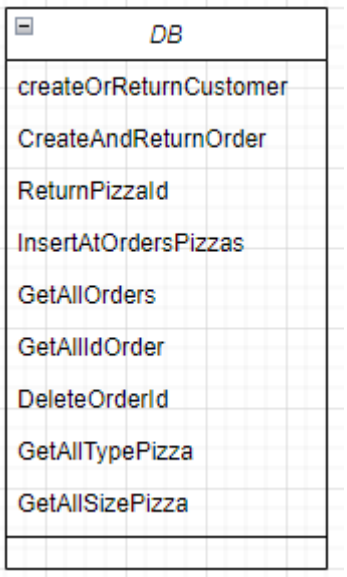

Рис.*27 Клас «DB»*

Методи:

createOrReturnCustomer (int Chat_id, string LastName, string FirstName) — додавання користувача до бази даних, або повернення його айді.

CreateAndReturnOrder (int Customer_idd, string Status_orderr) — додавання замовлення до бази даних, або повернення його айді.

ReturnPizzaId (string NName_pizza, string SSize_pizza) — повернення айді піцці з бази даних.

InsertAtOrdersPizzas (int Order_id,int Pizza_id) — внесення даних до бази даних з приводу формування замовлення.

GetAllOrders (int Customer_idd) — повернення усього списку замовлення з бази даних.

GetAllIdOrder (List<int> list_id,int Customer_idd) — повернення певного замовлення з бази даних, по атрибуту айді.

DeleteOrderId (int Order_id) — видалення замовлення з бази даних, по атрибуту айді.

GetAllTypePizza () — повернення усіх типів піцци з бази даних.

GetAllSizePizza () — повернення усіх розмірів піцци з бази даних.

Info — клас, виконуючий обробку проміжних значень, отриманих від користувача (див.Рис.28).

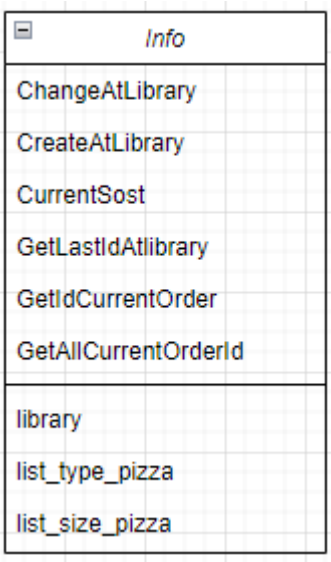

Рис.*28 Клас «Info»*

Методи:

ChangeAtLibrary (int Id, int Chat_idd,int Customer_idd,string Status_order,string Step_order,string Type_pizza,string Size_pizza) — зміна проміжних значень користувача

CreateAtLibrary (int Id, int Chat_idd, int Customer_idd, string Status_order, string Step_order, string Type_pizza, string Size_pizza) — створення проміжних значень користувача

CurrentSost (int Chat_idd, int Customer_idd) — отримання поточного кроку користувача, при створенні замовлення

GetLastIdAtlibrary () — отримання айді останніх проміжних значень користувача

GetIdCurrentOrder (int Chat_idd, int Customer_idd) — отримання айді поточних проміжних значень користувача

GetAllCurrentOrderId (List<int> spisok_order, int Chat_idd, int Customer_idd) — отримання всіх проміжних значень користувача.

Змінні:

library — змінна зберігаюча проміжні дані користувача.

list_type_pizza — змінна зберігаюча усі існуючи типи піцц.

list_size_pizza — змінна зберігаюча усі існуючи розміри піцц.

PersonalButtons — клас, зберігаючий всі можливі представлення (див.Рис.29)

Рис.*29 Клас «PersonalButtons»*

Методи:

GetButtonsCreate () — отримання представлення на вибір типу піцци.

GetButtonsSize () — отримання представлення на вибір розміру піцци.

GetButtonsFinish () — отримання представлення на завершення формування замовлення.

GetButtonsStart () — отримання представлення на початок роботи з ботом.

GetButtonsCancel () — отримання представлення на видалення замовлення.

GetButtonsFirstStep () — отримання представлення на початок формування замовлення.

Bot — клас, відповідальний за налаштування зв’язку з серверами месенджеру Telegram (див.Рис.30).

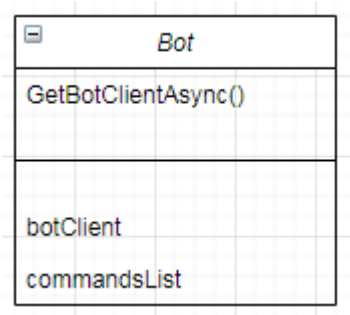

Рис.*30 Клас «Bot»*

Методи:

GetBotClientAsync () — отримання зв’язку з серверами месенджеру Telegram.

Змінні:

botClient — об’єкт зв’язку.

commandsList — список можливих команд.

WorkWithFiles — клас, відповідальний за зчитування даних з файлів (див.Рис.31).

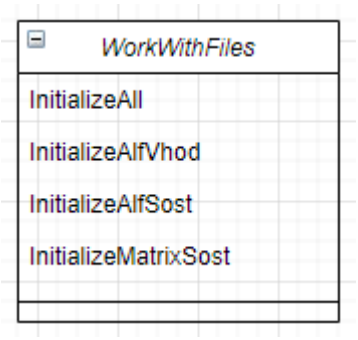

Рис.*31 Клас «WorkWithFiles»*

Методи:

InitializeAll () — зчитування всіх необхідних файлів

InitializeAlfVhod () — зчитування файлу, відповідного за вхідний алфавіт

InitializeAlfSost () — зчитування файлу, відповідного за алфавіт станів

InitializeMatrixSost () — зчитування файлу, відповідного за матрицю станів.

Command — абстрактний клас (див.Рис.32), є шаблоном для створених команд та класів (AnotherCommand, CancelCommand, CheckCommand, ContinueCommand, CreateCommand, FinishCommand, SizeCommand, TypeCommand, StartCommand).

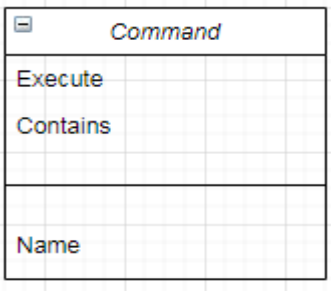

Рис.*32 Клас «Command»*

Методи:

Execute (Message message, TelegramBotClient client) — виконання команди

Contains (string command) — перевірка на належність до команди

Змінні:

Name — ім’я команди

3.7 База даних

База даних виконана за допомогою SQLite. Рішенням вибрати саме цю базу даних було завдяки:

- Достатньому рівню безпеки.
- Непотрібність в сервері SQL.
- Всі дані знаходяться в одному файлі.
- Простоті використання.
- Безкоштовному типу використання.

SQLite підтримує наступні типи даних:

- NULL (тип значення NULL).
- INTEGER (цілочисельне значення).
- REAL (значення з плаваючою точкою).
- TEXT (текстовий рядок).
- BLOB (значення зберігаються в блоці даних, таким чином, яким вони туди були занесені).

Загальний вигляд бази даних зображено на схемі, рисунок 33.

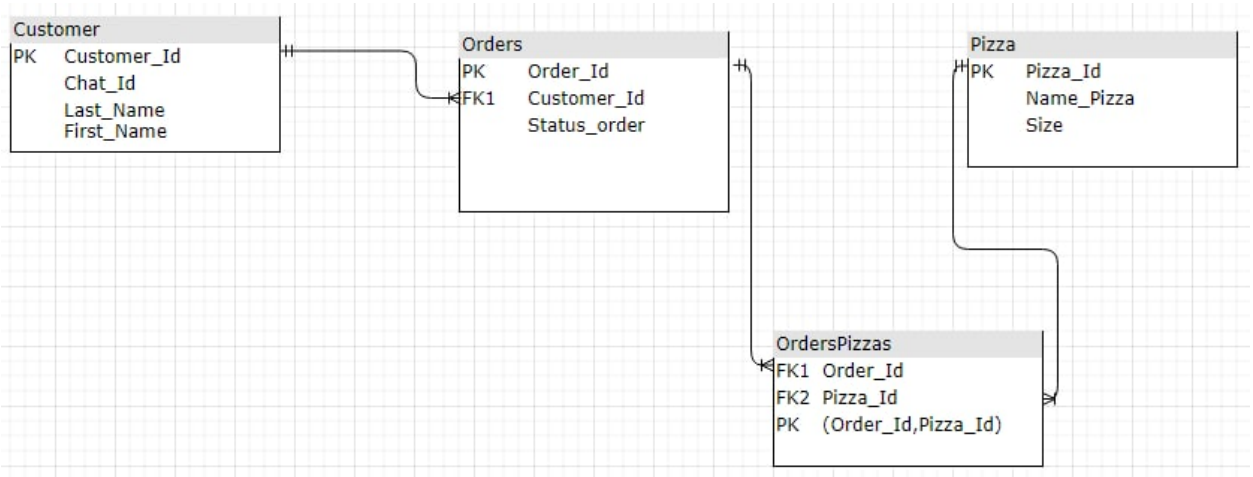

Рис.*33 Діаграма бази даних*

База даних складається з 4 таблиць: Customer, Orders,Pizza, OrdersPizzas — та має назву PizzaDB.

Таблиця Customer, відповідальна за зберігання даних про користувачів (див.Рис.34), має 4 атрибути:

1. Customer_id: має значення типу INTEGER, є первинним ключем, значення унікальне, значення NOT NULL.
2. Chat_id: має значення типу TEXT, значення NOT NULL, містить айді чату користувача (кожному користувачу месенджеру Telegram належить свій власний чат айді).
3. Last_name: має значення типу TEXT, значення NOT NULL, містить прізвище користувача.
4. First_name: має значення типу TEXT, значення NOT NULL, містить ім’я користувача.

PizzaDB0 Table name: Customer WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	Customer_id	INTEGER								NULL
2	Chat_id	TEXT								NULL
3	Last_name	TEXT								NULL
4	First_name	TEXT								NULL

Рис.*34 Таблиця Customer*

Таблиця Pizza, відповідальна за зберігання даних про піццу (див.Рис.35), має 3 атрибути:

1. Pizza_id: має значення типу INTEGER, є первинним ключем, значення унікальне, значення NOT NULL.
2. Name_pizza: має значення типу TEXT, значення NOT NULL, містить назву піцци.
3. Size: має значення типу TEXT, значення NOT NULL, містить розмір піцци.

PizzaDB0 Table name: Pizza WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	Pizza_id	INTEGER								NULL
2	Name_pizza	TEXT								NULL
3	Size	TEXT								NULL

Рис.*35 Таблиця Pizza*

Таблиця Orders, відповідальна за зберігання даних про замовлення користувача (див.Рис.36), має 3 атрибути:

1. Order_id: має значення типу INTEGER, є первинним ключем, значення унікальне, значення NOT NULL.
2. Customer_id: зовнішній ключ, типу INTEGER, NOT NULL, унікальне, має посилальний каскадний зв'язок з таблицею Customer на оновлення та видалення даних.
3. Status_order: має значення типу TEXT, значення NOT NULL, містить статус замовлення.

PizzaDB0 Table name: Orders WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	Order_id	INTEGER								NULL
2	Customer_id	INTEGER								NULL
3	Status_order	TEXT								NULL

Рис.*36 Таблиця Orders*

Таблиця OrdersPizzas, є проміжною таблицею між таблицями Pizza та Orders(див.Рис.37). Забезпечує тип зв'язку «many-to-many», одне замовлення може складатись з безліч піцц та одна піцца може відноситись до безліч замовлень, має 3 атрибути:

1. OP: має значення типу INTEGER, є первинним ключем, значення унікальне, значення NOT NULL.

2. Order_id: зовнішній ключ, типу INTEGER, NOT NULL, унікальне, має посилальний каскадний зв’язок з таблицею Orders на оновлення та видалення даних.
3. Pizza_id: зовнішній ключ, типу INTEGER, NOT NULL, унікальне, має посилальний каскадний зв’язок з таблицею Pizza на оновлення та видалення даних.

PizzaDB0 Table name: OrdersPizzas WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	OP	INTEGER								NULL
2	Order_id	INTEGER								NULL
3	Pizza_id	INTEGER								NULL

Рис.*37 Таблиця OrdersPizzas*

3.8 Інтерфейс

У системі використовується готовий інтерфейс месенджеру Telegram. Користування створеним ботом завжди супроводжується повідомленням та можливою відповідью на даний момент часу (у вигляді кнопки), щоб користувач розумів принцип роботи додатку.

Початок програми складається з привітання користувача та підказкою, що далі потрібно зробити, а саме відправидити команду «/start» (див.Рис.38).

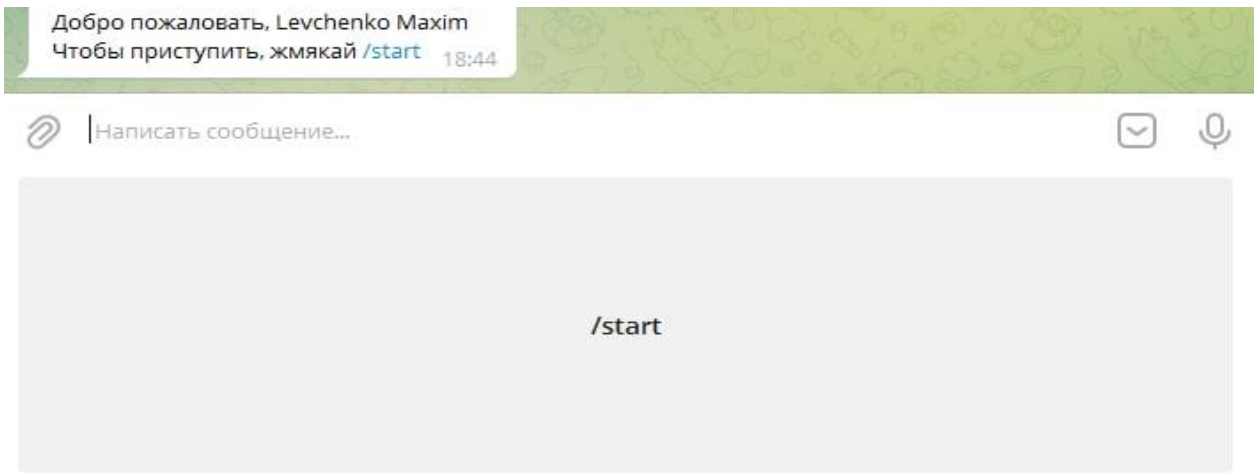

Рис.*38 Початок роботи програми*

Після відправлення необхідної команди, надається можливість почати формувати замовлення, перевірити вже створені замовлення користувачем та відминити існуюче замовлення (див.Рис.39).

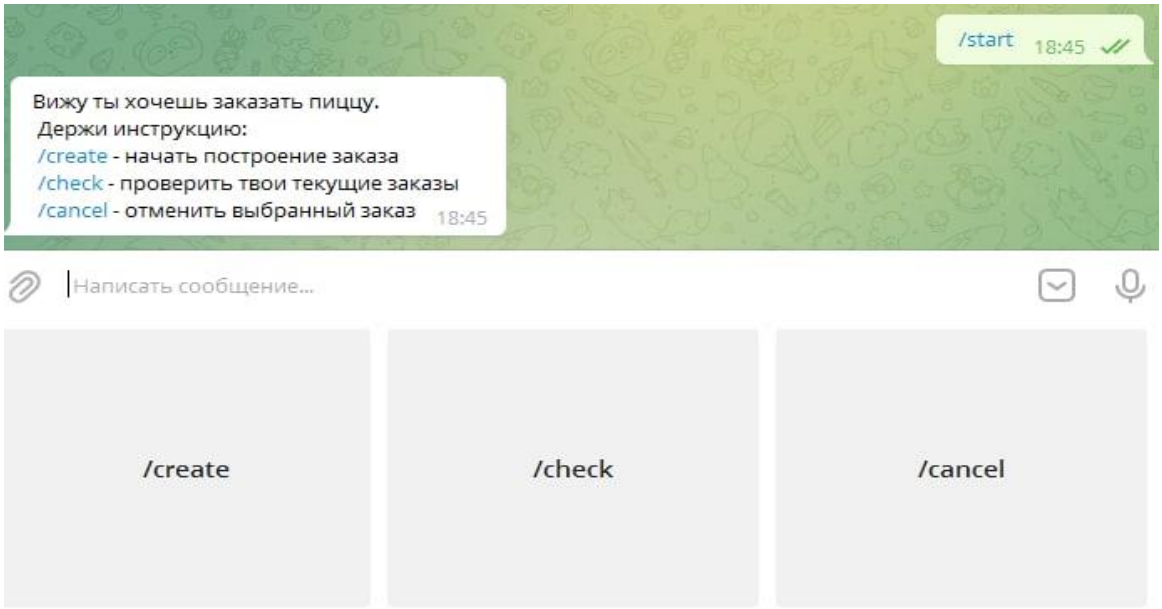

Рис.*39 Результат команди «/start»*

При бажанні почати формувати замовлення відправляється команда «/create» (див.Рис.40).

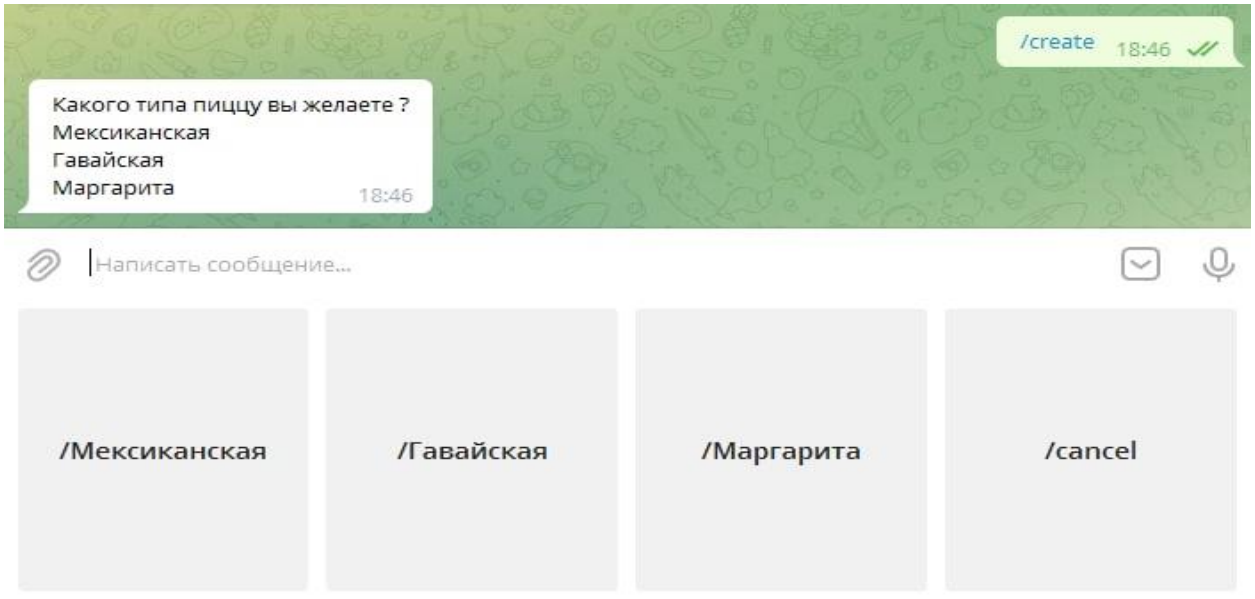

Рис.*40 Результат команди «/create»*

Визначившись з типом піцци, відправляється відповідна команда, наприклад «/Гавайская» (див.Рис.41). Після чого користувачеві потрібно вибрати підходящий розмір піцци (див.Рис.42).

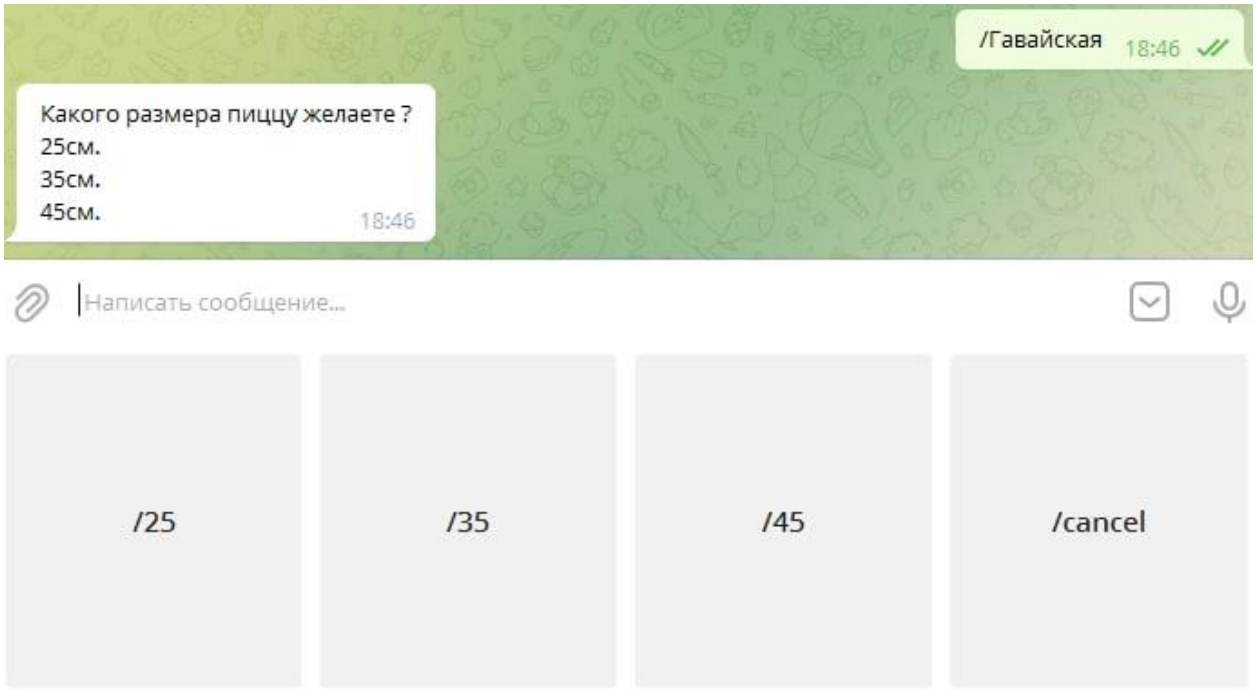

Рис.*41 Результат команди «/Гавайская»*

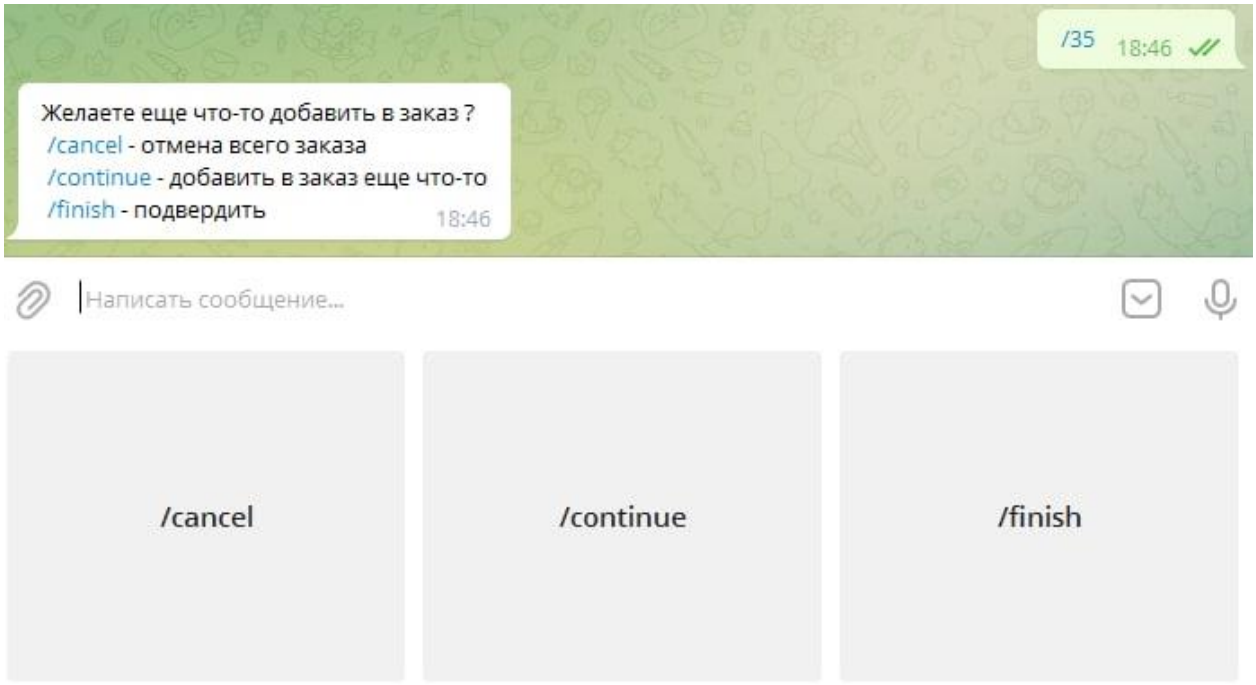

Рис.*42 Результат команди «/35»*

Після отриманного розміру, користувач підтверджує створене замовлення (див.Рис.43).

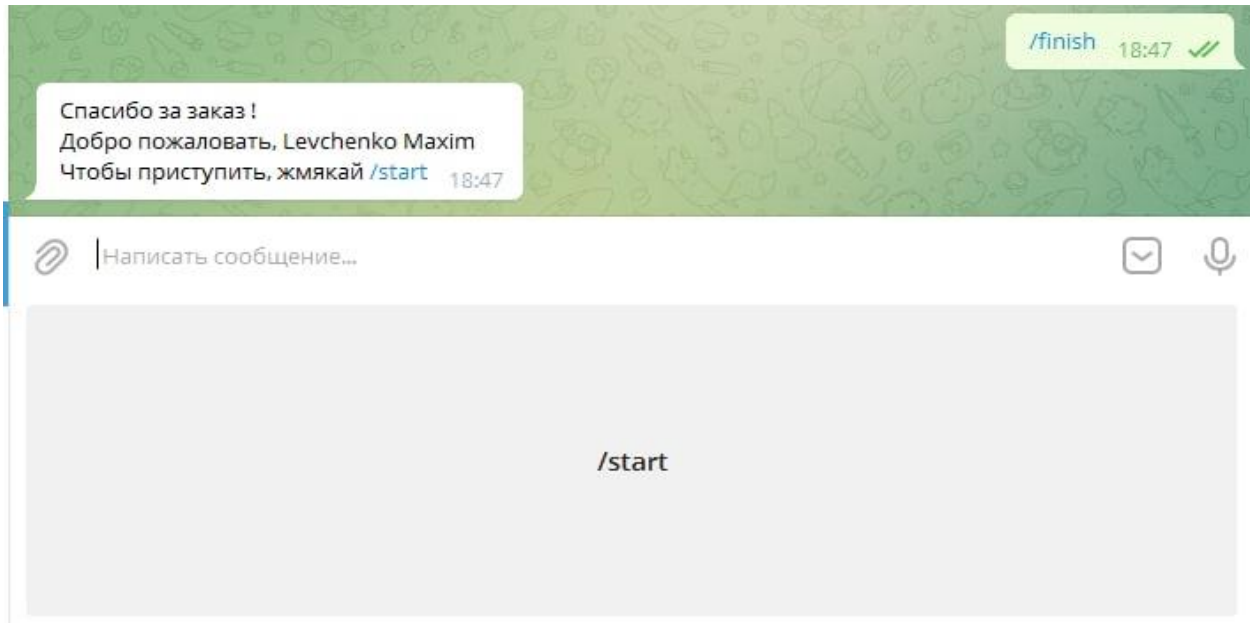

Рис.*43 Результат команди «/finish»*

Є можливість перевірити створене замовлення командою «/check» (див.Рис.44). Користувачеві буде видан перелік існуючих замовлень.

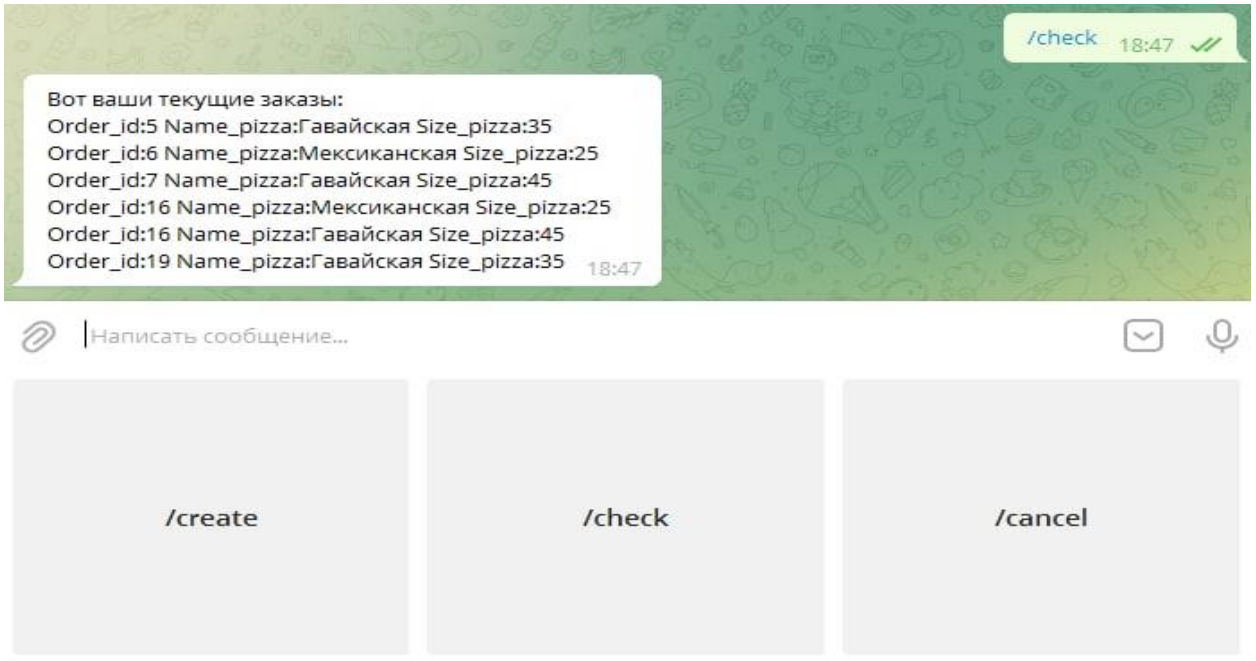

Рис.*44 Результат команди «/check»*

Якщо користувач хоче відмовитись від замовлення потрібно ввести команду «/cancel» (див.Рис.45).

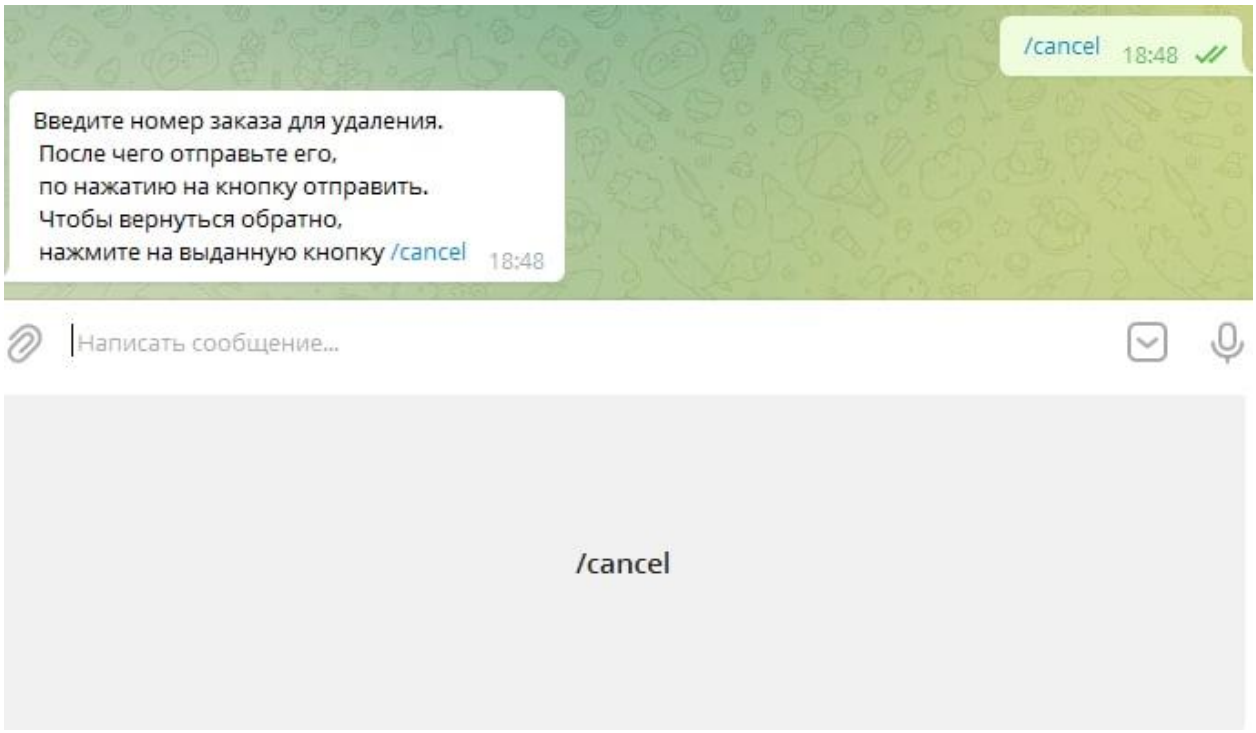

Рис.*45 Результат команди «/cancel»*

Після чого відправляється номер замовлення, котрий бажаємо видалити (див.Рис.46).

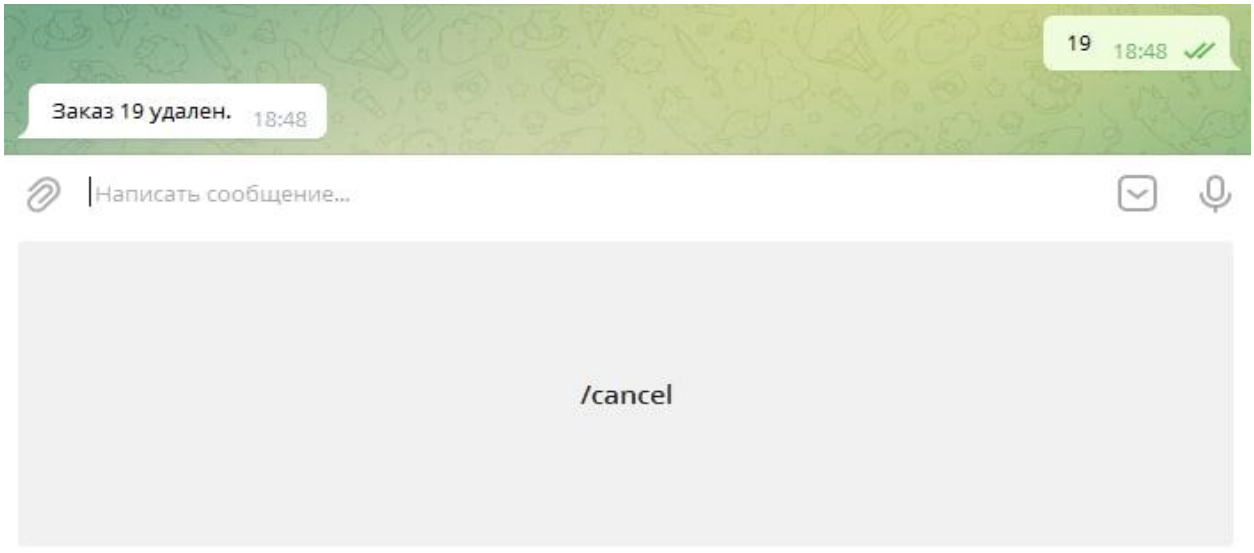

Рис.*46 Результат видалення замовленя*

РОЗДІЛ 4 ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ СИСТЕМИ ONLINE-ОБСЛУГОВУВАННЯ

4.1 Аналіз результатів

Результатом дослідження сфери online-обслуговування було прийнято рішення використовувати скінченний автомат для керуванням логікою програмної системи, а саме чат-ботом. Розроблене програмне забезпечення є ефективним рішенням поставленої задачі. Ефективність була досягнута завдяки використанню ефективних програмних засобів: Telegram Bot API, MVC.

Інструментарій Telegram Bot API дозволив створити чат-бота, та впровадити його в месенджер Telegram. Основною перевагою такого засобу є відстуність необхідності в власній реалізації шифрування даних, адже це вже реалізовано в додатку Telegram, це його обов’язок. В склад алгоритмів шифрвування входить: RSA-2048, DH-2048, SHA-256.

Патерн MVC дозволив раціонально розподілити обов’язки кожної частини програмного забезпечення. Контролер відповідає за прийняття нових даних від представлення, їх аналіз та вирішує який вигляд буде мати наступне представлення. Модель, котра викликається контролером, займається зміною даних та їх записом в базу даних. Представлення відображає оновленні дані від моделі.

Розроблене програмне забезпечення має можливість:

- отримувати вхідні дані;
- обробляти вхідні дані;
- надавати відповідь користувачеві, згідно вхідним даним;
- зберігати сформовані замовлення в базі даних;
- виконувати оновлення даних у базі даних;
- супроводжувати дії користувача додатковою інформацією.

Розроблений бот забезпечує формування замовлення без помилок у будь-якому випадку, це забезпечує бізнес-логіка програми, а саме скінченний автомат.

Можливі 3 сценарії роботи програми на основі автомату:

1. Повноцінне формування замовлення.
2. Формування замовлення з корекцією раніше введених даних.
3. Формування замовлення з обробкою помилково введених даних.

4.1.1 Повноцінне формування замовлення

Формування замовлення виконується відповідно концепції патерну MVC. Контролер перехоплює послідовність подій, отриманих від користувача, котра відноситься до частини представлення, виконує їх аналіз та вносить певні зміни завдяки виклику методів частини моделі. Після чого модель формує відповідну подію про виконані зміни, і на це реагує представлення відображенням нової інформації користувачеві. Перелічена послідовність спрямована на повноцінне формування замовлення та супроводжується роботою скінченного автомата препроцесора (див.Рис.47).

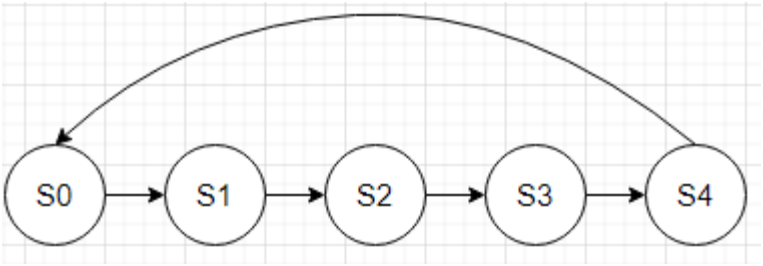

Рис.*47 Повноцінне формування замовлення*

4.1.2 Формування замовлення з корекцією раніше введених даних

Може утворитись ситуація, коли представлення сформує подію про необхідність повернення до одного з вже пройдених етапів формування замовлення з метою зміни деяких даних. Контролер зреагує на подію та виконає запит за допомогою елементів моделі, одним з котрих є робота автомату (див.Рис.48).

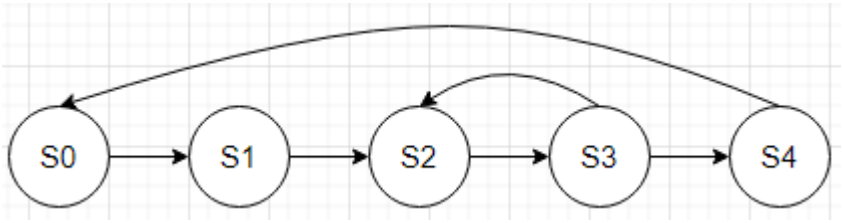

Рис.*48 Формування замовлення з корекцією*

4.1.3 Формування замовлення з обробкою помилково введених даних

Якщо представлення буде передавати таку послідовність даних, котра не передбачена в програмі, контролер виконає свої обов'язки по аналізу, але модель не зможе сформувати відповідну подію про виконання своєї роботи, через відсутність проходження одного з етапів роботи (див.Рис.49) — це свідчить про невідповідність концепції MVC. Без наявності концепції MVC немає можливості виконувати формування бажаного замовлення. В результаті не буде можливо закінчити процес формування замовлення, а отже неможлива робота програмної системи.

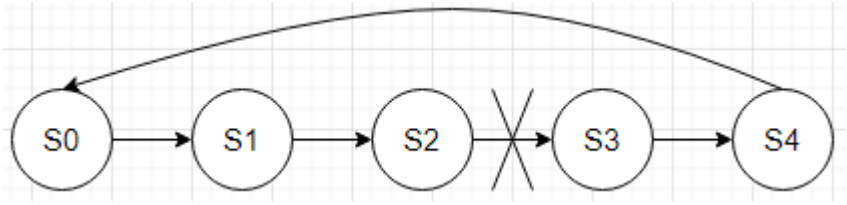

Рис.*49 Наявність недосягнення одного з етапів роботи*

Висновок: дослідження показали ефективність застосування скінченного автомату для керування чат-ботом, розробленим на основі технології MVC. Це підтверджується коректною обробкою online-замовлень користувачів.

ВИСНОВКИ

В рамках магістерської роботи отриманні наступні результати:

1. Проведено дослідження проблеми розробки чат-боту для обробки інформації на основі автоматного програмування.
2. Зроблено огляд технологій для розробки Web-застосунку.
3. Проаналізовані засоби розробки та існуючі рішення в розглянутій предметній області. Для реалізації застосунку використано: SQLite — база даних, Bot-API — засіб створення та впроводження чат-ботів у середі Telegram, патерн MVC, Web API — інтерфейс по розробці веб-додатків прикладного рівня, ASP.NET Core — фреймворк по розробці веб-додатків.
4. Розроблено обробник інформації в online-обслуговуванні на базі скінченного автомату у форматі чат-боту з дотриманням необхідних вимог.
5. Виконано дослідження впливу патерну MVC при проектуванні, розробці та роботі програмної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шилдт, Герберт. C# 4.0 : полное руководство : пер. с англ. Москва : ООО «И.Д. Вильямс», 2011. 1056 с.

2. Эндрю Троелсен. Язык программирования C# 5.0 и платформа .Net 4.5. Москва : Вильямс, 2013. 1312 с.

3. Попівщій В. І. Системне програмування та операційні системи : методи-ческький посібник з курсу : для студентів ЗДІА спеціальності 080403 "Програмне забезпечення автоматизованих систем" очної та заочної форм навчання. Запоріжжя, 2009. 250 с.

4. De Oliveira Jason, Bruchet Michel. Learning ASP.NET Core 2.0 : build modern web apps with ASP.NET Core 2.0, MVC, and EF Core 2. Packt Publishing, 2018. 508 p.

5. Telegram Bot Api Documentation. URL : https://core.telegram.org/bots/api (дата звернення: 07.03.2021)

6. Скрипник І. А., Пожуєв А.В. Комп’ютерная дискретна математика. Навчально-методичний посібник для студентів ЗДІА напряму 6.050103 «Програмна інженерія». Запоріжжя, 2016. 172 с.

7. Агибалов Г. П. Конечные автоматы в криптографии. 2009. URL: https://cyberleninka.ru/article/n/konechnye-avtomaty-v-kriptografii (дата звернення: 07.03.2021).

8. Казаков М. А., Корнеев Г. О., Шалыто А. А. Разработка логики визуализаторов алгоритмов на основе конечних автоматов. Санкт-Петербург, 2003. URL: http://www.softcraft.ru/auto/switch/vis/vis.pdf (дата звернення: 07.03.2021).

9. Савченко О. В., Куреннов Д.В. Система управления роботом на основе конечного автомата и нейройнной сети. 2014. URL: https://cyberleninka.ru/article/n/sistema-upravleniya-robotom-na-osnove-konechnogo-avtomata-i-neyronnoy-seti (дата звернення: 07.03.2021).

10. Бурдонов И. Б., Косачев А. С., Кулямин В.В. Использование конечних автоматов для тестирования програм. 2000. URL: http://www.burdonov.ru/doctor/papers_2000/Ispolzovanie_konechnykh_avtomatov_dlya_testirovaniya_programm/Ispolzovanie_konechnykh_avtomatov_dlya_testirovaniya_programm.pdf (дата звернення: 07.03.2021).

11. Никитин П. В., Горохова Р. И. Проектирование интеллектуальной обучающей системы на основе теори конечных автоматов. 2015. URL: https://cyberleninka.ru/article/n/proektirovanie-intellektualnoy-obuchayuschey-sistemy-na-osnove-teorii-konechnyh-avtomatov (дата звернення: 07.03.2021).

12. WhatsApp и Telegram : массовость против безопасности. URL: https://mbk-news.appspot.com/suzhet/whatsapp-i-telegram/ (дата звернення: 07.03.2021).

13. Почему Telegram (почти) всегда уделает WhatsApp. URL: https://vc.ru/marketing/174337-pochemu-telegram-pochti-vsegda-udelaet-whatsapp (дата звернення: 07.03.2021).

14. Telegram Messenger vs WhatsApp. URL: https://versus.com/ru/telegram-messenger-vs-whatsapp (дата звернення: 07.03.2021).

15. Почему Телеграмм? Сравнение Telegram vs Viber и WhatsApp. URL: https://stasfalkovich.com/pochemu-telegramm-sravnenie-telegram-vs-viber-i-whatsapp/ (дата звернення: 07.03.2021).

16. Harel D., Pnueli A. On the development of reactive systems : in «Logic and Models of Concurrent Systems». NATO Advanced Study Institute on Logic and Models for Verification and Specification of Concurrent Systems. Springer Verlag, 1985. P. 477–498.

17. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. 2008. 167 с. URL: http://is.ifmo.ru/books/_book.pdf (дата звернення: 27.06.2021).

18. Ахо А., Сети Р., Ульман Д. Компиляторы. Принципы, технологии, инструменты. Москва : Вильямс, 2003. 768 с.

19. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. Санкт-Петерубрг : Наука, 1998. URL: http://is.ifmo.ru/books/switch/1 (дата звернення: 27.06.2021).

20. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. Москва : ООО «И.Д.Вильямс», 2002. 528 с.

21. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Москва: Мир, 1978. 613 с.

22. Ник Рендольф, Дэвид Гарднер, Майкл Минутилло, Крис Андерсон. Visual Studio 2010 для профессионалов : пер. с англ. Москва : ООО «И.Д.Вильямс», 2011. 1184 с.

23. Алекс Макки. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов. Москва : ООО «И.Д.Вильямс», 2010. 416 с.

24. Jacobson Daniel, Woods Dan, Brail Greg. APIs : A Strategy Guide. Sebastopol, Calif. O'Reilly Media, 2011. 133 p.

25. Адам Фримен. ASP.NET Core MVC 2 с примерами на C# для профессионалов. Москва : ООО «И.Д.Вильямс», 2016. 1008 с.

26. Ngrok Documentation. URL: https://ngrok.com/docs (дата звернення: 13.11.2021).

27. SQLite Documentatio. URL: https://www.sqlite.org/docs.html (дата звернення: 13.11.2021).

28. Левченко М. Г., магістрант, Скрипник І. А., доцент, к.ф.-м.н. — науковий керівник. Особливості застосування скінченного автомата при розробці онлайн додатку для обслуговування людей. Збірник наукових праць студентів, аспірантів, докторантів і молодих вчених. *Молода наука-2021*. Запоріжжя: ЗНУ, 2021. Т.5. С. 89-90.

29. Левченко М. Г., магістрант, Скрипник І. А., доцент, к.ф.-м.н. — науковий керівник. Особливості застосування скінченного автомата для обробки інформації в online-обслуговуванні. Матеріали І Всеукраїнської науково-практичної конференції здобувачів вищої освіти, аспірантів та молодих вче-

них. Актуальні питання сталого наукового-технічного та соціально-економічного розвитку регіонів України. Запоріжжя : ЗНУ, 2021. С. 330-332.

30. История создания чат-ботов : от идеи до использования чат-ботов в бизнесе. URL: https://www.jivo.ru/blog/tutorials-jivo/chat-bot-history.html (дата звернення: 27.06.2021).

31. Все о чат-ботах: типы и примеры, какому бизнесу подойдет, список конструкторов для создания. URL: https://web-promo.ua/blog/vse-o-chat-botah-tipy-i-primery-kakomu-biznesu-podojdet-spisok-konstruktorov-dlya-sozdaniya/ (дата звернення: 27.06.2021).

32. Что такое чат-бот: определение и руководство. URL: https://sendpulse.ua/ru/support/glossary/chatbot (дата звернення: 27.06.2021).

33. Что такое чат-боты и какие они бывают. URL: https://www.carrotquest.io/chatbot/chatbot-types/ (дата звернення: 27.06.2021).

34. Что такое чат-бот ? URL: https://www.oracle.com/ru/chatbots/what-is-a-chatbot/ (дата звернення: 27.06.2021)

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Левченко Максим Геннадійович, студент_2_курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти levchenkomaksim1998@gmail.com, — підтверджую, що написана мною кваліфікаційна робота на тему «**Особливості застосування скінченного автомата для обробки інформації в online-обслуговуванні**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-систем, а також на архівування моєї роботи в базі даних цієї системи.

Дата 30.11.2021 Підпис_____ Левченко Максим Геннадійович

(студент)

Дата 30.11.2021 Підпис_____ Скрипник Ірина Анатоліївна

(науковий керівник)