ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи
на ступінь вищої освіти бакалавр

на тему: **«Створення чат-боту для лізингу автомобілів мовою C#»**

Виконав: студент 4 курсу, групи ПД-42
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Машлянка Д.В.
(прізвище та ініціали)
Керівник Коба А.Б
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)
Нормоконтроль
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність -121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
«____»_____2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ
МАШЛЯНКА ДЕНИС ВАЛЕРІЙОВИЧ

1.Тема роботи: «Створення чат-боту для лізингу автомобілів мовою C#»

Керівник роботи: Коба Андрій Борисович , старший викладач кафедри_ІПЗ__ затверджені наказом вищого навчального закладу від — «18» лютого 2022 року №22.
2. Строк подання студентом роботи_____03.06.2022_____

3. Вхідні дані до роботи:

3.1. Вибір мови програмування

3.2. Алгоритм дії бота

3.3. Aiogram Bot API

3.4. Науково-технічна література, пов’язана з розробкою ботів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1. Аналіз обов’язків розроблюваного бота

4.2. Аналіз та порівняння існуючих прототипів

4.3. Дослідження програмних засобів для розробки бота

4.4. Розробити функціонал створеного бота

5. Перелік графічного матеріалу

5.1.1. Популярність телеграм-ботів

5.1.2. Переваги та недоліки найвідоміших месенджерів

5.1.3. Огляд можливостей розробленого бота

5.1.4. Апробація результатів досліджень

6. Дата видачі завдання 11.04.2022

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.04.21 – 14.04.22	Виконано
2	Аналіз існуючих аналогів	15.04.21 – 17.04.22	Виконано
3	Дослідження програмних засобів	18.04.21 – 21.05.22	Виконано
4	Моделювання об’єкту проектування	26.04.21 – 05.05.22	Виконано
5	Розробка функціоналу застосунку	05.05.21 – 07.05.22	Виконано
6	Вступ, висновки, реферат	11.05.21 – 15.05.22	Виконано
7	Розробка презентації застосунку	16.05.21 – 01.06.22	Виконано
8	Попередній захист роботи	03.06.22	

Студент

Керівник роботи

РЕФЕРАТ

Текстова частина бакалаврської роботи 56с., 9 рис., 58 джерел.

Ключеві слова: C#, API Aiogram, Telegram, чат-бот.

Об'єкт дослідження – створення комплексного Telegram-бота для лізингу автомобілів на мові C#.

Предмет дослідження – методи та засоби створення чат-ботів на базі платформи Telegram.

Мета роботи – спрощення процесу лізингу автомобілів.

Методи дослідження – методи теорії інформації, методи структурного аналізу і проектування, методи розробки програмного забезпечення, методи тестування, валідації та верифікації програмного забезпечення.

Для реалізації поставленої мети потрібно вирішити наступні завдання:

1. Проаналізувати технічні засоби, що використовуються для розробки та обрати необхідні для створення чат-боту;
2. Розробити вимоги до застосунку на основі аналізу переваг та недоліків існуючих додатків;
3. Спроектувати та розробити систему на основі аналізу потреб користувачів;
4. Провести тестування чат-боту.

Практичне значення отриманих результатів: Даний продукт може бути використаний для спрощення лізингу автомобілів.

Проаналізовано можливості середовища розробки Visual Studio, мови програмування C#. Розроблено логіку практичних завдань та загальну концепцію представлення інформації для користувачів.

Галузь використання – лізинг автомобілів.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми та її актуальність. В сучасному світі популярність різного роду месенджерів, поштових клієнтів, чат – клієнтів і великого різноманіття подібних рішень важко недооціни.

В середньому, у кожної другої людини на комп'ютері або смартфоні встановлений той чи інший месенджер, який може нести в собі не тільки функцію спілкування, але й більш масштабні функції, реалізовані, наприклад, за допомогою ботів різної направленості.

Явище ботів в месенджерах набуває все більшої актуальності за рахунок зручності використання подібних рішень, адже вони не вимагають переходити на зовнішні ресурси, встановлювати додаткове програмне забезпечення, тощо.

З огляду на високу актуальність, кількість різноманітних ботів в месенджерах, зокрема, наприклад, в месенджері телеграм, росте кожного дня і набуває все більшої серйозності в масштабах функціональних можливостей.

Для реалізації поставленої мети потрібно вирішити наступні завдання:

1. Проаналізувати технічні засоби, що використовуються для розробки та обрати необхідні для створення чат-боту;
2. Розробити вимоги до застосунку на основі аналізу переваг та недоліків існуючих додатків;
3. Спроектувати та розробити систему на основі аналізу потреб користувачів;
4. Провести тестування чат-боту.

Практичне значення отриманих результатів: даний продукт може бути використаний для спрощення лізингу автомобілів.

Проаналізовано можливості середовища розробки Visual Studio, мови програмування C#. Розроблено логіку практичних завдань та загальну концепцію представлення інформації для користувачів.

Під час аналізу предметної області було обрано та описано програмні засоби для розробки чат-бота: асинхронний фреймворк Aiogram та середовище для розробки C#. Для системи управління базами даних використано Visual Studio та хмарна платформа Amazon AWS.

Додаток є актуальним для активних користувачів месенджера Telegram. Чат-бот націлений на спрощення та полегшення пошуку та оренди автіки, саме в тому місті, де потрібно для користувача. Галузь використання – лізинг автомобілів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Платформа телеграм

Telegram — це безкоштовне програмне забезпечення для миттєвих повідомлень (IM), яке працює на кількох платформах і є програмою миттєвого обміну повідомленнями. Послуга також пропонує наскрізне зашифроване відеодзвінки,[15] VoIP, обмін файлами та кілька інших функцій. Вперше він був запущений 14 серпня 2013 року, а Android – у жовтні 2013 року. Сервери Telegram розгортаються по всьому світу, щоб зменшити трафік даних, а операційний центр наразі розташований в Дубаї. [16][17][18][19][Різні клієнти Telegram доступні для настільних і мобільних платформ, включаючи офіційні програми для Android, iOS, Windows, macOS та Linux, а також Windows Phone, що припиняється. Також є офіційний веб-інтерфейс і багато неофіційних клієнтів, які використовують протокол Telegram. Усі офіційні компоненти Telegram відкриті[20], за винятком сервера, який є закритим і є власністю. [13]

Telegram пропонує наскрізне зашифровані голосові та відеодзвінки [21] та додаткові наскрізні зашифровані «секретні» чати. Хмарні чати та групи шифруються між програмою та сервером, тому провайдери та інші треті сторони в мережі не можуть отримати доступ до даних, але сервер Telegram має ключ дешифрування. Користувачі можуть надсилати текстові та голосові повідомлення, анімовані наклейки, здійснювати голосові та відеодзвінки, а також ділитися необмеженою кількістю зображень, документів (2 ГБ на файл), місцезнаходження користувача, контактів та музики. У січні 2021 року Telegram перевищив 500 мільйонів активних користувачів на місяць. [22] У січні 2021 року це був найбільш завантажуваний додаток у світі. [двадцять три]

Telegram був запущений у 2013 році братами Миколою та Павлом Дуровими. Раніше пара заснувала російську соціальну мережу VK, яку вони покинули в 2014 році після того, як перейшли під контроль союзників президента Путіна. Павло Дуров продав свій залишився пакет акцій у VK і залишив Росію після опору урядовому тиску [24]. Микола Дуров створив протокол MTProto, який є основою месенджера, тоді як Павло Дуров надавав фінансову підтримку та інфраструктуру через свій фонд Digital Fortress [25]. Telegram Messenger заявляє, що її кінцевою метою є не отримання прибутку [26] [27], але в даний час вона не структурована як некомерційна організація [28].

Telegram зареєстровано як UK LLP [29] та US LLC. [30] У ньому не розголошується, де він орендує офіси або які юридичні особи використовує для їх оренди, посилаючись на необхідність «захистити команди від неналежного впливу» та захистити користувачів від державних вимог щодо даних [31]. З 2014[32] до початку 2015 року служба базувалася в Берліні, Німеччина, але переїхала в іншу юрисдикцію після того, як не вдалося отримати дозволи на проживання для всіх членів команди, сказав Павло Дуров. Кажуть, що після того, як Павло Дуров покинув Росію, він переїжджав з країни в країну з невеликою групою з 15 ключових членів програмістів. [24] [34] Згідно з повідомленнями ЗМІ, у Telegram є співробітники в Санкт-Петербурзі. [33] Команда Telegram зараз знаходиться в Дубаї. [35]

У жовтні 2013 року Telegram оголосив, що має 100 000 щоденних активних користувачів.24 березня 2014 року Telegram оголосив, що досяг щомісяця 35 мільйонів користувачів та 15 мільйонів активних користувачів щодня [36]. У жовтні 2014 року плани урядового нагляду Південної Кореї спонукали багатьох її громадян перейти на Telegram. [32] У грудні 2014 року Telegram оголосила, що має 50 мільйонів активних користувачів, щоденно генеруючи 1 мільярд повідомлень, і що мільйон нових користувачів щотижня підписуються на її сервіс

[37], за 5 місяців трафік подвоївся за 2 мільярди щоденних повідомлень. [38] У вересні 2015 року Telegram оголосив, що додаток налічує 60 мільйонів активних користувачів і щодня передає 12 мільярдів повідомлень. [39]

У лютому 2016 року Telegram оголосив, що має 100 мільйонів активних користувачів щомісяця, 350 000 нових користувачів підписуються щодня і щодня надсилається 15 мільярдів повідомлень. У грудні 2017 року Telegram досяг 180 мільйонів активних користувачів щомісяця. [35] У березні 2018 року Telegram досяг 200 мільйонів активних користувачів щомісяця. [41] 14 березня 2019 року Павло Дуров заявив, що «3 мільйони нових користувачів підписалися на Telegram за останні 24 години». Дуров не сказав, що спричинило сплеск нових реєстрацій, але цей період часу пов’язаний з відповідає довгострокове технологічне відключення Facebook та його сімейства додатків, у тому числі Instagram. [43]

За даними Комісії з цінних паперів і бірж США, станом на жовтень 2019 року в усьому світі було 300 мільйонів щомісячних користувачів Telegram. [44]

24 квітня 2020 року Telegram оголосив, що він досяг 400 мільйонів активних користувачів щомісяця. [22] 8 січня 2021 р. Дуров у своєму повідомленні в блозі, що Telegram перебуває на рівні "близько 500 мільйонів" активних користувачів щомісяця [45].

1.2 Телеграм-боти

Інтернет-боти, веб-боти, боти або просто боти — це програми, які запускають автоматизовані завдання (скрипти) через Інтернет. [1] Загалом, роботи можуть виконувати прості й повторювані завдання набагато швидше, ніж люди. Найпоширеніші боти використовуються для веб-сканерів, де автоматизовані скрипти отримують, аналізують і зберігають інформацію з веб-серверів. Більше половини всього веб-трафіку генерується ботами. [2]

Спроби веб-серверів обмежити роботу ботів відрізняються. Деякі сервери мають файл robots.txt, який містить правила, які керують поведінкою роботів на цьому сервері. Теоретично, будь-який невідповідний бот може бути відхилений або видалений із уражених сайтів. Якщо в опублікованому текстовому файлі немає відповідної програми/програмного забезпечення/процедури, дотримання правил є цілком добровільним. Неможливість застосувати правило або переконатися, що розробник або реалізатор бота прочитає або підтвердить його файл robots.txt. Деякі боти "хороші" – напр. павуки в пошукових системах – тоді як інші використовуються для здійснення шкідливих атак, наприклад, на політичні кампанії. [2]

Деякі боти спілкуються з користувачами Інтернет-служб за допомогою миттєвих повідомлень (IM), Інтернет-ретрансляційного чату (IRC) або інших веб-інтерфейсів, таких як Facebook-боти та Twitter-боти. Ці чат-боти можуть дозволити людям задавати запитання простою англійською мовою, а потім формулювати відповідь. Такі боти часто можуть обробляти інформацію про погоду, інформацію про поштовий індекс, спортивні результати, конвертацію валют чи інших одиниць тощо [потрібне цитування] Інші використовуються для розваг, такі як SmarterChild на AOL Instant Messenger та MSN Messenger.

Додатковою роллю бота IRC може бути прослуховування каналу розмови, коментування певних фраз, виголошених учасниками (на основі відповідності шаблону). Це іноді використовується як служба довідки для нових користувачів або для цензури нецензурної лайки.

Боти для соціальних мереж – це набори алгоритмів, які беруть на себе обов'язки повторюваних наборів інструкцій з метою встановлення послуги або зв'язку між користувачами соціальних мереж. Серед різноманітних конструкцій мережевих ботів найпоширенішими є боти чату, алгоритми, розроблені для спілкування з користувачем-людиною, та соціальні боти, алгоритми, що імітують поведінку людини, щоб спілкуватися із шаблонами, подібними до схем користувача. Історію соціального боттингу можна простежити до Алана Тьюрінга в 1950-х роках та його бачення розробки наборів навчальних програм, затверджених тестом Тьюрінга. У 1960-х Джозеф Вайценбаум створив ELIZA, комп'ютерну програму для обробки природних мов. вважається раннім показником алгоритмів штучного інтелекту. Комп’ютерні програмісти ELIZA надихнули розробляти програми, що відповідають завданням, які можуть відповідати шаблонам поведінки їх набору інструкцій. В результаті обробка природної мови стала фактором, що впливає на розвиток штучного інтелекту та соціальних ботів. І оскільки інформація та думки бачать поступове масове поширення на веб-сайтах соціальних мереж [3], інноваційні технологічні досягнення досягаються за тією ж схемою.

Звіти про політичне втручання в останні вибори, включаючи загальні вибори у США та Великобританію у 2016 році [4], встановили, що поняття ботів є більш розповсюдженим через етику, яка оскаржується між дизайном бота та дизайнером бота. За словами Еміліо Феррари, комп’ютерного науковця з Університету Південної Каліфорнії, який звітує про зв’язок ACM, [5] відсутність ресурсів, доступних для здійснення перевірки фактів та перевірки інформації,

призводить до великих обсягів неправдивих повідомлень та заяв щодо них ботів на платформах соціальних медіа. У випадку з Twitter більшість із цих ботів запрограмовані з можливостями фільтрування пошуку, які націлюються на ключові слова та фрази, що надають перевагу політичним програмам, а потім ретвітують їх. Хоча увага ботів запрограмована на поширення неперевіреної інформації на всіх платформах соціальних медіа [6], це проблема, з якою стикаються програмісти у зв'язку з ворожим політичним кліматом. Ефект бота - це те, що Ferrera повідомляє про соціалізацію ботів та користувачів, що створює уразливість до витоку особистої інформації та поляризуючих впливів поза етикою коду бота. Це підтверджує Гіллорі Крамер у своєму дослідженні, де він спостерігає поведінку емоційно нестабільних користувачів та вплив ботів на користувачів, змінюючи сприйняття реальності.

Було багато суперечок щодо використання ботів у функції автоматизованої торгівлі. Аукціонний веб-сайт eBay розпочав судові дії, намагаючись придушити сторонні компанії використовувати ботів для обходу їх веб-сайту в пошуках вигідних пропозицій; такий підхід дав негативні наслідки на eBay і привернув увагу подальших ботів. Заснована у Великобританії біржа ставок Betfair побачила настільки великий обсяг трафіку, який надходить від ботів, що вона запустила API WebService, націлений на програмістів-ботів, за допомогою якого вона може активно керувати взаємодією ботів.

Відомо, що ферми ботів використовуються в Інтернет-магазинах додатків, таких як Apple App Store та Google Play, для маніпулювання позиціями [7] або для підвищення позитивних оцінок / відгуків. [8]

Чат-ботом є стрімко зростаюча, доброякісна форма інтернет-бота. З 2016 року, коли Facebook Messenger дозволив розробникам розміщувати чат-боти на своїй платформі, спостерігається експоненціальне зростання їх використання лише на цьому форумі. За перші шість місяців для Messenger було створено 30

000 ботів, які до вересня 2017 року зросли до 100 000 [9]. Аві Бен Езра, технічний директор SnatchBot, заявив Forbes, що дані використання їхньої платформи для створення чат-ботів вказують на найближчу перспективу, що заощаджує мільйони годин людської праці, оскільки "живий чат" на веб-сайтах замінюється ботами [10].

Компанії використовують Інтернет-ботів для збільшення зацікавленості в Інтернеті та спрощення спілкування. Компанії часто використовують ботів для скорочення витрат, замість того, щоб наймати людей для спілкування зі споживачами, компанії розробили нові способи бути ефективними. Ці чат-боти використовуються для відповіді на запитання клієнтів. Наприклад, Domino's розробив чат-бот, який може приймати замовлення через Facebook Messenger. Чат-боти дозволяють компаніям розподіляти час своїх співробітників на більш важливі справи. [11]

Одним із прикладів зловмисного використання ботів є координація та функціонування автоматизованої атаки на мережеві комп’ютери, наприклад атаки відмови в обслуговуванні ботнетом. Інтернет-боти або веб-боти також можуть бути використані для здійснення шахрайства із натисканням, і нещодавно вони з'явилися навколо ігор MMORPG як боти для комп'ютерних ігор. У наш час цей тип ботів також використовується у відеоіграх, таких як PUBG. Мобільні боти PUBG також відносяться до сімейства шкідливих ботів. Інша категорія представлена спам-ботами, інтернет-ботами, які намагаються спамувати велику кількість вмісту в Інтернеті, зазвичай додаючи рекламні посилання. Понад 94,2% веб-сайтів зазнали атаки ботів. [2]

1. Існують зловмисні боти (і бот-мережі) таких типів:
    - спам-боти, які збирають адреси електронної пошти зі сторінок контактів або гостьової книги;

- завантажуються програми, які висмоктують пропускну здатність, завантажуючи цілі веб-сайти;
- скребки веб-сайтів, які захоплюють вміст веб-сайтів і повторно використовують його без дозволу на автоматично згенерованих дверних сторінках;
- реєстраційні боти, які підписують певну електронну адресу для численних служб, щоб повідомлення про підтвердження заповнювали поштову скриньку та відволікали увагу від важливих повідомлень, що свідчать про порушення безпеки. [12]
- віруси та глисти;
- DDoS-атаки;
- ботнети, зомбі-комп'ютери тощо;
- спам-боти, які намагаються перенаправити людей на шкідливий веб-сайт, який іноді можна знайти в розділах коментарів або на форумах різних веб-сайтів;
- боти перегляду створюють підроблені подання. [13] [14]

2. Боти також використовуються для викупу більш затребуваних місць для концертів, зокрема, посередниками, що продають квитки [15]. Ці боти проходять процес придбання розважальних сайтів з продажу квитків та отримують кращі місця, відтягуючи стільки місць, скільки можна.
3. Боти часто використовуються в масових багатокористувацьких онлайнових рольових іграх для отримання ресурсів, для отримання яких інакше знадобився б значний час чи зусилля; це турбує більшість онлайн-ігрових економік.
4. Боти також використовуються для збільшення кількості переглядів відео на YouTube.

5. Боти використовуються для збільшення кількості трафіку в аналітичних звітах для вилучення грошей у рекламодавців. Дослідження, проведене компанією Comscore, показало, що більше половини оголошень, що відображаються в тисячах кампаній у період з травня 2012 року по лютий 2013 року, не відображалися для користувачів.
6. у 2012 р. журналіст Персі фон Ліпінський повідомив, що виявив мільйони ботів або переглянув або переглянув погляди на CNN iReport. CNN iReport тихо видалив мільйони переглядів з рахунку так званого суперзірки iReporter Кріса Моррoу. [17] Невідомо, чи доходи від реклами, отримані CNN від підроблених переглядів, колись повертались рекламодавцям.
7. Боти можуть використовуватися на форумах в Інтернеті для автоматичного розміщення запальних чи безглуздих дописів, щоб порушити форум і розсердити користувачів.

Найбільш широко використовувана техніка боротьби з ботами – це використання CAPTCHA, яка є формою тесту Тьюрінга, що використовується для розрізнення між користувачем людини та менш витонченим ботом на основі штучного інтелекту за допомогою графічно закодованого тексту, що читається людиною. Прикладами постачальників є Recaptcha та комерційні компанії, такі як Minteye, Solve Media та NuCaptcha. Однак капчі не є надійними у запобіганні ботам, оскільки їх часто можна обійти за допомогою комп’ютерного розпізнавання символів, дірок у безпеці та навіть передачі розв’язування капчі дешевим робітникам.

Взаємодія між людьми та соціальними роботами дуже поширена у повсякденному житті, наприклад, у школах, на робочих місцях, у відеоіграх та соціальних мережах [18]. Використання ботів має кілька переваг. Боти завжди

доступні, що забезпечує швидке та зручне обслуговування клієнтів. Боти можуть використовуватися для різних цілей, таких як чат / відповіді на запитання та реклама. Боти також зменшують витрати на робочу силу, що, в свою чергу, приносить більший прибуток компанії [5]

Компанії та клієнти можуть скористатися Інтернет-ботами. Інтернет-боти дозволяють клієнтам контактувати з компаніями, не потребуючи спілкування з людиною. KLM Royal Dutch Airlines створив чат-бот, який дозволяє клієнтам отримувати посадкові талони, нагадування про реєстрацію та іншу інформацію, необхідну для польоту. [11] Залучення клієнтів зросло з тих пір, як компанії розробили чат-боти, які можуть принести користь клієнтам.

Чат-боти використовуються щодня. Google Assistant і Siri вважаються формами чатових ботів. Google Assistant і Siri дозволяють людям задавати питання та отримувати відповіді за допомогою системи ШІ. Ці технологічні досягнення позитивно вплинули на повсякденне життя людей. [Цитування]

Більш нішевим варіантом використання інтернет-ботів є автоматичне придбання кросівок через Інтернет. Існує ряд різних ботів для придбання кросівок, які працюють на декількох різних веб-сайтах про кросівки. Часто людина використовує бота, щоб або отримати рідкісне / обмежене взуття, або отримати кілька пар взуття та перепродати їх, щоб отримати досить великий прибуток. [19]

У всьому світі боти викликають дві основні проблеми: ясність та особиста підтримка. Культурне походження людей впливає на спосіб спілкування з соціальними ботами. Багато людей вважають, що боти набагато менш розумні, ніж люди, і тому вони не гідні нашої поваги. [1]

Мін-Сун Кім запропонував п'ять проблем або питань, які можуть виникнути під час спілкування з соціальним роботом, і вони уникають шкоди

почуттів людей, мінімізують нав'язування, спростування з боку інших, проблеми з ясністю та наскільки ефективні їх повідомлення.]

Соціальні роботи також віддаляють від справжніх творінь людських стосунків. [1]

1.3 Telegram Bot API

У обчислювальних програмах інтерфейс прикладного програмування (API) – це інтерфейс, який визначає взаємодію між декількома програмними додатками або змішаними апаратно-програмними посередниками. [1] Він визначає види дзвінків або запитів, які можна зробити, спосіб їх здійснення, формати даних, які слід використовувати, домовленості, яких слід дотримуватися тощо. Він також може забезпечити механізми розширення, щоб користувачі могли розширити існуючі функціональні можливості різними способами та різною мірою. [2] API може бути повністю спеціальним, специфічним для компонента або розробленим на основі галузевого стандарту для забезпечення сумісності. Завдяки приховуванню інформації, API дозволяють модульне програмування, дозволяючи користувачам використовувати інтерфейс незалежно від реалізації.

Наразі посилання на веб-API є найпоширенішим використанням цього терміна. [3] Існують також API для мов програмування, бібліотек програмного забезпечення, комп’ютерних операційних систем та комп’ютерного обладнання. API виникли в 1940-х роках, хоча термін API з’явився лише в 1960-х і 70-х роках.

При побудові додатків API (інтерфейс програмування додатків) спрощує програмування, абстрагуючись від базової реалізації та виставляючи лише об'єкти або дії, необхідні розробнику. Хоча графічний інтерфейс для поштового клієнта може надавати користувачеві кнопку, яка виконує всі дії для отримання та виділення нових електронних листів, API для введення / виведення файлів

може дати розробнику функцію, яка копіює файл з одного місця в інше без вимагаючи, щоб розробник розумів операції з файловою системою, що відбуваються за кадром. [4]

Значення терміна API розширювалось за його історію. Спочатку він описав інтерфейс лише для програм, орієнтованих на кінцевих користувачів, відомих як прикладні програми. Це походження все ще відображається в назві "інтерфейс прикладного програмування". Сьогодні термін API є ширшим, включаючи також програмне забезпечення та навіть апаратні інтерфейси. [6]

Ідея API набагато старша за термін. Британські вчені-комп'ютеристи Уілкс і Вілер працювали над модульними бібліотеками програмного забезпечення в 1940-х роках для комп'ютера EDSAC. Їхня книга «Підготовка програм для електронного цифрового комп’ютера» містить першу опубліковану специфікацію API. Джошуа Блох стверджує, що Уілкс та Уілер "приховано" винайшли "API", оскільки це більше концепція, яка виявляється, ніж винаходить. [6]

Термін "інтерфейс прикладних програм" (без суфікса -ing) вперше зафіксовано в статті "Структури даних та техніки для віддаленої комп'ютерної графіки", представленій на конференції AFIPS у 1968 р. [8] [6] Автори цієї статті використовують термін для опису взаємодії програми - графічної програми в даному випадку - з рештою комп'ютерної системи. Послідовний інтерфейс програми (що складається з викликів підпрограми Fortran) мав на меті звільнити програміста від вирішення особливостей графічного пристрою відображення та забезпечити апаратну незалежність у разі заміни комп'ютера чи дисплея. [7]

Цей термін був введений у поле баз даних К. Дж. Дайтом [9] у роботі 1974 року, що називається "Реляційний та мережевий підходи: порівняння інтерфейсу прикладного програмування" [10]. API став частиною фреймворку ANSI / SPARC для систем управління базами даних. Цей фреймворк трактував інтерфейс

прикладного програмування окремо від інших інтерфейсів, таких як інтерфейс запиту. Професіонали баз даних у 1970-х роках спостерігали, що ці різні інтерфейси можна поєднувати; досить багатий інтерфейс програми може підтримувати й інші інтерфейси. [5]

Це спостереження призвело до API, які підтримували всі типи програмування, а не лише прикладне програмування. До 1990 року API був визначений просто як "набір послуг, доступних програмісту для виконання певних завдань" технологом Карлом Маламудом. [11]

Концепція API була знову розширена з початком веб-API. Дисертація Роя Філдінга "Архітектурні стилі та дизайн мережевих архітектур програмного забезпечення" в UC Irvine у 2000 р. Окреслила передачу представницького стану (REST) та описала ідею "мережевого інтерфейсу програмування програм", який Філдінг протиставив традиційній "бібліотеці" API. [12] Веб-API XML та JSON побачили широке комерційне прийняття, починаючи з 2000 року та продовжуючи з 2021 року.

Зараз веб-API є найпоширенішим значенням терміна API. [3] Вживаний таким чином, термін API має певне перекриття за значенням із термінами протокол зв'язку та виклик віддаленої процедури.

Семантична павутина, запропонована Тімом Бернерсом-Лі в 2001 році, включала "семантичні API", які переробляли API як відкритий, розподілений інтерфейс даних, а не як програмний інтерфейс поведінки. [13] Натомість запатентовані інтерфейси та агенти набули більшого поширення.

Інтерфейс бібліотеки програмного забезпечення є одним із типів API. API описує та прописує "очікувану поведінку" (специфікацію), тоді як бібліотека є "фактичною реалізацією" цього набору правил.

Один API може мати кілька реалізацій (або жодну, будучи абстрактною) у вигляді різних бібліотек, що мають один і той же інтерфейс програмування.

Відділення API від його реалізації може дозволити програмам, написаним однією мовою, використовувати бібліотеку, написану іншою. Наприклад, оскільки Scala та Java компілюються до сумісного байт-коду, розробники Scala можуть скористатися будь-яким Java API. [14]

Використання API може змінюватися залежно від типу мови програмування. API для процедурної мови, такої як Lua, може складатися в основному з базових підпрограм для виконання коду, маніпулювання даними або обробки помилок, тоді як API для об'єктно-орієнтованої мови, такої як Java, забезпечуватиме специфікацію класів та методів її класів. [15] [16]

Мовні прив’язки – це також API. Завдяки відображенню особливостей і можливостей однієї мови з інтерфейсом, реалізованим іншою мовою, прив’язка мови дозволяє використовувати бібліотеку чи послугу, написану однією мовою, при розробці на іншій мові. [17] Такі інструменти, як SWIG та F2PY, генератор інтерфейсу Fortran-to-Python, полегшують створення таких інтерфейсів. [18]

API також може бути пов'язаний із програмною структурою: структура може базуватися на декількох бібліотеках, що реалізують декілька API, але на відміну від звичайного використання API, доступ до поведінки, вбудованої в фреймворк, опосередковується шляхом розширення його вмісту новими класами підключений до самого фреймворку.

Більше того, загальний потік керування програмою може бути поза контролем абонента та в руках фреймворку шляхом інверсії управління або подібного механізму. [19] [20]

2 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Вибір мови програмування

C# (вимовляється C-Sharp) — це об'єктно-орієнтована мова програмування з безпечною системою типів для платформи .NET. Розроблено Андерсом Галесбергом, Скоттом Вілтамутом та Пітером Голдом під егідою Microsoft Research (належить Microsoft). Синтаксис C# подібний до C++ і Java. Мова має сувору статичну типізацію, підтримує поліморфізм, перевантаження операторів, покажчики на функції-члени класу, властивості, події, атрибути, винятки, анотації у форматі XML. Прийняття багатьох своїх попередників - C++, Object Pascal, Module і Smalltalk - C#, відповідно до їх практики використання, виключає деякі моделі, які виявилися проблематичними в розробці програмного забезпечення, наприклад C#, порівняно з C++, який не передбачає множинного успадкування класів. . Станом на 2021 рік поточна стабільна версія C# 10.0 випущена в 2021 році як частина платформи .NET 6.0.

Символ # у назві мови можна інтерпретувати як дві пари знаків плюс ++, що вказують на новий крок у розвитку мови щодо C++ (подібний кроку від C до C++), так і як музичний символ, з Разом з англійською літерою C назва ноти C-Diez. Останній називає мову. Хоча символ # (octotorpe) насправді є символом цифри на більшості клавіатур і не є таким же, як символ # (Unicode U+266F), Microsoft, як автор мови, неодноразово просила своїх клієнтів прийняти цю стилізацію.

C Sharp є близьким родичем мови програмування Java. Мова Java була створена компанією Sun Microsystems як глобальний Інтернет, розроблений для постановки задач розподілених обчислень. Java заснована на популярній мові C++ і виключає потенційно небезпечні речі (наприклад, покажчики без елементів

керування за межами). Для розподілених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп'ютера чи іншого інтелектуального пристрою.

Java завоювала значну популярність і також ліцензована Microsoft. Однак з часом Sun почала звинувачувати Microsoft у тому, що вона зробила свій клон Java сумісним із платформою Windows лише тоді, коли він його створив, що суперечило концепції машинно-незалежного середовища виконання та порушувало ліцензійну угоду. Microsoft відмовилася виконувати вимоги Sun, тому з'ясування відносин набуло судового статусу. Суд постановив, що позиція Sun була справедливою, і зобов’язав Microsoft відмовитися від неліцензійного використання Java.

У цьому випадку Microsoft вирішила використати своє домінування на ринку для створення власного аналога Java - мови, якою компанія буде володіти повністю. Ця новостворена мова називається C#. Він успадковує концепції віртуальної машини Java (середовище .NET), байт-коду (MSIL) і вищої безпеки вихідного коду програми, а також враховує досвід використання програм Java.Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проєктів. Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Сьогодні C# є флагманською мовою Microsoft, оскільки він використовує нові можливості .NET. Інші мови програмування, хоча й підтримуються, вважаються такими, що мають розрив у спадковості у використанні .NET. Рядки в C# є типами посилань.

C# стандартизовано в ECMA та ISO. У серпні 2000 року корпорація Microsoft, Hewlett-Packard і Intel Corporation спонсорували стандартизацію специфікації мови C# і загальномовної інфраструктури (CLI) в організації зі стандартизації ECMA International. У грудні 2001 року ECMA опублікувала специфікацію мови C# ECMA-334. C# став стандартом ISO в 2003 році (ISO/IEC 23270: 2006 - Інформаційні технології - Мови програмування - C#). До цього ECMA встигла адаптувати еквівалентну специфікацію до другого видання C# у грудні 2002 року. У червні 2005 року ECMA затвердила версію 3 специфікації C# і переглянула ECMA-334.

Додані розділи включають часткові класи, анонімні методи, порожні типи, і генерики (аналоги шаблонів C++).У липні 2005 ECMA подала стандарти і відповідні технічні умови на ISO/IEC JTC 1 через пришвидшену процедуру (Fast-Track). Цей процес звичайно займає 6-9 місяців.

Усі оператори мови С можна розділити на такі категорії:

- Умовні оператори, у тому числі умовні оператори if та оператори вибору перемикача;
- Оператори циклу (for, while, do while);
- оператори перетворення (break, continue, return, goto);
- інші оператори (оператор «вираз», нульовий оператор).

Оператори в програмі можна об’єднати в складені оператори за допомогою фігурних дужок. Будь-який оператор у програмі може бути позначений міткою, що складається з імені, за яким слід двокрапка. За винятком складених операторів, усі оператори С закінчуються крапкою з комою «;».

Нульовий оператор містить лише крапку з комою. Під час виконання цього оператора нічого не відбувається. Зазвичай він використовується в таких випадках: - оператор In do, for, while, якщо оператор не є обов'язковим у рядку, але синтаксично потрібен принаймні один оператор; - фігурні дужки

позначаються, якщо потрібно. Обов’язково слідкуйте за оператором. Фігурне ж дужка оператором не є. Тому, якщо вам потрібно передати керування фігурними дужками, ви повинні використовувати нульовий оператор. ;} Використання операторів для беззастережного перетворення goto настійно рекомендується в практиці програмування на С, оскільки це покращує розуміння програми та можливість її модифікації. Формат цього оператора наступний: goto name-tag; ...

тег імені: оператор;

Оператор goto передає управління позначеному оператору. Оператор doomed має виконувати ту ж функцію, що й оператор goto, а використовувані мітки повинні бути унікальними, тобто ім’я мітки не може використовуватися для різних операторів програми. Ім'я тега є ідентифікатором.

Будь-який оператор у складеному операторі може мати власну мітку. За допомогою оператора goto ви можете передати керування складеним операторам. Але вам потрібно бути обережними, вводячи складені оператори, які містять оголошення змінних ініціалізацією, оскільки оголошення розміщуються перед виконуваним оператором, а значення змінних, оголошених у цьому перетворенні, не визначаються.

Оператор if - Виконання оператора if починається з оцінки виразу.

Потім реалізуйте це так:

- Якщо вираз істинний (тобто відрізняється від 0), виконується оператор.
- Якщо вираз неправильний, виконайте наступний оператор if.

Після виконання оператора if значення передається до наступного оператора програми, якщо порядок виконання операторів програми не переривається примусово оператором розгалуження.

Приклад: якщо (i <j) i++:

Оператор If-else:

Виконання оператора *if* починається з обчислення *виразу.*

Далі виконання здійснюється за наступною схемою:

- Якщо вираз істинний (тобто відмінно від 0), то виконується *оператор-1*.
- Якщо вираз помилково (тобто дорівнює 0), то виконується *оператор-2*.

Після виконання оператора if значення передається до наступного оператора програми, якщо послідовність виконання операторів програми не переривається примусово оператором розгалуження.

приклад:

якщо (i < j) i++:

інакше {j = i-3; i++;}

Цей приклад також ілюструє той факт, що замість оператора-1 і замість оператора-2 можуть бути складні структури.

Вкладені оператори, якщо дозволені. Оператор if може міститися в операторі if або в операторі else іншого оператора if. Щоб зробити програми більш читабельними, рекомендується використовувати фігурні дужки для групування операторів і конструкцій у вкладених операторах if. Якщо фігурні дужки опущені, компілятор пов’язує кожне ключове слово else з найближчим словом if без else.

Приклад:

```
int main ()
{
int t = 2, b = 7, r = 3;
if (t> b)
{
if (b <r) r = b;
}
else r = t;
```

```
return (0);
}
```

В результаті виконання цієї програми *r* стане рівним 2

Якщо ж у програмі опустити фігурні дужки, що стоять після оператора *if*, то програма буде мати наступний вигляд:

```
int main ()
{
int t = 2, b = 7, r = 3;
if (a> b)
if (b <c) t = b;
else
r = t;
return (0);
}
```

У цьому випадку *r* отримає значення рівне 3, так як ключове слово *else* відноситься до другого оператору *if*, який не виконується, оскільки не виконується умова, що перевіряється в першому операторі *if*.

Оператор if-else if. Наступний фрагмент ілюструє вкладені оператори *if:*

```
char ZNAC;
int x, y, z;
:
if (ZNAC == '-') x = y - z;
else if (ZNAC == '+') x = y + z;
else if (ZNAC == '*') x = y * z;
else if (ZNAC == '/') x = y / z;
else ...
```

З розгляду цього прикладу можна зробити висновок, що конструкції використовують вкладені оператори *if,* є досить громіздкими і не завжди достатньо надійними. Іншим способом організації вибору з безлічі різних варіантів є використання спеціального оператора вибору *switch.*

Оператор *switch* призначений для організації вибору з безлічі різних варіантів. Формат оператора наступний:

switch (вираз)
{[Оголошення]
:
[Case константне-вираз1]: [список-операторов1]
[Case константне-вираз2]: [список-операторов2]
:
:
[Default: [список операторів]]
}

Вираз після ключового слова switch в дужках може бути будь-яким виразом, дозволеним у С, який має бути цілим числом.

Значення цього виразу є ключем до вибору з кількох варіантів. Тіло оператора smitch складається з кількох операторів, представлених ключовим словом case, за яким слідує постійний вираз. Слід зазначити, що використання цілочисельних константних виразів є істотним недоліком, притаманним розглянутому оператору.

Оскільки постійний вираз оцінюється під час перекладу, він не може містити змінні або виклики функцій. Цілі чи символічні константи часто використовуються як постійні вирази.

Усі вирази в операторі switch повинні бути унікальними. Крім операторів, позначених ключовим словом, може бути фрагмент, позначений ключовим словом за замовчуванням.

Список операторів може бути порожнім або містити один або кілька операторів. Крім того, оператори switch не вимагають, щоб послідовність операторів була укладена в фігурні дужки.

Також зауважте, що в операторі *switch* можна використовувати свої локальні змінні, оголошення яких знаходяться перед першим ключовим словом *case,* проте в оголошеннях не повинна використовуватися ініціалізація.

Схема виконання оператора перемикача така:

- оцінювати вирази в дужках;

- Обчислене значення послідовно порівнюється з константним виразом після ключового слова case;

- якщо один із постійних виразів узгоджується зі значенням оператора, передати керування оператору, зазначеному відповідним ключовим словом case;

- Якщо жоден з константних виразів не ідентичний виразу, передайте керування оператору, зазначеному за ключовим словом default, а за відсутності керування наступному оператору після перемикання.

Зверніть увагу на цікаву особливість використання оператора switch: структура зі словом default може бути не останньою в тілі оператора switch. Ключові слова case і default в тілі оператора комутатора є релевантними лише під час первинної перевірки, коли визначена точка початку виконання тіла оператора комутатора. Усі оператори між початковим оператором і кінцем тіла виконуються незалежно від ключового слова, якщо тільки якийсь з операторів не передасть управління з тіла оператора *switch.* Таким чином, програміст повинен сам подбає про вихід з *case,* якщо це необхідно. Найчастіше для цього використовується оператор *break.*

Для того, щоб виконати одні й ті ж дії для різних значень вираження, можна помітити один і той же оператор декількома ключовими словами *case.*

Приклад:

```
int i = 2;
switch (i)
{
case 1: i + = 2;
case 2: i *= 3;
case 0: i / = 2;
case 4: i -= 5;
default:;
}
```

Виконання оператора switch починається з оператора, відміченого case 2. Таким чином, змінна і отримує значення, рівне 6, далі виконується оператор, позначений ключовим словом *case 0,* а потім *case 4,* мінлива *i* прийме значення *3,* а потім значення - *2.* Оператор, позначений ключовим словом *default,* не змінює значення змінної.

Розглянемо раніше наведений приклад, в якому ілюструвалося використання вкладених операторів *if,* переписаної тепер з використанням оператора *switch.*

```
char ZNAC;
int x, y, z;
switch (ZNAC)
{
case '+': x = y + z; break;
case '-': x = y - z; break;
case '*': x = y * z; break;
```

```
case '/': x = u / z; break;
default:;
}
```

Використання оператора *break* дозволяє в необхідний момент перервати послідовність виконуваних операторів у тілі оператора *switch,* шляхом передачі управління оператору, наступного за *switch.*

Відзначимо, що в тілі оператора *switch* можна використовувати вкладені оператори *switch,* при цьому в ключових словах *case* можна використовувати однакові вирази.

Приклад:

```
switch (a)
{
case 1: b = c; break;
case 2:
switch (d)
{
case 0: f = s; break;
case 1: f = 9; break;
case 2: f-= 9; break;
}
case 3: b-= c; break;
:
}
```

Оператор *for* - це найбільш загальний спосіб організації циклу. Він має наступний формат:

for (вираз 1; вираз 2; вираз 3) тіло

Вираз 1 зазвичай використовується для встановлення початкового значення змінних, керуючих циклом. *Вираз 2* - це вираз, що визначає умову, за якої тіло циклу буде виконуватися. *Вираз 3* визначає зміна змінних, керуючих циклом після кожного виконання тіла циклу.

Схема виконання оператора for:

1. Обчислюється *вираз 1.*
2. Обчислюється *вираз 2.*
3. Якщо значення *виразу 2* відмінно від нуля (істина), виконується *тіло* циклу, обчислюється *вираз 3* і здійснюється перехід до пункту 2, якщо *вираз 2* дорівнює нулю (брехня), то управління передається на оператор, наступний за оператором *for*.

Суттєво те, що перевірка умови завжди виконується на початку циклу. Це означає, що тіло циклу може жодного разу не виконатися, якщо умова виконання відразу буде хибним.

Приклад:

```
int main ()
{
int i, b;
for (i = 1; i <10; i + +) b = "i * i;" return 0;}
```

У цьому прикладі обчислюються квадрати чисел від 1 до 9. Деякі варіанти використання оператора *for* підвищують його гнучкість за рахунок можливості використання декількох змінних, керуючих циклом.

Приклад:

```
int main ()
{
int top, bot;
char string [100], temp;
```

```
for (top = 0, bot = 100; top <bot; top + +, bot -)
{
temp = string [top];
string [bot] = temp;
}
return 0;
}
```

У цьому прикладі, реализующем запис рядка символів в зворотному порядку, для управління циклом використовуються дві змінні top і bot. Відзначимо, що на місці *вираз 1* і *вираз 3* тут використовуються декілька виразів, записаних через кому, і виконуваних послідовно.

Іншим варіантом використання оператора *for* є нескінченний цикл. Для організації такого циклу можна використовувати пусте умовний вираз, а для виходу з циклу зазвичай використовують додаткову умову і оператор *break.*

Приклад:

```
for (;;)
{
...
... break;
...
}
```

Так як згідно синтаксису мови С оператор може бути порожнім, тіло оператора *for* також може бути порожнім. Така форма оператора може бути використана для організації пошуку.

Приклад:

```
for (i = 0; t [i] <10; i + +);
```

У даному прикладі мінлива циклу *i* приймає значення номера першого елемента масиву *t*, значення якого більше 10.

Оператор циклу while називається циклом з передумовою і має наступний формат:

while (вираз) тіло;

В якості виразу допускається використовувати будь-який вираз мови Сі, а як тіла будь-який оператор, в тому числі порожній або складової.

Схема виконання оператора while наступна:

1. Обчислюється вираз.
2. Якщо вираз помилково, то виконання оператора while закінчується і виконується наступний по порядку оператор. Якщо вираз істинний, то виконується тіло оператора while.
3. Процес повторюється з пункту 1.

Оператор циклу виду

for (вираз-1; вираз-2; вираз-3) тіло;

може бути замінений оператором while наступним чином:

```
вираз-1;
while (вираз-2)
{
тіло
вираз-3;
}
```

Так само як і при виконанні оператора for, в операторі while спочатку відбувається перевірка умови. Тому оператор while зручно використовувати в ситуаціях, коли тіло оператора не завжди потрібно виконувати. Всередині операторів for і while можна використовувати локальні змінні, які повинні бути оголошені з визначенням відповідних типів.

Оператор циклу do while називається оператором циклу з постусловіем і використовується в тих випадках, коли необхідно виконати тіло циклу хоча б один раз.

Формат оператора має наступний вигляд: do тіло while (вираз);

Схема виконання оператора do while:

1. Виконується тіло циклу (яке може бути складеним оператором).
2. Обчислюється вираз.
3. Якщо вираз помилково, то виконання оператора do while закінчується і виконується наступний по порядку оператор. Якщо вираз істинний, то виконання оператора триває з пункту 1.

Щоб перервати виконання циклу до того, як умова стане хибним, можна використовувати оператор break.

Оператори while і do while можуть бути вкладеними.

Приклад:

```
int i, j, k;
...
i = 0; j = 0; k = 0;
do {i + +;
j -;
while (a [k] <i) k + +;
}
while (i <30 & & j <-30);
```

Хоча визначення C# і CLI стандартизовані ISO і Ecma для забезпечення розумного та недискримінаційного захисту ліцензій (RAND) для патентних претензій, Microsoft використовує C# і CLI у своїй бібліотеці базових класів (BCL), яка є її базовою власною платформою .NET Framework. , який надає багато нестандартних класів (Advanced I/O, Windows Forms GUI, Web Services тощо).

У деяких випадках патенти Microsoft стосуються стандартів, що використовуються в .NET Framework (задокументовані Microsoft), а патенти, на які подано заявку, доступні через інші умови RAND або через Microsoft Open Specification Promise (OSP), яка публікує патентні права. Але є застереження та обговорення, що інші аспекти патентів Microsoft не охоплені, що потенційно залишає незалежним розробникам повну структуру.

Microsoft також погоджується не судитися з розробниками програмного забезпечення з відкритим кодом, що порушує авторські права у неприбуткових проєктах для частини свого фреймворку, покритого OSP. Microsoft погодився не порушувати патентних вимог щодо продуктів Novell проти платних клієнтів Novell за винятком переліку продуктів, що явно не згадують C#, .NET чи реалізацію .NET від Novell (проєкт Mono). Проте Novell дотримується точки зору, Mono не порушує жодних патентів Microsoft. Microsoft також уклала спеціальну угоду не подавати до суду на плагін браузера Moonlight, заснований на Mono від Novell.

У коментарі у червні 2009 року на новинному сайті Free Software Foundation Річард Столлман попередив, що, на його думку, «Microsoft може планувати використовувати патенти на програмне забезпечення, щоб оголосити про всі безкоштовні впровадження C#», і порадив розробникам уникати того, що він назвав «безоплатним ризиком». " пов'язано з "залежністю вільного продажу C#".

Пізніше Фонд вільного програмного забезпечення повторив своє попередження про те, що розширення обіцянок спільноти Microsoft на специфікації ECMA C# і CLI може не захистити Microsoft від реалізацій C# з відкритим кодом, оскільки багато специфічних для Windows бібліотек, включених до .NET і Mono, не реалізовані. Таким чином, більшість провідних дистрибутивів Linux, за винятком Novell SUSE Linux, не включають Mono у свої інсталяції умовчанням (хоча його і можна завантажити з репозиторіїв).

2.2 Система управління базами даних Visual Studio

Microsoft Visual Studio — це сімейство продуктів Microsoft, яке включає інтегроване середовище розробки програмного забезпечення та багато інших інструментів. Ці продукти дозволяють розробляти консольні та графічні додатки, включаючи підтримку технології Windows Forms, а також веб-сайти, веб-додатки, веб-сервіси у рідному та керованому коді для Microsoft Windows, Windows Mobile, Windows Phone, Windows Усі підтримувані платформи CE , .NET Framework, .NET Compact Framework і Microsoft Silverlight. Найважливіші версії пакетів:

Visual Studio 97 (під кодовою назвою Boston) була першою випущеною версією Visual Studio, яка вперше об'єднала різноманітні інструменти розробки програмного забезпечення. Він випустив дві версії Professional Edition і Enterprise Edition, включаючи Visual Basic 5.0, Visual C++ 5.0, Visual J++ 1.1, Visual FoxPro 5.0, і вперше запустив середовище розробки ASP - Visual InterDev. Це перша спроба Microsoft створити єдине середовище розробки для різних мов програмування: Visual C++, Visual J++, Visual InterDev і MSDN використовують єдине середовище під назвою Developer Studio. Visual Basic і Visual FoxPro використовують різні середовища розробки.

Visual Studio 6.0 (кодова назва Aspen) - випущена в червні 1998 року - остання версія Visual Studio, що працює на платформі Win9x. Все ще популярний серед програмістів, які використовували Visual Basic. Цей випуск був основним середовищем розробки додатків Windows від Microsoft до платформи .NET. Цей випуск є основою на наступні чотири роки для розробників Microsoft. Visual

Studio 6.0 — остання версія, яка включає COM-версію Visual Basic. Це також остання версія, яка включає мову програмування Visual J++. Visual Studio 6.0 доступний у двох версіях: Professional і Enterprise. Enterprise включає додаткові плагіни, яких немає в Professional, зокрема Application Performance Explorer, Automation Manager, Microsoft Visual Modeler, RemAuto Connection Manager, Visual Studio Analyzer.

Visual Studio .NET 2002 (кодова назва Rainier; збірка 7.0) — випущена в лютому 2002 року (містить .NET Framework 1.0). Пакет оновлень 1 для Visual Studio .NET (2002) був випущений у березні 2005 року. Бета-версія була доступна в 2001 році. Найбільша зміна – це впровадження менеджера коду. Програми, розроблені за допомогою Visual Studio .NET, не компілюються на машинну мову, а перетворюються у формат, який називається Microsoft Intermediate Language (MSIL) або Common Intermediate Language (CIL). При використанні програми MSIL вона автоматично компілюється на машинну мову платформи, що робить код кросплатформним, що дозволяє йому працювати на різних платформах. Однак такі програми можна використовувати лише на платформах, які підтримують загальномовну інфраструктуру. Це дає можливість використовувати програми на операційних системах Linux або Mac OS за допомогою спеціальних програм, таких як Mono і DotGNU. Пакет випускається в чотирьох версіях: Academic, Professional, Enterprise Developer і Enterprise Architect. Вперше представлена нова мова програмування C# (c-sharp), призначена для використання у Visual Studio .NET. Також була представлена наступна версія Visual J++, яка називається Visual J#. Використовуючи Visual Studio .NET, можна створювати універсальні програми та веб-сайти (за допомогою ASP.NET та веб-сервісів). У травні 2005 року був випущений пакет оновлень для Visual Studio .NET.

Visual Studio .NET 2003 (кодова назва Everett; збірка 7.1) — випущена у квітні 2003 року (включає .NET Framework 1.1). Це перша версія, яка дозволяє розробляти мобільні додатки за допомогою ASP.NET або .NET Compact Framework. Номер збірки для Visual Studio .NET 2003 — 7.1, але версія файлу — 8.0. Visual Studio .NET 2003 також був випущений у чотирьох випусках: Academic, Professional, Enterprise Developer і Enterprise Architect. Версія Enterprise Architect містила спеціальний застосунок Microsoft Visio 2002, що використовувався для побудови UML об'єктів. Пакет оновлень для Visual Studio .NET 2003 було випущено 13 вересня 2006 року.

Visual Studio 2005 (кодова назва Whidbey; Build 8.0) — випущена в кінці жовтня 2005 року (включає .NET Framework 2.0). На початку листопада 2005 року також була випущена серія продуктів Express: Visual C++ 2005 Express, Visual Basic 2005 Express, Visual C# 2005 Express тощо. 19 квітня 2006 Express Edition є безкоштовним. Пакет оновлень 1 для VS2005 і всіх випусків Express [3] був випущений 14 грудня 2006 року. 3 червня 2007 р. було випущено додатковий пакет SP1 для вирішення проблем сумісності з Windows Vista.

Visual Studio 2005 підтримує ASP .NET версії 2.0 і дозволяє підтримувати онлайн-сервіси ASP .NET. Також підтримуються всі томи SQL Server до 2005 року. Надає можливість розробляти 64-розрядні програми. Ви можете скомпілювати код своєї програми до 32-розрядного або 64-розрядного. Visual Studio 2005 включає 64-розрядну версію стандартної бібліотеки. Є ще два продукти під назвою Visual Studio Tools for Applications (VSA) і Visual Basic for Applications (VBA). Це включає підтримку Microsoft Office 2007. Пізніше була додана підтримка таких програм, як WPF, WCF, WF, LINO та .NET Framework 3.5. Visual Studio 2008. У листопаді 2007 року Microsoft випустила [4] нові продукти для Visual Studio 2008 (під кодовою назвою Orcas) і .NET Framework 3.5. Visual Studio 2008 зосереджується на розробці програм для Windows Vista,

Microsoft Office 2007 та веб-додатків. Windows Presentation Foundation і новий редактор HTML/CSS надаються для візуальної розробки. Visual Studio 2008 має понад 250 нових функцій із серйозними покращеннями в кожному випуску, включаючи Visual Studio Express і Visual Studio Team System. Інтегрований мовний запит (LINQ) заповнює прогалину між об'єктним програмуванням і даними та дає змогу розробникам зосередитися не на доступі до даних, а на роботі з ними.

Visual Studio Team System підтримує програмне керування збірками, включаючи заплановані збірки та збірки, які є результатом безперервного процесу інтеграції. Team Build забезпечує інтегровану підтримку статичного аналізу коду під час виконання збірки та тестування контролю збірки.

Значно спрощує веб-розробку за допомогою нової технології спільного використання веб-серверів для веб-сайтів із підтримкою AJAX/JSON.

Нові елементи керування ASP.NET включають покращене керування сторінками та шаблони, а Windows Communication Foundation включає вбудовану підтримку RSS та REST. .NET Framework 3.5 також включає кілька нових функцій, включаючи можливості Web 2.0, сервісно-орієнтовану архітектуру (SOA) і програми на основі програмного забезпечення + служб. Служби підтримки послідовності надають новий клас моделей програмування, які спрощують створення сервісів з підтримкою послідовності операцій за рахунок використання Windows Communication Foundation і Windows Workflow Foundation. Це дозволяє розробникам на .NET Framework створювати бізнес-логіку сервісу, використовуючи WF, та організовувати обмін повідомленнями з цим сервісом за допомогою WCF.

Підтримка інших протоколів веб-сервісів у Windows Communication Foundation, включаючи атомну транзакцію веб-служб (WS-AtomicTransaction) 1.1, WS-

ReliableMessaging 1.1, WS-Secure Conversation та координацію веб-служб (WS-Coordination) 1.1.

Text Template Transformation Toolkit — це шаблонно-орієнтований генератор коду, що входить до складу середовища.

Visual Studio 2010. Запущено 12 квітня 2010 року. Включає .NET Framework 4.0. Нова мова F# Visual C++ підтримує стандарт C++0x.

Інструменти Visual Studio 2010 не тільки сприяють розробці універсальних додатків для мобільних телефонів і персональних комп'ютерів, але й для розробки хмарних додатків. Процес тестування, налагодження та розгортання програми в «хмарі» подібний до створення програми .NET. Ще одним чудовим доповненням до Visual Studio 2010 є багатопотокові інструменти розробки з використанням некерованого коду та .NET Framework. Visual Studio 2010 має повністю перероблений інтерфейс з використанням Windows Presentation Foundation (WPF), представляє наступне покоління інструментів ASP.NET, підтримує динамічні розширення в C# і Visual Basic, використовує нові шаблони проектів, інструменти для запису тестових сценаріїв і багато нових для Windows. 7 бібліотека Visual Studio Ultimate 2010, офіційно названа Visual Studio Team System 2010, під кодовою назвою Rosario, є новим інструментом для спільної розробки додатків.

Visual Studio 2012. Опубліковано 2 серпня 2012 року. Включає .NET Framework 4.5. З'явився новий тип проекту, який дозволяє писати нативні програми (у стилі Windows Metro) для операційної системи Windows 8, а також LightSwitch, інструмент для швидкої розробки та розгортання настільних бізнес-додатків, також включений у Visual Studio. Значно оновлено зовнішній інтерфейс програми, покращено додаток для перегляду проекту (браузер рішення англійською мовою) та тестування. Загалом вийшло 5 оновлень цієї версії, останнє датується 24 серпня 2015 року.

Visual Studio 2013. 26 червня 2013 року світ розглядався як стара версія, а 17 жовтня 2013 року була нарешті випущена нова версія стандартного кодового імені Dev12. Основні зміни включають покращення інтерфейсу користувача для розробників Teams, підтримку Windows 8.1 для настільних і мобільних платформ розробки та покращення для веб-розробників. Інструменти діагностики та налагодження були перероблені та вдосконалені, були представлені нові інструменти аналізу витоку пам'яті та багато іншого. До цієї версії було 5 оновлень, останнє 20 липня 2015 року.

Visual Studio 2015. Наступна версія Visual Studio під кодовою назвою Dev14 була випущена 20 червня 2015 року. Помітною зміною є підтримка багатьох цільових платформ: на додаток до базової Windows, тепер можна створювати проекти для iOS та Android. Додано підтримку програмного забезпечення Unity для розробників комп'ютерних ігор. Був оновлений механізм автентифікації: користувач під час запуску Visual Studio синхронізується з єдиним аккаунтом Microsoft. Версія має в собі .NET Framework 4.6 та підтримку універсальної платформи Windows 10. Розробників на мові C++ потішили новим функціоналом стандарту C++14, та навіть деякими поліпшеннями з C++17.

Останнім оновленням на сьогодні є оновлення 2 від 30 березня 2016 року, яке зосереджено на стабільності та продовжує підтримувати новий стандарт мови C++.

Visual Studio 2017. Перша нестабільна версія наступної версії програми під умовною назвою «15» була випущена 30 березня 2016 року. Основними змінами є багато невеликих покращень в інтерфейсі інсталятора та різних компонентах середовища розробки. Очікується підтримка мови програмування Solidity, і ця версія була нарешті випущена під назвою Visual Studio 2017 7 березня 2017 року. Visual Studio 2019. Перша версія нової Visual Studio (попередній перегляд) була випущена в грудні 2018 року. Має 5 етапів. RC (Release Candidate) було

підтверджено в березні. Отже, через місяць, 2 квітня, вийшли Visual Studio 2019 для Windows і Visual Studio для Mac Stable. Як і в попередніх версіях, є три типи (Community, Professional, Enterprise).

2.3 Хмарна платформа Amazon AWS

Amazon Web Services (AWS) — це філія Amazon, яка надає платформи хмарних обчислень на вимогу та API для окремих осіб, корпорацій та урядів. Ці мережеві служби хмарних обчислень забезпечують різноманітну базову абстрактну технічну інфраструктуру та розподілені обчислювальні блоки та інструменти. Однією з таких послуг є Amazon Elastic Compute Cloud (EC2), яка дозволяє користувачам мати кластер віртуальних комп'ютерів, які постійно доступні через Інтернет. Версія AWS емулює більшість властивостей реального комп'ютера, включаючи апаратний процесор і графічний процесор для обробки; локальну пам'ять/пам'ять із довільним доступом; накопичувач на жорсткому диску/SSD; вибір операційної системи; мережеве та попередньо встановлене програмне забезпечення, таке як веб-сервер , управління базою даних і взаємовідносинами з клієнтами (CRM).

Технологія AWS реалізована на серверах ферм по всьому світу і підтримується дочірніми компаніями Amazon. Комісійні засновані на комбінації вибраного користувачем використання (названого моделлю виставлення рахунків), апаратного забезпечення, операційної системи, програмного забезпечення або можливостей мережі, бажаної доступності, резервування, безпеки та параметрів обслуговування. Передплатники можуть оплатити віртуальну машину AWS, виділений фізичний комп'ютер або кластер будь-якого з них. Згідно з передплатним договором [10] Amazon забезпечує безпеку системи абонента. AWS працює в багатьох географічних регіонах по всьому світу, включаючи 6 у Північній Америці. [11]

Amazon продає AWS абонентам, щоб отримати більше обчислювальної потужності швидше і дешевше, ніж будувати реальну фізичну ферму серверів. [12] Усі послуги виставляються на основі використання, але кожна служба вимірює використання по-різному. Станом на 2017 р, AWS володіє домінуючим 33% всієї хмарності (IaaS, PaaS), тоді як у наступних двох конкурентів, Microsoft Azure та Google Cloud, відповідно 18% та 9%, згідно з даними Synergy Group. [13] [14]

Станом на 2021 рік AWS включає понад 200[15] продуктів і послуг, включаючи обчислення, сховище, мережу, базу даних, аналітику, послуги додатків, розгортання, керування, машинне навчання[16], мобільні пристрої, інструменти для розробників та засоби Інтернету речей. Найпопулярнішими є Amazon Elastic Compute Cloud (EC2), Amazon Simple Storage Service (Amazon S3), Amazon Connect і AWS Lambda (безсерверні функції, які дозволяють безсерверну ETL, наприклад, між екземплярами EC2 і S3). [17]

Більшість послуг не доступні безпосередньо кінцевим користувачам, а натомість надають функціональні можливості через API, які розробники можуть використовувати у своїх програмах. До продуктів Amazon Web Services можна отримати доступ через HTTP, використовуючи стиль схеми REST і SOAP для застарілих API, і лише JSON для нових API.

Станом на січень 2021 року AWS здійснює окремі операції в 25 географічних «регіонах»: [11] 7 у Північній Америці, 1 у Південній Америці, 6 у Європі, 1 на Близькому Сході, 1 у Африці та 8 у Азіатсько-Тихоокеанському регіоні.

AWS має намір запустити ще 15 доступних регіонів і ще 5 регіонів в Австралії, Індії, Індонезії, Іспанії та Швейцарії [11].

Кожен регіон повністю розташований в межах країни, і всі його дані та послуги залишаються в межах визначеного регіону. [10] Кожна область має ряд

«зон доступу» [103], які складаються з одного або кількох дискретних центрів обробки даних, кожен із резервним живленням, мережею та підключенням, розташованих у різних кімнатах. Зони доступу не забезпечують автоматично додаткову масштабованість або резервування всередині зон, оскільки вони навмисно ізольовані одна від одної, щоб запобігти поширенню збоїв між зонами. Деякі послуги можуть працювати в зонах доступності (наприклад, S3, DynamoDB), тоді як інші можна налаштувати на реплікацію між зонами, щоб розподілити попит і уникнути простоїв через збій.

Станом на грудень 2014 року Amazon Web Services керував приблизно 1,4 мільйонами серверів у 28 зонах доступності. [104] Глобальна мережа локацій AWS Edge складається з 54 пунктів присутності у всьому світі, включаючи локації в США, Європі, Азії, Австралії та Південній Америці [105].

У 2014 році AWS заявила, що прагне досягти 100% використання відновлюваної енергії в майбутньому. [106] У Сполучених Штатах AWS співпрацює з постачальниками відновлюваної енергії, включаючи Virginia Energy Community, щоб підтримати схід США; [107] Pattern Development, січень 2015 року, будівництво та експлуатація вітрової електростанції Amazon Fowler; [108] Iberdrola Renewables, LLC , липень 2015 р., для будівництва та експлуатації вітрової електростанції Amazon у східній частині США; EDP Renewables North America, листопад 2015 р. для будівництва та експлуатації вітряної електростанції Amazon Wind Farm US Central;[109] і Tesla Motors з використанням Tesla Motors, що використовує Північну Америку, листопад 2015 р. батареї Технологія зберігання для задоволення енергетичних потреб регіону США Захід (Північна Каліфорнія). [107]

AWS також має «спливаючі вікна» по всьому світу. [110] Вони надають AWS підприємцям і стартапам у різних галузях технологій у фізичних місцях. Відвідувачі можуть попрацювати або відпочити всередині горища або дізнатись

більше про те, що вони можуть робити з AWS. У червні 2014 року AWS відкрив свій перший тимчасовий спливаючий горище в Сан-Франциско. [111] У травні 2015 року вони розширилися до Нью-Йорка [112] [113], а у вересні 2015 року – до Берліна [114]. AWS відкрили свою четверту філію в Тель-Авіві з 1 березня 2016 року по 22 березня 2016 року [115]. Спливний лофт був відкритий у Лондоні з 10 вересня по 29 жовтня 2015 р. [116] Спливні лофти в Нью-Йорку [117] та Сан-Франциско [118] закриті на невизначений час через пандемію COVID-19, тоді як Токіо залишається відкритим обмеженою кількістю. [119]

AWS також має «спливаючі вікна» по всьому світу. [110] Вони надають AWS підприємцям і стартапам у різних галузях технологій у фізичних місцях. Відвідувачі можуть працювати або відпочивати в лофті або дізнатися більше про те, що вони можуть робити з AWS. У червні 2014 року AWS відкрила свій перший тимчасовий спливаючий лофт у Сан-Франциско. [111] У травні 2015 року вони розширилися до Нью-Йорка,[112] [113] і у вересні 2015 року до Берліна. [114] AWS відкрила свою четверту філію в Тель-Авіві з 1 березня 2016 року по 22 березня 2016 року. [115] Плавучий лофт відкрився в Лондоні з 10 вересня по 29 жовтня 2015 року. [116] Плаваючі лофти в Нью-Йорку [117] і Сан-Франциско [118] закриті на невизначений термін через пандемію COVID-19, тоді як Токіо залишається відкритим з обмеженою кількістю місць. Кількість. [119]

3 РЕАЛІЗАЦІЯ ЧАТ-БОТУ ДЛЯ ЛІЗИНГУ АВТОМОБІЛІВ

3.1 Діаграма варіантів використання

Діаграма використання найпростіша – це представлення взаємодії користувача із системою, яка показує взаємозв'язок між користувачем та різними випадками використання, в яких користувач бере участь. Діаграма випадків використання може ідентифікувати різні типи користувачів системи та різні випадки використання, і часто вона супроводжується також іншими типами діаграм. Варіанти використання представлені кругами або еліпсами.

Незважаючи на те, що сам випадок використання може детально вивчити кожну можливість, діаграма прикладів використання може допомогти забезпечити огляд системи на більш високому рівні. Раніше вже було сказано, що "схеми використання – це принципи вашої системи".

Через їх спрощений характер, схеми використання можуть бути хорошим інструментом комунікації для зацікавлених сторін. Креслення намагаються імітувати реальний світ і дають зацікавленій стороні уявлення про те, як буде розроблена система. Сіау та Лі провели дослідження, щоб визначити, чи взагалі існувала дійсна ситуація для схем використання або вони були непотрібними. Було виявлено, що діаграми випадків використання передають намір системи більш спрощеним чином зацікавленим сторонам і що вони "інтерпретуються більш повно, ніж діаграми класів".

Метою діаграми використання є відображення динамічного аспекту системи. Додаткові схеми та документація можуть бути використані для забезпечення повного функціонального та технічного уявлення про систему. Вони забезпечують спрощене та графічне представлення того, що система насправді повинна робити.

Елементи діаграми використання:

- рамки системи (англ. system border) – прямоугольник із назвою у верхніх частинах та еліпсами (прецедентами) всередині. Часто може бути опущено без корисної інформації про полезну інформацію;
- актор (англ. актор) – стилізований людський персонаж, обозначаючий набір ролей користувача (розуміється в широкому змісті: людина, внешняя сутність, клас, інша система), взаємодіючого з деякою сутністю (системною, підсистемою, класом). Актори не можуть бути пов'язані між собою з іншим (за вимкнення відносин щодо обробки / дослідження);
- прецедент – еліпс із надписом, що обозначає виконувану систематичну дію (може включати можливі варіанти), що призводить до спостерігаємих акторами результатів. Надпис може бути ім'ям або описом (з точки зору актора) того, "що" робить система (а не "як"). Ім прецедента зв’язано з неперервним (атомарним) сценарієм – конкретною послідовністю дій, ілюстріруючою повіддю. Під час сценарію актори обмінюються із систематичними повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у відео UML-коментарі. З одним прецедентом може бути пов'язано кілька різних сценаріїв

На рисунках 3.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

Рис. 3.1 — Діаграма варіантів використання користувача

3.2 Структурна діаграма

На рисунку 3.2 зображена структурна діаграма розроблюваного програмного продукту, яка описує взаємодію між внутрішніми компонентами системи та ілюструє внутрішню будову розроблюваного програмного продукту.

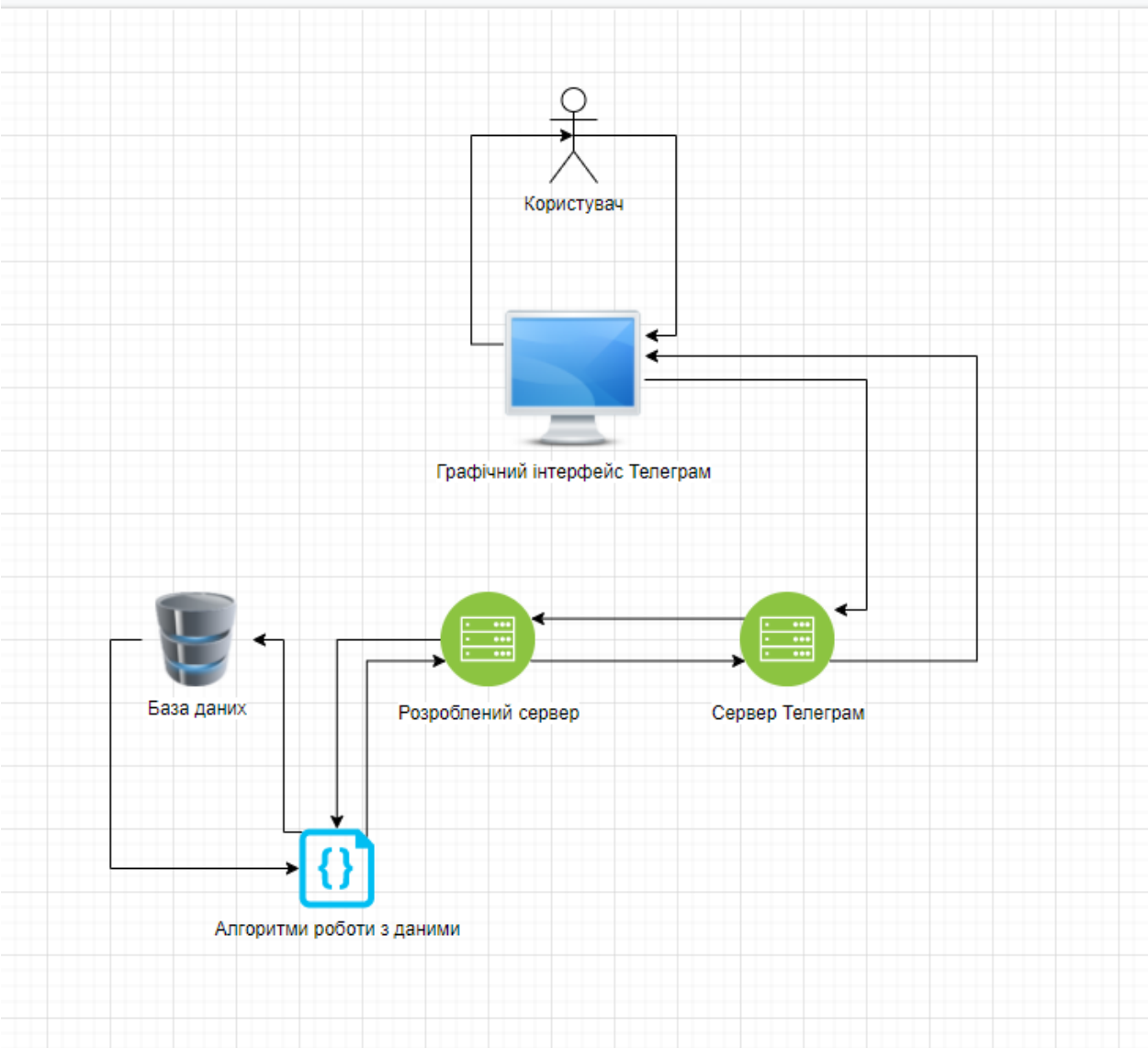

Рис. 3.2 — Структурна діаграма ІС

3.3 Тестування програмного забезпечення

Для тестування необхідно повністю перевірити функціонал програмного забезпечення.

Почати слід із заспуска бота, для цього потрібно натиснути на «Розпочати» на головному екрані (рис. 3.3), після чого ви побачите меню користувача зображено на рисунку 3.4.

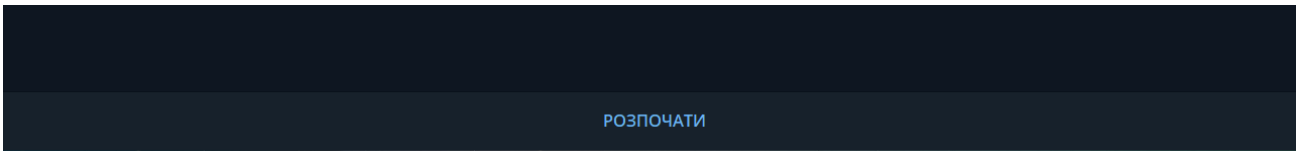

Рис. 3.3 — Заспуска бота

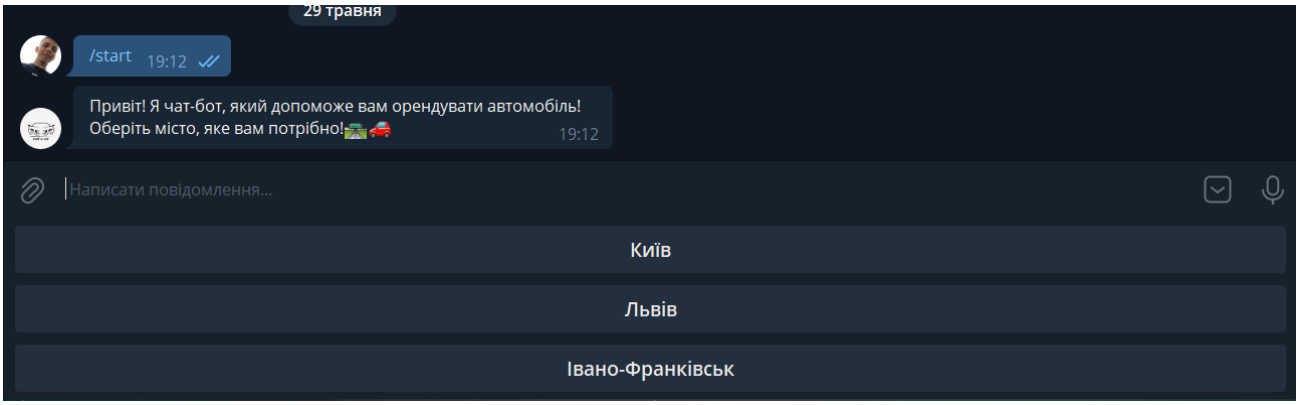

Рис. 3.4 — Меню користувача

Далі слід обрати місто в якому потрібна оренда автівки, це можна зробити натиснувши на кнопку(«Київ», «Львів», «Івано-Франківськ»). Для прикладу візьмемо місто Київ(інші міста працюють аналогічно). Натискаємо на кнопку «Київ». Після чого стають доступними ще три конопки «Всі варіанти» та «Фільтр», «Повернутись назад!» (рис. 3.5)

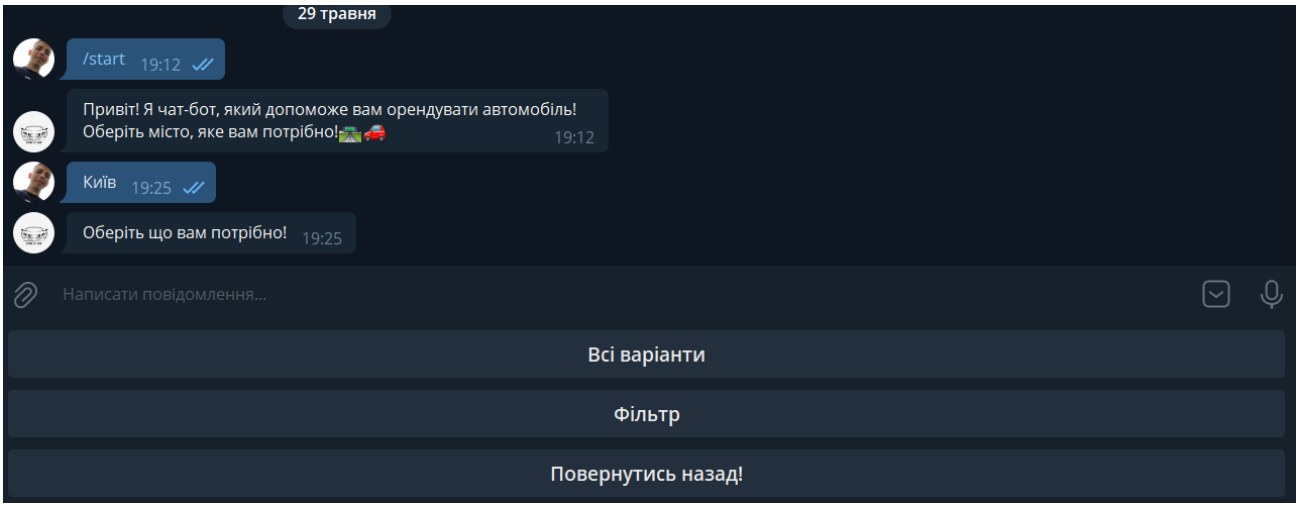

Рис. 3.5 — Меню для обраного міста

При натиску на кнопку «Всі варіанти», користувач отримає всі доступні варіанти для оренди авто. Після чого чат-бот надішле вам повідомлення з автівкою та двома кнопками «Далі» та «Повернутись назад!». Кнопка далі надає змогу переглянути інший варіант для оренди, а кнопка «Повернутись назад!» допоможе повернутись доголовного меню чат-бота(рис. 3.6 — рис. 3.6).

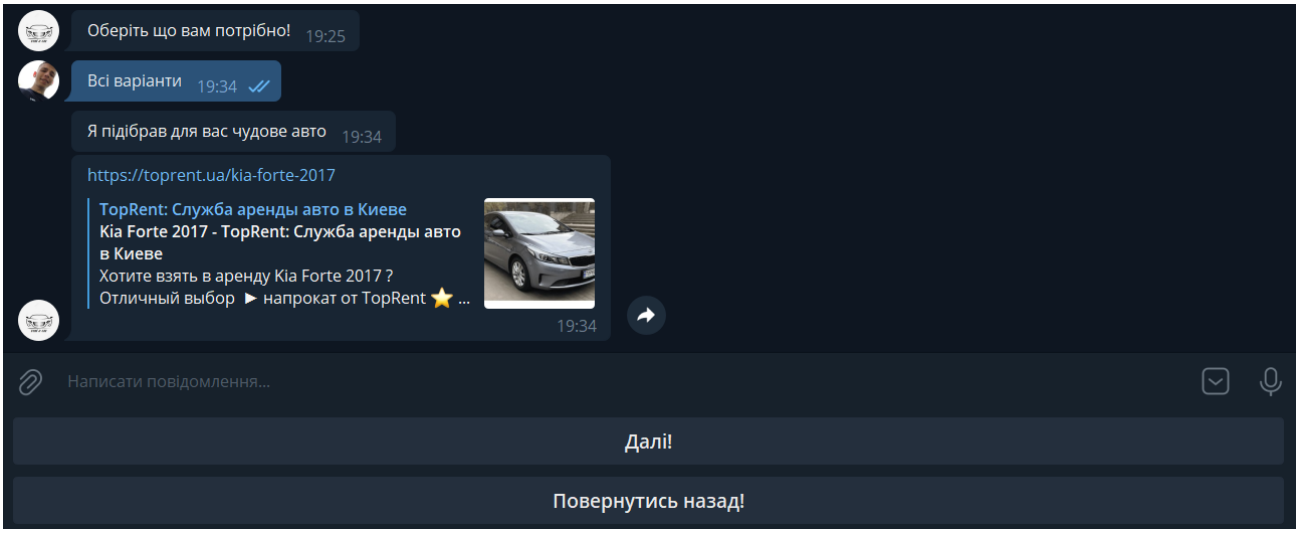

Рис. 3.6 — Робота кнопки «Всі варіанти»

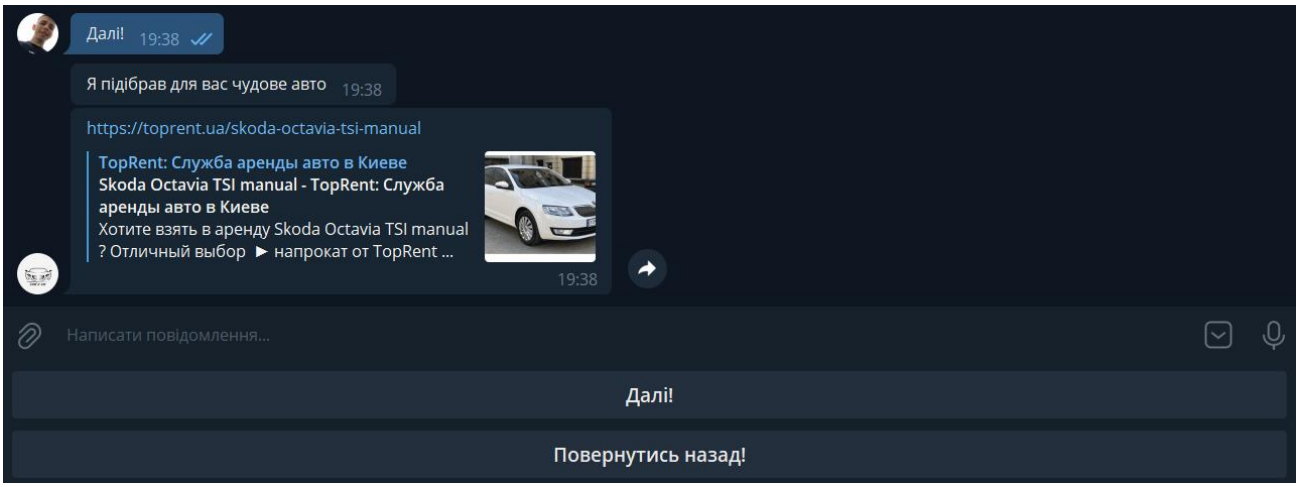

Рис. 3.7 — Робота кнопки «Далі!»

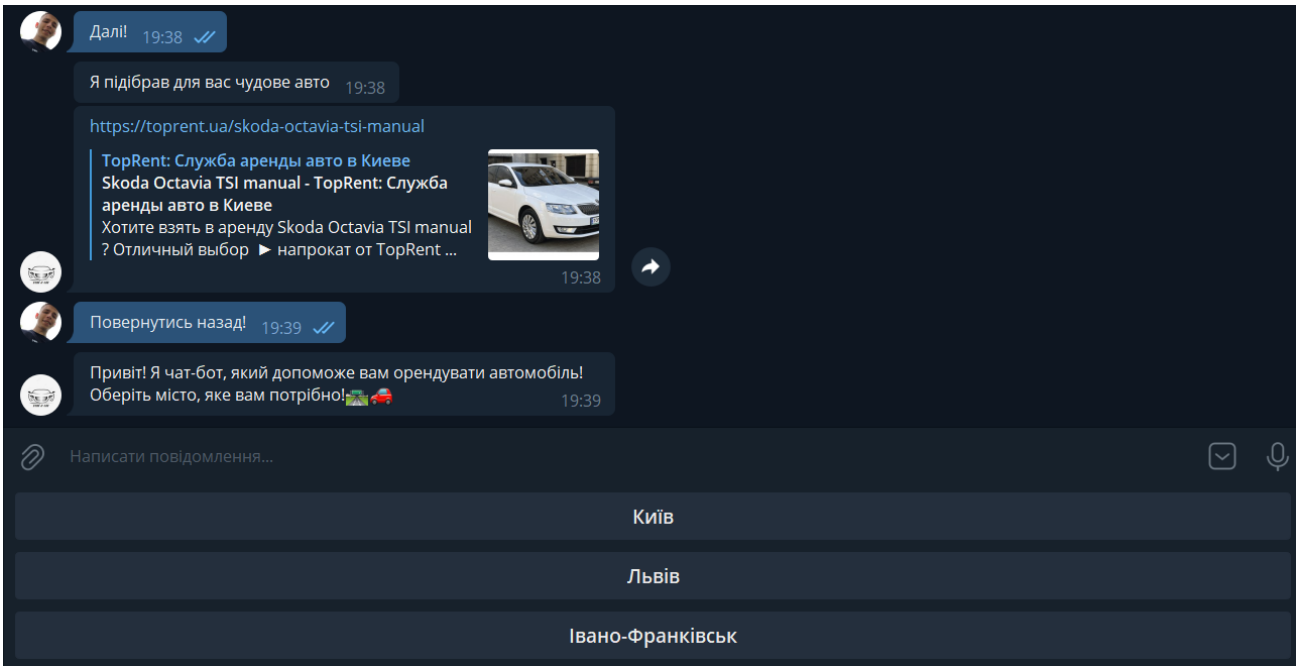

Рис. 3.8 — Робота кнопки «Повернутись назад!»

Кнопка «Фільтр» надає змогу обрати оренду авто в ціновому деапазоні який потрібний користувачу. При натиску на кнопка «Фільтр» отримаємо кнопки з ціновим діапазоном та кнопку «Повернутись назад!» зображено на рисунку 3.0

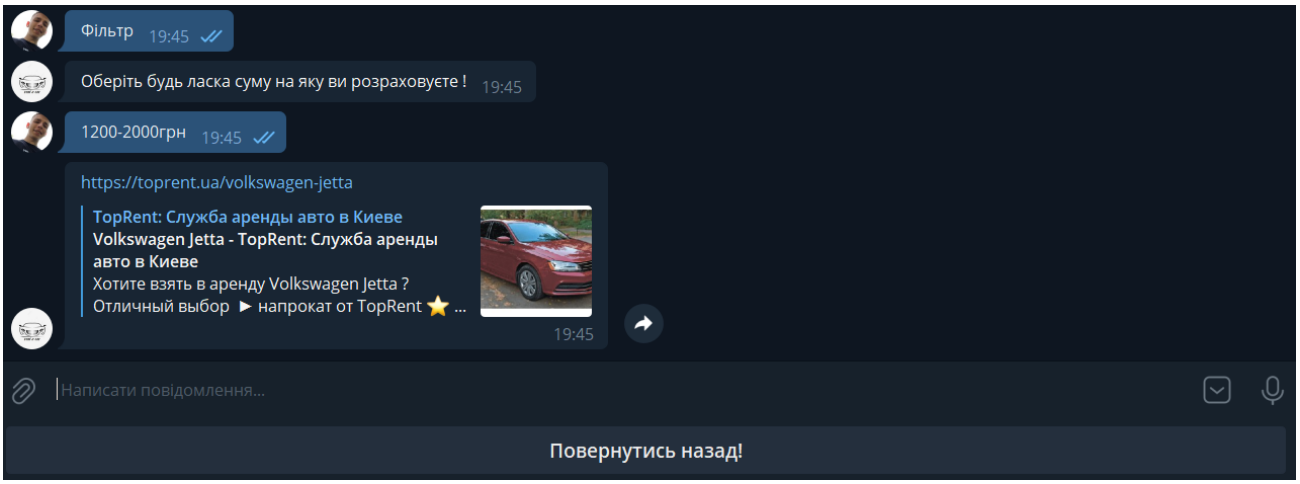

Рис. 3.9 — Функціонал кнопки «Фільтр»

3.5 Інструкція користувача

Для того, щоб орендувати автомобіль в боті слід виконати наступні дії:

- Ввести команду «старт»
- Обрати місто
- Натиснути кнопку «Всі варіанти» або «Фільтр»
- Обрати вподобаний варіант
- Перейти за посиланням
- Оформити лізинг

ВИСНОВКИ

В рамках випускної кваліфікаційної роботи був розроблений та впроваджений чат-бот. Ідея дипломного проекту полягала в створенні Telegram-боту для оптимізації та спрощення пошуку авто, які здаються в оренду.

На основі аналізу були сформовані наступні вимоги а саме: проаналізувати предметну область, обрати засоби реалізації, спроектувати і реалізувати телеграм-бота для оптимізації процесів лізингу автівок.

Під час аналізу предметної області було обрано та описано програмні засоби для розробки чат-бота: асинхронний фреймворк Aiogram та середовище для розробки C#. Для системи управління базами даних використано Visual Studio та хмарна платформа Amazon AWS.

Додаток є актуальним для активних користувачів месенджера Telegram. Чат-бот націлений на спрощення та полегшення пошуку та оренди автіки, саме в тому місті де потрібно для користувача .

Завдяки чіткому виконанню завдань, які були поставлені на початку роботи, було отримано вичерпні знання щодо методів і засобів розробки телеграм-ботів, за допомогою яких було спроектовано і розроблено повноцінного телеграм-бота для лізингу автомобілів, який повністю задовольняє вимоги, готовий до впровадження і використання в реальних умовах.

Робота пройшла апробацію на конференції : Застосування програмного забезпечення в інфокомунікаційних технологія // Розробка програмного забезпечення для інфокумунікації. Збірник тез.20.04.2022, ДУТ, м.Київ –К.: ДУТ 2022. – С.33 – 34.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Telegram Messenger". Google Play. Retrieved 18 March 2021.
2. "Telegram X". Google Play. Retrieved 25 February 2021.
3. "Telegram Messenger". App Store. Retrieved 26 March 2021.
4. "Telegram Desktop latest release". GitHub. Retrieved 20 March 2021.
5. "Telegram Desktop". Microsoft Store. Retrieved 20 March 2021.
6. "Telegram". Mac App Store. Retrieved 18 March 2021.
7. "Telegram Beta 2 – HockeyApp". rink.hockeyapp.net.
8. "Join the Telegram Messenger beta". testflight.apple.com.
9. "Version history". Telegram.
10. "Telegram Swift – HockeyApp". rink.hockeyapp.net.
11. "Telegram Messenger". Telegram Messenger LLP. Retrieved 25 February 2021.
12. "List of Telegram applications". 6 February 2014.
13. "Telegram FAQ". Telegram. Retrieved 29 March 2021.
14. "Telegram Company profile". 12 December 2019.
15. "Telegram introduces end-to-end encrypted video calls". The Next Web. Retrieved 29 March 2021.
16. EWDN, Editor (30 August 2013). "Russia's Zuckerberg launches Telegram, a new instant messenger service". Reuters. Retrieved 8 November 2020.
17. "Meet Telegram, A Secure Messaging App From The Founders Of VK, Russia's Largest Social Network". TechCrunch. Retrieved 8 November 2020.
18. "'Nobody can block it': how the Telegram app fuels global protest". The Guardian. Retrieved 7 November 2020.
19. "The Evolution of Telegram". Telegram. Retrieved 4 January 2021.
20. "Telegram Applications". Telegram.

21. "FAQ for the Technically Inclined". core.telegram.org. Retrieved 1 October 2017.
22. "Telegram hits 500M monthly active users". Telegram. Retrieved 12 January 2021.
23. "Durov Telegram". Telegram. 8 February 2021. Retrieved 10 February 2021.
24. Hakim, Danny (2 December 2014). "Once Celebrated in Russia, the Programmer Pavel Durov Chooses Exile". The New York Times. Retrieved 19 November 2015.
25. Shu, Catherine (27 October 2013). "Meet Telegram, A Secure Messaging App From The Founders Of VK, Russia's Largest Social Network". TechCrunch. Retrieved 18 March 2016.
26. Telegram F.A.Q, "...making profits will never be an end-goal for Telegram."
27. Why Telegram has become the hottest messaging app in the world, The Verge. Retrieved 25 February 2014. "Telegram operates as a non-profit organization, and doesn’t plan to charge for its services."
28. Dewey, Caitlin (23 November 2015). "The secret American origins of Telegram, the encrypted messaging app favored by the Islamic State". The Washington Post. Retrieved 31 March 2018.
29. "Telegram - Android Apps on Google Play". play.google.com. Retrieved 19 November 2015.
30. "Telegram Messenger on the App Store". App Store. Retrieved 19 November 2015.
31. Thornhill, John (3 July 2015). "Lunch with the FT: Pavel Durov". Financial Times. Retrieved 19 November 2015.
32. Brandom, Russell (6 October 2014). "Surveillance drives South Koreans to encrypted messaging apps". The Verge. Retrieved 19 November 2015.

33. Turton, William (29 September 2017). "What isn't Telegram saying about its connections to the Kremlin?". The Outline. Retrieved 11 October 2017.
34. "Telegram app free-speech advocate no stranger to Apple-FBI woes". 23 February 2016 – via www.reuters.com.
35. "This $5 Billion Encrypted App Isn't for Sale at Any Price". Bloomberg. 12 December 2017. Retrieved 22 December 2017.
36. Telegram Hits 35M Monthly Users, 15M Daily With 8B Messages Received Over 30 Days, TechCrunch, 24 March 2014
37. Telegram Reaches 1 Billion Daily Messages, Telegram, 8 December 2014
38. Telegram Hits 2 Billion Messages Sent Daily, Telegram, 13 May 2015
39. Lomas, Natasha (21 September 2015). "Telegram Now Seeing 12BN Daily Messages, up From 1BN in February". Techcrunch. Retrieved 19 November 2015.
40. Dunham, Ken; Melnick, Jim (2009). Malicious Bots: An outside look of the Internet. CRC Press. ISBN 9781420069068.
41. Zeifman, Igal. "Bot Traffic Report 2016". Incapsula. Retrieved 1 February 2017.
42. "Twitter Followers Guide". 20 November 2019
43. Howard, Philip N (18 October 2018). "How Political Campaigns Weaponize Social Media Bots". IEEE Spectrum.
44. Ferrara, Emilio; Varol, Onur; Davis, Clayton; Menczer, Filippo; Flammini, Alessandro (2016). "The Rise of Social Bots". Communications of the ACM. 59 (7): 96–104. arXiv:1407.5225. doi:10.1145/2818717. S2CID 1914124.
45. Alessandro, Bessi; Emilio, Ferrara (2016-11-07). "Social Bots Distort the 2016 US Presidential Election Online Discussion". SSRN 2982233.
46. "Touch Arcade Forum Discussion on fraud in the Top 25 Free Ranking".

47. "App Store fake reviews: Here's how they encourage your favourite developers to cheat". Electricpig. Archived from the original on 2017-10-18. Retrieved 2014-06-11.
48. "Facebook Messenger Hits 100,000 bots". 2017-04-18. Retrieved 2017-09-22.
49. Murray Newlands. "These Chatbot Usage Metrics Will Change Your Customer Service Strategy". Retrieved 2018-03-08.
50. "How companies are using chatbots for marketing: Use cases and inspiration - MarTech Today". MarTech Today. 2018-01-22. Retrieved 2018-04-10.
51. Dima Bekerman: How Registration Bots Concealed the Hacking of My Amazon Account, Application Security, Industry Perspective, December 1st 2016, In: www.Imperva.com/blog
52. Carr, Sam (July 15, 2019). "What Is Viewbotting: How Twitch Are Taking On The Ad Fraudsters". PPC Protect. Retrieved 19 September 2020.
53. Lewis, Richard (March 17, 2015). "Leading StarCraft streamer embroiled in viewbot controversy". Dot Esports. Retrieved 19 September 2020.
54. Safruti, Ido (June 19, 2017). "Why Detecting Bot Attacks Is Becoming More Difficult". DARKReading.
55. Holiday, Ryan (January 16, 2014). "Fake Traffic Means Real Paydays". BetaBeat. Archived from the original on 2015-01-03. Retrieved 2014-04-28.
56. von Lipinski, Percy (28 May 2013). "CNN's iReport hit hard by pay-per-view scandal". PulsePoint. Archived from the original on 18 August 2016. Retrieved 21 July 2016.
57. Heo, Hyun-Hee; Kim, Min-Sun (2013). "The effects of multiculturalism and mechanistic disdain for robots in human-to-robot communication scenarios". Interaction Studies. 14 (1): 81–106. doi:10.1075/is.14.1.06heo. ISSN 1572-0373.
58. "Bots to Buy Shoes With". GeoSurf. 2018-02-15. Retrieved 2020-04-26.

ДОДАТКИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ

Створення чат-боту для лізингу автомобілів мовою C#

Виконав студент 4 курсу
групи ПД-42
Машлянка Денис Валерійович
Керівник роботи
Коба Андрій Борисович

Київ-2022

МЕТА, ОБ'ЄКТ ТА ПРИДМЕТ ДОСЛІДЖЕННЯ

- Мета роботи - розробка комплексного Telegram-бота для оптимізації лізингу автомобілів на мові C#.
- Об'єкт дослідження - методи та засоби створення чат-ботів на базі платформи Telegram.
- Придмет дослідження - телеграм-бот для оптимізації лізингу автомобілів.

АНАЛОГИ

Rent a car

- Telegram Massenger;
- Швидкість обробки інформації;
- Особливості отримання данних;

АренДом

- Telegram, Facebook;
- Затримка в опрацюванні інформації;

ТЕХНІЧНІ ЗАВДАННЯ

Чат-бот має виконувати наступні функції:

- Можливість вибору міста;
- Зручний фільтр для оптимального пошуку;
- Надсилання даних для користувача;
- Повідомляти про оновлення об’єктів;

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

ДІАГРАМА КОРИСТУВАЧА

АПРОБАЦІЇ

Результати дослідження бакалавроської роботи:

1. Застосування програмного забезпечення в інфокомунікаційних технологія// Розробка програмного забезпечення для інфокумунікації. Збірник тез.20.04.2022, ДУТ, м.Київ –К.: ДУТ 2022. – С.33 – 34.
2. XIV науково-технічна конференція студентів та молодих вчених//Сучасні інформаційні технології. Збірник тез.19.05.2022 , м.Київ –К.: ДУТ 2022.

ВИСНОВКИ

В рамках випускної кваліфікаційної роботи був розроблений та впроваджений чат-бот. В процесі роботи над даним проектом було виконано наступні завдання:

- Проаналізовано предметну область;
- Обрати засоби реалізації;
- Спроектовано та реалізовано телеграм-бота;
- Побудовано діаграму варіантів використання;
- Оглянуто основні механізми роботи;
- Проведено тестування;

Дякую за увагу!