Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету(відділення))

Кафедра інформаційних систем і комп'ютерного моделювання
(повна назва кафедри (предметної циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)

(рівень вищої освіти)

на тему: «Розроблення голосового помічника для ІТ спеціаліста»

Виконав студент 2 курсу, групи ІСТС-21
спеціальності: 126
„Інформаційні системи та технології"
(шифр і назва напрямку підготовки спеціальності)

Марущак Микола Володимирович
(прізвище, ініціали)

Керівник: асист. Розумовський М.П.,
доц. Сторожук О.Л.
(прізвище, ініціали)

Рецензент: Кравецький І.М.
(прізвище, ініціали)

Львів-2023

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну

Кафедра Інформаційних систем і комп'ютерного моделювання

Рівень вищої освіти перший (бакаравський)

Спеціальність 126 *„Інформаційні системи та технології"*

ЗАТВЕРДЖУЮ:

В.о. завідувача кафедри ІСКМ

_____Сторожук О.Л.

"21" 11 2022.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Марущак Микола Володимирович

(прізвище, ім'я, по батькові)

1.Тема магістерської роботи: Розроблення голосового помічника для ІТ спеціаліста.

керівник роботи: асист. Розумовський М.П., доц. Сторожук О.Л. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "21" 11.2022 року, №С-521

2.Термін подання студентом проекту(роботи) 10 червня 2023р

3. Вихідні дані до проекту (роботи) Розробити програмне забезпечення «Голосовий помічник» мовою Python. Передбачити основні команди розробників.

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

Стан проблемної області

Інформаційне забезпечення

Програмне забезпечення

Висновки

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

Підготовка матеріалу до доповіді

6. Консультанти розділів проекту (роботи)

_____

7. Дата видачі завдання 23 листопада 2022р.

КАЛЕНДАРНИЙ ПЛАН

№, з/п	Етапи бакалаврської роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	23.11-20.12	Виконано
2.	Постановка задачі і її формалізація	20.12-13.01	Виконано
3.	Виконання вхідного етапу технології	13.01-14.04	Виконано
4.	Реалізація головних класів проекту	14.04-20.05	Виконано
5.	Виконання етапу відлагодження проєкту	20.05.-25.05.	Виконано
6.	Виконання етапу впровадження та випуску бета-версії.	25.05.-02.06.	Виконано
7.	Оформлення записки до дипломного проекту.	02.06.-10.06.	Виконано

Студент _____ Марущак М.О.
(підпис) (прізвище та ініціали)

Керівник роботи
_____ асист. Розумовський М.П.,
_____ доц. Сторожук О.Л.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 41 сторінок пояснювальної записки, 16 рисунків, 15 джерел, 1 додатку.

Розроблене програмне рішення голосового помічника для інтегрованих середовищ. Система складається з клієнта, API-шлюзу і 7 сервісів, кожен з яких відповідає за свою частину програми.

Клієнт представлений у вигляді плагіна для IntelliJ IDEA. Система розпізнає команди користувача українською мовою і надає можливість керувати середовищем розробки. Завдяки використанню модифікованого алгоритму розпізнавання іменованих сутностей, що враховує контекст користувача, точність розпізнавання команд є досить досить високою.

Ключові слова: Python, фреймворк, голосовий помічник, програмне забезпечення.

ABSTRACT

The thesis contains 41 pages of an explanatory note, 16 figures, 15 sources, 1 appendix.

A developed software solution of a voice assistant for integrated environments. The system consists of a client, an API gateway and 7 services, each of which is responsible for its own part of the program.

The client is presented as a plugin for IntelliJ IDEA. The system recognizes user commands in Ukrainian and provides an opportunity to manage the development environment. Thanks to the use of a modified algorithm for recognizing named entities that takes into account the user's context, the accuracy of command recognition is quite high.

Keywords: Python, framework, voice assistant, software.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити програмне забезпечення голосового помічника для інтегрованих середовищ з використанням мікросервісної архітектури. Система спроектувати із наяності таких частин: клієнта, API-шлюзу і 7 сервісів.

Клієнта представити у вигляді плагіна для IntelliJ IDEA. Система повинна розпізнавати команди користувача українською мовою і надавати можливість керувати середовищем розробки.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

НЛП - Обробка природної мови (Natural Language Processing).

SDK - Комплект засобів розробки програмного забезпечення (Software Development Kit).

BDD - Розробка через поведінку (Behavior Driven Development).

ВСТУП

Останнім часом використання віртуальних помічників для управління оточенням стає звичайною практикою. Ми використовуємо Google AI, Siri, Alexa, Cortana та багато інших подібних віртуальних помічників, щоб виконувати завдання за нас за допомогою простої голосової або аудіокоманди. Ви можете попросити їх відтворити музику або відкрити певний файл або будь-яке інше подібне завдання, і вони з легкістю виконували б такі дії.

Хоча ці пристрої круті, також інтригуюче розробити власного голосового автоматизованого помічника AI, який ви можете використовувати для керування робочим столом лише за допомогою голосу. Ми можемо використовувати такий ШІ, щоб спілкуватися з вами, відкривати відео, відтворювати музику та багато іншого.

Об’єктом дослідження використання Python для створення сервісу «Голосовий помічник».

Метою роботи є поступове полегшення роботи розробника прозрамного забезпечення.

Предметом дослідження є використання всього об’єму бібліотек Python для розробки сервісу наближеного до сервісу із елементами штучного інтелекту.

Практичне значення пришвидшити процес написання коду

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Огляд проблемної області.

Голосові помічники або персональні голосові помічники - це програми, що використовують обробку природної мови (НЛП) і синтез мови для виконання певних завдань за командою користувача. В даний час вони були основною частиною наших смартфонів, комп'ютерів протягом останніх кількох років, Якщо ви користувач iPhone, безумовно, ви використовуєте Siri, або якщо ви користувач Android, ви знаєте свого Google Assistant.

Голосові помічники були випущені основними гравцями за останні кілька років як Apple, Microsoft, Amazon, Google і Facebook. Тепер у нас є open-source проекти, що дають можливість людям створити власних особистих помічників і впровадити це в своїх розумних будинках.

Чим відрізняється голосовий помічник від інтелектуального помічника?

Інтелектуальний помічник - це програма, створена для допомоги користувачеві у виконанні завдань, часто використовує обробку природної мови та штучний інтелект. Однак голосовий помічник сам по собі є інтелектуальним помічником, але орієнтованим на технології, оскільки він використовує розпізнавання голосу, синтез мови поряд з обробкою природної мови (НЛП).

Як працює голосовий помічник?

Голосовий помічник відповідає на певне ключове слово як "OK Google", "Hey Siri", "Alexa" і "Hey Cortana", активує помічника, щоб чекати подальших інструкцій або завдань.

Голосові помічники – це виробництво інтеграції штучного інтелекту в нашу повсякденну рутину, вони досить розумні, щоб розпізнавати голоси своїх користувачів, а деякі створені для виконання складних завдань на основі алгоритмів машинного навчання.

Що вміє голосовий помічник?

- Створення нотаток
- Створення календаря
- Бронювання зустрічей
- Виклик на телефон
- Пошук в Інтернеті
- Створення нагадувань
- Зробити бронювання
- Пошук місцезнаходження
- Читання заміток, книг і статей

Технології з відкритим вихідним кодом надають гнучкі можливості творцям, стартапам і розробникам створювати власні споживчі продукти. Це дозволяє розробникам експериментувати, розширювати зручність використання на іншу сферу та створювати продукти, орієнтовані на сектор.

1.2. Огляд конкурентів

Mycrof - це голосовий помічник з відкритим вихідним кодом, створений для машин Linux, може бути встановлений на настільних комп'ютерах Linux і Raspberry Pi. Mycroft орієнтований на конфіденційність, який не збирає та не монетизує ваші дані.

Рис. 1.1. Майкрофт Марк 1

Mycroft AI випустив Mycroft Mark 1, який є пристроєм із підтримкою Mycroft, призначеним для розробників як вдосконалений прототип, Mycroft Mark II має бути випущений цього місяця (грудень 2018 року), який є готовим до споживачів динаміком для кінцевих користувачів.

Рис. 1.2. Майкрофт Марк II

Mycroft надав розробникам документацію для створення своїх продуктів за допомогою Mycroft на пристрої на базі Linux, тому розробники не обмежуються Raspberry Pi, але будь-якою машиною, готовою до Linux, та одноплатним комп'ютером.

В даний час Mycroft не підтримує Windows і Mac OSX, я підписався на їх список розсилки, щоб отримувати повідомлення про підтримку Mac OSX і Windows більше року немає. Сподіваємося, що незабаром будуть підтримуватися інші операційні системи.

- Платформи: Linux, Raspberry Pi
- Зручний для розробника : так
- Статус розробки: активний, на базі громади

Kalliope - це персональний помічник з голосовим управлінням з відкритим вихідним кодом, розроблений спеціально для домашньої автоматизації. Він був створений для роботи на телефонах Linux, Raspberry Pi та Android.

Kalliope створено для розробників з потужною багатою документацією, API та інструментами, зручною для розробників.

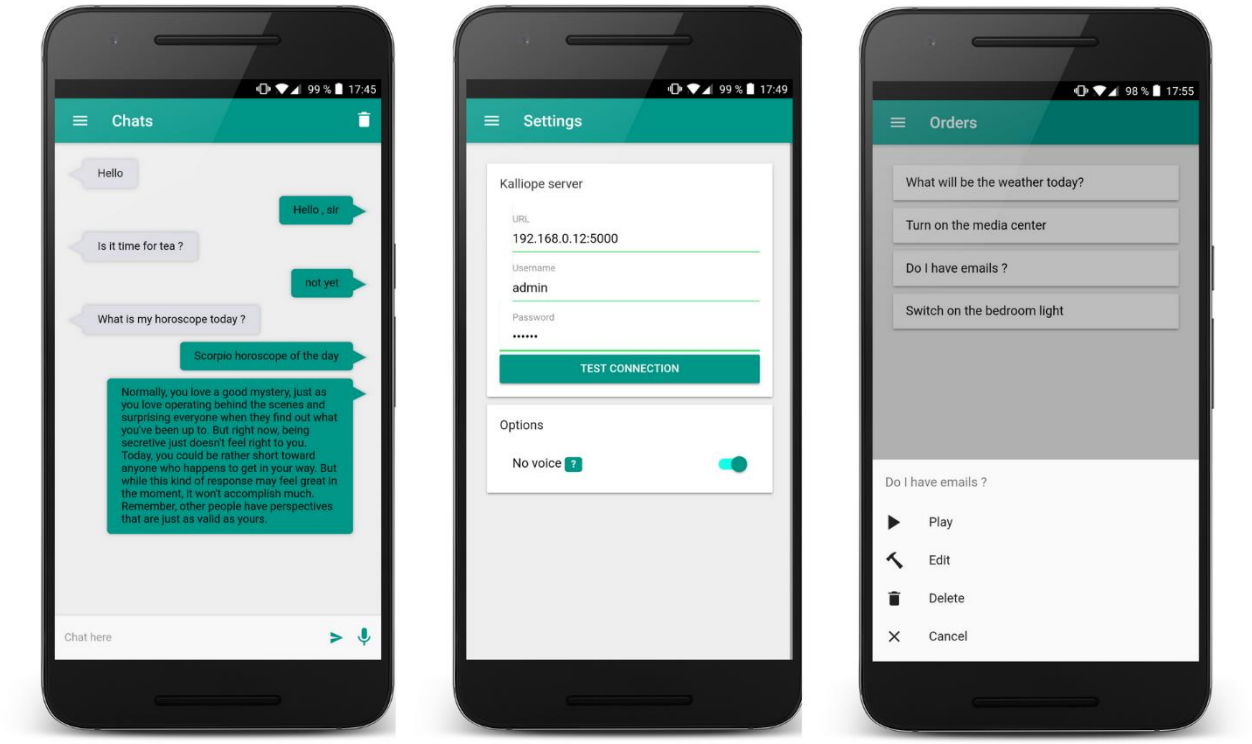

Рис. 3.3. Клієнт Kalliope Android

Оскільки Kalliope побудований на модульній архітектурі, у нього є ринок сигналів, які викликають відповідь, нейрони, які є плагінами для виконання певних дій. Ринок нейронів має основні нейрони та нейрони спільноти розробниками спільноти.

Демо голосового помічника Kalliope

Інші інструменти розробника:

- Kalliope REST API
- Інтерактивна оболонка голосового помічника Kalliope
- Веб-інтерфейс користувача Kalliope
- Додаток Kalliope для Android

Платформи: Linux/ Raspberry Pi та Android

Stephanie - це платформа з відкритим вихідним кодом, створена спеціально для додатків з голосовим управлінням, а також для автоматизації щоденних завдань, що імітують більшу частину роботи віртуального помічника.

Голосовий помічник Stephanie був випущений під ліцензією MIT, з багатою документацією, призначеною для кінцевого користувача та розробників для його встановлення, налаштування та використання. Розробники можуть легко розширити його, створивши свої модулі.

Рис. 1.4. Документація Стефані

Рис. 1.5. Open Assistant

Open Assistant - це персональний помічник з відкритим вихідним кодом. Він відповідає на голосові команди в розмовному багатому діалозі, як у демонстраційному відео. Він так довго перебував у фазі створення прототипів, але йому вдається стежити за розробниками, і як його багато разів розгалужували.

Open Assistant легко встановити, однак у ньому відсутня документація розробника. Враховуючи, що є лише один основний розробник, проект все ще знаходиться в активній розробці. Платформи: Windows, Linux і macOS.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Python

Python - це мова програмування загального призначення, яка працює майже на всіх системних архітектурах і може використовуватися для широкого кола додатків у різних областях, від веб-розробки до машинного навчання. На додаток до своєї універсальності, мова також зручна для початківців, що робить її однією з найпопулярніших доступних мов програмування.

Python – це мова з відкритим вихідним кодом, якою керує Python Software Foundation – некомерційна корпорація, яка володіє інтелектуальною власністю, пов'язаною з Python.

Люди можуть використовувати та розповсюджувати вихідний код Python безкоштовно, навіть у комерційних цілях. Сьогодні кожен, хто має комп'ютер і сильне бажання вчитися, може навчитися кодувати на Python.

Щоб завантажити останню версію Python, просто перейдіть на офіційний веб-сайт Python, натисніть Завантаження на панелі навігації та натисніть кнопку Завантажити Python. Версія за замовчуванням призначена для Windows, але Python також доступний для інших операційних систем, включаючи Linux і macOS.

Станом на 2022 рік Python 3 вважається найновішою мовною версією. Тим часом, його попередник Python 2 був закритий у січні 2020 року і більше не оновлюється виправленнями помилок, новими функціями чи виправленнями безпеки.

Python легко вивчити

Навчаючись кодувати, новачкам може бути важко зрозуміти, як працює мова програмування, особливо якщо він сильно відрізняється від їх рідної мови.

Як одна з найпростіших мов програмування для вивчення, Python використовує простий синтаксис з великою кількістю англійських ключових слів. Він був розроблений, щоб бути лаконічною мовою з високою читабельністю. Це робить його більш зручним для початківців порівняно з іншими мовами, такими як C++ та Haskell.

На додаток до простоти та послідовності, ще одним фактором, що сприяє простоті використання Python, є те, що це інтерпретована мова програмування, на відміну від скомпільованої, такої як C або C++.

Це означає, що ви можете запустити кожен рядок коду, як тільки закінчите його написати, і побачити негайні результати, вносячи корективи, якщо це необхідно. В результаті це економить програмістам багато часу, допомагаючи їм швидко і легко виявляти помилки.

Багато вакансій з високими зарплатами для розробників Python

Python є однією з найбільш затребуваних мов програмування завдяки своїй простоті та універсальності. Крім того, це одна з найбільш високооплачуваних мов програмування у 2022 році. Насправді, середня зарплата розробника Python у Сполучених Штатах становить $ 108,043 / рік.

Інше дослідження, яке оцінює вакансії LinkedIn у США та Європі, показує, що Python є номером один за попитом на роботу у 2022 році, з понад 200 000 пропозицій роботи в США та 50 000 у Європі.

Якщо ви плануєте стати веб-розробником-фрілансером, Python також може привести вас до вигідних можливостей роботи. Згідно з опитуванням про найбільш високооплачувані мови програмування для фрілансерів у 2022 році, Python є номером один у списку, із середньою ставкою 55-60 доларів на годину.

Python популярний у спільноті

Згідно з індексом спільноти програмування TIOBE, який вказує на популярність мов програмування, Python займає позицію номер один станом на квітень 2022 року.

Apr 2022	Apr 2021	Change	Programming Language	Ratings	Change
1	3	▲	Python	13.92%	+2.88%
2	1	▼	C	12.71%	-1.61%
3	2	▼	Java	10.82%	-0.41%
4	4		C++	8.28%	+1.14%
5	5		C#	6.82%	+1.91%
6	6		Visual Basic	5.40%	+0.85%
7	7		JavaScript	2.41%	-0.03%
8	8		Assembly language	2.35%	+0.03%
9	10	▲	SQL	2.28%	+0.45%
10	9	▼	PHP	1.64%	-0.19%

Рис. 2.1. Рейтинг мов програмування

Величезна спільнота Python означає, що програмісти мають багато людей, до яких можна звернутися, коли у них виникнуть запитання та проблеми з кодуванням.

Наприклад, на платформі кодування запитань і відповідей Stack Overflow є понад 1 мільйон питань, що несуть тег Python. Існує також понад 2 мільйони репозиторіїв, позначених Python на GitHub, платформі хостингу коду, яку програмісти використовують для контролю версій та спільної роботи. Користувачі Python також можуть відвідувати офіційні форуми для обговорення різних тем, пов'язаних з мовою, спільнотою та Python Software Foundation.

На додаток до своїх великих онлайн-спільнот, багато груп користувачів Python проводять щомісячні неформальні офлайн-зустрічі, щоб поділитися порадами та рекомендаціями. Існує понад 1 000 груп з більш ніж 800 000 користувачів Python по всьому світу.

Python універсальний

Окрім простоти вивчення, Python також популярний завдяки своїй універсальності. Використання мови охоплює кілька областей, включаючи науку про дані, веб-розробку та машинне навчання. Python також є кросплатформною мовою, що вказує на те, що він може працювати в різних операційних системах, таких як Windows, Linux та macOS. Крім того, ця популярна мова кодування може працювати поряд з іншими мовами.

Еталонна реалізація CPython, наприклад, написана на C і Python. Інші приклади реалізації включають Jython, написаний на Java і Python, і IronPython, створений на Python і C# і інтегрований з .NET framework.

Крім того, існує понад 100 000 бібліотек Python. Бібліотека - це набір попередньо написаного коду, який програмісти можуть використовувати для виконання певних повторюваних завдань.

Численні бібліотеки або фреймворки використовуються для різних цілей у різних областях. Програмісти можуть заощадити час, використовуючи їх замість написання та переписування часто використовуваних послідовностей коду.

Для чого використовується Python?

Деякі з найпоширеніших випадків використання мови програмування Python включають веб-розробку, автоматизацію, тестування програмного забезпечення, аналіз даних, машинне навчання та розробку ігор.

Веб-розробка

Веб-розробка - це практика створення веб-сайтів та їх підтримки. Існує дві основні частини веб-сайту – front-end і back-end. Також відомий як клієнтська сторона програми, інтерфейс відноситься до частини веб-сайту, з якою відвідувачі безпосередньо взаємодіють. Це включає такі елементи, як зображення, кнопки та навігаційне меню.

З іншого боку, серверна частина відноситься до частини веб-сайту, яку відвідувачі безпосередньо не бачать. Серверна сторона або серверна частина зберігає дані веб-сайту та гарантує, що все на інтерфейсі працює безперебійно.

Python - це внутрішня мова програмування. Веб-розробники можуть використовувати код Python для передачі даних на сервери та з них, взаємодії з базами даних та управління безпекою сайту.

Існує багато фреймворків Python, які часто використовуються у веб-розробці, включаючи:

1) Джанго. Веб-фреймворк з відкритим вихідним кодом, високим рівнем та на основі Python для швидкої розробки безпечних веб-сайтів з чистим дизайном;
2) Колба. Мікрофреймворк, написаний на Python для забезпечення простого, але розширюваного ядра, практично без залежностей від зовнішніх бібліотек;
3) CherryPy. Відомий своєю простотою, CherryPy дозволяє розробникам створювати веб-додатки за допомогою об'єктно-орієнтованого програмування, створюючи менший вихідний код за менший час;
4) web2py. Безкоштовний повноцінний фреймворк, який чудово підходить для розробки портативних веб-додатків на основі баз даних;
5) Піраміда. Виступаючи золотою серединою між мікрофреймворком і мегафреймворком, Pyramid пропонує гнучкість, контроль і розширюваність за допомогою доповнень і середовищ розробки.

Автоматизація та системні сценарії

Автоматизація означає змусити комп'ютери або машини виконувати завдання без участі людини. Тим часом, скрипти відносяться до написання коду для створення автоматизованої системи.

Як скриптова мова, Python може використовуватися для автоматизації різних завдань як для програмістів, так і для непрограмістів. Ось деякі приклади завдань, які можуть виконувати сценарії автоматизації:

- Розв'язування простих математичних задач;
- Перевірка на помилки і дублікати в файлах;
- Перейменування файлів;
- Перетворення файлів;
- Введення даних в електронну таблицю Excel;
- Відправка HTTP-запитів;
- Розрахунок курсів валют;
- Завантаження контенту;
- Сортування, надсилання та відповіді на текстові повідомлення та електронні листи;
- Скрейпінг даних з веб-сайтів;
- Тестування програмного забезпечення.

Однією з найважливіших частин розробки програмного забезпечення є процес забезпечення якості. Щоб підтримувати відмінну задоволеність клієнтів, компанії-розробники програмного забезпечення повинні забезпечити, щоб їх продукт пропонував високоякісну, послідовну та безперебійну роботу користувачів.

Однак вони також повинні швидко та ефективно випускати своє програмне забезпечення та його оновлення, щоб не відставати від конкурентів. Тут на допомогу приходить автоматизація тестування.

Автоматизоване тестування або автоматизація тестування означає використання комп'ютерів для запуску тестів, управління тестовими даними та автоматичного аналізу результатів тестування для покращення якості програмного забезпечення. Це схоже на автоматизацію, яку ми описали в попередньому розділі, але більш специфічно для сфери розробки програмного забезпечення.

Автоматизація тестування відмінно підходить для повторюваних завдань, таких як регресія і функціональне тестування. З іншого боку, тести, що вимагають суджень і думок реальних людей, таких як зручність використання, бета-тестування та A/B-тестування, краще проводити вручну.

Універсальність, масштабованість та популярність Python роблять його ідеальним для створення рішень автоматизації тестування для розробки програмного забезпечення. Ось деякі модулі Python, які інженери програмного забезпечення часто використовують для тестування програмного забезпечення:

Рамковий робот. Платформа автоматизації з відкритим вихідним кодом та розширювана для автоматизації тестових та роботизованих процесів із простим для розуміння синтаксисом із використанням ключових слів, зрозумілих людині.

pytest. Фреймворк для тестування програмного забезпечення, що дозволяє користувачам писати різні типи тестового коду на Python. Приклади включають модульні, інтеграційні, функціональні та наскрізні тести.

юніттест. Unittest, також відомий як PyUnit, є стандартною структурою модульного тестування для Python. Цей фреймворк вбудований у стандартну бібліотеку Python. Він був створений на основі JUnit для мови програмування Java.

Поводьтеся. Фреймворк Python для поведінкової розробки (BDD). Це відноситься до техніки гнучкої розробки програмного забезпечення, орієнтованої

на створення програмного забезпечення відповідно до поведінки, очікуваної користувачами, які взаємодіють з ним.

Селен. Набір різних інструментів і модулів для полегшення автоматизації браузера. Він забезпечує єдиний інтерфейс для написання тестових сценаріїв кількома мовами, включаючи Python. Існує три основні проекти селену - Selenium WebDriver, Selenium IDE та Selenium Grid.

Аналіз даних і машинне навчання

Python став невід'ємною частиною науки про дані та штучного інтелекту, оскільки мова проста у вивченні, універсальна та гнучка.

Вчені та аналітики використовують код Python для видобутку великих даних, побудови алгоритмів машинного навчання, маніпулювання та аналізу даних, а також виконання складних статистичних розрахунків. Люди також можуть використовувати цю популярну мову для створення різних типів графічних відображень даних, включаючи секторні діаграми, лінійні та гістограми, 3D-графіки та гістограми.

Мова Python також має величезну колекцію бібліотек, корисних для аналізу даних та машинного навчання. Ось кілька прикладів:

Панди. Бібліотека Python, що пропонує інструменти для маніпулювання та аналізу структур даних, числових таблиць та часових рядів.

Матплотліб. Люди використовують цю кросплатформенну бібліотеку для створення інтерактивної візуалізації даних та графічного побудови графіків для мови Python та її чисельного розширення NumPy.

NumPy. Скорочення від Numerical Python, ця бібліотека з відкритим кодом підтримує багатовимірні масиви та матриці та надає різні процедури для математичних операцій над масивами.

SciPy. Побудований на основі NumPy, SciPy або Scientific Python - це бібліотека, корисна для науково-технічних обчислень. Він містить алгоритми для

вирішення інженерних та математичних задач, таких як оптимізація, лінійна алгебра, інтерполяція та інтеграція.

TensorFlow. Бібліотека Python для розробки та навчання моделей машинного навчання з використанням високорівневих API. Він може запускати глибокі нейронні мережі для різних цілей, таких як розпізнавання зображень і обробка природної мови.

PyTorch. Бібліотека машинного навчання для Python на основі Torch, ще одна бібліотека машинного навчання для мови програмування Lua. В основному він використовується в дослідженнях глибокого навчання.

Скрепі. Фреймворк Python з відкритим вихідним кодом для сканування веб-сайтів та вилучення структурованих даних з їхніх сторінок.

Розробка ігор

Простота Python робить його чудовим для створення базових ігор або швидкого створення прототипів складних. Популярні ігри, написані на Python, включають The Sims 4, World of Tanks, EVE Online та Civilization IV.

Зручний для початківців і простий синтаксис означає, що розробники ігор-новачків можуть швидко вивчити Python і використовувати його для створення графічних інтерфейсів користувача, побудови 2D і 3D ігор, а також створення візуальних новел і ігор на основі фізики.

Професійні розробники ігор можуть використовувати код Python для швидкого створення прототипів своїх ігор та представлення ігрової візуалізації інвесторам для збору фінансування.

Щоб створити прототипи ігор якнайшвидше та найефективніше, розробники ігор можуть скористатися багатьма фреймворками Python для розробки ігор. Ось деякі з найбільш часто використовуваних з них:

Pygame. Безкоштовний набір модулів Python для написання відеоігор або створення мультимедійних програм, побудований поверх бібліотеки Simple

DirectMedia Layer (SDL). Він портативний і працює майже на кожній платформі та операційній системі.

карлик. Написаний на чистому Python, pyglet - це бібліотека, що надає об'єктно-орієнтований API для створення ігор та мультимедійних додатків. Він не має зовнішніх вимог до встановлення та пропонує вбудовану підтримку вікон та стандартних форматів зображень та аудіо.

Ківі. Безкоштовний, кросплатформенний фреймворк Python з відкритим кодом для розробки додатків з природним інтерфейсом користувача (NUI), таких як ігри з використанням технології мультитач.

Panda3D. Ігровий рушій, що надає різні підпрограми, корисні для 3D-рендеринга та розробки ігор. Він пропонує такі функції, як перегляд графіка сцени, оптимізація анімації, моніторинг продуктивності та стійкість до помилок.

Рен'Пі. Ігровий рушій для інтерактивного цифрового оповідання за допомогою слів, зображень та звуків, зосереджений на візуальних романах та іграх-симуляторах важкого життя. Крім підтримки Python, він має власну мову сценаріїв.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Архітектура програмного забезпечення.

Як архітектурний підхід до розроблення голосового помічника для інтегрованих середовищ було обрано мікросервісний підхід. Використовуючи мікросервіси, система ділиться на невеликі функціональні компоненти, які не залежать один від одного. Порівняно з традиційною монолітною архітектурою, в якій система складалася з одного великого модуля, мікросервіси розділені та працюють разом для виконання одного й того ж завдання.

Використовуючи цей підхід, система отримає такі переваги:

Швидкий час розгортання та внесення змін до конкретного сервісу.

Крім того, ці процеси стають більш керованими та передбачуваними порівняно з традиційними підходами.

Важливою перевагою з погляду розроблення продукту, що використовує моделі машинного навчання, є те, що кожний мікросервіс може бути написаний з використанням технологій, які найкраще підходять для розв'язання завдання, за яке він відповідає.

При створенні голосового помічника для розв'язання задачі розпізнавання й оброблення тексту доцільно використовувати бібліотеки та мову програмування Python, а для вирівнювання інформації краще підійде стек ELK, написаний на Java.

Кожен із сервісів може масштабуватися горизонтально, що підвищує продуктивність і стабільність системи, оскільки вона зможе коректно працювати, навіть якщо конкретний екземпляр екземпляр служби вийде з ладу.

Оскільки сервіси не пов'язані між собою тісно і взаємодіють через API-інтерфейс, це дає змогу замінювати реалізації сервісів незалежно від інших.

Архітектура розробленої системи голосового помічника для інтегрованих середовищ розроблення представлена на рис.7 і складається з 7 сервісів: клієнт - плагін для конкретного середовища розроблення програмного забезпечення

(Client), сервіс розпізнавання мовлення користувача та перетворення його на текст (Speech Recognition service), сервіс ідентифікації іменованих сутностей (Named Entities Recognition service), сервіс протоколювання (Logging service), сервіс конфігурації (Configuration service), сервіс виявлення сервісів (Discovery service) та API-шлюз (API Gateway).

Проаналізувавши наведену архітектуру, можна побачити, що користувачі взаємодіють безпосередньо з клієнтом (мобільним пристроєм або плагіном IDE), який, своєю чергою, взаємодіє з хмарним сервером, який виконує всі важкі обчислення, як-от розпізнавання команд користувача та ідентифікація іменованих сутностей тексту програми.

Кожен із сервісів має кілька екземплярів, що забезпечує відмовостійкість системи, оскільки якщо один із сервісів вийде з ладу, його замінить інший.

Аналогічно, завдяки створеній архітектурі системи, для додавання підтримки нового інтегрованого середовища розроблення програмного забезпечення або нової мови програмування достатньо створити новий клієнт, який відповідатиме за виконання розпізнаних команд користувача.

ElasticSearch використовується для забезпечення протоколювання, а також для зберігання конфіденційної інформації.

3.1.1. Клієнт

Під час розроблення програмного забезпечення для цієї дипломної роботи клієнт являє собою плагін для обраного інтегрованого середовища розроблення (IntelliJ IDEA) і мови програмування Java. Однак, у зв'язку з розділенням логіки клієнтської та серверної частин, клієнтом може бути мобільний пристрій або плагіни, розроблені для інших середовищ і мов програмування. Зокрема, плагіни можуть бути створені для таких інтегрованих середовищ розробки програмного забезпечення та текстових редакторів, як Eclipse, NetBeans, PyCharm, Sublime, Microsoft Visual Code, CLion та інших.

IntelliJ IDEA - це інтегроване середовище розробки програмного забезпечення, створене компанією JetBrains, що підтримує велику кількість мов

програмування, таких як Python, Java та JavaScript. Середовище дає змогу розробникам швидко реорганізувати вихідний код програми та має широкий набір інструментів рефакторингу. Візуальний інтерфейс системи орієнтований на підвищення продуктивності праці розробників, дозволяючи їм зосередитися на розробці функціональності. Продукт доступний у вигляді безкоштовної версії (Community) з урізаною функціональністю і у вигляді преміум-версії (Ultimate).

Середовище надає можливість розширити його функціональність шляхом створення користувацьких плагінів. Так розроблений голосовий помічник був інтегрований в обране середовище за допомогою клієнта.

При створенні плагіна для IntelliJ IDEA використовувалася мова програмування Python і SDK обраного інтегрованого середовища розробки ПЗ. Комплект засобів розробки програмного забезпечення (SDK) - це набір інструментів, необхідних для того, щоб розроблення додатка або плагіна для наявного фреймворка або платформи. Наприклад, під час розроблення програмного забезпечення з використанням Python. SDK містять двійкові файли, вихідний код програми для двійкових файлів і документацію до них для вихідного коду програми. Як правило, SDK є глобальними. Це означає, що один SDK може бути використаний у кількох проєктах і модулях.

Клієнт відповідає за отримання голосової команди від користувача, взаємодіє з інтегрованим середовищем розроблення для створення доступних іменованих сутностей, які існують у контексті проєкту, та надсилає їх за протоколом HTPP з використанням архітектурного стилю REST на віддалений сервер для виконання ресурсомістких обчислювальних операцій. На хмарний сервер надсилається голосова команда користувача. Команда користувача записується за допомогою Python Sound API.

Після отримання відповіді від сервера клієнт знову взаємодіє з SDK інтегрованого середовища для виконання розпізнаної користувацької команди.

Система дає змогу користувачеві створювати та модифікувати змінні, методи, файли, класи, а також виконувати основні операції з ними та обраним

інтегрованим середовищем розробки ПЗ. Описаний алгоритм взаємодії клієнта системи з користувачем та інтегрованим середовищем представлено на діаграмі послідовності на рис. 8.

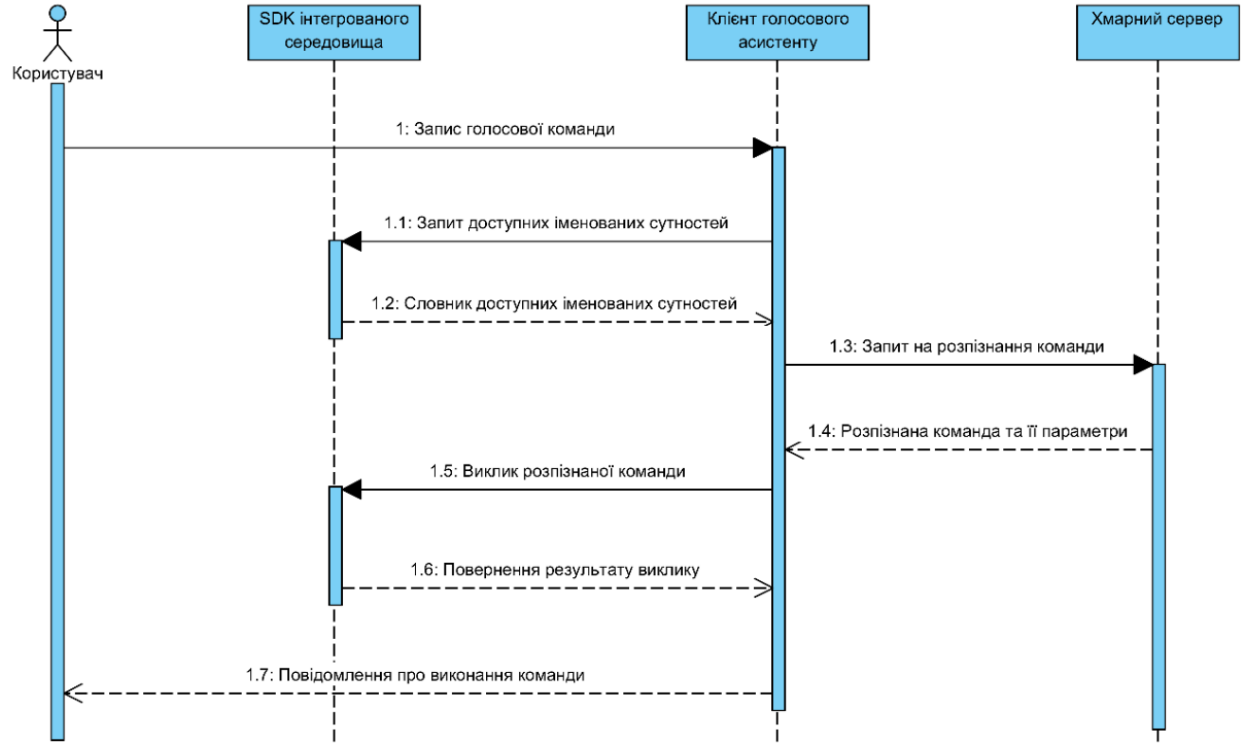

Рис. 3.1. Обробка голосової команди користувача

Уточнення розпізнаної команди користувача також відбувається на стороні клієнта, з використанням словника наявних іменованих сутностей за модифікованим алгоритмом. Це дає змогу уникнути зайвого навантаження на мережу та розв'язує питання конфіденційності, оскільки вся інформація про текст користувацької програми залишається на локальній машині користувача і не передається зовнішнім сервісам.

3.1.2. Шлюз API

Шлюз API відповідає за агрегацію отриманих даних і маршрутизацію запитів до системних сервісів. Він утворює проміжний шар між мікросервісами та клієнтами системи. Будь-які запити від користувачів або інших систем спочатку надходять до шлюзу API, який потім перенаправляє їх до кількох або одного конкретного мікросервісу.

Результати, отримані від сервісів, агрегуються і надсилаються назад клієнту у відповідь.

Теоретично, клієнт може надсилати запити напряму до будь-якого мікросервісу, але такий підхід внесе певні обмеження та проблеми в архітектуру системи. Однією з проблем, яка може виникнути, є невідповідність між інформацією, що надається кожним мікросервісом, і потребами клієнта. Залежно від розміру системи, клієнту може знадобитися виконати велику кількість звернень до різних мікросервісів для отримання даних, які його цікавлять.

У цьому випадку клієнту доведеться самостійно комбінувати дані, тому такий підхід значно ускладнює програмний код на стороні клієнта. Систему стає складніше змінити, оскільки вона безпосередньо взаємодіє з сервісами.

Великою перевагою підходу API-шлюзу є інкапсуляція внутрішньої структури системи. Замість того щоб звертатися до конкретних сервісів, клієнт робить запити тільки до API шлюзу. У цьому разі текст програми на стороні клієнта матиме набагато простіший вигляд.

З вищевказаних причин у даній магістерській дисертації ми обрали підхід використання API шлюзу замість прямої взаємодії клієнта з сервісами.

Для реалізації API шлюзу ми використовували Spring Cloud Gateway, який є прикладом реалізації патерну Gateway у середовищі мікросервісів. Він використовує можливості сервера Netty для перенаправлення запитів до внутрішніх сервісів. Spring Gateway є неблокуючим, оскільки він побудований поверх Spring WebFlux.

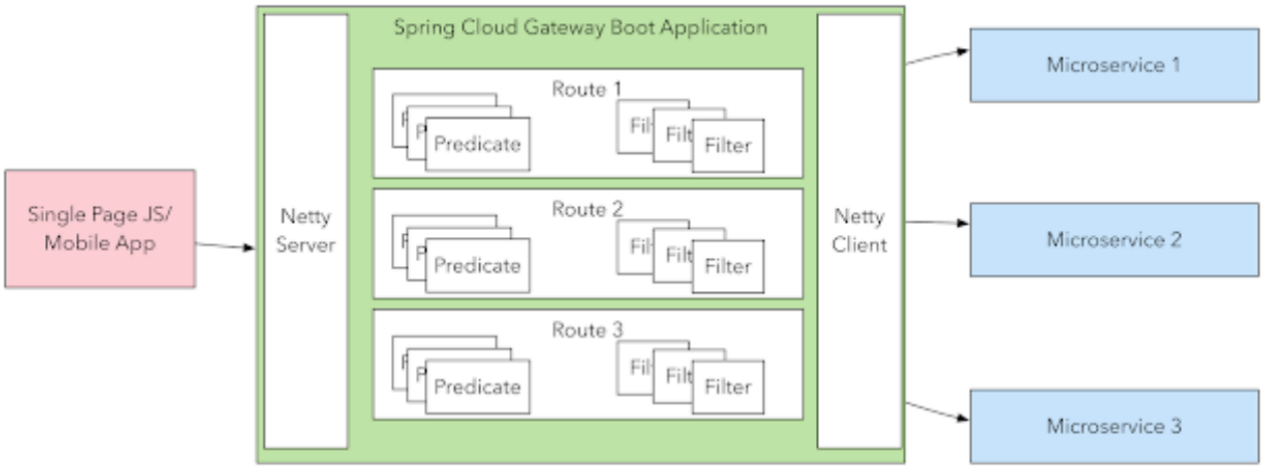

Рис. 3.2. Використання Spring API шлюзу

Шлюз Spring обробляє деякі запити, просто переспрямовуючи їх на певний мікросервіс, тоді як для інших запитів він звертається до багатьох сервісів, які містять дані, і об'єднує результати, отримані від них. Шлюз API також аутентифікує користувачів за допомогою служби авторизації та аутентифікації, перш ніж перенаправити запит на інші внутрішні сервіси.

Щоб підвищити загальну продуктивність системи, ми додали кешування користувацьких запитів на рівні шлюзу за допомогою бази даних Redis і Spring Caching. Для збору метрик і моніторингу стану системи ми використовували Spring Actuator, який доповнює API-інтерфейс шлюзу технічними характеристиками.

Redis - це система управління базами даних NoSQL, що працює зі структурами "ключ-значення". Вона використовується для кешування та реалізації брокерів повідомлень, а також як база даних.

3.1.3. Служба розпізнавання мови

Служба розпізнавання мови призначена для перетворення голосової команди користувача в текст. У створеній архітектурі голосового помічника ця служба працює на хмарному сервері та взаємодіє з ним через API-шлюз.

Системи розпізнавання мови досить складні в реалізації, оскільки існує безліч джерел варіативності, що впливають на точність розпізнавання речень.

Нижче перелічено найважливіші фактори, які необхідно враховувати при створенні сервісу для розпізнавання мовлення користувачів. Перш за все, це стиль мовлення користувача.

Кожна людина має свій власний стиль мовлення, включно з різними акцентами. Наприклад, для англійської мови такими акцентами є американська англійська, британська англійська, австралійська англійська та багато інших, оскільки це найпоширеніша мова у світі. Вимова певних слів або звуків також ускладнює розпізнавання мови системою. Навколишнє середовище також впливає на точність розпізнавання, оскільки воно впливає на рівень фонового шуму під час запису команд. Запис голосової команди, створений в ізольованій кімнаті, міститиме набагато більше шуму порівняно з аудиторією, оскільки навіть відлуння може внести шум у систему.

Також необхідно враховувати, що кожна людина має свої особливості людської мови, які залежать від багатьох чинників, зокрема від тембру і чистоти голосу. Система розпізнавання мови повинна враховувати і вирішувати перераховані вище проблеми для досягнення достатнього рівня точності.

Мова програмування Python містить безліч модулів і бібліотек, що розв'язують задачу розпізнавання мови та її перетворення на текст, зокрема ApiAI, SpeechRecognition, Google Speech Cloud, AssemblyAI, PocketSphinx, Watson Developer Cloud, WIT тощо. Для реалізації сервісу розпізнавання мови ми використовували мову програмування Python і модуль SpeechRecognition. Цей модуль надає широкий спектр інструментів від різних виробників для розпізнавання мови та перетворення її на текст.

Сервіс розпізнавання мови використовує модель Speech to Text від IBM. Рішення, що використовується, засноване на методах глибокого машинного навчання, а саме на використанні рекурентної нейронної мережі з блоком LSTM і моделі WaveNet.

Використовуване рішення забезпечує можливість розпізнавання голосових команд багатьма мовами та аудіоформатами. У результаті сервіс розпізнавання

мови повертає відповідь у форматі JSON із розпізнаним контентом у кодуванні UTF-8 за протоколом HTTP REST.

3.1.4. Служба ідентифікації іменованих сутностей

Служба ідентифікації іменованих сутностей відповідає за розпізнавання команд користувача та їхніх параметрів. Для реалізації розпізнавання використовують рекурентну нейронну LSTM-мережу та бібліотеку SpaCy, реалізовану мовою програмування Python.

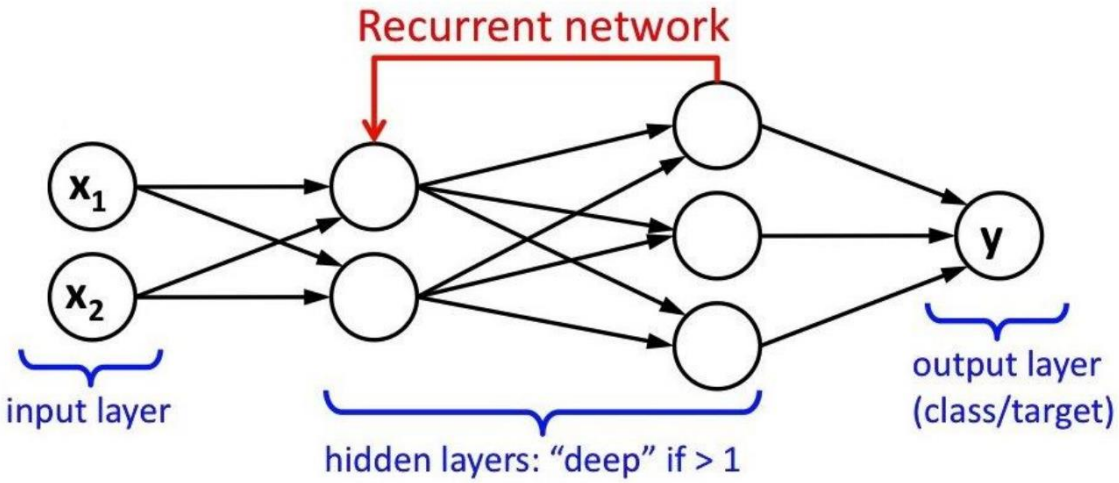

Рис. 3.3. Рекурентна нейронна мережа

SpaCy - це популярна бібліотека для аналізу тексту та обробки природної мови мовою Python. Вона розроблена на основі мови програмування Cython, яка є підмножиною Python, але має набагато кращу продуктивність. Саме тому багато компаній віддають перевагу саме їй під час реалізації під час реалізації власних проєктів.

Бібліотека містить інструменти для токенізації та класифікації тексту, визначення частин мови для слів, визначення залежності між словами в реченні, розпізнавання іменованих сутностей, інтеграції з інструментами глибокого машинного навчання та візуалізації даних.

Сервіс використовує протокол HTTP і підхід REST для передачі повідомлень. Кожен вхідний запит і розпізнані іменовані сутності реєструються за допомогою служби реєстрації подій. Таким чином, можна відстежити, які іменовані сутності були для конкретної команди користувачів.

3.1.5. Служба конфігурації

Служба конфігурації - це централізоване місце для управління зовнішньою конфігурацією для кожного сервісу незалежно від середовища. Як його реалізацію було обрано Spring Cloud Config, який спрощує управління зовнішньою конфігурацією для великих розподілених систем як на стороні сервера, так і на стороні клієнта.

До того, як мікросервіси набули широкого поширення, конфігураційні файли створювали таким чином, що при внесенні в них змін потрібно було перезапускати контейнер. Зазвичай під час розроблення системи використовується підхід із кількома різними оточеннями, кожне з яких має підтримувати свій власний конфігураційний файл, і щоб переключитися між ними, необхідно призупинити роботу всієї системи. Тому дуже важливою проблемою є те, що кодова база тісно пов'язана з конфігураційними файлами, і, відповідно, будь-які внесені зміни тягнуть за собою розгортання і повторне складання всієї системи.

Всі перераховані вище проблеми вирішуються за допомогою хмарної служби конфігурації. Перевага такого централізованого підходу до конфігурування полягає в тому, що при внесенні змін до налаштувань мікросервісу вони обробляються "на льоту", а значить, немає необхідності перебудовувати мікросервіс.

Служба конфігурації зберігає всі налаштування в SVN (система контролю версій), які можуть бути змінені під час роботи системи.

Мікросервіси, якщо їм потрібна певна конфігурація, отримують до неї доступ за допомогою REST API. Схема роботи вищевказаного сервісу показана на рис. 11.

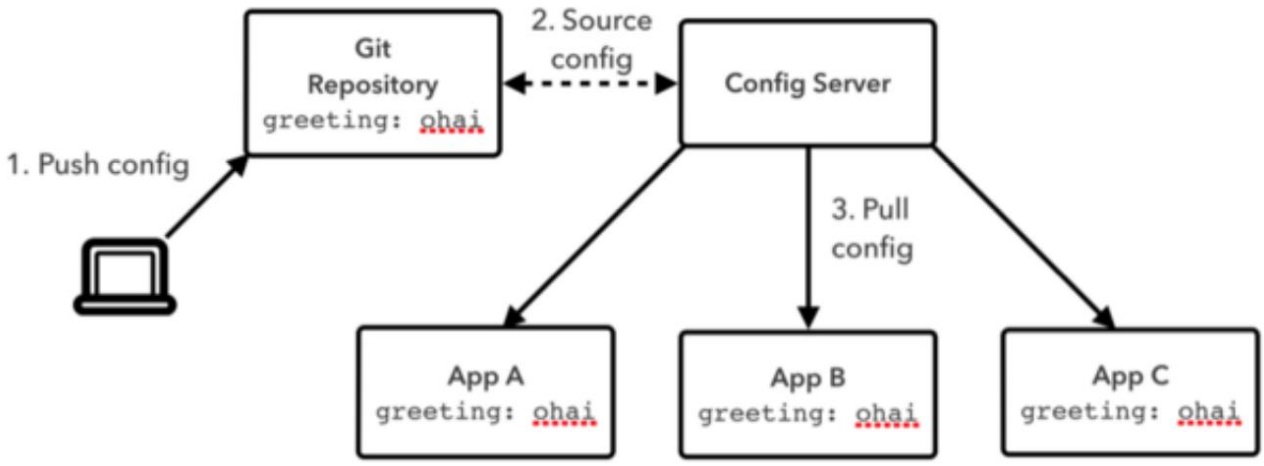

Рис. 3.4. Принцип роботи сервісу конфігурації

3.1.6. Сервіс виявлення сервісів

Під час використання мікросервісної архітектури важливу роль відіграє сервіс виявлення сервісів, оскільки кожний сервіс існує в багатьох екземплярах, тому необхідний механізм визначення адрес інших сервісів для їхнього подальшого виклику, без явного введення хоста та порту для кожного мікросервісу, що викликається. Ба більше, під час роботи системи, адреси мікросервісів можуть змінюватися в реальному часі.

Наприклад, можуть створюватися нові екземпляри мікросервісів або виходити з ладу старі. Тому механізм автоматичної реєстрації дуже важливий для виявлення сервісів. Для реалізації служби виявлення та реєстрації сервісів були використані рішення Netflix Eureka і Spring Cloud.

Netflix Eureka використовується для створення реєстру мікросервісів і автоматичного запису в нього кожного мікросервісу під час запуску. Після цього всі інші сервіси мають можливість викликати його, використовуючи ідентифікатор сервісу, який був введений під час реєстрації та є унікальним.

Для спрощення виявлення сервісів і створення реєстру використовується Spring Cloud, а як балансувальник - Ribbon. Він балансує запити до сервера на клієнті. Перш ніж зробити запит, клієнт повинен звернутися до мікросервісу виявлення сервісів, щоб дізнатися необхідну інформацію, таку як імена хостів, IP-адреси, типи портів тощо.

Після цього Ribbon вибирає один екземпляр з побудованого списку за допомогою Round-Robin, до якого і виконується запит. Щоб не робити щоразу запит до служби ідентифікації сервісу, отримані від неї дані записуються в кеш для кожного клієнта. Метод розподілу запитів від клієнта показано на рис. 12.

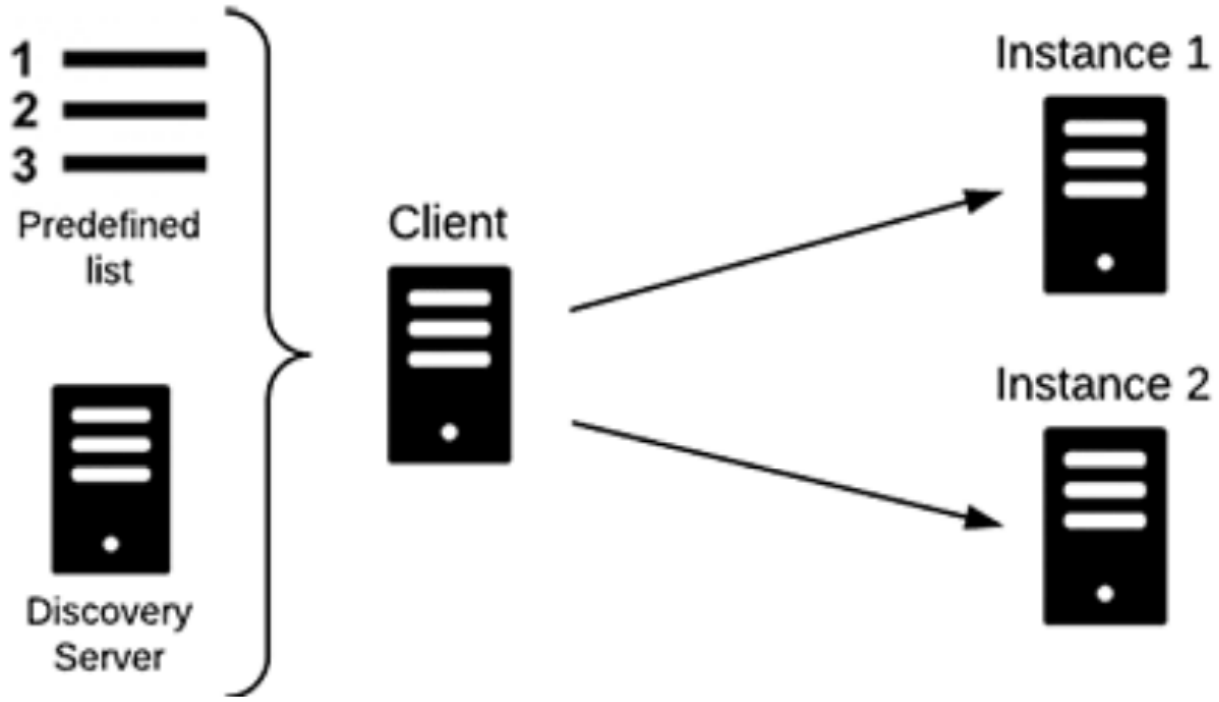

Рис. 3.5. Схема балансування на стороні клієнта

Перевагами балансування на стороні клієнта є стійкість і децентралізація, тому цей підхід було обрано для реалізації в цьому дипломному проєкті.

3.1.7. Служба ведення журналу

Оскільки всі служби працюють на різних портах і хостах, вам потрібне централізоване, зовнішнє місце для зберігання інформації про події, що відбуваються в голосовому помічнику. Розподілена система протоколювання дає змогу переглядати й аналізувати інформацію про всі події, що відбуваються в багатьох різних мікросервісах системи, і спрощує процес налагодження та виявлення помилок.

Для передачі повідомлень сервіс використовує протокол HTTP і підхід REST. Кожен вхідний запит і розпізнані іменовані сутності реєструються за

допомогою служби реєстрації подій. Таким чином, можна відстежити, які іменовані сутності були для певної користувацька команда. Щоб вирішити проблему реєстрації подій, ELK обрав набір технологій, а саме Logstash, ElasticSearch та Kibana.

Filebeat + Logstash - це динамічний перетворювач інформації з широким набором доповнень і потужною взаємодією з ElasticSearch. Оскільки Filebeat збирає файли журналів від кожного сервісу в системі та записує їх у сховище ElasticSearch, він має бути встановлений на всіх серверах з екземплярами мікросервісів. Filebeat - це програма, єдиним завданням якої є надсилання журналів від кожного сервісу в систему Logstash, що дає змогу встановити компонент Logstash тільки в одному місці. Вона має невеликий розмір і використовує мало системних ресурсів.

ElasticSearch - це пошукова, розподілена та аналітична система, розроблена для максимальної надійності, горизонтальної масштабованості та простого управління інформацією.

Kibana використовується для забезпечення візуалізації даних через користувацький інтерфейс. На рис. 13 показано принципи взаємодії між компонентами набору ELK. Кожен сервіс записує події, що відбуваються під час його роботи, у певний файл. Filebeat, який встановлюється на кожному сервері, де потрібен збір даних для подальшого перенаправлення, передає дані з цих файлів компоненту Logstash. Він приймає передані дані, конвертує їх у формат JSON і відправляє в ElasticSearch для подальшого зберігання. Kibana отримує доступ до записів в ElasticSearch і виконує візуалізацію для клієнта.

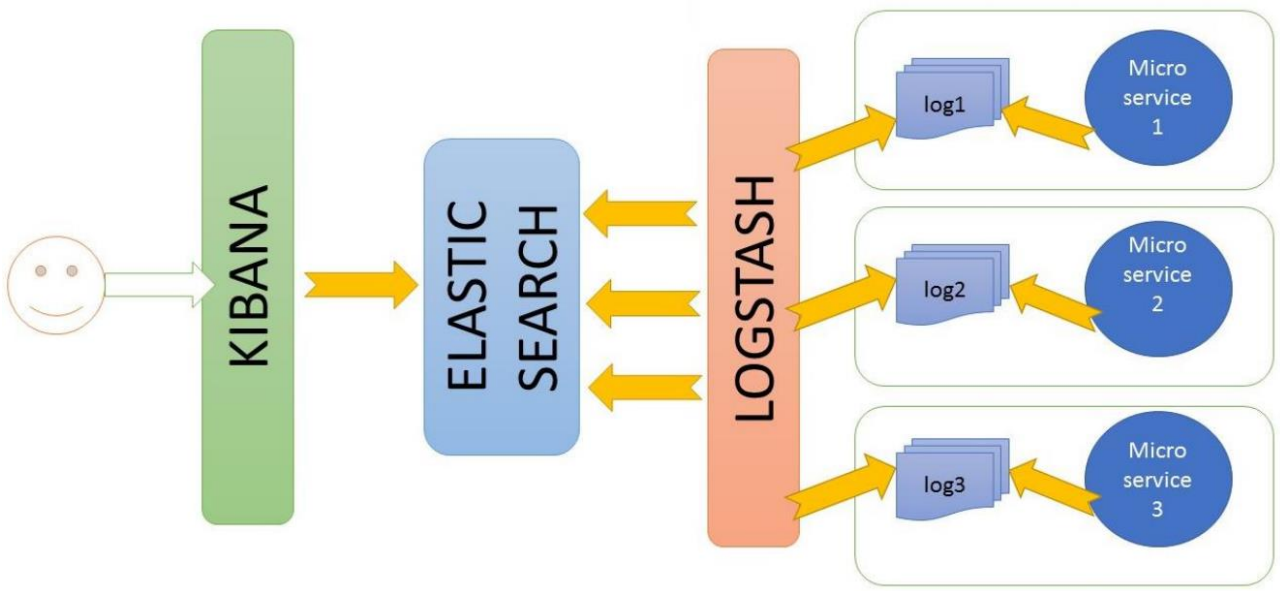

Рис. 3.6. Стек ELK для ведення логів

3.2. Тестування проєкту

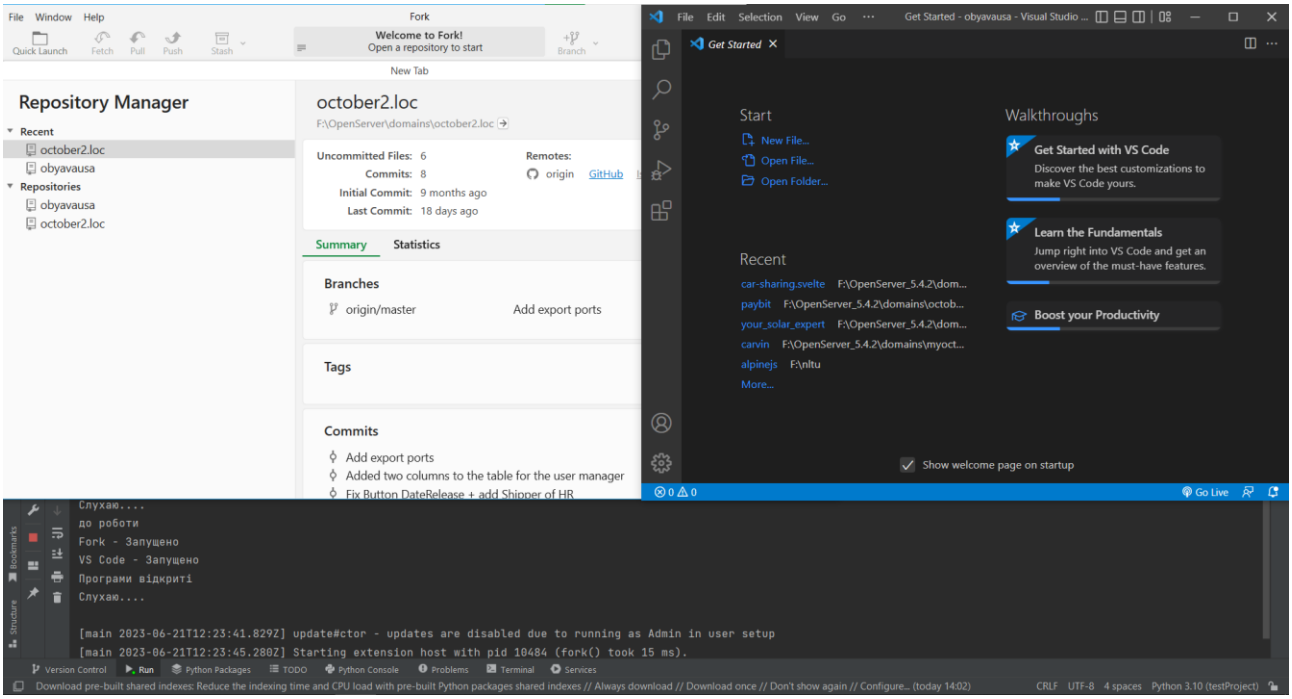

Рис. 3.7. Команда «до роботи» відкриває необхідні програмні

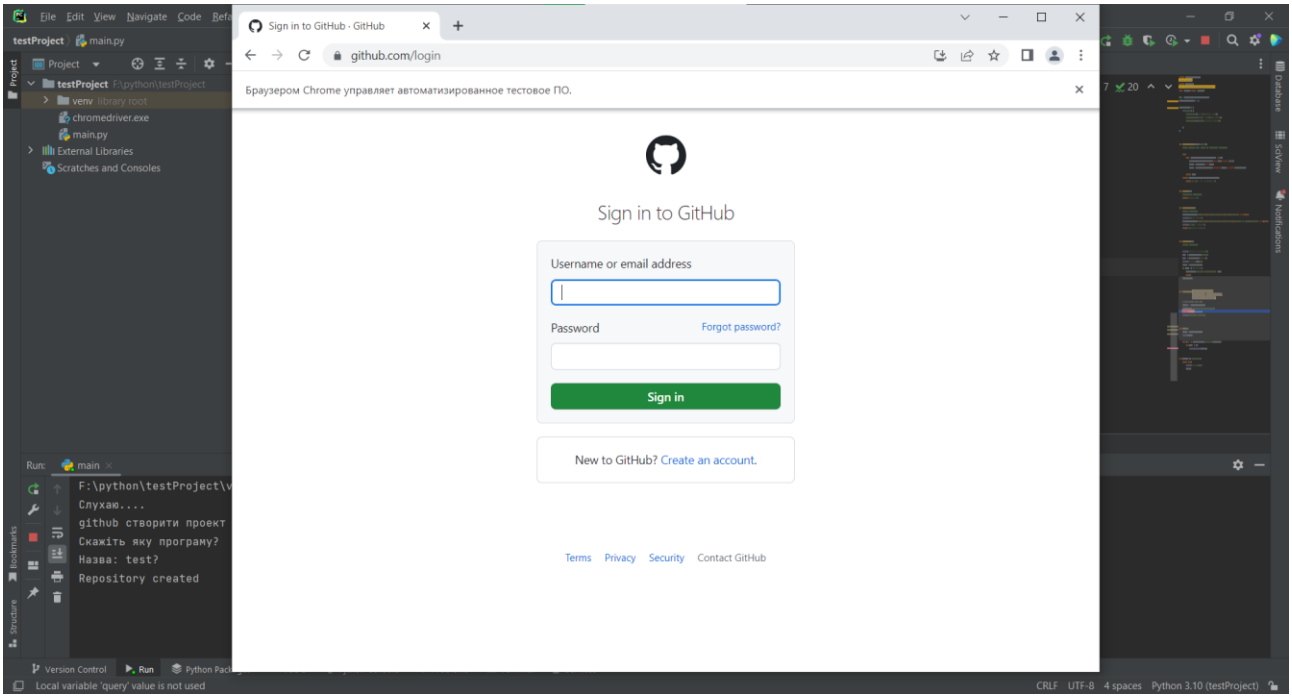

Рис. 3.8. Команда «github створити проект» створює новий репозиторій

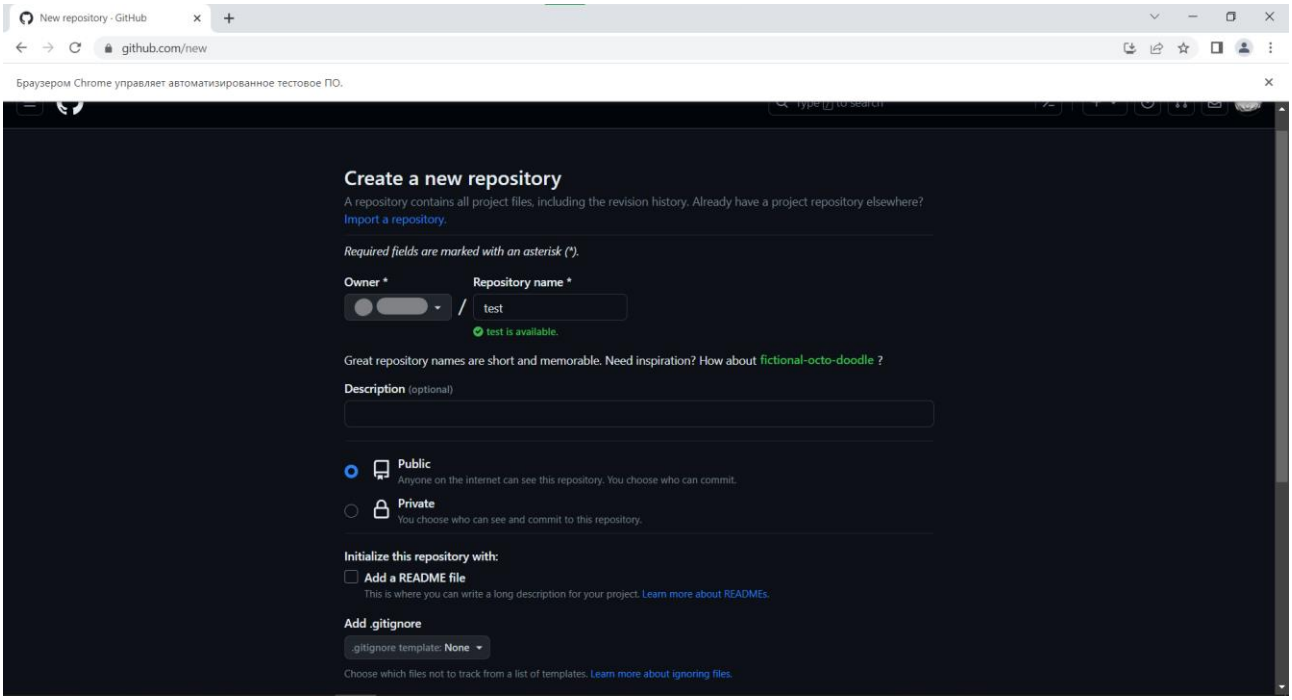

Рис. 3.9. Приклад створення нового репозиторія

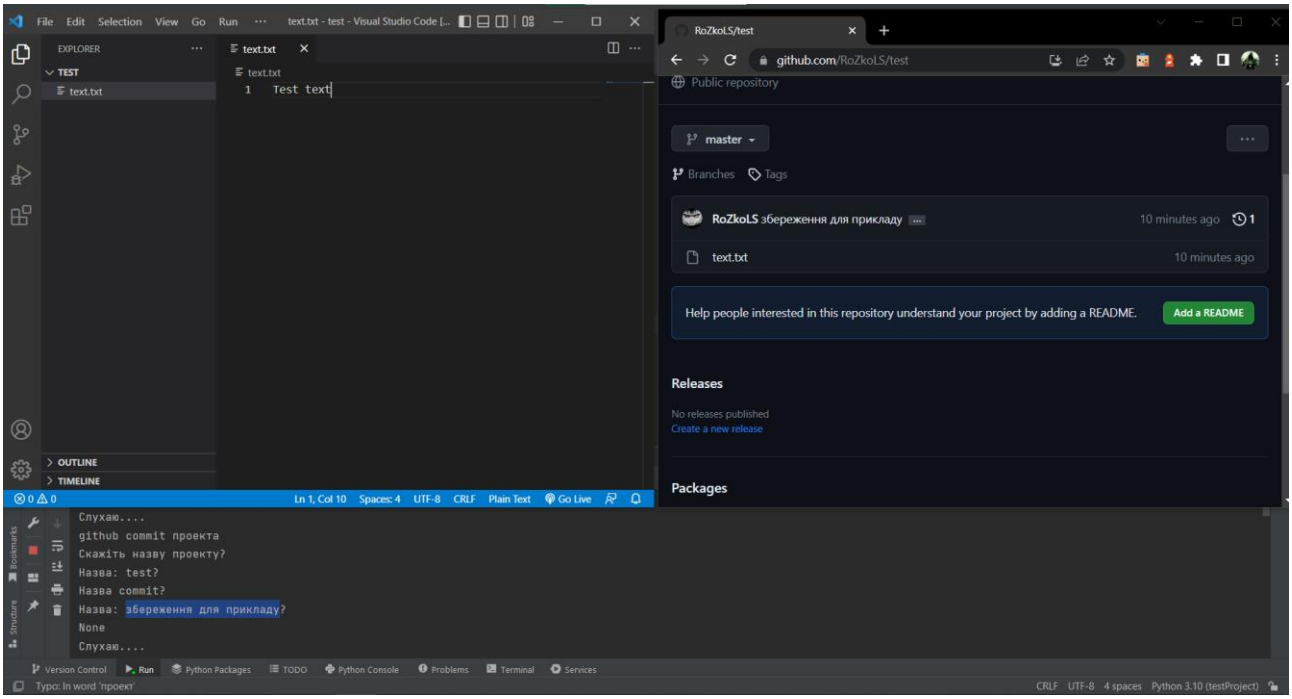

Рис. 3.10. Команда «github commit проекта» зберігає зміни на сервері

ВИСНОВКИ

Голосовий помічник, розроблений для інтегрованих середовищ розробки, допомагає підвищити продуктивність праці розробників. Програмне забезпечння такого типу відповідає всім критеріям оцінки, а саме: він орієнтований і спеціально розроблений для інтегрованих середовищ розроблення програмного забезпечення, дає змогу користувачам віддавати команди високого рівня на звичайній розмовній мові, є вільним у використанні, містить програмний текст із відкритим вихідним кодом, розроблений із використанням програмного тексту, а також розроблено з використанням мікросервісної архітектури, що дає змогу легко вносити зміни в систему.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lutz M. Python Crash Course: A Hands-On, Project-Based Introduction to Programming. – No Starch Press, 2019.
2. Sweigart A. Automate the Boring Stuff with Python: Practical Programming for Total Beginners. – No Starch Press, 2019.
3. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. – O'Reilly Media, 2019.
4. Grus J. Data Science from Scratch: First Principles with Python. – O'Reilly Media, 2019.
5. Hughes B. Fluent Python: Clear, Concise, and Effective Programming. – O'Reilly Media, 2022.
6. Russell T. Head First Python: A Brain-Friendly Guide. – O'Reilly Media, 2020.
7. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. – Pearson, 2020.
8. Goldberg Y. Neural Network Methods for Natural Language Processing. – Morgan & Claypool, 2019.
9. Manning C. D., Schütze H. Foundations of Statistical Natural Language Processing. – MIT Press, 2021.
10. O'Reilly Media. Designing Voice User Interfaces: Principles of Conversational Experiences. – O'Reilly Media, 2020.
11. Mycroft AI. Mycroft Documentation and Developer Guide. – Mycroft AI, 2022. – [Електронний ресурс]. – Режим доступу: https://mycroft.ai/docs (Дата звернення: 05.06.2023).
12. Kalliope Project. Kalliope: Programmable Voice Assistant. – Community documentation, 2022. – [Електронний ресурс]. – Режим доступу: https://kalliope.ai (Дата звернення: 05.06.2023).

13. Tepper J. Building Intelligent Chatbots: Deploying Voice-Enabled Assistants. – Apress, 2020.
14. Michael R., et al. Voice Recognition Systems and Applications. Springer, 2021.
15. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. – Prentice Hall, 2020.

ДОДАТКИ

Main.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# GUI
from .ui import design
from PyQt5 import QtWidgets
import sys
from qt_material import apply_stylesheet

# Core
from .core import (speak, talk, recognise, defaults)
import vasisualy.utils as utils
import random
import os

# Skills
from .skills import (skill_loader, time_date, exit, weather, music, open, screenshot, search, powerof
f, ytvideo,
                     resay, map, wiki, location, weather_no_city, translate, news, coin, upd_upg, shoplist, to
dolist,
                     netconnection, record, guess_num, rulette, math, audio, crystal_ball, random_num, tim
er,
                     show_about, show_settings, old_skills)

styles = {
    "Dark Amber": "dark_amber.xml",
    "Dark Blue": "dark_blue.xml",
    "Dark Cyan": "dark_cyan.xml",
    "Dark Lightgreen": "dark_lightgreen.xml",
    "Dark Pink": "dark_pink.xml",
    "Dark Purple": "dark_purple.xml",
    "Dark Red": "dark_red.xml",
    "Dark Teal": "dark_teal.xml",
    "Dark Yellow": "dark_yellow.xml",
    "Light Amber": "light_amber.xml",
    "Light Blue": "light_blue.xml",
    "Light Cyan": "light_cyan.xml",
    "Light Lightgreen": "light_lightgreen.xml",
    "Light Pink": "light_pink.xml",
    "Light Purple": "light_purple.xml",
    "Light Red": "light_red.xml",
    "Light Teal": "light_teal.xml",
    "Light Yellow": "light_yellow.xml",
}
```

```
randnum = -1
isGuessNum = False
isRuLette = False


class Main(QtWidgets.QMainWindow, design.Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)

        self.lineEdit.setFocus()
        speak.speak("Привет, меня зовут Васисуалий. Чем могу быть полезен?", self.listWidget)

        skill_loader.load()  # Импорт "новых" навыков

        self.lineEdit.editingFinished.connect(self.vasmsg)
        self.pushButton.clicked.connect(self.recogniser)
        self.aboutMenu.triggered.connect(self.showAboutDialog)
        self.settingsMenu.triggered.connect(self.showSettingsDialog)

    def showAboutDialog(self):
        self.dialog = show_about.ShowAboutWindow()
        self.dialog.show()

    def showSettingsDialog(self):
        self.dialog = show_settings.ShowSettingsWindow()
        self.dialog.show()

    def vasmsg(self):
        self.say = self.lineEdit.text()
        self.lineEdit.clear()
        self.listWidget.scrollToBottom()
        self.say = self.say.capitalize()
        self.program()

    def recogniser(self):
        self.say = recognise.recognise(self, self.listWidget)  # Вызов функции распознавания речи
        self.program()

    def program(self):
        tmp = utils.tmp

        say = self.say
        skillUse = False

        if say == '' or say == ' ':
            pass
```

```
        else:
            # Вывод сообщения пользователя
            item = QtWidgets.QListWidgetItem(say)
            item.setTextAlignment(0x0002)
            self.listWidget.addItem(item)

        if os.path.exists(f"{tmp}/.skill_lock"):
            skill_loader.run_looped(say, self.listWidget)
            skillUse = True

        elif skill_loader.run_skills(say, self.listWidget):
            skillUse = True

        elif old_skills.old_skills_activate(say, self.listWidget, self):
            skillUse = True

        elif guess_num.isTriggered(say):
            skillUse = True
            global randnum, isGuessNum
            randnum = guess_num.getRandomNum()
            isGuessNum = guess_num.startGame(self.listWidget)

        elif rulette.isTriggered(say):
            skillUse = True
            global isRuLette
            isRuLette = rulette.startGame(self.listWidget)

        elif say == 'stop' or say == 'Stop' or say == 'Стоп' or say == 'стоп':
            speak.tts_d.stop()

        elif isGuessNum:
            isGuessNum = guess_num.game(say, randnum, self.listWidget)

        elif isRuLette:
            isRuLette = rulette.game(say, self.listWidget)

        else:
            if talk.talk(say) and not skillUse:
                speak.speak(talk.talk(say), self.listWidget)
            elif not skillUse:
                if say != "":
                    # Фразы для ответа на несуществующие команды
                    randwrong = random.choice(wrong)
                    speak.speak(randwrong, self.listWidget)


def main():
    app = QtWidgets.QApplication(sys.argv)
```

```
    window = Main()

    try:
        theme = defaults.get_value("theme")  # Установка темы Qt5-приложения
    except FileNotFoundError:
        theme = defaults.defaults["theme"]

    if theme != "System":
        apply_stylesheet(app, theme=styles[theme])

    window.show()
    app.exec_()
    speak.tts_d.close()


if __name__ == '__main__':
    main()
```

recognise.py

```
from . import speak
import speech_recognition
from PyQt5 import QtGui


def recognise(_cls, widget):
    mic_on = QtGui.QIcon.fromTheme("mic-on")  # Иконка для состояния распознавания речи
    mic_off = QtGui.QIcon.fromTheme("mic-off")  # Иконка для состояния покоя
    _cls.pushButton.setIcon(mic_on)  # Установка иконки во время распознавания речи

    recognizer = speech_recognition.Recognizer()
    recognizer.pause_threshold = 0.5
    mph = speech_recognition.Microphone()

    print("[sys] Слухаю...")

    with mph as source:
        say = recognizer.listen(source)

    try:
        say = recognizer.recognize_google(say, language="uk-UA")
        say = say.capitalize()
        print("[sys] Речь распознана.")
    except Exception:
        say = ''
        speak.speak("Хмм... Я вас не зрозумів :/", widget)
    return say
```

speak.py

```
import speechd
from . import defaults

tts_d = speechd.SSIPClient('Vasisya')
tts_d.set_output_module('rhvoice')
tts_d.set_language('ru')

try:
   voice = defaults.get_value("voice")
   speed = defaults.get_value("speed")
   pitch = defaults.get_value("pitch")
except FileNotFoundError:
   voice = defaults.defaults["voice"]
   speed = defaults.defaults["speed"]
   pitch = defaults.defaults["pitch"]

tts_d.set_synthesis_voice(voice)
tts_d.set_rate(speed)
tts_d.set_punctuation(speechd.PunctuationMode.SOME)
tts_d.set_pitch(pitch)


def speak(message, widget):
   tts_d.speak(message)
   widget.addItem(message)
   widget.scrollToBottom()
```