

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

РОЗРОБКА ЧАТ-БОТА РОЗКЛАДУ НАВЧАЛЬНИХ ЗАНЯТЬ ДЛЯ
TELEGRAM-МЕСЕНДЖЕРА

Здобувача вищої освіти першого (бакалаврського) рівня,
4 курсу, групи КБ-19-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»


(підпис)

Гречаного Максима Тарасовича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Філоненко Михайло
Миколайович, к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Федотова - Півень Ірина
Миколаївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри
протокол від «26» травня 2023 № 13

Завідувач кафедри


(підпис)

Філоненко Михайло Миколайович,
к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене
звання)

Ірпінь – 2023

Реєстрація № 29

06.06.2023

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту на доопрацювання)

06.06.2023

(дата)



(підпис)

Результат захисту

4 (добре)
(оцінка)

15.06.2023

(дата)



(підпис)

Голова комісії:



(підпис)

В.М. Франчук

Члени комісії:



(підпис)

М.М. Філоненко



(підпис)

О.В. Редич



(підпис)

Є.В. Котух

БЛАНК ОЦІНЮВАННЯ ВИПУСКНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	26		15.06.2023 Франчук В.М.  Філоненко М.М.  Редич О.В.  Котух Є.В.  Суховерхня Л.П. 
2. Дотримання вимог щодо оформлення	від 0 до 10	7		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	41		

Оцінювання кваліфікаційної роботи науковим керівником - 43 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
84	B	4 (добре)	15.06.2023	Франчук В.М.  Філоненко М.М.  Редич О.В.  Котух Є.В.  Суховерхня Л.П. 

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій

Кафедра комп'ютерних та інформаційних технологій і систем

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 122 «Комп'ютерні науки»

Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри

 Філоненко М.М.

«16» 11 2022

Завдання на кваліфікаційну роботу

здобувачу вищої освіти

Гречаному Максиму Тарасовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи

Розробка чат-бота розкладу навчальних занять для telegram-месенджера

Затверджена наказом ректора від «15» листопада 2022 № 1205 (зв змінами наказ від «04» квітня 2023 №361)

2. Строк здачі закінченої роботи «26» травня 2023

3. Цільова установка застосування знань, вмінь та навичок, здобутих під час навчання на першому (бакалаврському) рівні за спеціальністю 122 «Комп'ютерні науки»

Мета роботи дослідити сучасні технології розробки чат-ботів, дослідити мови програмування та інтегровані середовища розробки чат-ботів, розробити чат-бот розкладу навчальних занять, проаналізувати переваги й недоліки чат-боту та продемонструвати його роботу.

Об'єкт дослідження – чат-бот в соціальній мережі Telegram.

Предмет дослідження - програмні середовища, серверне обладнання, бази даних, мова програмування Python як складові фактори для реалізації проекту по створенню чат-боту.

База дослідження telegram-месенджер.

4. Вихідні дані до роботи – розробка чат-боту

5. Перелік графічного матеріалу 28 рисунків

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРИТИЧНІ ПОЛОЖЕННЯ РОЗРОБКИ ЧАТ-БОТІВ.....

ТИПИ ТА АРХІТЕКТУРА ЧАТ-БОТІВ.....

1.1. Призначення та актуальність чат-ботів.....

1.2. Telegram як месенджер та соціальна мережа.....

РОЗДІЛ 2. ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ ПРОЕКТУ.....

2.1. Інтегроване середовище розробки чат-боту.....

2.2. Мова програмування Python.....

РОЗДІЛ 3. ЕТАПИ РЕАЛІЗАЦІЇ ПРОЕКТУ.....

3.1. Вибір засобів реалізації проекту. Веб-сервіс «BotMother».....

3.2. Головний кореневий бот «Father».....

ВИСНОВКИ.....

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактичн
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.11.2022	15.11.2022
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	01.02.2023	01.02.2023
3	II розділ (аналіз експериментальної частини про методики досліджень)	03.04.2023	04.04.2023
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	10.05.2023	09.05.2023
5	Висновки	15.05.2022	15.05.2022
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	26.05.2023	26.05.2023
7	Отримання відгуку керівника	06.06.2023	06.06.2023
8	Отримання зовнішньої рецензії	08.06.2023	08.06.2023

8. Дата видачі завдання «16» листопада 2022

9. Керівник випускної кваліфікаційної роботи


(підпис)

Філоненко
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Федотова-Пі
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти



Гречаний М.

АНОТАЦІЯ

Розробка чат-бота розкладу навчальних занять для telegram-месенджера

Метою роботи є дослідити сучасні технології розробки чат-ботів, дослідити мови програмування та інтегровані середовища розробки чат-ботів, розробити чат-бот розкладу навчальних занять, проаналізувати переваги й недоліки чат-боту та продемонструвати його роботу.

Завдання роботи включає проаналізувати термін «чат-бот» в загальному аспекті та в мережі Telegram; дослідити мову програмування Python та інтегровані середовища розробки чат-ботів; дослідити особливості соціальної мережі Telegram та надійність захисту даних; на основі дослідження визначити основний функціонал чат-боту; реалізувати функціонал чат-боту, оцінити перспективи розробки.

Предметом дослідження програмні середовища, серверне обладнання, бази даних, мова програмування Python як складові фактори для реалізації проекту по створенню чат-боту.

Об'єктом дослідження є чат-бот в соціальній мережі Telegram.

Базою дослідження є telegram-месенджер.

За результатами дослідження розроблено чат-бот для отримання розкладу навчальних занять у Telegram-месенджері.

Кваліфікаційна робота містить 56 сторінок, 28 рисунків, список використаних джерел 20 найменувань.

Ключові слова чат-бот, розклад занять, є telegram-месенджер.

Рік виконання кваліфікаційної роботи: 2023 р.

Рік захисту роботи: 2023 р.

ABSTRACT

Development of a chatbot for the schedule of educational classes for Telegram messenger

The aim of the study is investigate modern chatbot development technologies, research programming languages and integrated chatbot development environments, develop a class schedule chatbot, analyze the advantages and disadvantages of the chatbot and demonstrate its operation..

The task of work includes is analyzing the term "chatbot" in a general aspect and in the Telegram network; explore the Python programming language and integrated chatbot development environments; explore the features of the Telegram social network and the reliability of data protection; on the basis of research, determine the main functionality of the chatbot; implement the functionality of the chatbot, evaluate the development prospects.

The subject of research is software environments, server equipment, databases, Python programming language as constituent factors for the implementation of a project to create a chatbot.

The object of the study is a chatbot in the Telegram social network. The basis of the study is Telegram messenger.

The basis of the study is Telegram messenger

According to the results of the research of the research, a chatbot was developed to receive the schedule of training classes in the Telegram messenger..

The qualification work contains 56 pages, 28 images, a list of 20 references.

Keywords chatbot, class schedule, telegram messenger.

Year of qualification work: 2023.

Year of job protection: 2023

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1. ТЕОРИТИЧНІ ПОЛОЖЕННЯ РОЗРОБКИ ЧАТ-БОТІВ.....	12
ТИПИ ТА АРХІТЕКТУРА ЧАТ-БОТІВ.....	12
1.1. Призначення та актуальність чат-ботів.....	12
1.2. Telegram як месенджер та соціальна мережа.....	14
РОЗДІЛ 2. ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ ПРОЕКТУ.....	29
2.1. Інтегроване середовище розробки чат-боту.....	29
2.2. Мова програмування Python.....	33
РОЗДІЛ 3. ЕТАПИ РЕАЛІЗАЦІЇ ПРОЕКТУ.....	45
3.1. Вибір засобів реалізації проекту. Веб-сервіс «BotMother».....	45
3.2. Головний кореневий бот «Father».....	47
ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	55

ВСТУП

Одне з недавніх інновацій у галузі інформаційних технологій - це технологія чат-ботів, що представляє програму, яка забезпечує можливість віртуального спілкування з користувачем мережі. Чат-бот може спілкуватися з користувачем автоматично за допомогою тексту або голосу. Використання чат-ботів є надзвичайно різноманітними: автоматизація бізнес-процесів, підвищення їх ефективності, економія на персоналі тощо.

Актуальність теми. Розробка чат-боту розкладу навчальних занять для Telegram-месенджера має велику актуальність у зв'язку зі зростанням популярності онлайн-навчання та використання месенджерів як зручного інструменту для комунікації. Крім того, чат-бот може бути корисним інструментом для викладачів, які зможуть вносити зміни у розклад, інформувати про заміни, скасування або додаткові заняття. Викладачі зможуть також вносити інші корисні повідомлення для студентів, наприклад, нагадування про дедлайни здачі робіт або інформацію про важливі події. А ще, розробка чат-боту розкладу навчальних занять може знизити витрати на зберігання та обробку інформації про розклад занять, що є особливо актуальним у великих установах зі значною кількістю студентів та викладачів.

Чат-бот - це програма, здатна вести автоматичну розмову з людиною через текстове або голосове повідомлення. Вона може запам'ятовувати попередні запитання та відповіді, щоб забезпечити функціональність віртуального спілкування. Вони легкі у використанні і зазвичай працюють за принципом запитання-відповіді, звертаючись до бази знань.

Чат-бот, розроблений для університетів, надасть користувачам швидкий доступ до інформації про розклад занять. Тому розробка такого чат-боту буде корисним сервісом. Система реагує на запити користувачів, надаючи релевантну відповідь згідно їх потреб. Користувач може використовувати різні форми запитів, такі як текст, цифри, літери або вибір із запропонованого переліку. Основна мета чат-бота - надати студентам та викладачам корисну та реальну інформацію, яка допоможе їм у навчальному процесі.

Для створення чат-боту можуть використовуватися різні технології та програмні засоби, включаючи мови програмування, такі як Python, Java або JavaScript, та платформи для розробки чат-ботів, наприклад, Dialogflow, Botpress, чи Bot Framework.

Враховуючи зростання популярності месенджерів та онлайн-навчання, розробка чат-боту розкладу навчальних занять для Telegram-месенджеру є важливим кроком у поліпшенні комунікації та забезпеченні зручності користувачів.

Метою дипломної роботи є дослідити сучасні технології розробки чат-ботів, дослідити мови програмування та інтегровані середовища розробки чат-ботів, розробити чат-бот розкладу навчальних занять, проаналізувати переваги й недоліки чат-боту та продемонструвати його роботу.

Завдання дипломної роботи. Відповідно до поставленої мети визначено наступні завдання: проаналізувати термін «чат-бот» в загальному аспекті та в мережі Telegram; дослідити мову програмування Python та інтегровані середовища розробки чат-ботів; дослідити особливості соціальної мережі Telegram та надійність захисту даних; на основі дослідження визначити основний функціонал чат-боту; реалізувати функціонал чат-боту, оцінити перспективи розробки.

Об'єкт дослідження – чат-бот в соціальній мережі Telegram.

Предмет дослідження – програмні середовища, серверне обладнання, бази даних, мова програмування Python як складові фактори для реалізації проекту по створенню чат-боту.

Практичне значення дипломної роботи полягає в тому, що у перспективі за результатами роботи можна розробити технічне завдання для розробника та інструкцією користувача з експлуатації чат-боту розкладу для закладу освіти:

- для попереднього аналізу як цілісної системи керування даних користувачів та обміну даних з інтернет джерелами через «API»;
- для запобігання використанню чат-боту шахраям або особам, котрі можуть недоцільно використовувати цей проект;
- для розроблення нових інструментів захисту;

- для попередження та запобігання від спам-атак та крадіжок інформації або особистих даних користувача;
- для подальших досліджень у галузі інформаційних технологій.

РОЗДІЛ 1. ТЕОРИТИЧНІ ПОЛОЖЕННЯ РОЗРОБКИ ЧАТ-БОТІВ, ТИПИ ТА АРХІТЕКТУРА ЧАТ-БОТІВ

1.1. Призначення та актуальність чат-ботів

Існує безліч чат-ботів, розроблених з різних причин, від простих програм з закодованими шаблонами до складних систем, що використовують вбудовані алгоритми навчання і постійно поповнюють свою базу знань. Традиційні технології зазвичай використовуються для реагування на запити користувачів, шляхом видачі відповідей з баз даних. Однак я хочу розробити бота, що взаємодіє з користувачем інтерактивно, тобто забезпечує зручне використання для нього. Крім того, якщо відповідь на запит користувача відсутня у базі даних системи, бот повинен знайти відповідь в Інтернеті. Чат-боти зазвичай надають текстовий інтерфейс, що дозволяє користувачам вводити команди та отримувати текстові або голосові відповіді.

Chatbot додатки допомагають покращити взаємодію між людьми та сервісами, сприяючи покращенню взаємодії з клієнтами. Крім того, вони надають компаніям нові можливості для поліпшення процесу залучення клієнтів та оптимізації операційної ефективності, знижуючи витрати на обслуговування клієнтів. Щоб досягти успіху в цих завданнях, рішення чат-бота повинно мати здатність ефективно виконувати обидві функції. Тут людська підтримка відіграє ключову роль: незалежно від типу підходу та платформи, втручання людини є необхідним для налаштування, навчання та оптимізації системи чат-ботів.

Існує різноманітність підходів та інструментів, які можна використовувати при розробці чат-ботів. В залежності від сфери застосування, деякі технології чат-ботів можуть бути більш ефективними, ніж інші.

Переваги використання чат-ботів:

- Цілодобова підтримка. Багато компаній замінюють свої служби підтримки та довідкові служби на чат-ботів, що дозволяє клієнтам отримати відповідь на свої запитання 24/7. Крім того, чат-бот може відповідати швидше, ніж оператор колл-центру, що покращує якість обслуговування клієнтів.

- Економія ресурсів. Використання чат-ботів може допомогти оптимізувати ресурси компанії або установи. Рутинні запитання можна автоматизувати, виявивши поширені питання, на які зазвичай звертаються до служби підтримки, та створивши стандартні відповіді для включення в базу даних та інтерфейс чат-боту.

- Залучення аудиторії або клієнтів. Компанія швидко та ефективно надає інформацію про свої продукти за допомогою чат-ботів, що допомагає привернути зацікавлених осіб та збільшити кількість клієнтів. Якісний сценарій діалогу між клієнтом та ботом є ключовим для успіху, оскільки він має бути побудований таким чином, щоб не допускати глухих кутів відповіді. Якщо бот не може відповісти на конкретне запитання, він повинен допомогти клієнту порадою, як іншим способом розв'язати проблему або куди звернутися. Тон та манера спілкування бота також впливає на привернення користувачів, тому важливо створити неформальну атмосферу розмови, щоб збільшити ефект присутності справжньої людини у чаті, що є популярним підходом у соціальних мережах та месенджерах.

- Розширення цільової аудиторії, генерація лідів. Шляхом вмілого структурування запитань, чат-бот може ефективно перетворити звичайного користувача на потенційного клієнта, який буде зацікавлений у придбанні продукту. Крім того, збір інформації від клієнтів за допомогою опитувань є важливим кроком для покращення роботи компанії. За допомогою зворотного зв'язку, компанія може здійснювати персоналізовану розсилку рекламних повідомлень та відстежувати показники їх відправки та ефективності привернення користувачів.

1.2. Telegram як месенджер та соціальна мережа

Telegram є месенджером та соціальною мережею, яка була створена у 2013 році. Це один з найбільш популярних месенджерів у світі, який відрізняється високою ступенем безпеки та конфіденційності, а також широким спектром можливостей.

Telegram доступний для використання на різних платформах, таких як Android, iOS, Windows, macOS та Linux. Для того, щоб скористатися Telegram, користувачам потрібно завантажити додаток та створити акаунт за допомогою номера телефону.

Telegram пропонує широкий набір функцій для обміну повідомленнями, таких як можливість відправляти текстові повідомлення, фотографії, відео, аудіозаписи та документи. Крім того, в месенджері є можливість створювати групові чати, в яких можуть брати участь декілька користувачів.

Однією з ключових особливостей Telegram є його високий рівень безпеки. Для захисту даних використовуються різні технології, такі як шифрування транспортного рівня та шифрування на рівні клієнта. Крім того, Telegram пропонує функцію "секретних чатів", в яких повідомлення автоматично видаляються після того, як їх прочитано.

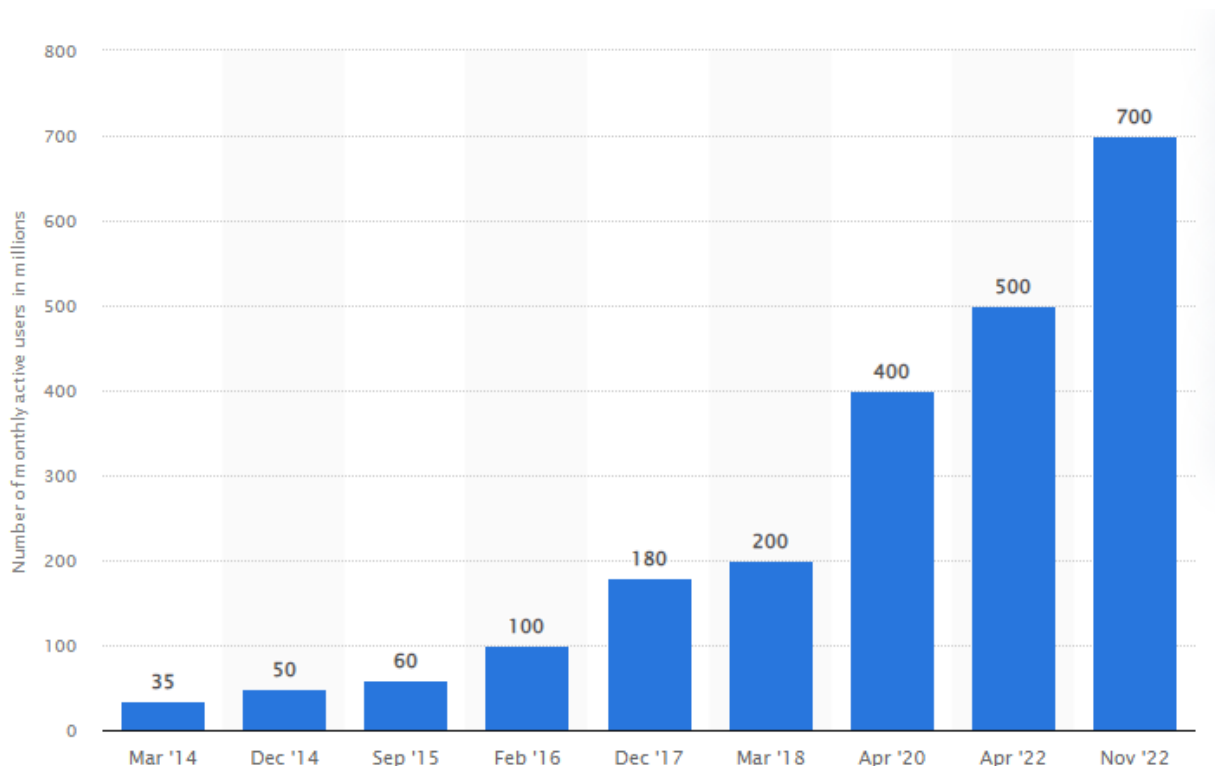
Окрім того, Telegram є соціальною мережею, де користувачі можуть створювати та приєднуватися до каналів, які надають різноманітну інформацію. Канали можуть бути засновані на різних темах, від новин до розваг, тому користувачі можуть знайти для себе цікавий контент. Крім того, користувачі можуть створювати власні групи для спільної комунікації з друзями, колегами чи зацікавленими людьми.

Однією з переваг Telegram як соціальної мережі є те, що вона не має алгоритмів, які визначають, що користувачі повинні бачити. Замість цього, користувачі можуть обирати канали та групи, які їм цікаві, та отримувати тільки ту інформацію, яку вони хочуть отримати.

Окрім цього, Telegram пропонує різні інструменти для взаємодії з користувачами, такі як опитування та голосування. Це дозволяє користувачам більш ефективно взаємодіяти зі своєю аудиторією та збирати відгуки.

У Telegram також є можливість створювати ботів, які можуть виконувати різні завдання. Наприклад, бот може надсилати користувачам новини, розклади занять, погоду та іншу інформацію. Крім того, боти можуть бути корисні для бізнесів, які хочуть автоматизувати свої процеси та полегшити комунікацію з клієнтами.

У загальному, Telegram є потужним месенджером та соціальною мережею, який продовжує набувати великої популярності у світовому масштабі, про що свідчить статистика встановлень месенджеру на пристроях (рисунк 1.1). Він дозволяє користувачам комунікувати між собою та отримувати інформацію про різні теми. Його безпека та конфіденційність дозволяють користувачам відчувати себе захищеними під час обміну повідомленнями, а широкий набір функцій та можливостей роблять його привабливим для різних користувачів - від особистих



користувачів до бізнесів.

Рис.1.1. – кількість активних користувачів Telegram, млн.

Типи чат-ботів, примітивні чат-боти, базові команди

Функціональність ботів сильно залежить від рівня їх підтримки командами. Базові боти обмежені тільки можливістю виконувати команди, які спеціально запрограмовані в їх систему. Це означає, що такі боти не можуть обробляти більш складні запити або виконувати різноманітні дії, які виходять за межі їх базової функціональності.

Зазвичай, доступні команди відображаються на екрані месенджера у вигляді кнопок, щоб зробити їх використання більш зручним для користувача. Це дозволяє легко та швидко виконувати команди, просто натискуючи відповідну кнопку. (рис 1.2)

У більш складних ботів підтримка команд грає значну роль у їхній функціональності. За допомогою програмування боти можуть взаємодіяти з користувачами та виконувати різні завдання, що робить їх корисними в багатьох галузях, включаючи бізнес, медицину, освіту та інші.

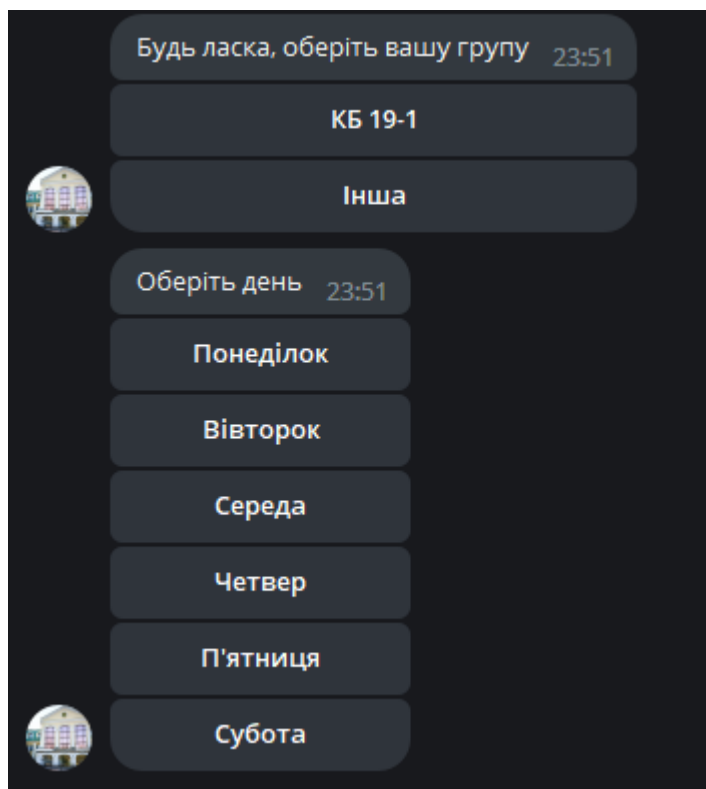


Рис. 1.2. - Приклад командних кнопок в інтерфейсі бота для Telegram

Однією з поширених практик є написання команд для ботів з фіксованим префіксом, який зазвичай відображається у вигляді символу «/». Наприклад, команда `"/help"` є стандартною для багатьох ботів і надає користувачам доступ до списку доступних команд або інших функцій, необхідних для початку роботи з ботом. Рисунок 1.3 показує приклад відповіді бота BotFather на таку команду, що дозволяє створити бота для месенджера Telegram. Важливо зазначити, що командам цього типу не дозволяється містити пропуски (пробіли) між словами. Наприклад, команда `"/set name"` буде вважатися недійсною. Просто натиснувши на кнопку на екрані, користувач може надіслати відповідну команду боту, який здійснить пов'язану з нею дію. Таким чином, з використанням фіксованих команд з префіксом користувач може легко і швидко взаємодіяти з ботами та отримувати від них потрібну інформацію.

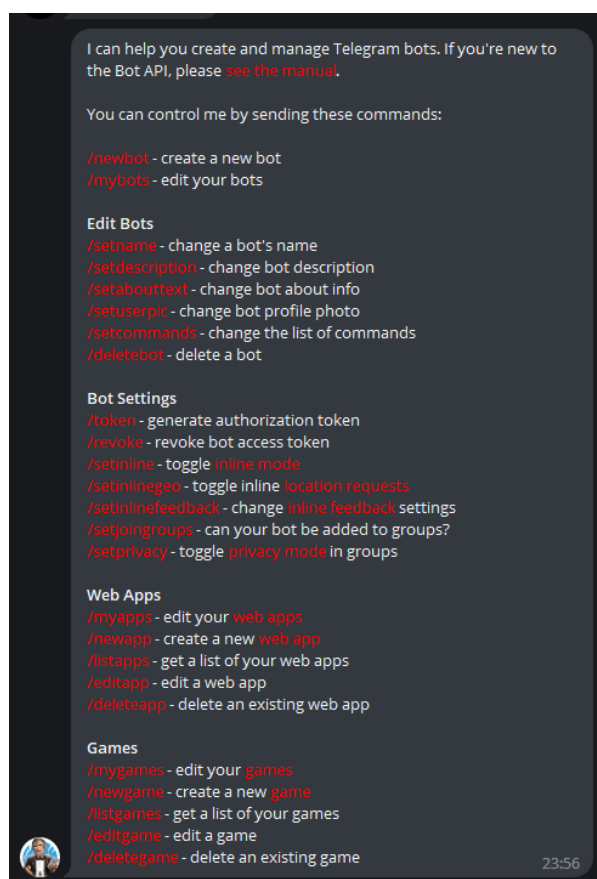


Рис. 1.3. - Результат обробки команди «/help» ботом BotFather

Також існує можливість для розробників додавати власний текст до таких команд. Наприклад, можна виконати команду «/встановити будильник о 10 ранку»,

що дозволяє створювати найпростіші боти, які можна легко розширювати шляхом додавання нових команд в програму.

Чат-боти, які можуть відокремити команду від повідомлення

Відокремлення команд від повідомлень користувачів є важливим кроком у підвищенні інтелектуального рівня бота і тому дедалі більш популярною функцією. Боти з такою функціональністю можуть розуміти різні варіанти команд, які подібні до розмови між людьми, наприклад:

"Не можеш встановити будильник на 06:00 5 вересня?" або

"Постав будильник на завтра о 08:30".

Це вимагає використання засобів обробки "природньої" мови, оскільки людська мова може бути неточною та специфічною, а одне слово може мати багато різних значень, наприклад, слово "run" має 645 різних значень в англійській мові, а слово "розклад" має чотири значення згідно з тлумачним словником[2].

Для обробки природного мовлення користувача програмні засоби використовують розпізнавання іменованих об'єктів та токенизацію, щоб точно зрозуміти, що саме користувач бажає. Розпізнавання іменованих об'єктів є типом пошуку інформації, який спрямований на ідентифікацію та класифікацію об'єктів за певними категоріями, такими як компанії та організації, місцезнаходження, імена людей або назви продуктів, вказівка часу тощо. На основі вилученої інформації програма формує правильну відповідь.

Для того, щоб уникнути плагіату, чат-бот повинен розбити введені речення на окремі «жетони», або лексеми, які можуть бути у будь-якій формі - словами, числами або розділовими знаками. Цей процес відомий як сегментація речення. Потім бот аналізує всі ці лексеми за контекстом.

Існують боти, які розрізняють ключові слова в фразах користувача і підготували відповіді на запитання з певним набором цих ключових слів. Наприклад, якщо на сайті роздрібної компанії користувач вводить слово "просування", бот використовує заздалегідь зареєстроване значення для цього слова, наприклад: "Сьогодні взуття зі знижкою!". Але якщо запитання користувача

складніше або містить кілька запитань, бот може попросити уточнень або дати відповідь, яка не відповідає очікуванням користувача.

Еволюція чат-ботів. Боти, здатні зрозуміти контекст діалогу

Чат-боти мають на меті досягти здатності зрозуміти загальний контекст діалогу, що є наступним етапом їх еволюції. Це означає, що бот буде здатний зрозуміти зміст попередніх реплік і відповідно до цього взаємодіяти з користувачем. Наприклад, якщо користувач просить додати банани до списку покупок, а потім просить додати упаковку борошна, бот буде здатний зрозуміти, що обидва запити стосуються списку покупок і додати їх до нього. Результатом буде список покупок, який міститиме як банани, так і упаковку борошна.

Ще одним важливим кроком у розвитку чат-ботів є забезпечення можливості отримання відповідей ботом у різних варіаціях. Це дозволяє наблизити штучний діалог до реального, що ведуть люди між собою. Для досягнення цієї гнучкості та здатності зрозуміти зв'язок між повідомленнями необхідне використання нейронних мереж та машинного навчання. Наприклад, бот може аналізувати попередній діалог з користувачем, щоб зрозуміти його потреби та відповісти на його запит у персоналізованому форматі. Наприклад, якщо бот з'ясував, що користувачі віддають перевагу процесорам Intel, він може дати відповідь на запит про купівлю процесора, надаючи інформацію саме про продукти цієї торгівельної марки[13].

Сучасні боти можуть бути також інтегровані з сторонніми сервісами, що дозволяє їм взаємодіяти з іншими програмами та виконувати різноманітні завдання. Наприклад, якщо ви скажете голосом своєму персональному помічнику, щоб він увімкнув музику, то бот запустить її у вашому улюбленому музичному програвачі. Один з прикладів успішного та відомого персонального помічника - це SIRI (Speech Interpretation and Recognition Interface), що є частиною IOS, IpadOS, WatchOS, MacOS та TVOS від компанії Apple. Цей додаток обробляє голосові команди та надає користувачеві відповіді на запитання та рекомендації, при цьому він адаптується до потреб та смаків кожного окремого користувача.

Ще одним прикладом є Google Assistant - віртуальний помічник для Windows, що може працювати на смартфоні та управляти відтворенням відео на телевізорі в режимі діалогу з ботом.

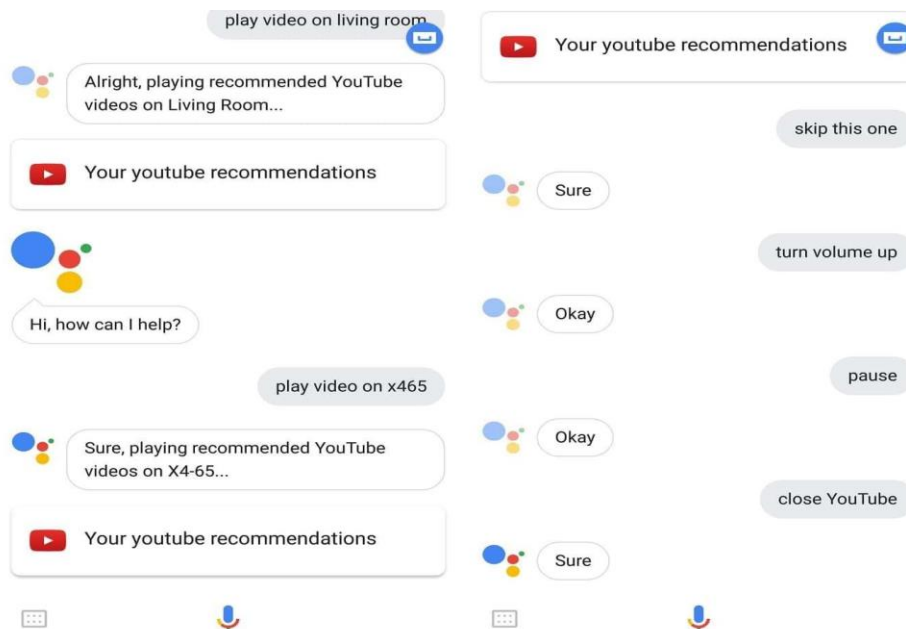


Рис. 1.4. - Використання Google Assistant на смартфоні для відтворення відео по телевізору

Створення серверної частини бота

Давайте розглянемо загальний приклад серверної архітектури бота, який показано на рисунку 1.5.

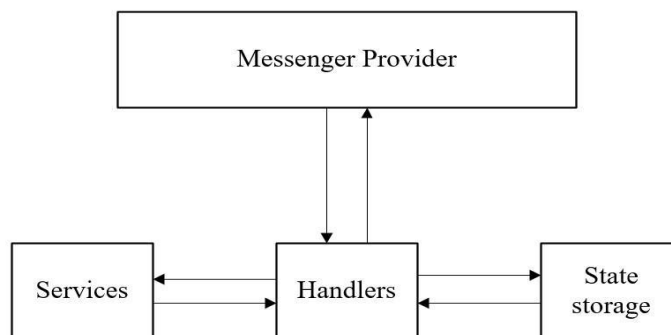


Рис. 1.5. - Узагальнений приклад архітектури бот-сервера

Бот складається з чотирьох основних компонентів:

- а) постачальника месенджерів;
- б) обробників;
- в) сервісів;
- г) контейнера стану.

Постачальник месенджерів обгортає API Telegram Bot та пропонує універсальний інтерфейс для отримання, обробки та відправлення повідомлень. Провайдер месенжера визначає, що користувач хоче зробити, та направляє це до відповідного обробника. Метою цієї частини є забезпечення рівня абстракції над API обміну повідомленнями. Іншими словами, якщо необхідно підтримувати більше СМС-провайдерів або перейти на використання іншого месенжера, то потрібно буде змінити лише цю частину коду. На рисунку 1.6 показано конструктор та функцію прослуховування класу Messenger, який є постачальником месенджерів. Наприклад, бот "JonathanZWhite/бот-сервер" може бути розглянутий як приклад такої архітектури.

```

class Messenger {
  constructor() {
    this.bot = new TelegramBot('Your token here', { polling: true });
  }

  listen() {
    this.bot.on('text', this.handleText.bind(this))
  }

  ...
}

```

Рис. 1.6. - Частина класу Messenger

У конструкторі класів "Messenger" ми ініціалізуємо бібліотеку Bot API. У даному випадку використовується API Telegram, але його можна легко замінити на інший. Далі ми створюємо функцію прослуховування, яка отримує повідомлення з вхідного чату та передає їх до функції "handleText", яка зображена на рисунку 1.7.

```

class Messenger {
  ...

  handleText(msg) {
    const message = new Message(Message.mapMessage(msg))
    const text = message.text

    if (inputParser.isAskingForGenreList(text))
      return handlers.music.getGenreList(message, this.bot)

    if (inputParser.isAskingForNumberOfRec(text, store.getState(message.from).command))
      return handlers.music.getNumOfRec(message, this.bot)

    if (inputParser.isAskingForRecommendation(text, store.getState(message.from).command))
      return handlers.music.getRecommendation(message, this.bot)

    // default
    return handlers.casual.getHelp(message, this.bot)
  }
}

```

Рис. 1.7. - Функція «HandleText» класу «Messenger»

В середині функції «handleText» ми передаємо повідомлення та останню інформацію про статус зберігання до контейнера стану до класу «InputParser» (рис. 1.8). Він визначає, яку саме дію користувач бажає здійснити та допомагає надіслати відповідну команду до правильного обробника.

```

class InputParser {
    isAskingForGenreList(text) {
        const pattern = /music|recommendation/i

        return text.match(pattern)
    }

    isAskingForNumberOfRec(text, prevCommand) {
        return prevCommand === commands.GET_GENRE_LIST
    }

    isAskingForRecommendation(text, prevCommand) {
        return prevCommand === commands.SET_NUMBER_OF_REC
    }
}

```

Рис. 1.8. - Клас «InputParser»

У класі «InputParser» застосовується простий регулярний вираз або аналіз стану, щоб визначити, яку команду хоче отримати користувач, і передати її до класу Messenger для обробки. Крім того, «InputParser» дозволяє додавати інтеграції з іншими мережевими інструментами для обробки природної мови. Обробники, зображені на Рис. 1.9, відповідають за обробку повідомлень, які надсилаються постачальником месенджеру. Вони делегують завдання службам, оновлюють контейнер стану, якщо це потрібно, і генерують повідомлення для відправлення користувачеві. Клас, що відповідає за музику, представлений на Рис. 1.9, відповідає за виконання всіх дій та команд, пов'язаних з отриманням рекомендацій Spotify для користувача. Функція «getGenreList» приймає повідомлення та екземпляр бота, що надісланий йому з класу Messenger. Об'єкт повідомлення містить ідентифікатор користувача та текст, який він надіслав. Бот використовується для надсилання повідомлень користувачеві.

```

class Music {
  getGenreList(message, bot) {
    // clears store for new command tree
    store.clearState(message.from)
    store.update(message.from, { command: commands.GET_GENRE_LIST })

    bot.sendMessage(message.from, 'Choose a genre you would like a recommendation for 🎵', {
      reply_markup: {
        keyboard: genresJSON.genres,
        one_time_keyboard: true
      }
    })
  }
}

...
}

```

Рис. 1.9. - Клас «Music»

Зазвичай розробники оновлюють контейнер стану, щоб відстежувати, де знаходиться користувач у дереві розмов. У нашому прикладі ми також оновлюємо контейнер стану, і після цього функція «getGenreList» надсилає користувачеві список жанрів Spotify для вибору[4].

Проте іноді потрібно використовувати більш складну бізнес-логіку, щоб зрозуміти повідомлення та надіслати користувачеві відповідь. У таких випадках необхідно передати обробку послуги іншим сервісам (див. рис. 1.10).

На початку функції «getRecommendations», яка є аналогічною до «getGenreList», також оновлюється контейнер стану. Тут ми оновлюємо ще один параметр, який зберігає кількість музичних рекомендацій, які користувач замовив. Тому, щоб отримати рекомендації від Spotify API, викликається функція «getRecommendations» з класу «MusicService», який відповідає за цей процес. Після цього отримані рекомендації можна надіслати користувачеві.

```

class Music {
  ...

  getRecommendation(message, bot) {
    store.update(message.from, {
      command: commands.GET_RECOMMENDATION,
      spotify: {
        numberOfRecs: message.text
      }
    })

    const state = store.getState(message.from)
    const selectedGenre = state.spotify.genre
    const numberOfRecs = state.spotify.numberOfRecs

    return MusicService.getRecommendation(selectedGenre, numberOfRecs)
      .then(sendRecommendation)

    function sendRecommendation(resp) {
      if (!resp.length) return bot.sendMessage(message.from, 'Sorry, looks like I don\'t have anything')

      bot.sendMessage(message.from, 'Why not give this a listen 🎵')

      resp.forEach((song) => {
        bot.sendMessage(message.from, song)
      })

      store.clearState()
    }
  }
  ...
}

```

Рис. 1.10. - Клас «Music»: використання MusicService

MusicService - це приклад того, як служби повинні виконувати складні та довгострокові операції, тоді як обробники повинні виконувати простіші завдання. Служба включає запити до сторонніх API, обробку даних та форматування відповідей для обробників. Це чітко розподіляє відповідальність між різними компонентами системи, що спрощує тестування та масштабування коду.

У фрагменті коду на малюнку 1.11, MusicService звертається до Spotify API, отримує результати запиту та генерує відповідь на основі цих результатів, передаючи її назад до обробника[10].

```

(function(module) {

    function getRecommendation(selectedGenre, limit) {
        ...

        return Promise.resolve()
            .then(getSpotifyAccessToken)
            .then(getRecommendation)

        function getRecommendation() {
            return spotify.getRecommendations({
                seed_genres: selectedGenre,
                limit: limit
            })
            .then((resp) => {
                return _parseForTrackList(resp.body.tracks, limit)
            })
        }
    }

    ...
})(exports));

```

Рис. 1.11. - Клас MusicService

Контейнери стану - це складова системи, яка дозволяє зберігати та змінювати стан програми відповідно до контексту розмови з користувачем. Ця програма зберігає дані, які стосуються запитань користувача, такі як бажаний жанр музики та кількість рекомендацій. Контейнери стану контролюють стан системи та дозволяють зберігати дані розмови. На малюнку 1.12 показана діаграма стану, яка імітує всі можливі стани системи рекомендацій. Кожен стан має асоційовані дані, які ми отримуємо від користувача. Для цього можна використовувати додаткові повідомлення для опитування користувача.

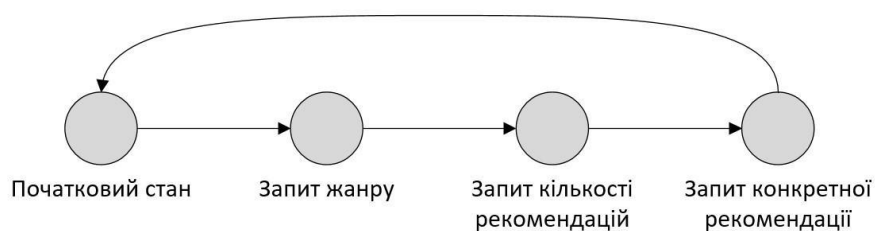


Рис. 1.12. - Діаграма станів ЧБ для рекомендації музики

Наприклад, в стані «Запит на кількість рекомендацій» повинно бути відомо, який жанр обрав користувач. Точно так само, стан «Запит на конкретну рекомендацію» повинен успадкувати дані попереднього стану і отримати додаткові дані про кількість рекомендацій, які бажає користувач.

Контейнер станів містить сутність «_users», яка є користувачами (рис. 1.13). У цьому полі зберігається хешована таблиця, де ключем є ідентифікатор користувача, отриманий з повідомлення, а значенням є поточний стан користувача.

```

let store = {
  _users: {},

  // creates state tree
  _initializeUserState: function(hash) {
    this._users[hash] = {
      command: '',
      spotify: {
        genre: '',
        numberOfRecs: 0
      }
    }
  }
},
...
}
  
```

Рис. 1.13. - Контейнер станів

Функція «initializeUserState» відповідає за ініціалізацію початкового стану користувача та викликається при першій взаємодії користувача з системою. Функція «update» (рис. 1.14), відповідає за оновлення статусу і приймає старий стан та новий стан, які зливаються разом.

```

let store = {
  ...

  update: function(hash, data) {
    if (!this._users[hash]) this._initializeUserState(hash)

    console.log('🕒 => PREVIOUS STATE')
    console.log(this.getState(hash))

    this._users[hash] = _.merge(this._users[hash], data)

    console.log('🕒 => NEXT STATE')
    console.log(this.getState(hash))
  }
}

```

Рис. 1.14. - Функція «update» в контейнері стану

Функції "getState" та "clearState" (зображені на рис. 1.15) входять до складу контейнера стану. Функція "getState" дозволяє отримати поточний стан користувача, тоді як "clearState" використовується для скидання розмови до початкового стану.

```

let store = {
  ...

  getState: function(hash) {
    if (!this._users[hash]) return this._initializeUser(hash)

    return this._users[hash]
  },

  clearState: function(hash) {
    this._users[hash] = this._initializeUser()
  }
}

```

Рис. 1.15. - Функції «getState» і «clearState» у контейнері стану

РОЗДІЛ 2. ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ ПРОЕКТУ

2.1. Інтегроване середовище розробки чат-боту

WebStorm - це інтегроване середовище розробки, призначене для програмування на JavaScript та суміжних технологіях, таких як TypeScript, React, Vue, Angular, Node.js, HTML та CSS. Це програмне забезпечення забезпечує зручний та ефективний процес розробки, автоматизуючи рутинні задачі та допомагаючи виконувати складні завдання. Кожен проект у WebStorm - це набір файлів з вихідним кодом, бібліотеками та інструментами, що використовуються у проекті, а також різноманітними конфігураційними файлами програми, наприклад, файлом `package.json` або `.eslintrc`.

Після відкриття проекту в WebStorm, програма створює папку `.idea`, де зберігаються внутрішні налаштування конфігурації WebStorm для проекту, такі як налаштування стилю коду або системи контролю версій. Привітальний екран WebStorm надає користувачам можливість легко створювати нові проекти, перевіряти та відкривати існуючі проекти.

Узагальнюючи, WebStorm - це потужний інструмент для розробників JavaScript, який забезпечує ефективну та зручну розробку, надаючи можливості для автоматизації рутинних задач та допомагаючи виконувати складні завдання. Кожен проект у WebStorm - це набір файлів з вихідним кодом, бібліотеками та інструментами, що використовуються у проекті, а також різноманітними конфігураційними файлами програми. Користувачам надається можливість легко створювати нові проекти та працювати з існуючими проектами через привітальний екран WebStorm[7].

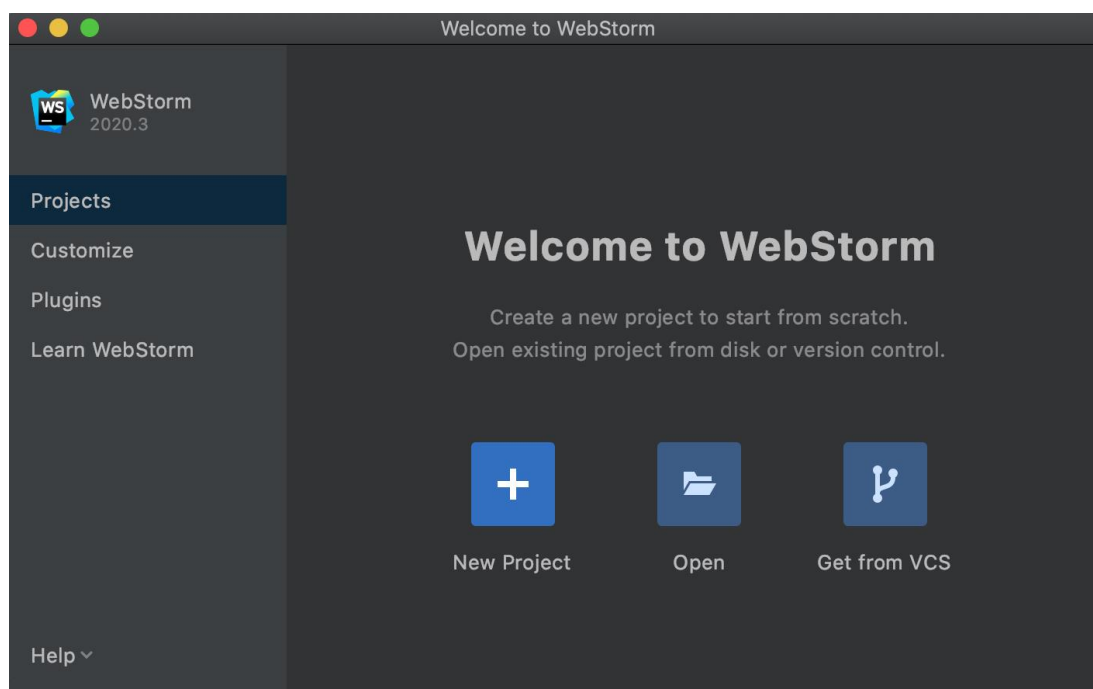


Рис. 2.1. - Вітальне вікно програми

Щоб створити новий файл у проекті, потрібно відкрити вікно інструмента «Проект», обрати папку, в якій необхідно створити новий файл, а потім натиснути комбінацію клавіш `Alt + Insert`. Альтернативою є вибір опції Створити в контекстному меню, після чого слід вибрати потрібний тип файлу зі списку (див. рис. 2.1).

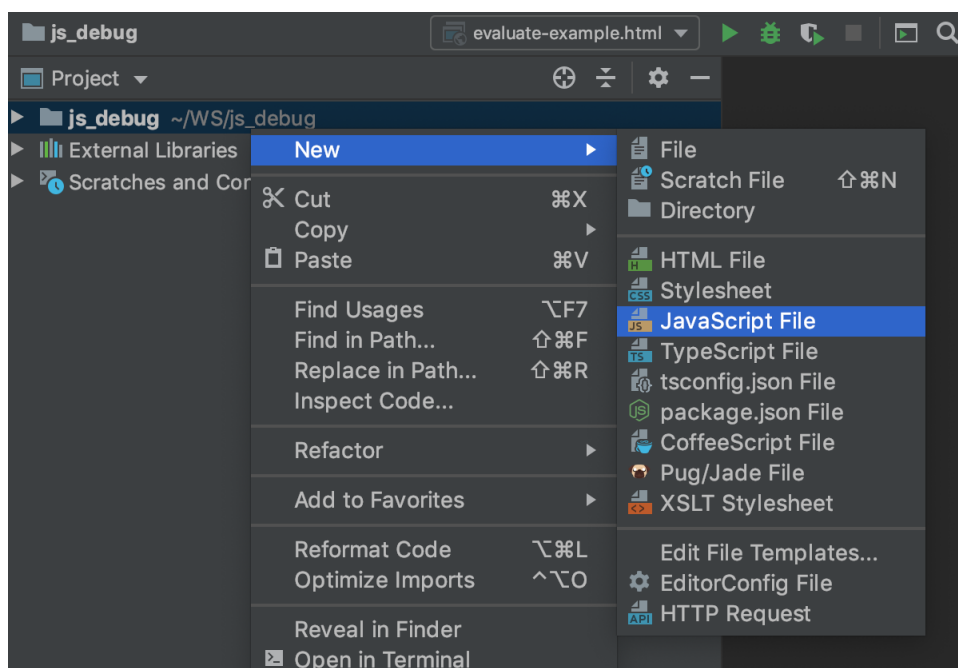


Рис. 2.2. - Створення нового файлу в проекті

WebStorm має багатoelementний інтерфейс, що складається з редактора коду, де ви можна переглядати, створювати та змінювати код, меню та панелі інструментів, панелі навігації, рядка стану та декількох інших вікон інструментів. Останні розташовані знизу та по боках робочої області і дають можливість здійснювати налагодження коду, запускати тести, взаємодіяти з системою контролю версій та іншими функціями (див. рис. 2.3).

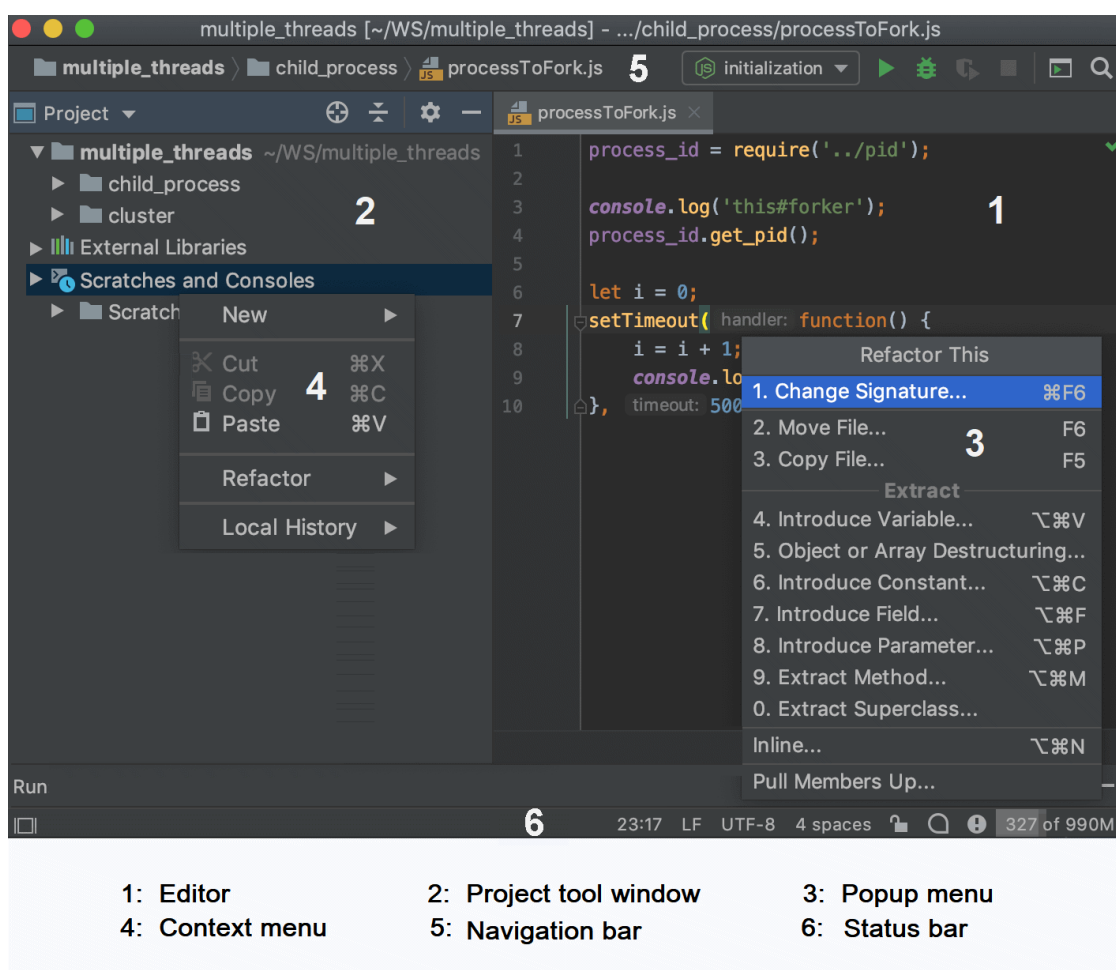


Рис. 2.3. - Панель інструментів та дій в коді

Ви можете налаштувати макет WebStorm за своїм власним смаком. Наприклад, якщо ваша мета - зосередитись на програмуванні, ви можете спробувати Режим вільного відволікання. Цей режим видаляє всі панелі інструментів, вікна інструментів та вкладки редактора, щоб забезпечити більше вільного місця. Щоб увійти в цей режим, вам потрібно відкрити меню "Перегляд", вибрати "Зовнішній вигляд", а потім вибрати "Без уваги".

WebStorm моніторить ваш код і намагається підтримувати його точність та чистоту. Він знаходить потенційні помилки та проблеми і пропонує швидкі способи їх усунення. Кожного разу, коли WebStorm виявляє невикористаний код, нескінченний цикл, відсутність оператора імпорту символу та багато інших речей, які можуть вимагати вашої уваги, ви побачите виділення та лампочку.

WebStorm позначає проблемні символи і надає підказки з запропонованими швидкими виправленнями:

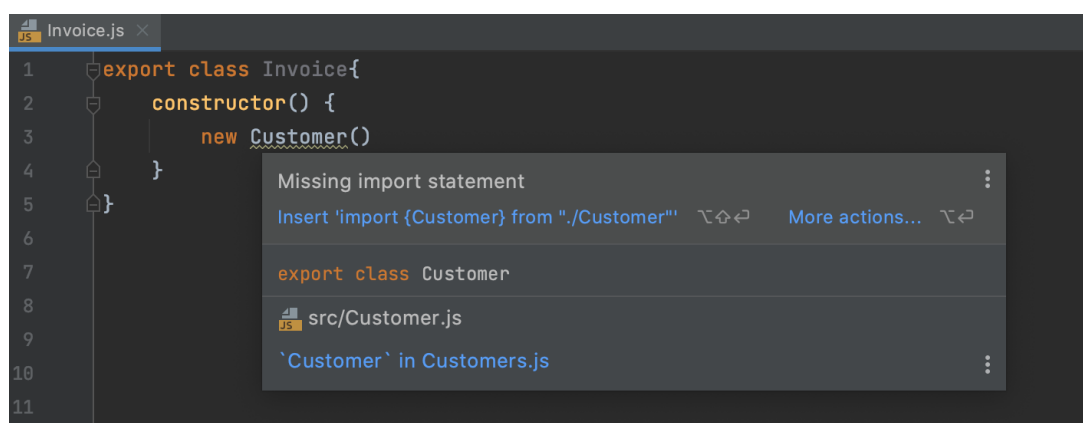


Рис. 2.4. - Підказки щодо виправлення помилок у коді

2.2. Мова програмування Python

Python є мовою програмування високого рівня з динамічною семантикою та об'єктно-орієнтованою парадигмою, яка пропонує вбудовані структури даних високого рівня та динамічне прив'язування, що робить її дуже привабливою для швидкої розробки програм та використання в ролі мови сценаріїв або склеювання існуючих компонентів. Легкий у вивченні синтаксис Python підкреслює читабельність та зменшує витрати на обслуговування програми. Python також підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python та його велика стандартна бібліотека є безкоштовними для всіх основних платформ та доступні у вихідній або двійковій формі[15].

Багато програмістів віддають перевагу Python за підвищену продуктивність, яку вона забезпечує. Відсутність кроку компіляції робить цикл редагування-тестування-налагодження надзвичайно швидким. Налагодження програм на Python дуже просте, оскільки помилки не спричиняють помилок сегментації, але спричиняють винятки, що дозволяє легко виявляти та виправляти помилки. Python має вбудований налагоджувач, який дає можливість перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати точки зупинки, переходити через код за рядком за один раз тощо. Налагоджувач написано на Python, що свідчить про його інтроспективну силу. Також, часто швидким та ефективним способом налагодження програм є додавання декількох операторів друку до джерела: швидкий цикл редагування-тестування-налагодження робить цей простий підхід дуже ефективним.

Python часто порівнюють з іншими мовами програмування, які також є інтерпретованими, наприклад, Java, JavaScript, Perl, Tcl та Smalltalk. Порівняння з мовами, такими як C++, Common Lisp та Scheme, також можуть бути корисними для розуміння особливостей Python. В даному розділі будуть коротко порівняні особливості Python з кожною з цих мов, зосереджуючись на мовних питаннях. Проте, у практиці вибір мови програмування часто залежить від різних обмежень, таких як вартість, доступність, навчання та попередні інвестиції, або навіть від

емоційних уподобань. Тому, розглядати багато таких аспектів для порівняння мов програмування може бути марною тратою часу.

Наприклад, в порівнянні з JavaScript, можна сказати, що вона є "об'єктною" підмножиною Python, більш-менш еквівалентною. Як і в JavaScript (і на відміну від Java), Python підтримує стиль програмування, що використовує прості функції та змінні, без визначення класів. Однак, для JavaScript це все, що доступно, в той час як Python підтримує написання набагато більших програм та забезпечує краще повторне використання коду, завдяки справжньому об'єктно-орієнтованому стилю програмування, в якому класи та спадкування відіграють важливу роль.

Java. Зазвичай програми на Python працюють повільніше, ніж програми на Java, але їх розробка займає значно менше часу. Програми на Python, як правило, коротші в 3-5 разів, ніж еквівалентні програми на Java. Це можна пояснити наявністю в Python вбудованих типів даних високого рівня та динамічним набором тексту. Наприклад, у програміста Python не виникає потреби в оголошенні типів аргументів чи змінних, і користування потужними поліморфними списками та словниками в Python, які мають багату синтаксичну підтримку вбудовану в мову, є поширеним явищем. Це дає можливість Python працювати ефективніше під час виконання завдань, ніж Java. Наприклад, при обчисленні виразу $a + b$ в Python, мова спочатку перевіряє типи об'єктів a та b , які не відомі під час компіляції, а потім викликає відповідну операцію додавання, яка може бути перевизначена користувацьким методом. З іншого боку, Java може виконувати ефективне додавання цілих або дійсних чисел, але вимагає оголошень змінних для a та b та не допускає перевизначення оператора $+$ для екземплярів класів, визначених користувачем[6].

Python чудово підходить як мова для зв'язування («клеюва» мова), тоді як Java краще підходить для розробки компонентів низького рівня. Однак їх поєднання може бути дуже ефективним. Можна розробляти компоненти на Java та використовувати їх у програмах на Python, або використовувати Python для створення прототипів компонентів, які в подальшому будуть реалізовані на Java. Для забезпечення такої розробки існує реалізація Python, написана на Java, яка

дозволяє викликати код Python з Java і навпаки. Ця реалізація використовує бібліотеку, яка перекладає вихідний код Python в байт-код Java для підтримки динамічної семантики Python.

Sublime Text як середовище розробки бота на Python

Sublime Text є популярним текстовим редактором, який часто використовується для написання коду на Python. Редактор має зручний інтерфейс та безліч розширень для виділення синтаксису, пошуку вихідних файлів та аналізу метрик коду, що дозволяє новим програмістам легко розпочати роботу в ньому, порівняно з іншими програмами, такими як Vim та Emacs. У Sublime Text немає різних режимів навчання, як у Vim, тому користувач може зосередитися на вивченні комбінацій клавіш поступово. Також використання Sublime Text як редактора Python має ще один бонус - плагіни для нього можна писати мовою Python, що дозволяє розробникам Python не вчити нову мову, таку як Emacs або Vimscript в Vim, а зосередитися на розробці. З часом, коли рівень кваліфікації розробника зростає, Sublime Text може бути розширений численними функціями, які надає IDE, наприклад PyCharm[10].

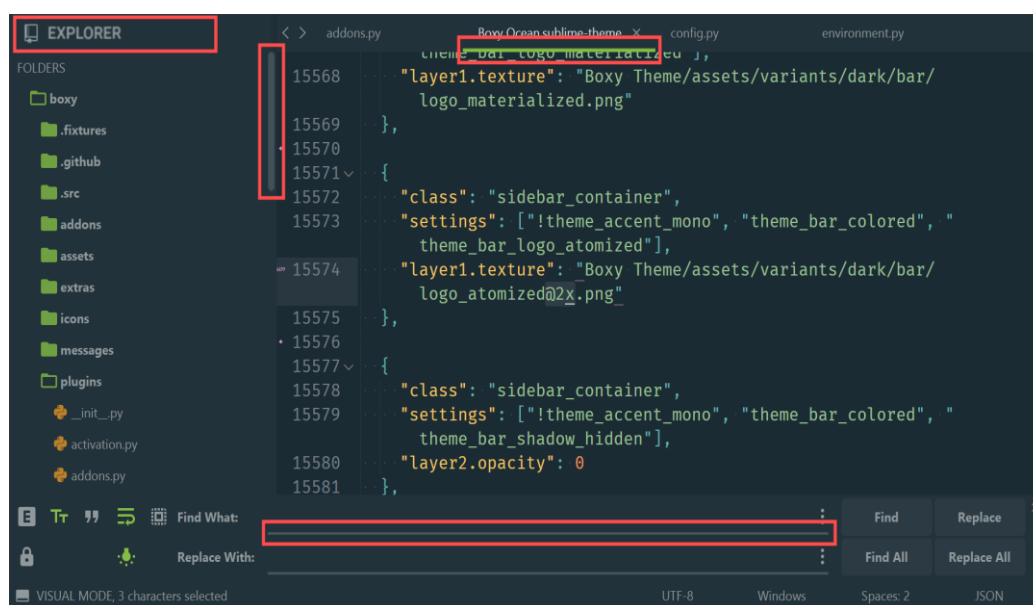


Рис. 2.5. - Інтерфейс програми

Кожен розробник має особисті уподобання щодо вибору між текстовим редактором та інтегрованою середовищем розробки (IDE), але використання IDE може бути доцільним, якщо ви плануєте проводити багато часу в обраному середовищі щодня. Тому розумним вибором буде переконатися, що вибране

середовище є приємним та продуктивним. Кінцевий вибір залишається за розробником, який повинен оцінити переваги та недоліки кожного редактора.

Немає єдиного "найкращого" редактора - краще вибрати той, який вам найбільше підходить і детально ознайомитися з ним. Для досягнення найвищої продуктивності при програмуванні, важливо освоїти свій інструмент та використовувати його для економії часу.

Якщо ви використовуєте Sublime Text для програмування на Python, то є багато підручників та ресурсів, які допоможуть налаштувати ваш редактор, забезпечуючи такі функції, як метрики коду, перевірку синтаксису та багато інших функцій IDE[8].

Sublime Text пропонує різні методи редагування тексту, які можуть заощадити час розробки інтерфейсу. Наприклад, методи, такі як бульбашки рядків, нерівний вибір рядків, AceJump та транспонування, можна використовувати для зручності редагування тексту. Відеокліпи з цими методами можуть бути корисними для тих, хто використовує Sublime Text.

База даних PostgreSQL

PostgreSQL є об'єктно-реляційною системою баз даних з відкритим кодом, яка використовує мову SQL разом з безліччю функцій для безпечного зберігання та масштабування найскладніших навантажень даних. Він походить від проекту POSTGRES, який був створений у 1986 році в Університеті Каліфорнії в Берклі, і зараз має більше 30 років активного розвитку. PostgreSQL відомий своєю надійністю, цілісністю даних, розширюваністю та підтримкою відкритого коду. Ця система баз даних працює на всіх основних операційних системах, сумісна з ACID з 2001 року та має ряд потужних доповнень, таких як PostGIS, популярний розширювач геопросторових баз даних. PostgreSQL є незамінним інструментом для багатьох людей та організацій, які шукають реляційну базу даних з відкритим кодом.

PostgreSQL містить безліч функцій, які допомагають розробникам створювати додатки, адміністраторам забезпечувати цілісність даних та створювати відмовостійкі середовища, а також керувати даними будь-якого обсягу. Крім того, PostgreSQL є безкоштовним та відкритим кодом і має високу розширюваність, що дозволяє визначати власні типи даних, створювати власні функції і навіть писати код з різних мов програмування без перекомпілювання бази даних.

PostgreSQL дотримується стандарту SQL, якщо така відповідність не суперечить традиційним особливостям або може призвести до неправильних архітектурних рішень. Багато функцій, які вимагає стандарт SQL, підтримуються з дещо різними синтаксисом або функцією. PostgreSQL постійно покращується у відповідності до стандарту SQL і починаючи з версії 13, випущеної у вересні 2020 року, підтримує принаймні 170 з 179 обов'язкових функцій для SQL: 2016 Core conformance. Зараз немає жодної реляційної бази даних, яка повністю відповідала б цьому стандарту на момент написання статті.

Нижче подано неповний перелік різноманітних функцій, які можна знайти в PostgreSQL. Кожного разу в нових випусках додаються нові функції до наступних категорій:

Типи даних:

- Примітивні: цілі числа, дійсні числа, рядки, булеві значення.
- Структуровані: дата/час, масиви, діапазони, UUID.
- Документи: JSON/JSONB, XML, ключ-значення (Hstore).
- Геометрія: точки, лінії, кола, багатокутники.
- Нестандартні типи.

Цілісність даних:

- Унікальність, не нульове значення.
- Первинні ключі.
- Іноземні ключі.
- Обмеження перевірки.
- Явні та консультативні блокування.

Паралельність та продуктивність:

- Індування: В-дерева, багатоклонкові, виразові та часткові.
- Розширене індування: GiST, SP-Gist, KNN Gist, GIN, BRIN, індекси покриття, фільтри Bloom.
- Складний планувальник/оптимізатор запитів, сканування з індексом, багатостовпчаста статистика.
- Транзакції, вкладені транзакції (через точки збереження).
- Багатоверсійний паралельний контроль (MVCC).
- Паралельна обробка запитів на читання та побудову індексів В-дерева.
- Розбиття таблиці.

- Всі рівні ізоляції транзакцій, визначені стандартом SQL, включаючи серіалізацію.
- Складання виразів точно вчасно (JIT).

Надійність, відновлення після катастрофи:

- Запис журналу попереднього запису (WAL)
- Реплікація: асинхронна, синхронна, логічна
- Відновлення до певної точки в часі (PITR), активні режими очікування
- Таблицеві простори

Безпека:

- Різноманітні методи аутентифікації, такі як GSSAPI, SSPI, LDAP, SCRAM-SHA-256, сертифікати тощо.
- Надійна система контролю доступу
- Захист на рівні стовпців і рядків
- Багатофакторна автентифікація за допомогою сертифікатів та додаткових методів

Розширюваність:

- Збережені функції та процедури
- Процедурні мови: PL/PGSQL, Perl, Python (і багато інших)
- Вирази шляху SQL/JSON
- Іноземні обгортки даних: можна підключати до інших баз даних або потоків за допомогою стандартного інтерфейсу SQL
- Налаштовуваний інтерфейс зберігання для таблиць

- Багато розширень, що надають додаткові можливості, включаючи PostGIS[16].

Інтернаціоналізація та пошук тексту:

- Підтримка міжнародних наборів символів за допомогою інтерактивної транслітерації
- Незалежний від регістру та акцентів пошук тексту
- Повнотекстовий пошук

Крім цього, документація PostgreSQL містить багато інших можливостей. PostgreSQL є дуже розширюваним, наприклад, багато функцій, таких як індекси, мають визначені API, щоб ви могли створювати свої власні рішення з використанням PostgreSQL.

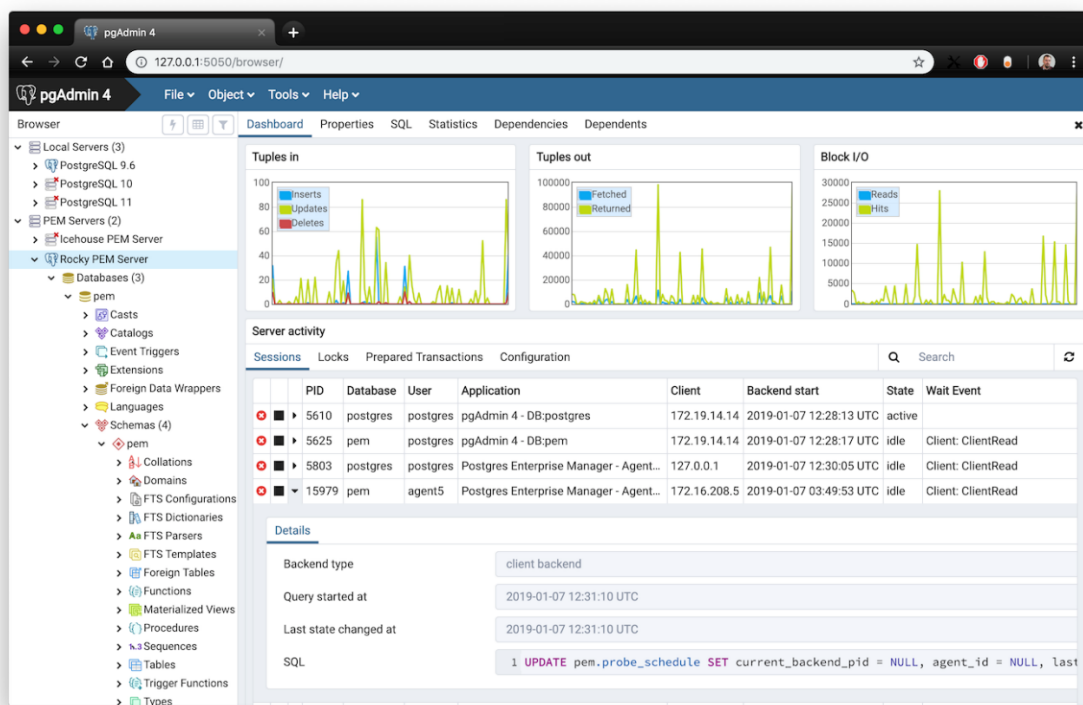


Рис. 2.6. - Інтерфейс бази даних, інструменти

Було підтверджено, що PostgreSQL може ефективно масштабуватися як за обсягом даних, які може обробляти, так і за кількістю користувачів, які можуть використовувати його одночасно. У виробничих середовищах застосовуються активні кластери PostgreSQL, які забезпечують управління багатотерабайтними

обсягами даних, а також спеціалізовані системи, які здатні керувати петабайтами даних.

Docker як інструментарій управління контейнерами

Docker - це відкрита платформа, яка дозволяє розробляти, доставляти та запускати програми. За допомогою Docker ви можете відокремити ваші додатки від інфраструктури та швидко доставляти програмне забезпечення. Docker дозволяє керувати як вашою інфраструктурою, так і вашими програмами. Використання методологій Docker для швидкої доставки, тестування та розгортання коду допоможе значно скоротити затримки між написанням коду та його запуском у виробництво.

Docker дозволяє упаковувати та запускати додатки у вільно ізольованому середовищі, яке називається контейнером. Ізоляція та безпека дозволяють запускати багато контейнерів одночасно на даному хості. Контейнери легкі та містять усе необхідне для запуску програми, тому вам не потрібно покладатися на те, що зараз встановлено на хості. Ви можете легко ділитися контейнерами та бути впевненими, що всі, з ким ви ділитесь, отримують той самий контейнер, який працює однаково.

Docker надає інструменти та платформу для управління життєвим циклом ваших контейнерів, дозволяючи розробляти програму та допоміжні компоненти за допомогою контейнерів. Контейнер стає пристроєм для розповсюдження та тестування вашої програми.

Коли ви готові, використовуйте свою програму виробничому середовищі, використовуючи контейнер або організовану службу. Це підходить для будь-якого типу виробничого середовища - локального, хмарного постачальника або гібридного середовища.

Docker допомагає організувати цикл розробки, надаючи розробникам стандартизовані середовища за допомогою локальних контейнерів, які містять програми та послуги. Контейнери ідеально підходять для безперервної інтеграції та безперервної доставки робочих процесів (CI / CD).

Ось приклад сценарію: ваші розробники пишуть код локально та спільно працюють за допомогою Docker контейнерів. Вони використовують Docker для перенесення своїх програм в тестове середовище та виконання автоматизованих та ручних тестів. Якщо розробники знаходять помилки, вони можуть виправити їх у

своєму розробницькому середовищі та перенести їх у тестове середовище для тестування та перевірки.

Щойно закінчивші тестування можна надати зміни клієнту швидко і просто, подібно до того, як ви надсилаєте оновлення зображень у виробниче середовище. Використання контейнерної платформи Docker дозволяє створювати дуже портативні робочі навантаження, які можуть працювати на різних пристроях, від локальних комп'ютерів розробників до фізичних або віртуальних машин у центрі обробки даних, або в хмарних сервісах. Docker також допомагає управляти динамічним розгортанням робочих навантажень, масштабуванням та знищенням програм і сервісів з урахуванням потреб бізнесу. Це означає, що ви можете запускати більше робочих навантажень на тому ж обладнанні, використовуючи Docker, що є легким і швидким. Docker дозволяє зробити більше за менше ресурсів, що ідеально підходить для середовищ із високою щільністю, а також для малих та середніх розгортань[14].

Docker використовує архітектуру клієнт-сервер, де клієнт Docker взаємодіє з демоном Docker через REST API, сокети UNIX або мережевий інтерфейс. Docker Compose є іншим клієнтом Docker, який дозволяє працювати з програмами, що складаються з набору контейнерів.

Контейнер в Docker є запущеним екземпляром зображення, який можна створити, запустити, зупинити, перемістити або видалити за допомогою Docker API або CLI. Контейнери за замовчуванням добре ізольовані від інших контейнерів та хост-машини. Можна контролювати рівень ізоляції мережі контейнера, сховища та інших базових підсистем від інших контейнерів та хост-машини[15].

З інтерфейсом програми можна ознайомитися на рисунку 2.7.

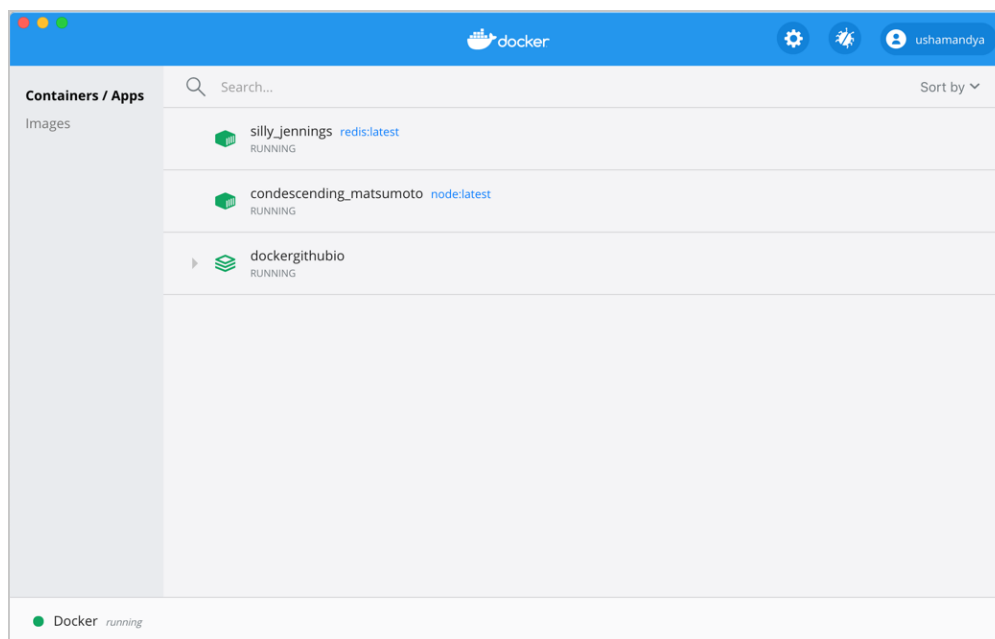


Рис. 2.7. - Інтерфейс програми Docker

Контейнер є комбінацією зображення та конфігураційних параметрів, які ви задаєте при його створенні чи запуску. Після видалення контейнера будь-які зміни, які не були збережені в постійному сховищі, втрачаються.

РОЗДІЛ 3. ЕТАПИ РЕАЛІЗАЦІЇ ПРОЕКТУ

3.1. Вибір засобів реалізації проекту. Веб-сервіс «BotMother»

Botmother - хмарний конструктор чат-ботів, один з найпопулярніших і перших у Східній Європі. Має великий набір функцій і графічний інтерфейс із drag-and-drop.

На сайті сервісу доступний зручний інтерфейс для налаштування розсилок та інтерфейсу, що виглядає наступним чином:

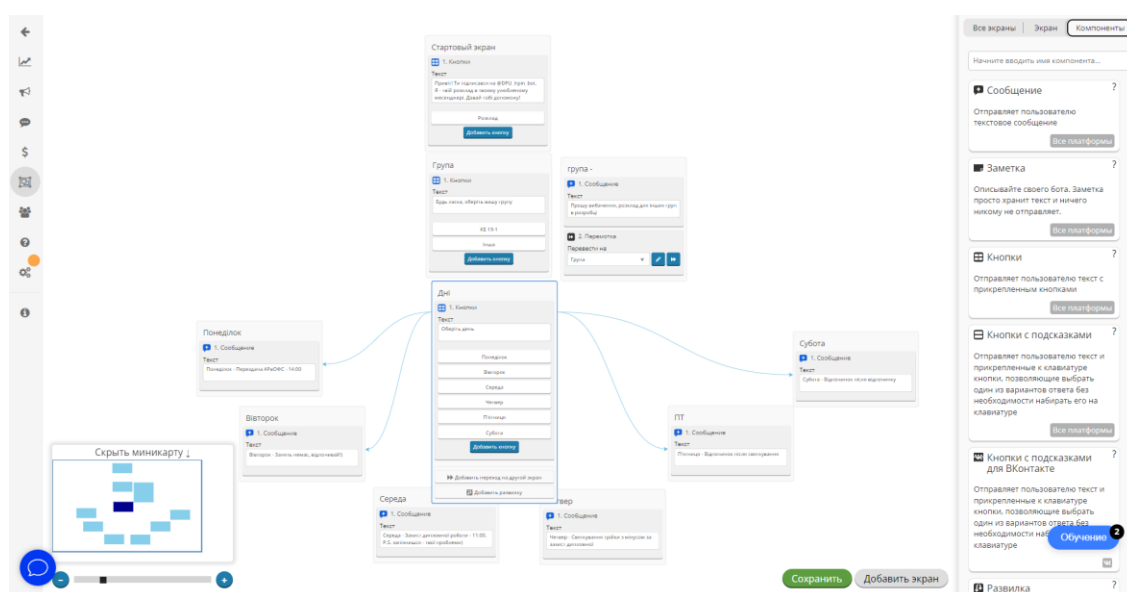


Рис. 3.1. - Інтерфейс веб-сайту налаштування чат-боту «BotMother»

1. Мультиплатформність. Botmother дозволяє створювати та налаштовувати чат-ботів для різних месенджерів, таких як Facebook Messenger, Telegram, Viber, Slack, Line та інші.

2. Швидке створення ботів. Завдяки зручному інтерфейсу та готовим шаблонам Botmother, ви можете створити та запустити свій чат-бот всього за кілька хвилин.

3. Легка інтеграція з іншими сервісами. Botmother дозволяє інтегрувати чат-бота з іншими сервісами, такими як CRM-системи, Google Sheets, Slack, Mailchimp та інші.

4. Підтримка AI-моделей. Botmother має можливість налаштування AI-моделей, які дозволяють ботам розуміти контекст та відповідати на запитання користувачів, що допомагає покращити взаємодію з клієнтами.¹

5. Аналітика. Botmother дозволяє відслідковувати показники ефективності вашого чат-бота, такі як кількість підписників, активність, конверсія та інші.

6. Підтримка 24/7. Ваш чат-бот може працювати цілодобово без зупинок та вихідних, що дозволяє вам завжди бути на зв'язку зі своїми клієнтами.

7. Надійна і безпечна інтеграція з месенджерами. Botmother пропонує безпечні методи авторизації та обміну даними з месенджерами, такими як Telegram, Viber, Facebook Messenger та інші.

8. Аналітика та звіти про результативність. Botmother дозволяє відстежувати ефективність вашого чат-бота, включаючи кількість повідомлень, кількість користувачів, рівень взаємодії та інші метрики. Це дозволяє вам аналізувати результативність вашого чат-бота та вносити зміни для покращення його ефективності.

9. Простота використання. Botmother має інтуїтивно зрозумілий і легкий у використанні інтерфейс для створення та налаштування чат-бота, що дозволяє вам зосередитися на створенні вмісту, а не на технічних деталях.

10. Підтримка клієнтів. Команда підтримки Botmother завжди готова надати необхідну допомогу, відповісти на питання та допомогти вирішити будь-які проблеми, пов'язані з налаштуванням чат-бота.

11. Широкий спектр можливостей. Botmother має безліч корисних функцій, таких як підтримка інтерактивних кнопок, інтеграція з CRM-системами, автоматичне перекладання повідомлень на інші мови та багато іншого, що дозволяє створювати потужні та ефективні чат-боти для вашого бізнесу.

12. Навчання. Сервіс має короткі відео-навчання, які ви можете подивитись за 10 хвилин і створити свій перший чат-бот.

3.2. Головний кореневий бот «Father»

BotFather є унікальним ботом, який забезпечує управління всіма іншими ботами. Це дає можливість створювати нових ботів та змінювати налаштування для вже існуючих.

Для створення нового бота використовується команда `/newbot` у BotFather. Після цього BotFather запитує вас про ім'я бота та ім'я користувача. В результаті отримуємо токен авторизації для нашого нового бота.

Ім'я нашого бота відображається в контактних даних та інших розділах. Ім'я користувача є коротким ідентифікатором, який використовується у згадках та посиланнях на соціальній мережі Telegram (t.me). Імена користувачів повинні бути довжиною від 5 до 32 символів і не розрізняють регістр. Вони можуть містити лише латинські літери, цифри та підкреслення. Ім'я користувача нашого бота має закінчуватися на "bot", наприклад "DPU_Irpin_bot" або "DPU_Rozklad_bot".

Маркер - це рядок, який складається з символів, наприклад "166700989:AFjfFFH1vGWJxfSeofSAsSanLSPDsaw", і він необхідний для авторизації бота та виконання запитів до API бота. Важливо зберігати свій токен у безпеці та зберігати його конфіденційно, оскільки цим токеном може скористатися будь-хто, хто має доступ до вашого бота.

Якщо ваш поточний маркер став недійсним або ви його втратили з якихось причин, ви можете створити новий маркер, використовуючи команду `/token`. Це дозволить вам отримати новий токен для авторизації та продовжувати керувати вашим ботом.

Команди BotFather мають простий зміст:

- `/mybots` - відображає список ваших ботів з зручними елементами керування для редагування налаштувань.

- `/mygames` - виконує ту саму функцію для ваших ігор.

Основні команди для редагування ботів:

- `/setname` - змінює ім'я вашого бота.

- /setdescription - змінює опис бота, короткий текст до 512 символів, який пояснює, що ваш бот може зробити. Користувачі побачать цей текст на початку розмови з ботом під назвою "Що може зробити цей бот?".

- /setabouttext - змінює інформацію про бота, ще коротший текст до 120 символів. Цей текст відображається на сторінці профілю бота. Коли користувачі поділяють вашого бота з іншими людьми, цей текст надсилається разом з посиланням.

- /setuserpic - змінює зображення профілю бота. Чудово мати обличчя на ім'я.

- /setcommands - змінює список команд, які підтримує ваш бот. Користувачі побачать ці команди як пропозиції, коли вони вводять "/" у чаті з вашим ботом. Кожна команда має назву (повинна починатися з "/", складатися з літер, цифр та підкреслень, не довше 32 символів, без урахування регістру), параметри та текстовий опис. Користувачі будуть бачити список команд кожного разу, коли вони вводять "/" у розмові з вашим ботом.

- /deletebot - видалить свого бота та звільнить його ім'я користувача.

- Зміна налаштувань:

- /setinline - перемикає вбудований режим для вашого бота.

- /setinlinegeo - запитує дані про місцезнаходження для надання вбудованих результатів на основі розташування.

- /setjoiningroups - дозволяє додавати вашого бота до груп або вимкнути цю можливість. Кожен бот повинен мати здатність обробляти приватні повідомлення, але якщо ваш бот не призначений для роботи в групах, ви можете вимкнути цю функцію.

- /setprivacy - встановлює, які повідомлення отримуватиме ваш бот при додаванні до групи. При вимкненому режимі приватності бот

отримуватиме всі повідомлення. Рекомендується залишити ввімкненим режим приватності. Після внесення змін ви повинні повторно додати бота до наявних груп, щоб зміни набрали чинності.

Ви можете переглянути зовнішній вигляд цих команд на прикладі нижче:

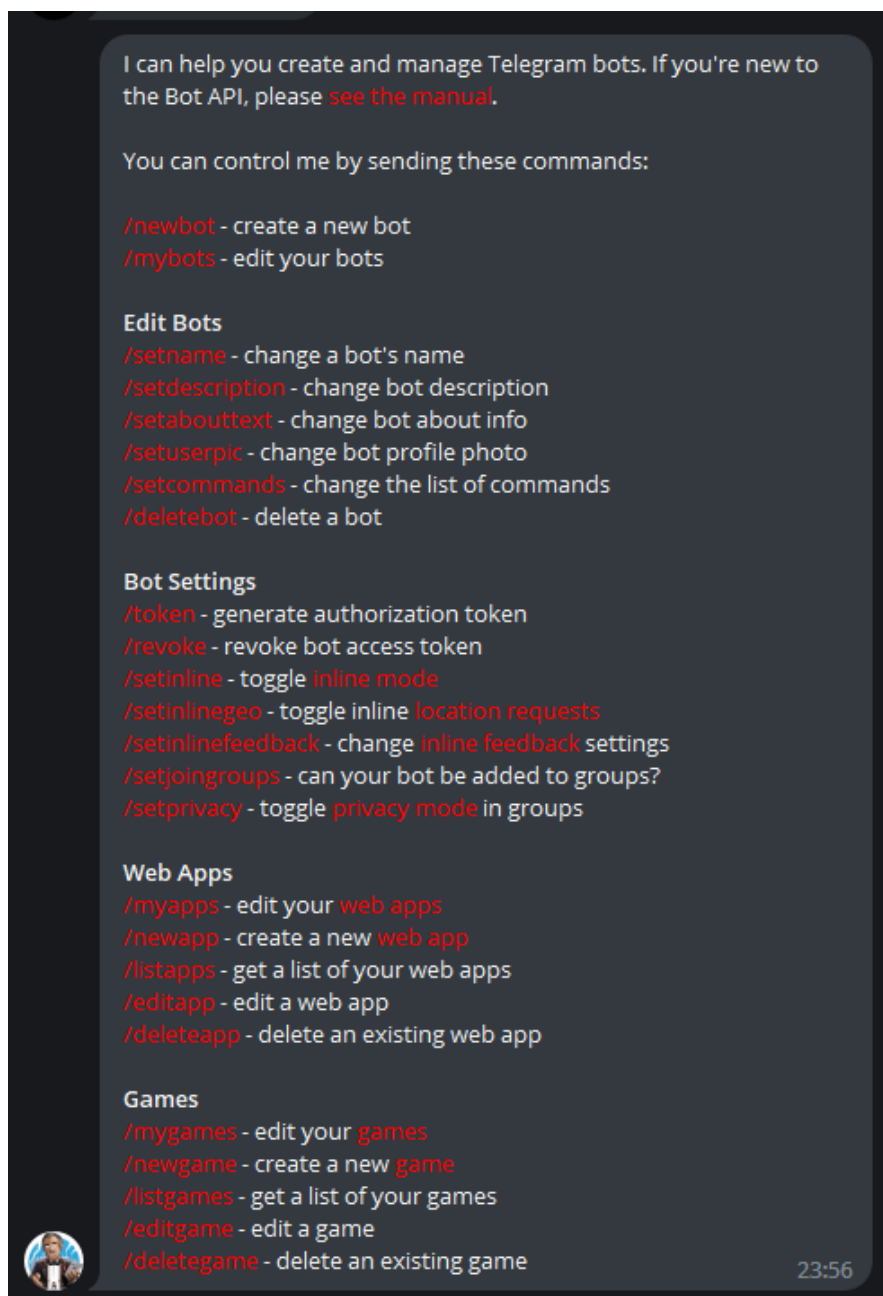


Рис. 3.2. - Вигляд команд кореневого бота

Проектування та розробка директорії файлів проекту

Я створив директорію файлів, використовуючи вищезгадані програмні засоби, що в результаті запускає проект університетського розкладу у вигляді чат-бота. У цій директорії містяться такі складові:

- "parser" - це компонент, який виконує парсинг розкладу для тестування;
- "tests" - це набір юніт-тестів для бота;
- "config" - це файл, в якому знаходяться параметри і загальна конфігурація бота;
- "statistic" - це процес відправки та отримання статистики на платформі chatbase.com;
- "scheduleCreator" - це набір функцій для генерації повідомлень з розкладом;
- "scheduledb" - це класи, які використовуються для роботи з базою даних;
- "setup" - це скрипт для створення файлу налаштувань;
- "setup_db" - це скрипт для налаштування бази даних;
- "UniversityScheduleBot" - це файл для запуску функціонуючого бота;
- "auto_posting_thread" - це скрипт для автоматичного надсилання певного повідомлення;
- "schema.sql" - це схема бази даних;
- "commandsList" - це список команд для "BotFather".

На рисунку 3.3 нижче представлені папки та важливі файли:

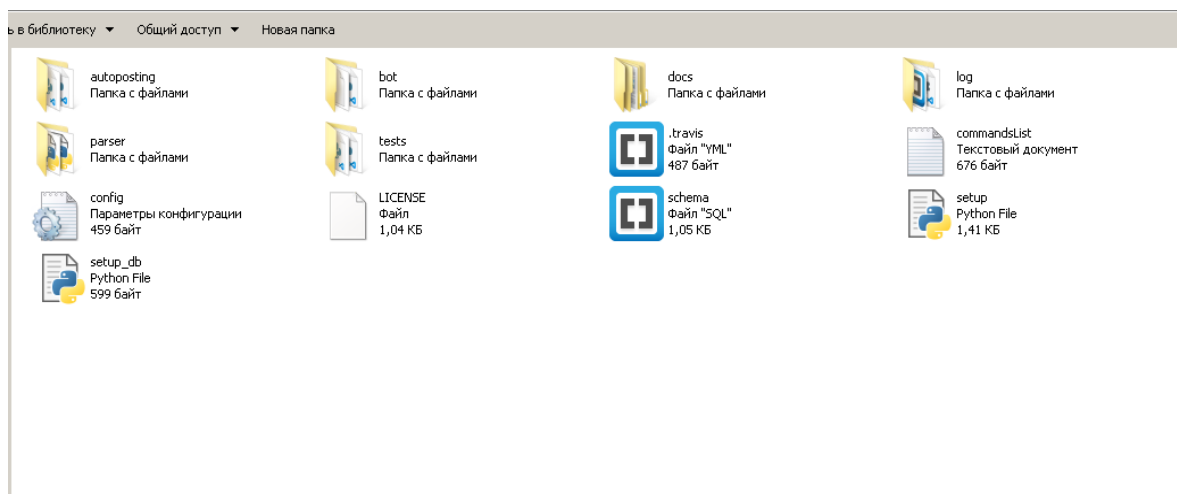


Рис. 3.3. - Вигляд директорії файлів проекту чат-бота

Для продовження роботи з готовими файлами конфігурації необхідно мати дві обов'язкові залежності, про які я вже згадував у попередніх розділах - це Python-середовище (мінімум версії 3) і PostgreSQL.

Саме ці два фактори визначають результат, який ми отримуємо у вигляді чат-бота. Обираючи цей шлях розробки, ви отримуєте початкові навички програмування та легкість у реалізації своїх ідей.

Процес запуску бота в реальному проекті досить простий. Вам потрібно ознайомитись з кількома етапами використання бота та його командами.

Наприклад, етап реєстрації виглядає так:

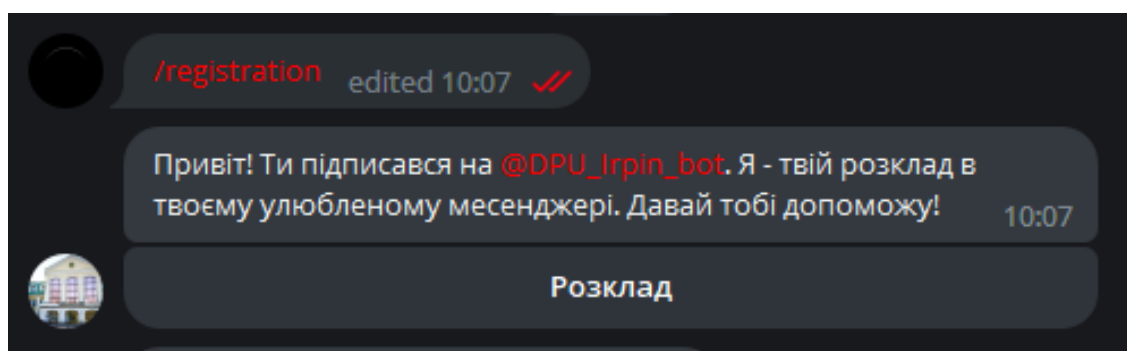


Рис. 3.4. - Реєстрація користувача в системі

І коли користувач буде запитувати систему, вона надаватиме йому правильну інформацію відповідно до його потреб у розкладі. Ось приклад такої ситуації:

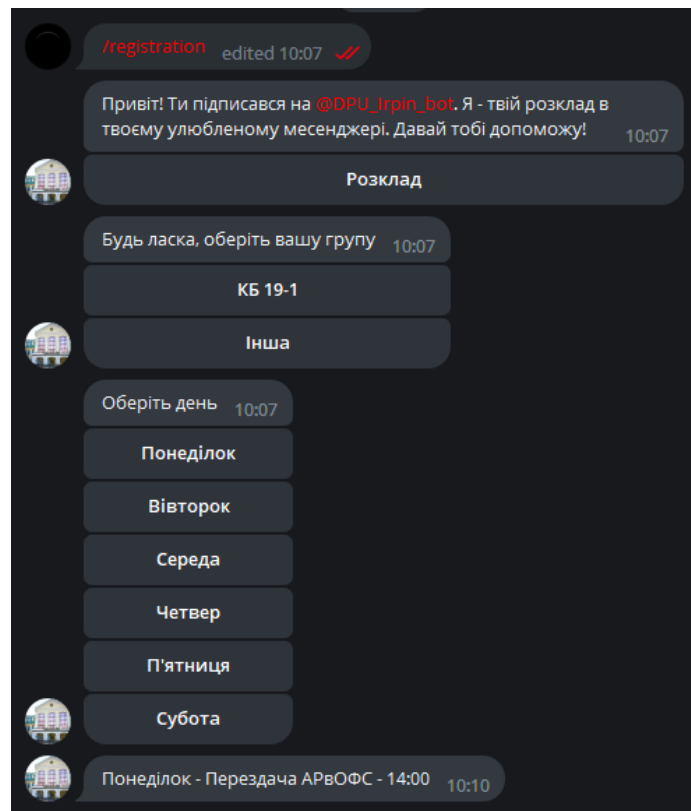


Рис. 3.5. - Демонстрація роботи чат-бота на запит користувача
Надаю скріншот коду, який представляє частину чат-бота:

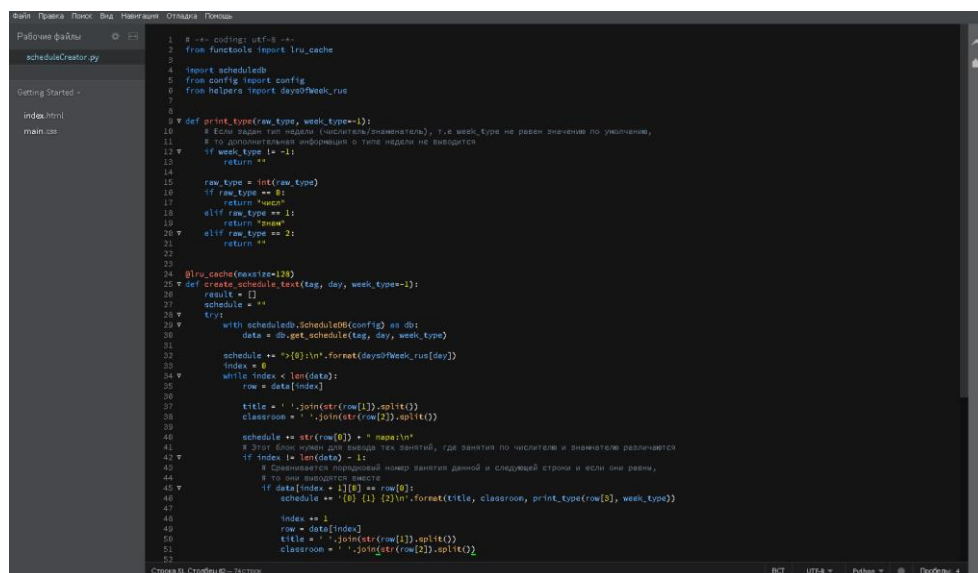


Рис. 3.6. - Приклад програмування одного з файлів розкладу

В моєму проекті директорія файлів містить кілька кодових файлів, які в результаті дають необхідний результат. Цей результат дозволяє користувачу отримати необхідну для нього інформацію або ознайомитися з роботою проекту.

ВИСНОВКИ

У процесі написання моєї випускної кваліфікаційної (дипломної) роботи були проведені наступні дослідження:

✓ Було досліджено месенджер Telegram як сучасний засіб комунікації та платформу для розробки "чат-ботів". Були детально розглянуті типи чат-ботів та приклади їх використання.

✓ Були описані загальні принципи побудови комунікаційного програмного сервісу для платформи Telegram.

Також було проаналізовано сучасні програмні засоби розробки чат-ботів та вибрані необхідні інструменти для ефективної та гнучкої реалізації проекту. Для створення служби тематичного аналізу інформації для платформи Telegram були обрані наступні інструменти: мова програмування Python разом з редактором Sublime Text, використання BotMother та Brackets, а також Telegram.Bot, який використовує API Telegram Bot.

У результаті моєї роботи був розроблений чат-бот для отримання розкладу навчальних занять у Telegram-месенджері. Цей проект є перспективним і може бути розвинутий та модернізований відповідно до нових вимог.

Для подальшого розвитку проекту необхідно спочатку дослідити наявну документацію (інструкції) з використання API системи Moodle та модуля "Розклад" автоматизованої системи управління в Державному Податковому Університеті, щоб ознайомитися з доступними функціями.

Для розробників, які працюють над подібними проектами, важливим є здобуття остаточного досвіду експлуатації продукту користувачем. Для цього важливо слідкувати за відгуками користувачів, аналізувати недоліки та виправляти їх, враховуючи сучасні тенденції та максимізуючи позитивний досвід. В майбутньому можна розглянути використання нейронних мереж для подальшого розширення функціональності.

Практична цінність цієї роботи полягає в тому, що вона надає розробнику досвід, навички і знання, які стосуються створення та підтримки подібних проектів.

Незалежно від результатів роботи, вона допоможе розробнику отримати цінний досвід і ознайомитися з процесом розробки та підтримки таких проектів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Академічний тлумачний словник української мови [Електронний ресурс] - Режим доступу: <http://sum.in.ua/>
2. Довідник Telegram (FAQ) [Електронний ресурс]. - Режим доступу: <https://telegram.org/faq>
3. Обробка природної мови [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Обробка_природної_мови
4. Прикладний програмний інтерфейс [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Прикладний_програмний_інтерфейс
5. Ратушняк Т.В. **Програмування мовою Java: практикум** / Т.В. Ратушняк, Ірпінь: Університет ДФС України, 2016. 214 с.
6. Скороход В. ВИЗНАЧЕННЯ ЗАСОБІВ РОЗРОБКИ ЧАТ-БОТА «ПОМІЧНИК АБІТУРІЄНТА» ДЛЯ СУЧАСНИХ МЕСЕНДЖЕРІВ [Електронний ресурс] / Володимир Скороход – Режим доступу: <https://phm.kspu.kr.ua/nauka/konferentsii/fizyka-tekhnolohii-navchannia/99-2017/kompiuterni-nauky-ta-informatsiini-tekhnolohii/1118-vyznachennya-zasobiv-rozrobky-chatbota-pomichnyk-abituriyenta-dlya-suchasnykh-mesendzheriv.html>
7. Смолинець О.Т. Можливість використання нейронних мереж у системах обміну миттєвими повідомленнями [Електронний ресурс] / О. Т. Смолинець – Режим доступу: <https://nauka-online.com/ua/publications/informatsionnye-tehnologii/2018/5/vozmozhnostispolzovaniya-nejronnyh-setej-v-sisteme-obmena-mgnovennymi-soobshheniyami/>
8. Типові архітектури нейронних мереж [Електронний ресурс] – Режим доступу: <https://studfiles.net/preview/5740125/page:4/>
9. Типові архітектури нейронних мереж [Електронний ресурс] – Режим доступу: <https://studfiles.net/preview/5740125/page:4/>
10. A Modern Bot Architecture [Електронний ресурс] – Режим доступу: <https://guild.beach.io/t/a-modern-bot-architecture/84>
11. Best Telegram Bot Examples to Get Inspired by in 2021 [Електронний ресурс]. - Режим доступу: <https://sendpulse.com/blog/telegram-bot-examples>

12. Bot API 2.0: информация для разработчиков [Электронный ресурс] – Режим доступа: <https://tlgrm.ru/docs/bots/2-0-intro>
13. Bots: An introduction for developers [Электронный ресурс] – Режим доступа до ресурсу: <https://core.telegram.org/bots>
14. Making a telegram bot [Электронный ресурс] – Режим доступа: <https://www.sohamkamani.com/blog/2016/09/21/making-a-telegram-bot/>
15. Most popular global mobile messenger apps [Электронный ресурс]. – 2018. – Режим доступа: <https://www.statista.com/statistics/258749/most-popular-globalmobile-messenger-apps/>
16. Neural-Networks-part-2 [Электронный ресурс] – Режим доступа: <http://www.marekrei.com/blog/neural-networks-part-2-the-neuron/>
17. PLATO (комп'ютерна система) [Электронный ресурс] – Режим доступа: [https://uk.wikipedia.org/wiki/PLATO_\(комп'ютерна_система\)](https://uk.wikipedia.org/wiki/PLATO_(комп'ютерна_система))
18. Radziwill N. Evaluating Quality of Chatbots and Intelligent Conversational Agents [Электронный ресурс] / N. Radziwill, M. Benton. – 2017. – Режим доступа: <https://arxiv.org/ftp/arxiv/papers/1704/1704.04579.pdf>
19. Server-side architecture when bots invade [Электронный ресурс] – Режим доступа: <https://medium.com/@JonathanZWhite/server-side-infrastructurewhen-bots-invade-a2252e9d4bc9>
20. V-Soft Consulting. Chatbots 101 [Электронный ресурс] – Режим доступа: <https://cdn2.hubspot.net/hubfs/1629777/Chatbots%20101.pdf?t=1526138378633>