МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Литвиненко О.Є.

«___» _____2021 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО–КВАЛІФІКАЦІЙНОГО РІВНЯ "БАКАЛАВР"

Тема: Телеграм-бот попереднього запису пацієнтів поліклініки

Виконавець: Грицюк П.Ю.

Керівник: Глазок О.М.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління

Освітньо-кваліфікаційний рівень бакалавр
Спеціалізація 6.050102 "Системне програмування"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Литвиненко О.Є.
«_____» _____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)
Грицюка Павла Юрійовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проекту (роботи): Телеграм-бот попереднього запису пацієнтів поліклініки
затверджена наказом ректора від "04" лютого 2021 року №135/ст.

2. Термін виконання проекту (роботи): з 17.05.2021 до 20.06.2021

3. Вихідні дані до проекту (роботи): чат-бот; *Telegram*; автоматизація; модуль мови програмування *Python Telebot*; інструмент для візуального проектування баз даних *MySQL Workbench.*

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) огляд поняття чат-бот та порівняння мов програмування;
2) проектування чат-бота;
3) розробка чат-бота.

5. Перелік обов'язкового графічного матеріалу:
1) перспективна діаграма варіантів використання;
2) діаграма варіантів використання;
3) діаграма станів;
4) діаграма діяльності.

6. Календарний план

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Аналіз літератури і технічної документації	17.05.2021-18.05.2021	
2	Аналіз основних завдань та цілей проектування	19.05.2021-20.05.2021	
3	Підготування першого розділу.	21.05.2021-22.05.2021	
4	Проектування структури чат-бота	23.05.2021-24.05.2021	
5	Підготування другого розділу	25.05.2021-26.05.2021	
6	Розробка коду чат-бота	27.05.2021-28.05.2021	
7	Підготування третього розділу	29.05.2021-30.05.2021	
8	Оформлення пояснювальної записки та проходження нормоконтролю	31.05.2021-02.06.2021	
7	Підготовка графічного демонстраційного матеріалу	03.06.2021-13.06.2021	

7. Дата видачі завдання «17» травня 2021 р.

Керівник дипломного проекту _____ Глазок О.М.
(підпис)

Завдання прийняв до виконання _____ Грицюк П.Ю.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Телеграм-бот попереднього запису пацієнтів поліклініки»: 54 с., 38 рис., 2 таблиці, 13 використаних джерел.

Ключові слова: ЧАТ-БОТ, БАЗА ДАНИХ, *TELEGRAM, API, BOTFATHER, PYTHON.*

Об’єкт дослідження – процес запису пацієнтів на прийом до лікарів поліклініки.

Предмет дослідження – чат-бот для попереднього запису пацієнтів поліклініки.

Мета дипломного проекту – розробити чат-бота для виконання запису на консультацію на базі мессенджера *Telegram*.

Отримані результати – в ході виконання дипломного проекта було розроблено чат-бот для запису пацієнтів на прийом до лікарів поліклініки. Завдяки роботі чат-бота клієнти мають можливість дистанційно, у режимі он-лайн записатись до необхідного спеціаліста.

Прогнозні припущення щодо розвитку об’єкта дослідження – впровадження створеної системи в реальній поліклініці, доповнення чат-бота функціоналом для лікарів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

API	–	*Application Programming Interface*
IDE	–	*Integrated Development Environment*
CRM	–	*Customer Relationship Management*
URL	–	*Uniform Resource Locator*
FTP	–	*File Transfer Protocol*
FAQ	–	*Frequently Asked Questions*
SQL	–	*Structured Query Language*

ВСТУП

Ринкові умови, що постійно змінюються, висока швидкість прийняття рішень, необхідність зниження ризиків вимагають сучасних підходів до організації підприємницької діяльності. Рішення проблем зовнішніх та внутрішніх операцій підприємства полягає в комплексній автоматизації бізнес-процесів. Це дозволяє вивільнити цінні ресурси для стратегічного планування та концентрації управління в ключових сферах діяльності компанії.

Необхідність автоматизації інформаційних процесів обумовлена збільшенням обсягу інформації в інформаційних системах (ІС) організацій, необхідністю прискорення і використання більш складних методів їх обробки.

Автоматизація бізнесу – це частковий або повний переклад рутинних операцій і бізнес-задач під управління спеціалізованої інформаційної системи або складного апаратного і програмного забезпечення. В результаті цього відбувається вивільнення людських і фінансових ресурсів для підвищення продуктивності праці і ефективності стратегічного управління.

Основними завданнями автоматизації інформаційних процесів є:

– усунення рутинних операцій;

– зниження витрат людської праці при виконанні традиційних процесів і операцій;

– збільшення швидкості обробки інформації; забезпечення більшої ефективності та якості обслуговування клієнтів;

– надання широких можливостей для статистичного аналізу і підвищення точності обліку та звітності інформації;

– надання більших можливостей для організацій та ефективніше використання інформаційних ресурсів за рахунок використання інформаційних технологій.

Таким чином, автоматизація бізнес-процесів – важливий аспект для компаній, оскільки в результаті автоматизації процесів підвищується загальна ефективність компанії.

Мета дипломного проекту – розробити чат-бота для виконання запису на консультацію на базі мессенджера *Telegram*.

Для виконання поставленої мети необхідно вирішити наступні задачі:

– ознайомитись з поняттям чат-бота і його застосуванням;

– розглянути мови програмування і вибрати мову для розробки чат-бота;

– описати процес створення бота.

Предметом дослідження є процес обробки звернень клієнтів в чат відділу консультації клієнтського сервісу поліклініки.

Об’єкт дослідження – процес запису пацієнтів на прийом до лікарів поліклініки.

Предмет дослідження – чат-бот для попереднього запису пацієнтів поліклініки.

Для розробки бота було використано наступні програмні засоби:

– *MS Visio;*

– *Excel;*

– *Notepad++;*

– *MySQL Workbench;*

– *Python 3.*

Практична значимість дослідження полягає в створенні чат-бота, що автоматизує процес обробки звернень клієнта в чат для поліпшення якості обслуговування і скорочення витрат на оплату праці, а також в тому, що даний чат-бот можна застосувати в будь-якій компанії, в якій є схожий бізнес процес.

РОЗДІЛ 1
ОГЛЯД ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ ЧАТ-БОТІВ ТА ЗАСОБІВ ЇХ РОЗРОБКИ

1.1. Поняття та класифікація ботів

Бот, чат-бот – програма, що виконує певні дії у відношенні до користувача. Найчастіше, це зарезервований системою аккаунт, за яким не закріплено будь-яку людину, а повідомлення, що відправляються від імені бота, обробляються зовнішньою системою.

До основних застосувань ботів можна віднести наступне:

– клієнтський сервіс.

За допомогою ботів у користувача з'являється можливість отримувати послуги та здійснювати покупки. До таких послуг можна віднести отримання інмормації, виконання фінансових операцій, перегляд асортименту продукта та навіть обробку графічних файлів.

– підтримка клієнтів.

Чат-бот може надати користувачам відповіді на часті питання, що в свою чергу економить робочий час працівників служби підтримки, до того ж бот може функціонувати цілодобово, а компанія може отримати з його використання фінансову вигоду, оскільки витрати на розробку бота необхідно зробити лише один раз, а працівникам потрібно платити зарплатню кожен місяць.

– рекрутинг.

Завдяки чат-боту можна наймати нових співпрацівників, отримуючи заявки від них через систему. На основі співбесіди кандидата з ботом менеджер з персоналу отримує інформацію щодо рівня знань кандидата та приймає рішення, чи слід запрошувати кандидата на подальші «інтерв'ю».

– маркетинг.

Бот має можливість в автоматичному порядку поширювати контент, підтримувати лояльність клієнтів та збирати аналітичні метрики. За допомогою бота можна робити розсилки, інформувати клієнтів про знижки, акції, будь-які зміни, отримувати зворотній зв’язок про якість надання послуг.

– корпоративні функції.

При підтримці бота співробітники компанії можуть отримати такі привілеї, як інформування про дати відпусток, бронювання переговорів, сроки зарплатні та графік корпоративного

Проаналізувавши класифікації ботів, можна виділити два основних види: бізнес-класифікація (рис. 1.1) та технічна класифікація (рис. 1.2).

Рис. 1.1. Бізнес-класифікація ботів

Бізнес-класифікація ботів. Розглянемо кожен тип більш детально.

– боти-асистенти.

Отримуючи інформацію від користувача та аналізуючи їх запити, витягують необхідні дані.

– розмовні боти.

Створені для комунікації, подібної до спілкування зі звичайною людиною.

– *FAQ*-боти

Принцип функціонування наступний: одне запитання – одна відповідь.

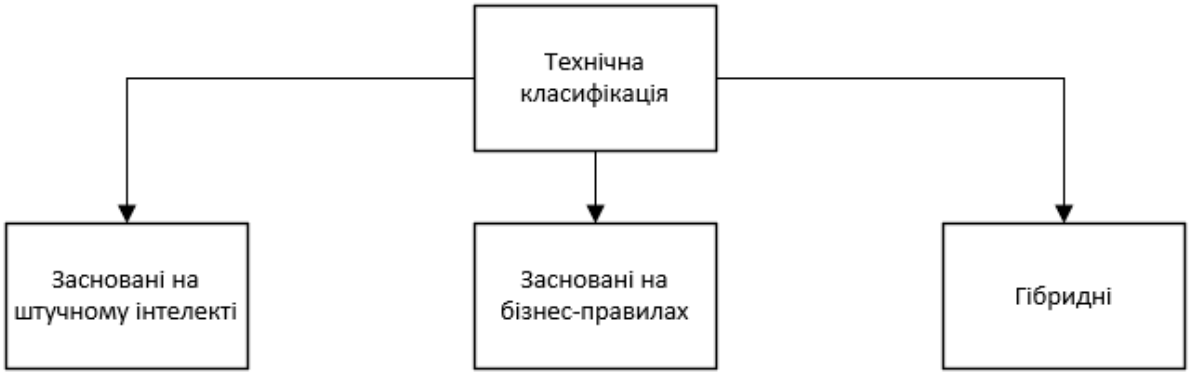

Рис. 1.2. Технічна класифікація ботів

Технічна класифікація ботів. Розглянемо кожен тип більш детально.

– засновані на штучному інтелекті.

Такі боти мають визначену структуру. Шлях діалогу визначається в результаті навчання моделі машинного інтелекту на основі тестових (навчальних) даних. Такі боти, як правило, повинні мати великий обсяг вхідних даних для якісної роботи.

– засновані на бізнес-правилах.

Дані боти мають заздалегідь прописаний сценарій діалогу бота з людиною та репрезентують дерево-подібну структуру бізнес-логіки. Завдяки великій кількості кнопок людина приходить до певної відповіді. Питань з відповіддю у вільній формі в такому чат-боті існувати не може.

– гібридні.

Цей тип об’єднує в собі взаємодію обої типів. Тобто, розмова з користувачем йде за заздалегідь визначеною структурою, але також використовується штучний інтелект для визначення намірів користувача.

1.2. Огляд можливостей мессенджера *Telegram* в порівнянні з іншими

Telegram – кроссплатформенний мессенджер, який надає користувачам можливість обмінюватись повідомленнями в персональному та груповому порядку,

зберігати файли, вести канали (блоги), а також створювати ботів на базі існуючого *API*. Вигляд інтерфейсу мессенджера *Telegram* подано на рис. 1.3, 1.4.

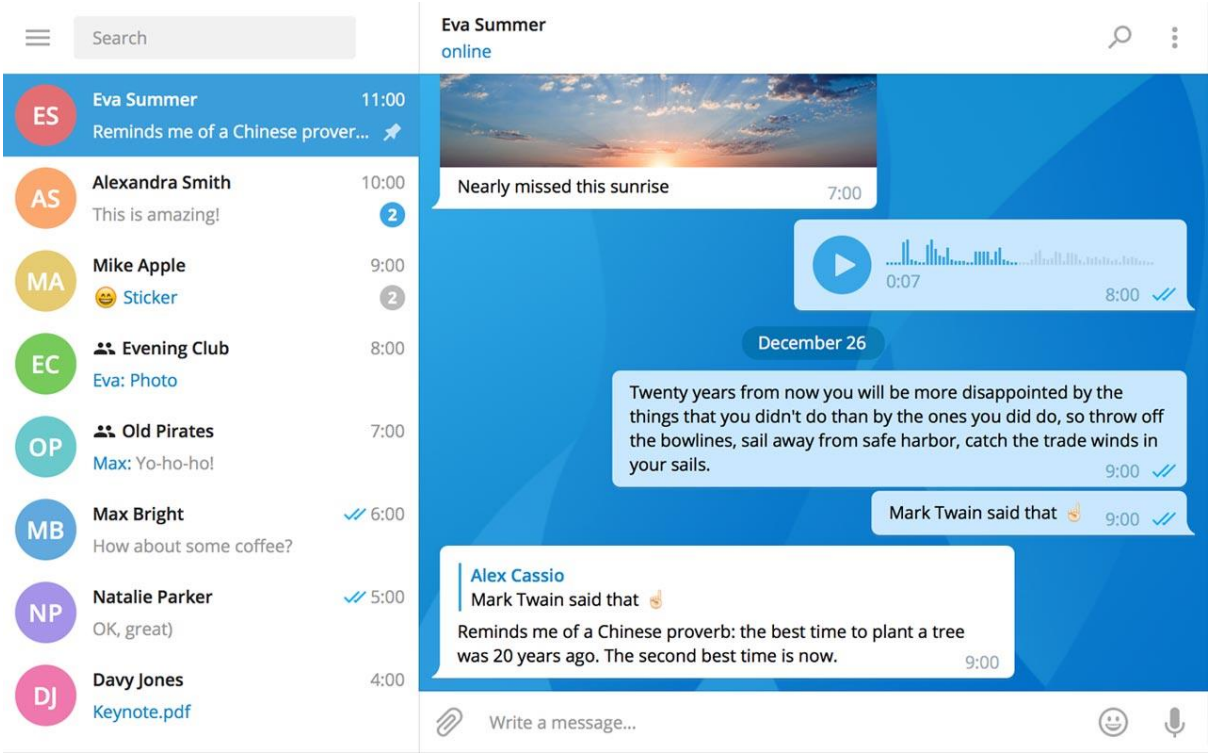

Рис. 1.3. Інтерфейс *Telegram* на *Windows 10*

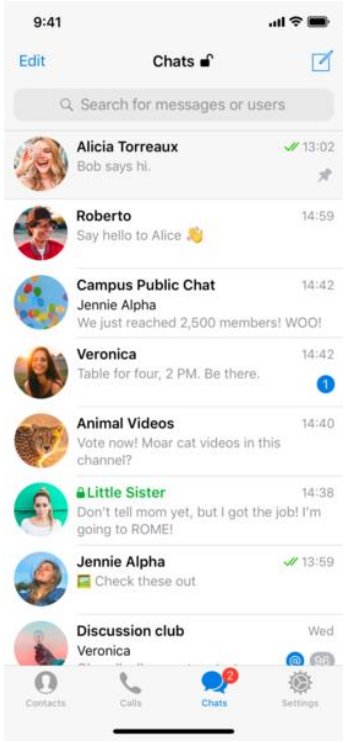

Рис. 1.4. Інтерфейс *Telegram* на *iOS 13*

В *Telegram* реалізована функція «*socks proxy*», що дозволяє користуватись месенджером у тяжких для підключення умовах. Головною перевагою даної функції є можливість роботи серверних додатків за межами міжмережевого екранування, тобто, *socks proxy* може отримати запит від клієнта, що, наприклад, знаходиться за файрволом, переглянути його права доступу, та передати запит на зовнішній сервер.

Telegram одні з найперших відкрили можливість створення ботів у месенджері. Всього, за даними *Forbes* за 2018 рік, у системі вже існувало понад 100 тисяч ботів,

Наведу кілька популярних прикладів: чат-бот Укрпошти та Нової Пошти для відслідковування посилок, бот *iGov* для моніторингу місця в черзі на отримання закордонного паспорту, *RailwayBot* для отримання інформації про вільні місця у поїздах, бот для відстеження курсу валют, чат-бот Приватбанку, що дозволяє клієнтам виконувати транзакції не виходячи з месенджеру, бот під назвою «*To PDF*», що надає користувачам можливість конвертувати файли у формат .pdf, *Opendatabot*, що допомагає шукати інформацію з державних реєстрів: судові рішення, реєстр МВС, перелік юридичних осіб, *vodokanalkiev_bot*, за допомогою якого можна передати показання обсягу лічильника води, *KyivMetropolitenBot* надає можливість придбання *QR*-квитків для проїзду у київському метрополітені, а також повідомляє про розклад руху поїздів, *ZHOVTEN*, робот, котрий бронює квитки в однойменному кінотеатрі «Жовтень», а також *MedicalCorps_bot*, що може надати інформацію про те, як правильно надати першу медичну допомогу.

Facebook Messenger – чат-бот платформа з аудиторією більш 1,2 мільярда активних користувачів на місяць і більше 100 тисяч активних ботів по всьому світу. Провідна платформа для служб обміну миттєвими повідомленнями в США, а набір функцій є найбільш просунутим для розробників: широкий спектр елементів взаємодії (шаблони списку продуктів, розширення, елементи управління, шаблони реєстрації на рейси), аутентифікація, прийом платежів, відправка сервісних повідомлень за номером телефону (поки тільки в США).

Viber – паблік акаунти для бізнесу стали доступні зовсім недавно. Компанії можуть встановити діалог з передплатниками свого аккаунта – через бота або *CRM*. Чат-боти, створені на цій платформі: *Glamour, App in The Air, Aviasales*.

Для середнього і великого бізнесу зробили *WhatsApp Business API* – прямий доступ до *API* месенджера, що дозволяє реалізовувати служби підтримки в месенджері, ботів і т.п.

Розглянемо, чим відрізняється створення чат-ботів в *Telegram, WhatsApp, Viber та Facebook*.

1. Чат-бот в *Telegram*:

Основною перевагою цього додатка для обміну повідомленнями в порівнянні з *WhatsApp* та *Facebook Messenger* є його *API*, відкритий для всіх, який пропонує більше можливостей і дозволяє реалізувати безліч креативних ідей.

Telegram – відмінна безпечна платформа для найкреативніших та складних роботів, наприклад, для інтеграції криптовалютних гаманців, обробки банківської інформації та фінансових транзакцій. Команда *Telegram* також підкреслює простоту інтеграції онлайн-запитів, ігрових роботів *HTML5* і призначених для користувача клавіатур.

2. Чат-бот у *Viber*:

Viber присутній у всьому світі, і багато великих брендів почали використовувати його для спілкування з клієнтами. Представники компанії стверджують, що у месенджера є близько одного мільярда унікальних користувачів по всьому світу, хоча деякі сторонні дослідження показують, що 260 мільйонів є більш близьким до істини числом.

Як і у випадку з *Telegram*, *Viber* не має обмежень *API*, які є у *WhatsApp*. Навпаки, *Viber* пропонує деякі досить витончені функції, яких немає у інших месенджерів. Наприклад, можна повністю відмалювати власний дизайн кнопок, шпалер та ін.

Приклади іноземних viber-ботів: *Mica, Tech Talk, Queen.gr.*

3. Чат-бот в *Facebook Messenger*:

Facebook Messenger налічує 1,3 мільярда користувачів і більше 100 000 активних ботів. Щодня ці боти збирають дані, рекомендують продукти, приймають замовлення і надають підтримку користувачам.

Користувачі можуть відповідати ботам за допомогою тексту, смайликів, *GIF*-файлів, зображень, аудіо, відео, а також перетягувати чат-роботів в групові чати.

Хоча *Facebook Messenger* має широкий спектр інструментів і функцій, він не має команд у вигляді кнопок, що є дивним і незручним. Крім того, обмеження на відправку повідомлень в *Facebook Messenger* можуть розглядатися рекламодавцями як незручність.

Приклади іноземних брендів, які використовують чат-боти в *Facebook Messenger*: *Kindred Bravely, Spotify, TechCrunch*.

4. Чат-бот в *WhatsApp*:

API для цього месенджера платний, а можливість платити за нього не гарантує доступ. Крім того, чат-боти в цьому додатку засновані на принципі відправки повідомлень (немає ніяких команд у вигляді кнопок). Фактично, це не чат-бот в класичному сенсі цього слова, а платформа для автоматичних повідомлень і миттєвих привітань.

Все це робить *WhatsApp* найменш популярною платформою для запуску чат-бота. Мессенджер є відносним новачком в області чат-ботів, рішення впроваджуються повільно, тому неможливо передбачити, коли з'являться нові функції і чи відбудеться це взагалі.

Приклади зарубіжних ботів *Whatsapp*: *KLM Royal, RedBus, Sabrina*.

За переглянутими характеристиками можна зробити висновок, що зараз існує два найцікавіших, що мають багато функцій і можливостей мессенджера – *Telegram* і *Facebook Messenger*. Для реалізації проекту було обрано месенджер *Telegram*, так як він має відкритий *API* та можливість реалізовувати команди у вигляді кнопок, саме на цьому механізмі і буде заснований бот.

1.3. Порівняння існуючих рішень для запису на отримання консультаційних послуг

На сьогоднішній день існує багато варіантів запису клієнтів в медичні установи. Деякі з них більш нові, деякі більш застарілі, тому розглянемо кожен варіант окремо.

1. Фізичний запис у поліклініці.

Цей варіант є найбільш старим. До переваг такого методу можна віднести те, що немає необхідності десь реєструватись, кудись дзвонити та відкривати будь-які сайти. Ти просто приходиш та записуєшся. Проте, до недоліків такого методу відноситься те, що вам скоріше за все прийдеться стояти в черзі, а також ви ризикуєте взагалі не отримати свою послугу, оскільки може виявитись, що необхідний вам лікар сьогодні не приймає або повністю зайнятий.

2. Запис по телефону через контактний центр.

Цей спосіб є одним із самих найпоширеніших, оскільки не потребує фізичної присутності клієнта. Часто трапляється так, що лінія, на яку телефонує клієнт є зайнятою іншими клієнтами, а також контакт-центри мають обмежений графік роботи, що, в теорії, зменшує пропускну спроможність клініки.

3. Запис через веб-сайт.

Для виконання такого запису необхідно мати пристрій, який надасть можливість зайти на сайт. Цей спосіб може бути занадто важким для людей похилого віку, оскільки потребує знань в області технологій.

4. Замовлення зворотнього зв’язку.

Щоби замовити зворотній зв’язок, необхідно або як у запису через контактний центр зателефонувати туди та залишити заявку, або зайти на сайт та замовити його там. Цей метод об’єднує недоліки попередніх методів і його перевага лиш в тому, що він підходить тільки тим, хто має вдосталь часу, щоб чекати на дзвінок.

5. Запис через мобільний додаток.

Цей варіант найбільш схожий на запис через чат-бота, проте розробка додатку потребує набагато більше ресурсів та являється більш довготривалим процесом.

Тому, з точки зору ергономічності та економії ресурсів, цей варіант виглядає як надмірність. Також, мобільний додаток необхідно буде завантажувати та він буде займати місце на пристрої, що може влаштувати далеко не всіх клієнтів.

6. Запис через чат-бота.

Даний спосіб не потребує фізичного контакту з іншою людиною, чат-бот працює цілодобово та не потребує встановлення сторонніх додатків або відвідувань сайтів. Для того, щоб записатися, клієнту необхідно буде лише відкрити бота у месенджері, яким він повсякденно користується, та відповісти на його питання.

З огляду на переваги та недоліки перелічених варіантів, чат-бот виглядає як найбільш просте, економне та ефективне рішення.

1.4. Вибір програмного забезпечення для розробки бота

1.4.1. Основні поняття та види мов програмування

Мова програмування – набір конструкцій та правил, що визначають, як буде виглядати створена комп'ютерна програма та які властивості або функціонал вона буде мати. Програма – код, створений відповідно до парадигм та правил даної мови програмування. Вихідний код – код, з якого складається «скелет» програми.

Мови програмування – штучні мови. Проте, як і природні мови, вони включають в себе такі поняття, як алвафіт, словарний запас, граматику, синтаксис, а також семантику.

Алфавіт – набір символів, дозволений до використання, за допомогою якого можуть бути утворені слова даної мови.

Семантика – система правил тлумачення всіх мовних конструкцій, що дозволяє виконувати процес обробки даних.

Синтаксис – система правил, що визначає допустимі конструкції мови програмування з символів алфавіту.

Всі мови програмування можна поділити на два основних види: мови низького та високого рівня:

– Мови низького рівня жорстко орієнтовані на конкретний тип обладнання. Загалом, мови низького рівня – спосіб написання комп'ютерних інструкцій на апаратній мові, тобто, в машинних кодах.

– Мови високого рівня – це мови, що дозволяють записувати програми в зручній для людини формі. Вони орієнтовані не на систему інструкцій апаратного забезпечення, а на систему операторів (інструкцій), що характерна для написання певного класу алгоритмів.

Мови високого рівня більш прості у використанні, оскільки їх завдання – обслуговувати потреби програміста, а не визначати можливості комп'ютера. Програми, написані на цих мовах, повинні бути перекодовані - тобто, переведені на машинну мову, щоб перед запуском програми комп'ютер міг їх зрозуміти. Тому системи програмування, наприклад на *Java* включають в себе або інтерпретатор мови, або компілятор.

Мови низького рівня, близькі до машинної мови, дозволяють створювати програми, які працюють швидше і дозволяють більш ефективно використовувати ресурси комп'ютера.

1.4.2. Статистика та аналіз мов програмування для розробки бота на базі мессенджера *Telegram*

В сучасному світі майже кожна людина використовує додатки для полегшення життя, додатки, що роблять повсякденні задачі менш нудними та підвищують загальну ефективність людини.

На сьогоднішній день існує велика кількість різноманітних мов програмування, у кожної з них своя область застосування, проте задля проведення аналізу з метою вибору найкращої мови програмування для написання бота необхідно обрати декілька самих популярних мов програмування, щоб між порівняти кожну з них.

На рис. 1.5 відображено відсоток використання мов програмування за інформацією dou.ua в комерційних робочих проектах 2021 року.

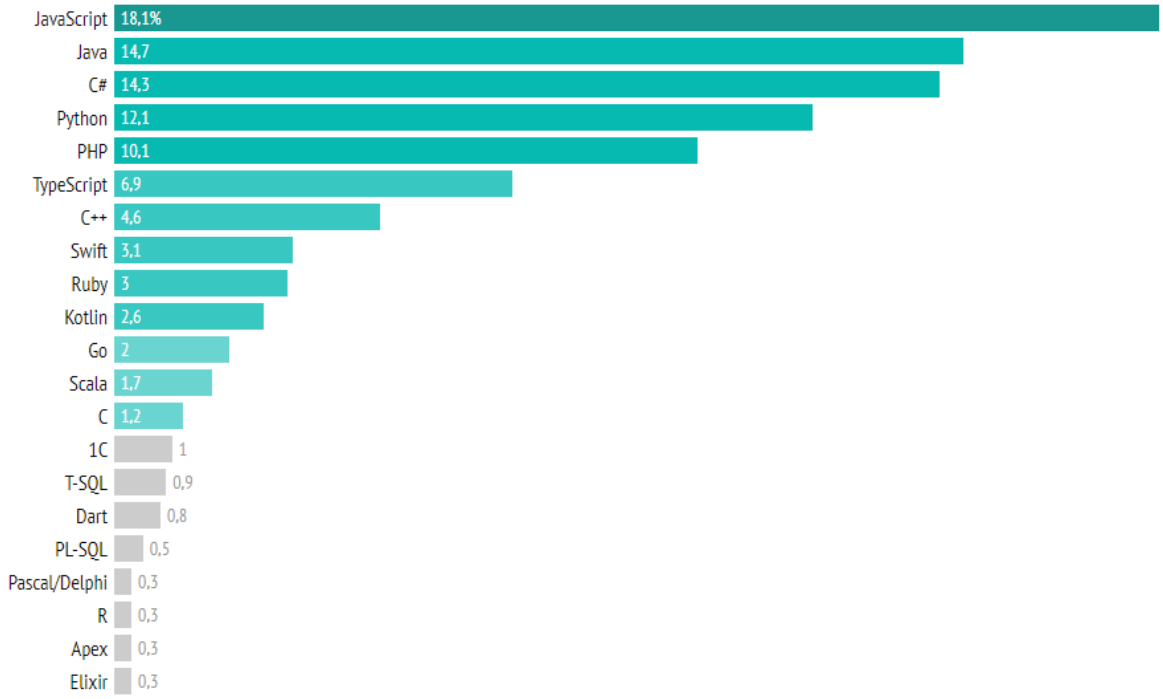

Рис. 1.5. Відсоток використання мов програмування в комерційних проектах

За даними напрошується висновок, що *JavaScript* значно випереджає *Java* і зараз є найпопулярнішою мовою програмування. У п'ятірку кращих мов увійшли також: *C#, Python, PHP*. На рис. 1.6 можна подивитися, як змінювалися дані з 2012 по 2021 рік.

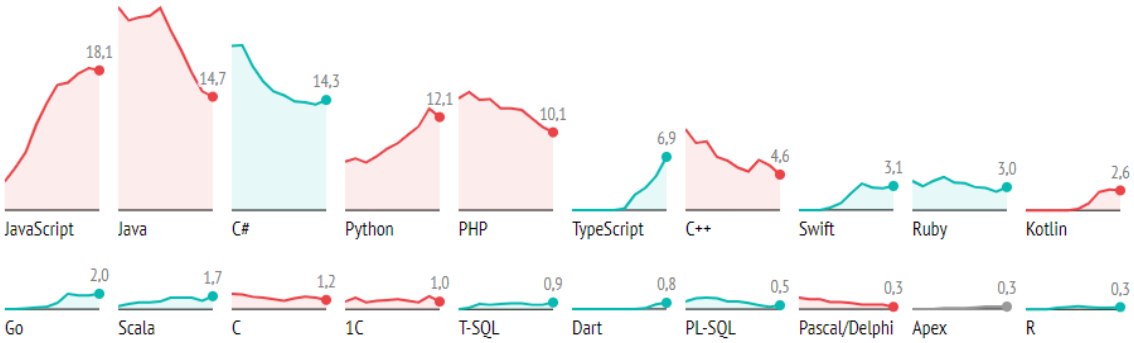

Рис. 1.6. Графік зміни відсотку використання мов програмування в комерційних проектах за період з 2012 по 2021 рік

За рис. 1.6 можна зробити висновок, що популярність мови *Java* і *C#* сильно падає, а популярність таких мов як *JavaScript*, *TypeScript* і *Python* продовжує зростати.

Далі розглянемо рейтинг популярності по особистим уподобанням, який представлений на рис. 1.7.

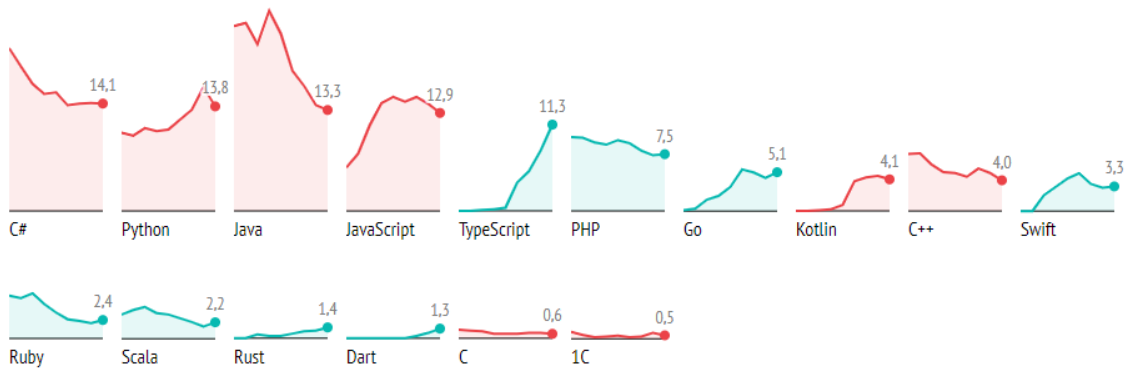

Рис. 1.7. Графік зміни відсотку популярності мов по особистим уподобанням

За рис. 1.7 можна зробити висновок, що за особистим уподобанням люди обирають *C#*, далі *Python*, а потім *Java*.

Ще в цьому розділі хотілося б подивитися статистику з пріоритетів вивчення нових мов, рис. 1.8.

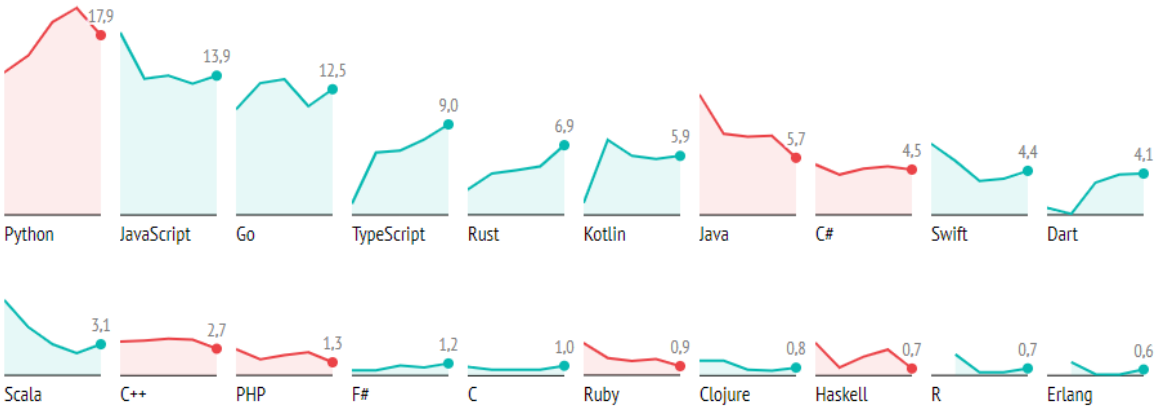

Рис. 1.8. Відсоток обрання мов для вивчення з наступного року

За рис. 1.8 наочно видно, що *Python*, хоч його популярність трохи і впала, збирається вивчити найбільша кількість людей.

Висновок: отже, за даними, представленим вище найпопулярнішими мовами на даний момент, є *JavaScript*, *Python* і *C#*, тому для аналізу і порівняння виберемо саме ці мови.

1.4.3. Порівняльний аналіз мов програмування

Python – це високорівнева мова програмування загального призначення, яка також використовується для розробки веб-додатків. Мова націлена на підвищення продуктивності праці розробників і читабельності коду.

Python підтримує різні парадигми програмування: структурну, об'єктно-орієнтовану, функціональну, імперативну і аспектно-орієнтовану. Мова включає в себе динамічну типізацію, автоматичне керування пам'яттю, повне самоспостереження, механізм обробки винятків, підтримку багатопотокових обчислень і практичні структури даних високого рівня.

Переваги *Python*:

– відкрита розробка;

– дегкий у вивченні на початковому рівні;

– особливості синтаксису стимулюють програміста писати код, що добре читається;

– надає засоби швидкого прототипування і динамічної семантики;

– має широке ком’юніті, позитивно налаштоване по відношенню до новачків;

– безліч корисних бібліотек і розширень мови можна легко використовувати в своїх проектах завдяки гранично-уніфікованому механізму імпорту та програмним інтерфейсами;

– механізми модульности добре продумані і можуть бути легко використані;

– абсолютно все в *Python* є об'єктами в сенсі ООП, але при цьому об'єктний підхід не нав'язується програмісту.

Недоліки *Python*:

– не сама вдала підтримка багатопоточності;

– на *Python* створено не так вже й багато якісних програмних проектів у порівнянні з іншими універсальними мовами програмування, наприклад, з *Java*;

– відсутність комерційної підтримки засобів розробки (хоча ця ситуація з часом змінюється);

– початкова обмеженість засобів для роботи з базами даних;

– бенчмарки показують меншу продуктивність *Python* в порівнянні з основними *Java VM*, що створює цій мові репутацію повільної.

JavaScript – мультипарадигменна мова програмування. Підтримує об'єктно-орієнтований, імперативний і функціональний
стилі.

Переваги *JavaScript*:

– жоден сучасний браузер не обходиться без підтримки *JavaScript*;

– з використанням написаних на *JavaScript* плагінів і скриптів впорається навіть не фахівець;

– корисні функціональні налаштування;

– мова постійно вдосконалюється - зараз розробляється бета-варіація проекту, *JavaScript2*;

– взаємодія з додатком може здійснюється навіть через текстові редактори.

– перспектива використання мови в процесі навчання програмуванню та інформатиці.

Недоліки *JavaScript*:

– знижений рівень безпеки через доступ до вихідного коду популярних скриптів.

– безліч дрібних дратівливих помилок на кожному етапі роботи. Більша частина з них легко виправляється, але їх наявність дозволяє вважати цю мову менш професійною, порівняно з іншими.

– повсюдне поширення. Своєрідним недоліком можна вважати той факт, що частина активно використовуваних програм (особливо додатків) перестануть існувати при відсутності мови, оскільки цілком базуються на ній.

C# – мова програмування, що поєднує об'єктно-орієнтовані і
контекстно-орієнтовані концепції.

Переваги *C#*:

– для малих підприємств і деяких окремих розробників безкоштовні інструменти включають *Visual Studio, Azure Cloud, Windows Server* і багато інших;

– велика кількість синтаксичних конструкцій, розроблених для кращого розуміння написання коду;

– дуже простий у вивченні;

– є ціла спільнота з досвідчених програмістів.

Недоліки *C#*:

– пріоритетна орієнтованість на *Windows* платформу;

– мова безкоштовна тільки для невеликих фірм, індивідуальних програмістів, стартапів і учнів. Великій компанії покупка ліцензійної версії цієї мови обійдеться в кругленьку суму;

– в мові залишилася можливість використання оператора безумовного переходу.

Вибір мови програмування для чат-бота потрібно зробити з урахуванням призначення розробки. Чат-бот – це програма, яка з'ясовує потреби користувачів, а потім допомагає задовольнити їх.

Кілька прикладів переваг створення бота на *Python*:

– велика кількість модулів та бібліотек для розробки під *Telegram*;

– простий синтаксис, що допомагає уникати великої кількості помилок;

– одразу створюється код, що легко читати.

Недоліки:

– обмеженість при роботі з базами даних.

Підбивши підсумки, у кожної мови програмування є свої плюси і мінуси, кожна мова підходить для конкретних цілей, наприклад якщо створювати сайт, то за описаними критеріями потрібно вибрати *JavaScript*, а для написання чат-бота в месенджері *Telegram* більше підійде *Python* так як він більш простий в написанні, має багато документації і має великий вибір бібліотек, які допоможуть створити хороший сервіс.

1.4.4. Вибір текстового редактора для роботи з кодом

Для написання коду нам потрібен текстовий редактор – незалежна комп'ютерна програма або компонент програмного комплексу, призначена для створення і зміни текстових даних і текстових файлів. Код пишеться, налагоджується і виконується з використанням текстового редактора.

Вибір ідеального редактора для роботи – складне завдання, яке включає в себе: тестування, особисті уподобання і остаточне рішення. Перш ніж зробити вибір, нам потрібно проаналізувати деякі з них.

Visual Studio Code (рис. 1.9) – це продукт корпорації *Microsoft*, розроблений для того, щоб дозволити розробникам писати код без завантаження масивної *Visual Studio* (3 *ГБ* +). *Visual Studio Code* – простий редактор з відкритим вихідним кодом, який однаково добре працює в *Windows, OS X* і *Linux*. Основні функції *Visual Studio Code* включають підтримку більше 30 мов, автозаповнення, просту навігацію і т. д. Він також включає засоби налагодження і *Git* для полегшення розробки.

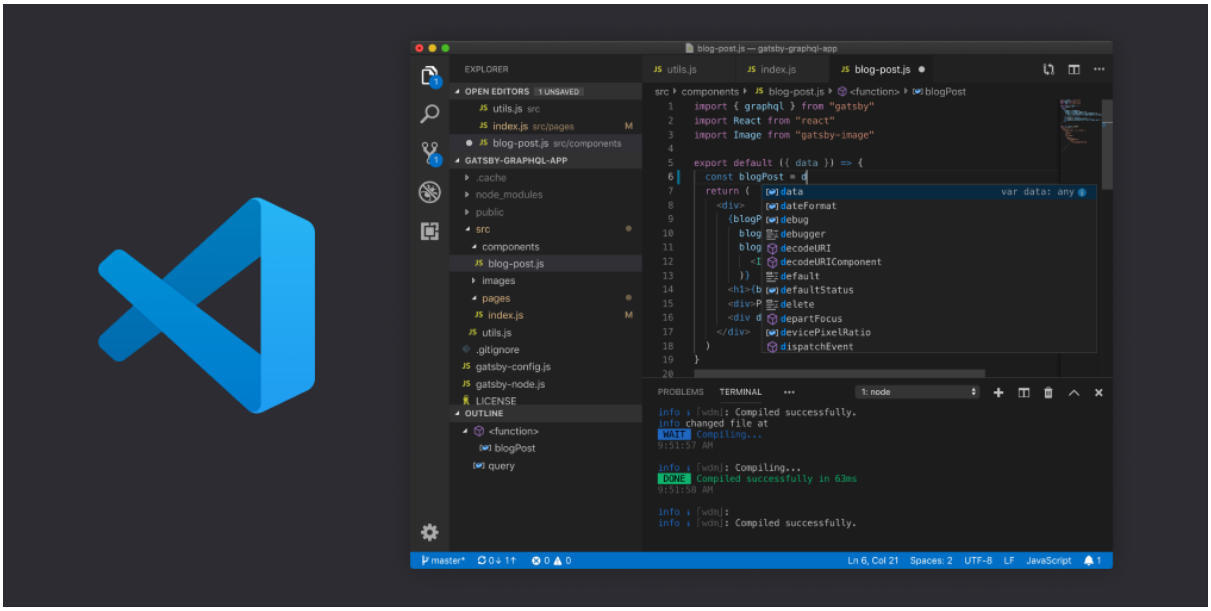

Рис. 1.9. *Visual Studio Code*

Переваги:

– підтримка більш ніж 30 мов, а також основні мови *Microsoft*, такі як *ASP.NET, C#* і т.д.;

– невеликий розмір гарантує швидку установку і швидке використання.

Недоліки:

– підтримка розширень потребує поліпшення;

– оновлення на *Linux* може бути непростим завданням.

Visual Studio Code – відмінний вибір для програмістів, які не хочуть завантажувати і використовувати громіздкі інтегровані середовища розробки.

Notepad++ (рис. 1.10) – це редактор з відкритим вихідним кодом. Він схожий на блокнот, який підтримує масу мов програмування. Найбільшою перевагою *Notepad++* є те, що він може обробляти великі файли без тривалих затримок або збоїв.

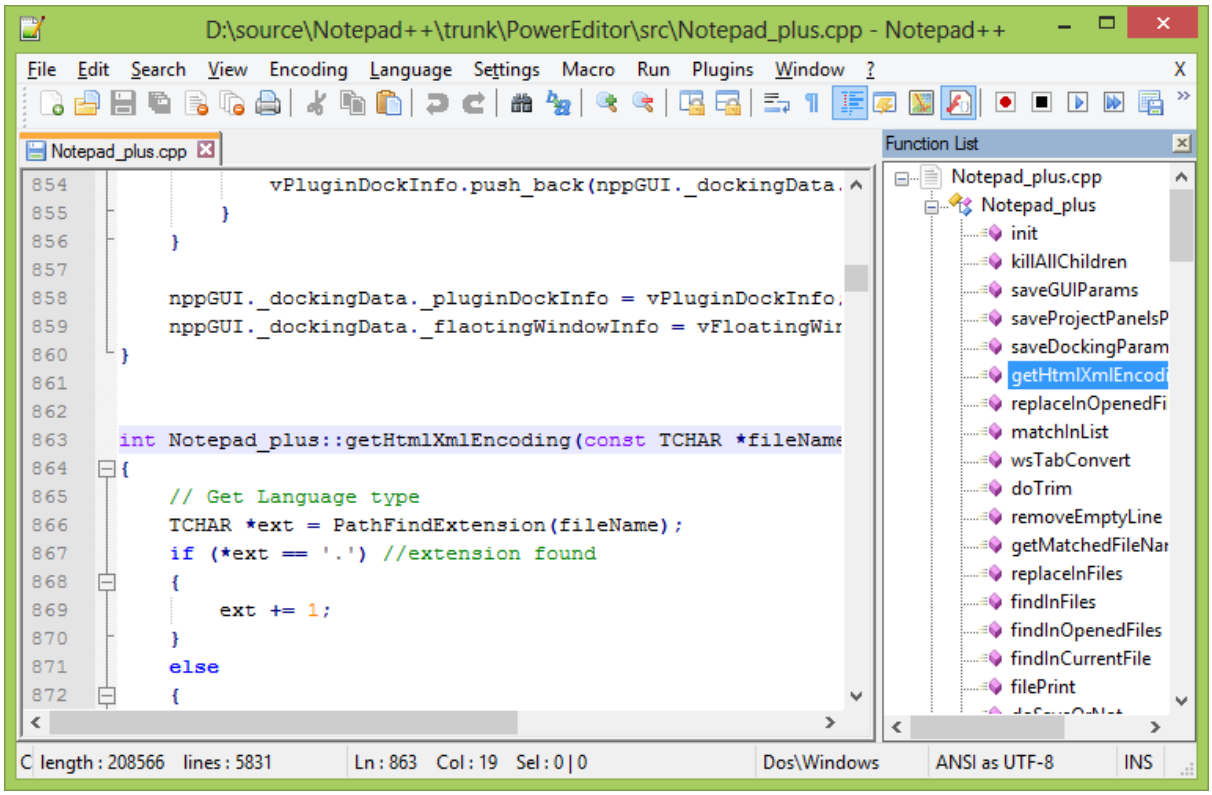

Рис. 1.10. *Notepad++*

Переваги:

– підтримка великої кількості кодувань;

– підсвічування синтаксису;

– паралельне редагування документів;

– порівняння документів;

– пошук і автозаміна за регулярними виразами;

– робота з файлами на *FTP* сервері.

Недоліки:

– не найкращий для користувача інтерфейс;

– занадто простий.

Notepad++ відмінно підійде тим, хто задоволений простим інтерфейсом і набором функцій. Він допоможе вам швидко і легко створювати нові продукти.

Sublime Text editor (рис. 1.11) – один з кращих текстових редакторів на сьогоднішній день. Це відмінна альтернатива потужної *IDE*, вона легка і виконує свою роботу з високою ефективністю і точністю.

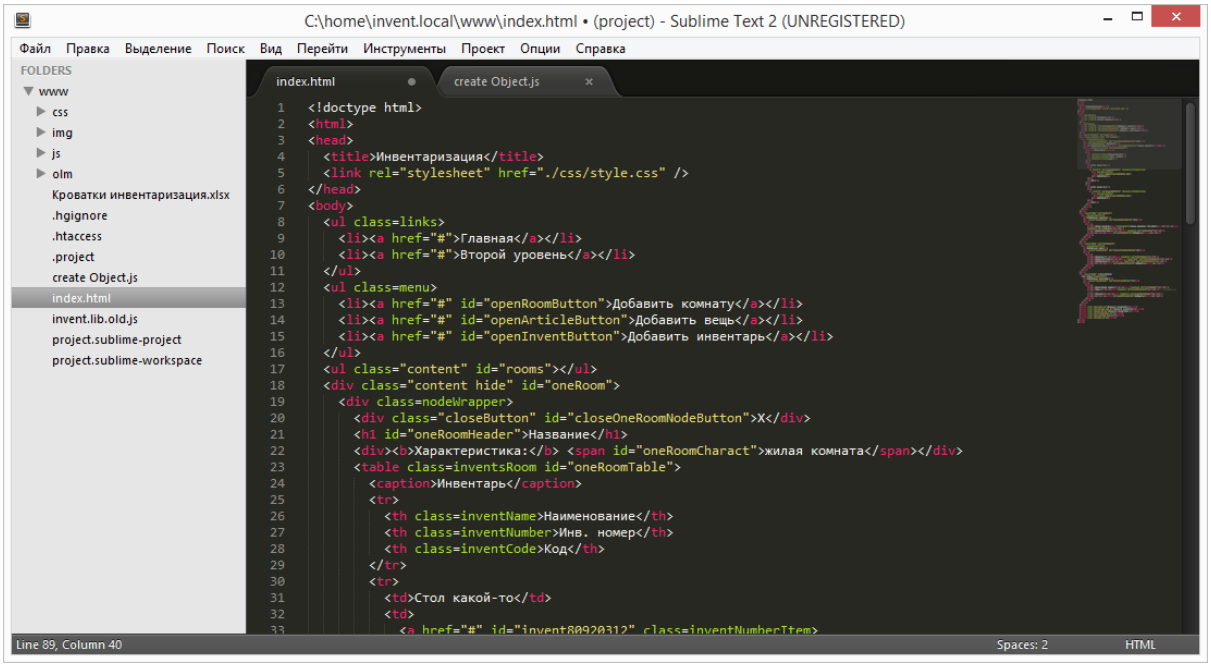

Рис. 1.11. *Sublime Text editor*

Sublime Text надає багато можливостей, але, як і будь-який інструмент він не є досконалим. Розглянемо плюси і мінуси використання *Sublime Text editor*.

Переваги:

– приємний, легкий, мінімалістичний інтерфейс;

– дуже гнучкі налаштування. Множинне виділення;

– можливість створення будь-яких заготовлених фрагментів і вставки їх хоч по гарячих клавішах, хоч по літерним скороченням;

– можливість призначення гарячих клавіш абсолютно на будь-яку дію;

– наявність міні-карти коду для зручного переміщення;

– можливість відображення прихованих символів.

Недоліки:

– *Sublime text* платний;

– час завантаження довше в порівнянні з *Notepad++*;

– в цілому, *Sublime text* – відмінний редактор для роботи. Він може бути використаний безкоштовно з нескінченним пробним періодом.

Для написання сервісу був обраний *Notepad++*, враховуючи всі переваги і недоліки, він виявився найзручнішим для роботи.

1.4.5. База даних для бота

MySQL Workbench – інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення та експлуатацію БД в єдине оточення для системи баз даних *MySQL*.

Перша версія *MySQL Workbench* була випущена у вересні 2005 року. *MySQL Workbench* був першим сімейством продуктів, який був доступний в двох варіантах. Щоб залучити розробників в основну команду розробки, комерційна стандартна версія програми (англ. *Standard Edition*) пропонується поверх вільної версії (англ. *Community Edition*), що розповсюджується під ліцензією *GNU GPL*. «*Community Edition*» є повнофункціональним продуктом, що володіє всіма основними можливостями комерційного варіанту. Будучи основою для всіх

майбутніх релізів, він буде отримувати користь від усіх майбутніх зусиль, прикладених для розвитку продукту. «*Standard Edition*» розширює «*Community Edition*» серією модулів і плагінів, що дозволяють оптимізувати робочий процес і, тим самим, заощадити час і уникнути помилок.

Можливості програми:

– дозволяє наочно представити модель бази даних в графічному вигляді;

– наочний і функціональний механізм установки зв'язків між таблицями, в тому числі «багато до багатьох» зі створенням таблиці зв'язків;

– *Reverse Engineering* – відновлення структури таблиць з уже існуючою на сервері БД (зв'язки відновлюються в *InnoDB,* при використанні *MyISAM* - зв’язки необхідно встановлювати вручну);

– зручний редактор *SQL* запитів, що дозволяє відразу ж відправляти їх на сервер і отримати відповідь у вигляді таблиці;

– можливість редагування даних в таблиці у візуальному режимі.

1.5. Висновки до розділу

У цьому розділі було розглянуто загальні поняття чат-бота, показані функції, наведено класифікацію всіх існуючих на даний момент чат-ботів. Також були показані переваги використання месенджерів в бізнесі. Був проведений вибір програмного забезпечення для реалізації чат-бота, за результатами якого було обрано такі засоби: мова програмування *Python,* текстовий редактор *Notepad++*, месенджер *Telegram* і база даних *MySQL Workbench*.

Чат-бот для запису на консультацію в поліклініку вирішує проблему великих черг у поліклініках. Автоматизивуваши цей процес, поліклініка зможе зменшити навантаження на консультантів, оскільки клієнти отримають можливість записатись самостійно. Для клієнтів зручність заключається в тому, що їм не потрібно буде нікуди йти або дзвонити, треба лише виконати декілька натискань на смартфоні.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ЧАТ-БОТА

2.1. Модулі *Python*, необхідні для розробки сервісу

В першому розділі було з’ясовано, що мова програмування *Python* добре підійде для написання чат-бота для мессенджера *Telegram*. Розглянемо модулі даної мови, необхідні для написання чат-бота.

Модуль *random* надає змогу генерації випадкових чисел, букв, випадкового вибору елементів послідовності.

Модуль *datetime* надає класи для обробки часу і дати різними способами. Підтримується і стандартний спосіб представлення часу, проте більший акцент зроблено на простоту маніпулювання датою, часом і їх частинами.

JSON (JavaScript Object Notation) – простий формат обміну даними, заснований на підмножині синтаксису *JavaScript*. Модуль *json* дозволяє кодувати і декодувати дані в зручному форматі.

Time – модуль для роботи з часом в *Python*.

Модуль *telebot* необхідний для створення і підключення чат-бота.

Модуль *messagebox* надає базовий клас шаблону, а також безліч зручних методів для часто використовуваних конфігурацій. Поля повідомлень є модальними і повертатимуть підмножину («*True*», «*False*», «*OK*», «*None*», «*Yes*», «*No*») на основі вибору користувача.

Requests – це модуль *Python*, який можна використовувати для відправки всіх видів *HTTP*-запитів. Це проста у використанні бібліотека з безліччю функцій, від передачі параметрів в *URL* до відправки призначених для користувача заголовків.

Модуль *sys* забезпечує доступ до деяких змінних і функцій, взаємодіє з інтерпретатором python.

Модуль *subprocess* відповідає за виконання наступних дій: породження нових процесів, з'єднання с потоками стандартного введення, стандартного виводу, стандартного виводу повідомлень про помилки і отримання кодів повернення від цих процесів.

Модуль *xlsxwriter* – це модуль *Python* для запису файлів у форматі файлів *Excel* 2007+ *XLSX*. *XlsxWriter* можна використовувати для написання тексту, чисел, формул та гіперпосилань на декілька аркушів, і він підтримує такі функції, як форматування, об’єднання комірки, діаграми, автофільтри, зображення робочого аркушу.

Модуль *MySQL Connector* / *Python* надає програмам *Python* доступ до баз даних *MySQL*, використовуючи *API*, який відповідає специфікації *API* бази даних *Python*.

2.2. *Telegram Bot API*

Bot Api представляє собою *HTTP*-інтерфейс для роботи з ботами в *Telegram*. Кожен бот є спеціальним аккаунтом, створеним для автоматичної обробки та відправлення повідомлень. Існує два протилежних за логікою способи отримання оновлень від бота:

– *long pooling* – додаток автоматично надсилає запити серверам *Telegram* на наявність будь-яких оновлень для бота. За замовчуванням стоїть проміжок в 100 мс.

– *webhook* – сервера *Telegram* самостійно сповіщають додаток на сервері, як тільки з’явиться будь-яке оновлення.

Вхідні оновлення будуть зберігатися на сервері до тих пір, поки їх не оброблять, проте не довше 24 годин. Незалежно від способу отримання оновлень, у відповідь відправляється *Update*, серіалізований в *JSON*. Всі запити до *Telegram Bot API* повинні бути виконані через *HTTPS* у наступному вигляді:

https://api.telegram.org/bot<token>/НАЗВА_МЕТОДУ. Принцип взаємодії чат-бота та користувача зображено на рис. 2.1.

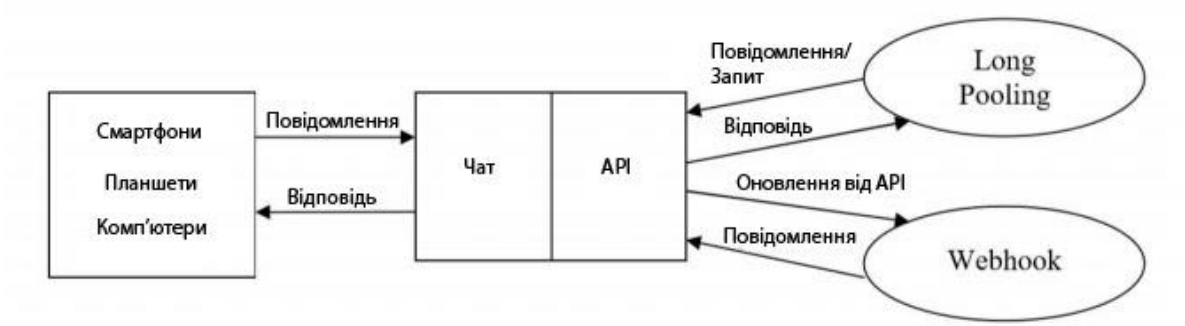

Рис. 2.1. Принцип роботи чат-бота на платформі *Telegram*

setWebhook – метод прив'язує до бота *URL* домена де міститься запущений бот;

sendMessage – метод відправляє текстове повідомлення в клієнт *Telegram*;

sendLocation – метод відправляє повідомлення з координатами в клієнт *Telegram*;

getFile – метод повертає завантажений файл за його іменем.

2.3. *BotFather*

BotFather – це асистент, який допомагає користувачам керувати ботами.

BotFather відповідає за реалізацію оболонки та видає токен (ключ) для роботи з *Telegram API*. У разі необхідності токен може бути змінений на інший.

Для зміни зовнішнього вигляду чат-бота використовуються різноманітні команди, представлені в таблиці 2.1. Для виконання цих команд їх необхідно вводити в чат *BotFather* в якості повідомленнь.

Таблиця 2.1

Команди для зміни чат-бота

Команда	**Опис**
/*setname*	Змінює існуюче ім’я
/*setdescription*	Встановлює текст, що відображається при першому відкритті чат-бота
/*setabouttext*	Встановлює текст в полі «Про чат-бота»
/*setuserpic*	Встановлює обране зображення в профілі бота
/*setcommands*	Дозволяє створити список обраних команд
/*deletebot*	Видаляє обраного чат-бота

Окрім команд для зміни основних параметрів бота існує ряд команд, які дозволяють виводити параметри, що не змінюються. Для зміни параметрів токена використовуються різноманітні команди, представлені у таблиці 2.2.

Таблиця 2.2

Команди для додаткового налаштування чат-бота

Команда	**Опис**
/*token*	Повертає отриманий раніше токен у обраного бота
/*revoke*	Анулює токен доступу до бота
/*setinline*	Вмикає або вимикає можливість викликати бота із інших чатів

Закінчення таблиці 2.2

Команда	**Опис**
/*setinlinegeo*	Вмикає або вимикає можливість передавати боту місцеположення користувача із іншого чату
/*setinlinefeedback*	Дозволяє отримувати інформацію про кількість вибраних користувачем команд
/*setjoingroup*	Визначає, чи може бот бути доданий в групові чати
/*setprivacy*	Вмикає режим конфіденційності. В цьому режимі бот отримує, обробляє і відсилає назад інформацію окремо для кожного користувача в чаті

2.4. Діаграми

2.4.1. Діаграма варіантів використання

Проектуєма система представлена у вигляді множини сутностей або акторів (*actor*), взаємодіючих з системою за допомогою прецедентів. При цьому, актором або діючою особою зветься будь-яка сутність, що взаємодіє з системою ззовні. Іншими словами, кожний варіант використання визначає деякий набір дій, що виконується системою при діалозі з актором.

В даній системі розглядаються такі ролі, як клієнт, лікар та консультант. Клієнт за допомогою бота може виконати запис, відмінити, переглянути розклад та свої записи, лікар та консультант мають можливість переглянути розклад, щоб з’ясувати актуальність наявних вільних місць.

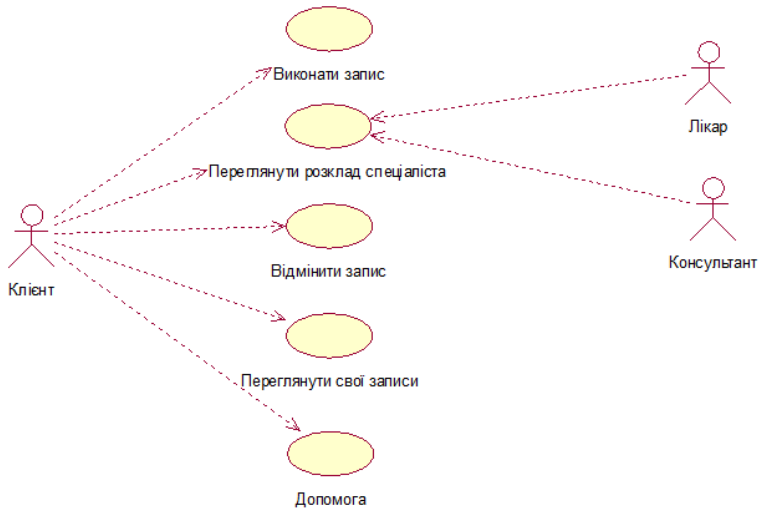

Рис. 2.2. Діаграма варіантів використання чат-бота

Для кращої відповідності системи потребам замовника слід додати можливість для лікарів отримувати інформацію про клієнтів, що збираються їх відвідати. На рис. 2.3 зображена перспективна діаграма використання.

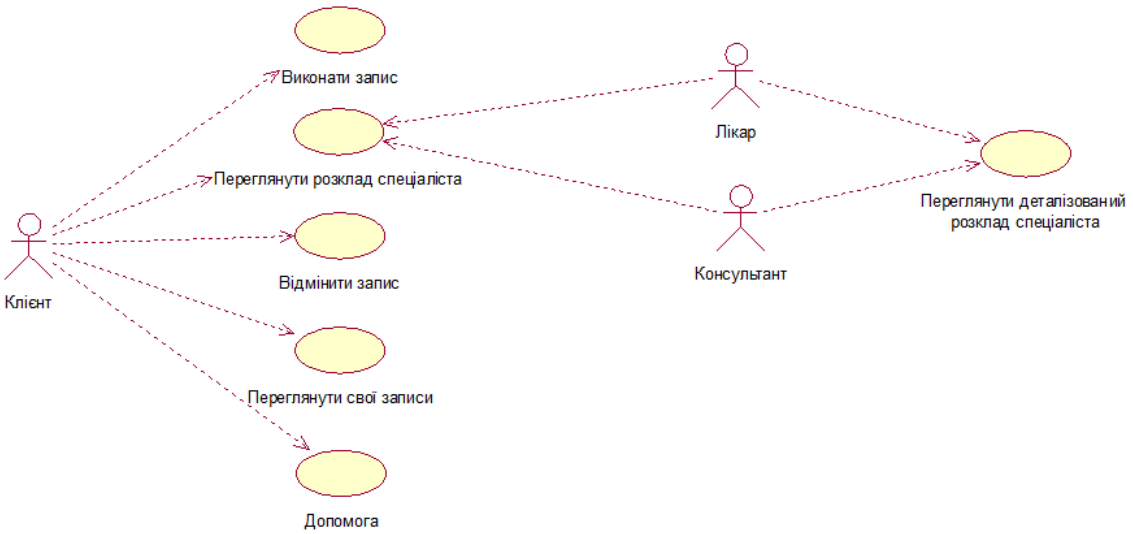

Рис. 2.3. Перспективна діаграма використання чат-бота

2.4.2. Діаграма станів

Дана діаграма надає змогу зрозуміти, яким чином поводять себе об’єкти при впливі на них певних подій ззовні.

Діаграма станів містить в блоках стани, переходи від одного стану до іншого являються діями. Перехід відбувається під дією події, що вказується на діаграмі (рис. 2.4).

В даній діаграмі розглядається робота чат-бота та його реакція на дії користувача, що виконує необхідні йому дії.

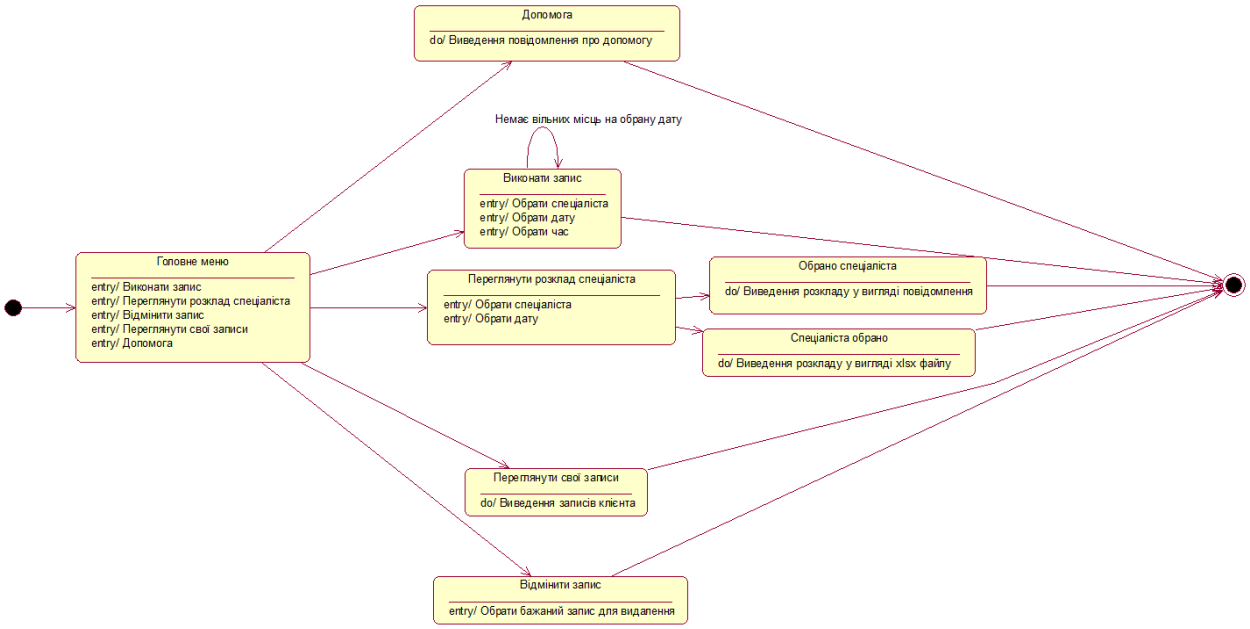

Рис. 2.4. Діаграма станів

2.4.3. Діаграма сутностей

Діаграми «сутність-зв’язок» (*Entity Relation Diagram*) призначені для того, щоб можна було поглянути на систему в цілому. На такій діаграмі розглядаються сутності з усіма атрибутами і взаємозв’язок між цими сутностями. Діаграма визначає дані та відношення між ними.

Існує набір стандартних визначень та правил для того, щоб можна було вірно відобразити дані.

Для того, щоб зрозуміти, як будуються ці діаграми, необхідно зрозуміти, що мається на увазі під «сутністю» та «зв'язком».

Сутність – множина реальних або абстрактних об’єктів, які об’єднані загальними характеристиками та володіють одними й тими ж атрибутами (ознаками). Кожний об’єкт може відноситись тільки до однієї сутності і при цьому він повинен мати унікальне ім’я і щось, що буде відрізняти його від інших об’єктів даної сутності.

Зв’язок об’єднує між собою сутності, будучи відношенням або асоціацією між цими сутностями. Зв’язки бувають трьох видів: «один до одного», «один до багатьох» та «багато до багатьох».

Для того, щоб зрозуміти, який саме зв’язок зображено, необхідно подивитись на початок та кінець стрілки. В *MySQL Workbench* діють наступні позначення: 1 (один) на початку та 1 на кінці відображає зв’язок «один до одного», 1 на початку та стрілка з рискою на кінці відображає зв’язок «один до багатьох», а дві паралельні риски на початку та стрілка з рискою на кінці відображає зв’язок «багато до багатьох».

Діаграма сутностей відображає всі об’єкти (таблиці) бази даних, відповідні їм атрибути (поля) та зв'язки між ними.

В кожній таблиці міститься основна інформація для необхідної системи. Наприклад, в таблиці «Спеціаліст» міститься основна інформація про ім’я спеціаліста та порядковий номер.

Діаграма «сутність-зв’язок» зображена на рис. 2.5.

2.4.4. Контекстна діаграма

На основі даних вище була створена контекстна діаграма, що зображена на рис. 2.6.

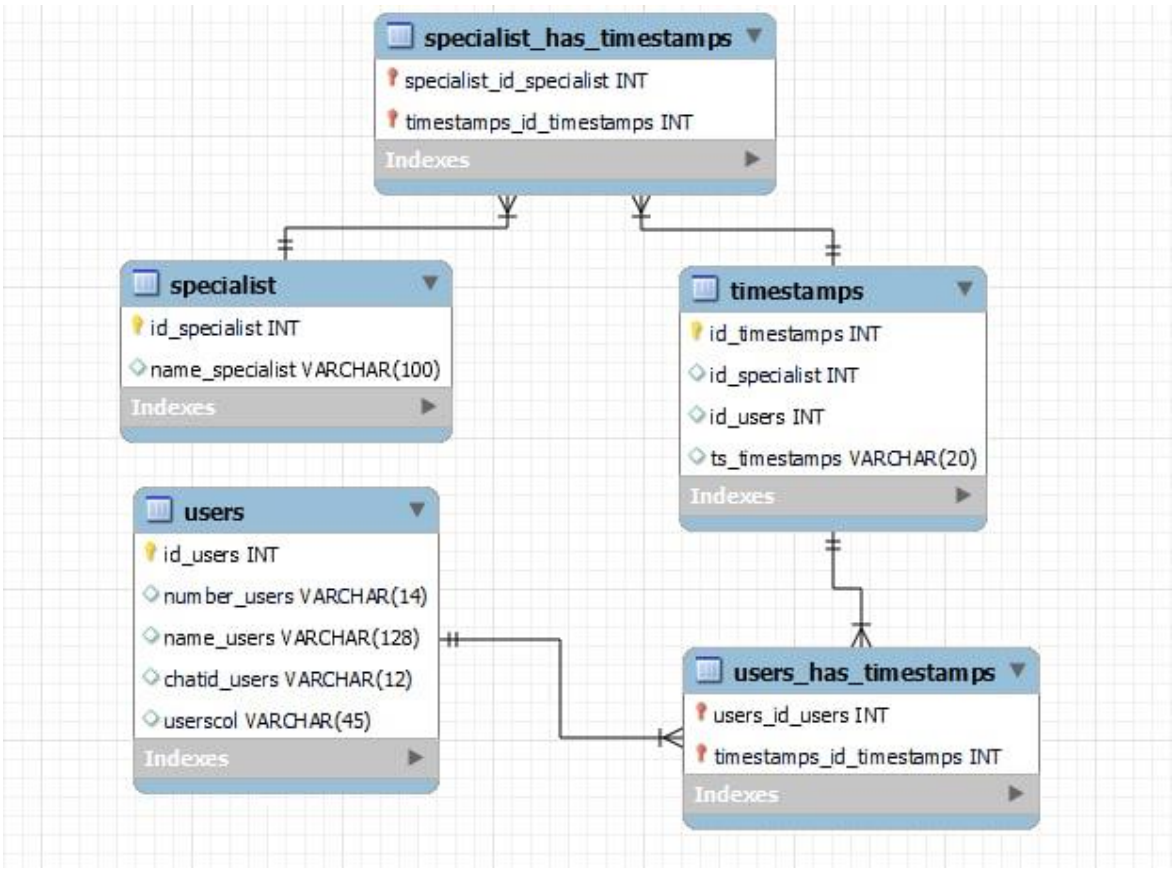

Рис. 2.5. Діаграма «сутність-зв’язок»

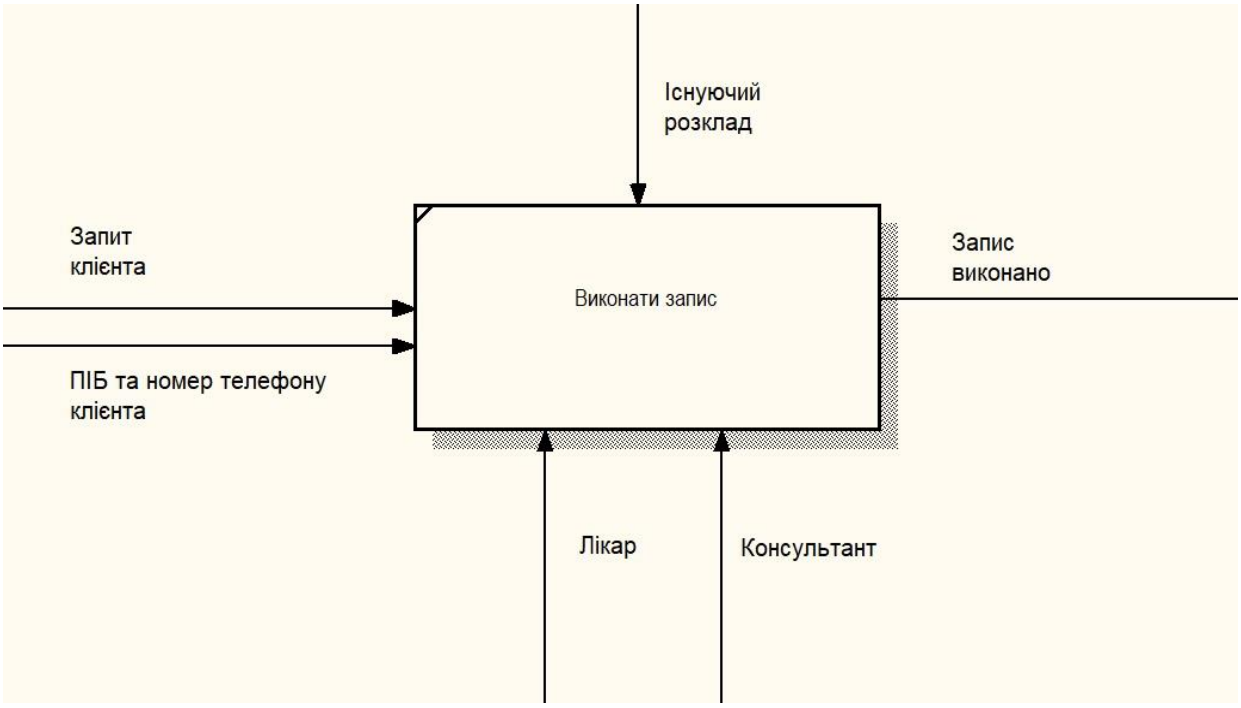

Рис. 2.6. Контекстна діаграма *IDEF0*

Контекстна діаграма відображає основну функцію системи та її взаємодію із зовнішнім середовищем. Основна функція, виконувана системою – запис клієнтів. Вхідними даними цього процесу є запит клієнта та інформація про нього, а сам ПІБ та номер телефону. Кінцевий результат виконуваної функції – виконаний запит клієнта.

2.4.5. Діаграма діяльності

Клієнт виконує запит через бота, запит поступає на сервер, де виконується аналіз та пошук необхідної відповіді. При знаходженні підходящої відповіді, бот виводить відповідь на екран, продовжуючи при необхідності діалог, відповідаючи на інші запити користувача. При помилковому запиті бот видає повідомлення про помилку. Діаграма зображена на рис. 2.7.

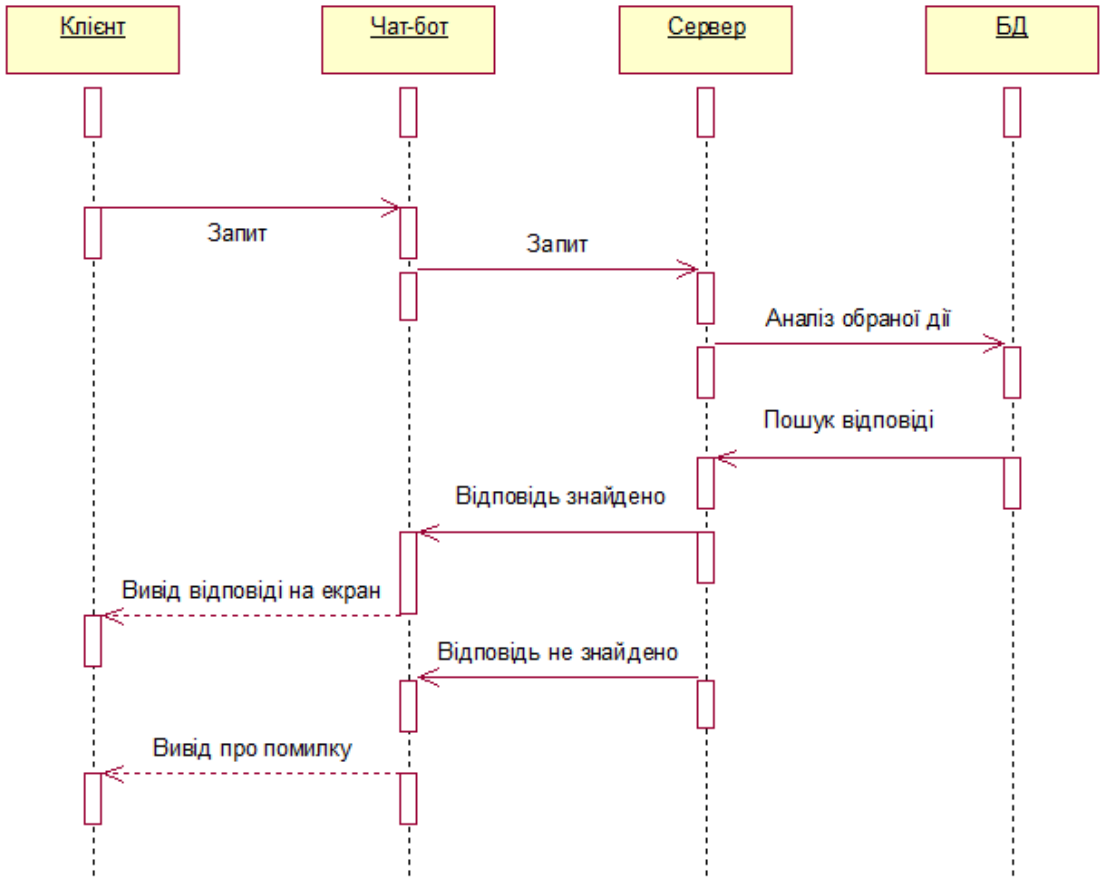

Рис. 2.7. Діаграма діяльності

2.5. Висновки до розділу

У цьому розділі було описано етап проектування чат-бота, а саме розглянуто головний програмний засіб розробки чат-бота – модулі, що необхідні для розробки в *Python*. Однією із головних переваг обраної мови є наявність модулів, що полегшують написання коду, дозволяючи програмісту більше поглибитись в логіку додатку, що розробляється.

Найбільш наочно модель розроблюваного бота можна реалізувати у вигляді uml-діаграм. Дані діаграми мають різні особливості і задачі, що допомагає більш детально розглянути бота. Було розглянуто наступні діаграми: діаграма варіантів використання, контекстна діаграма, діаграма сутностей, діаграма станів та діаграма діяльності.

Також було описано принцип функціонування чат-ботів у *Telegram*, розглянуто його *API* і взаємодію з ним завдяки *BotFather*.

РОЗДІЛ 3
РОЗРОБКА ЧАТ-БОТА

3.1. Реєстрація бота в *Telegram*

Розробка бота умовно ділиться на дві частини:

– створення оболонки;

– програмування функціоналу.

Першим кроком розробки додатку є реєстрація у спеціального чат-бота «*BotFather*». Даний бот допомагає створити оболонку чат-бота. Завдяки цьому немає необхідності створювати чат-бота з нуля.

Реєстрація починається з команди «*/newbot*» (рис. 3.1), після чого пропонується ввести назву бота з обов'язковою умовою: в кінці назви повинно бути вказано «*Bot*» або «*_bot*». Ім'я для бота слід обирати просте, для зручності та простоти пошуку користувачами.

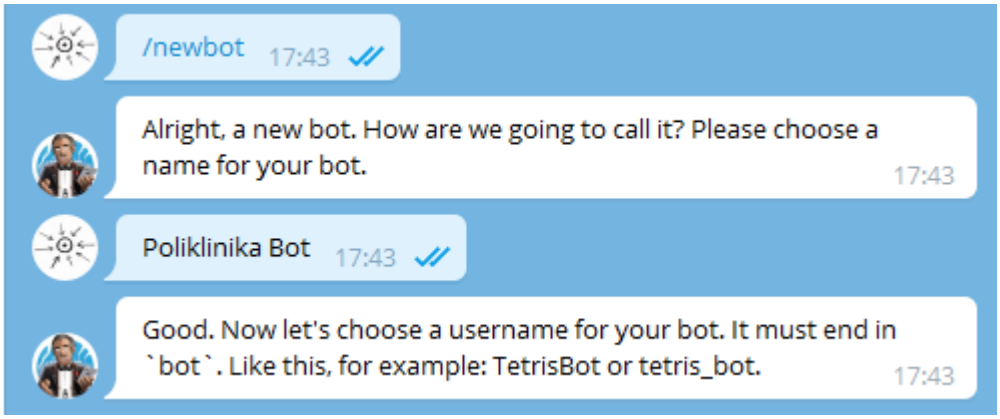

Рис. 3.1. Початок роботи з *BotFather*

При виконанні всіх умов, а саме введенні унікального імені для бота, «*BotFather*» видасть токен (спеціальний набір символів для доступа до *API Telegram Bot* та *URL*-посилання для доступу до чат-бота. Далі заповнюється профіль та налаштовується меню.

За допомогою «*BotFather*» можна додати опис створеного чат-бота, додати зображення профілю, змінити ім'я, присвоїти текст, що буде відображатись при першому відкритті чат-бота.

Після налаштувань *Telegram* та отримання токену можна приступити до розробки програмної частини бота. Для розробки було використано описані вище бібліотеки *Python*.

В першу чергу ці бібліотеки потрібні для можливості підключення чат-бота до *Telegram*. Під час розробки в коді вказується токен створеного чат-бота (рис. 3.2).

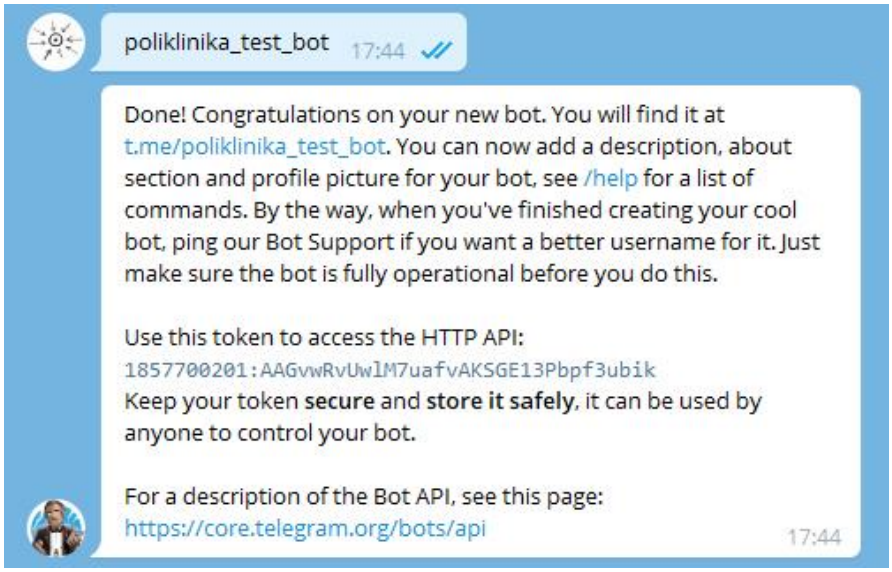

Рис. 3.2. Створення бота в *Telegram*

Впевнитися у створенні бота можна за допомогою команди «/*mybots*», що зображена на рис. 3.3. Дана команда виводить список ботів, підвладних користувачу, що надсилає запит.

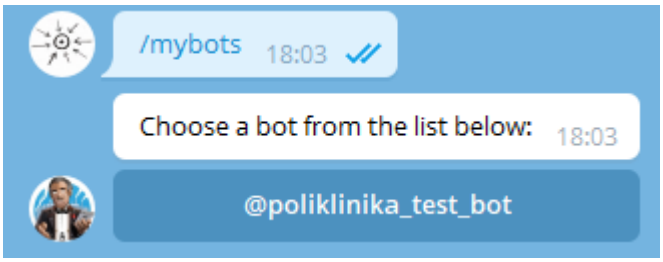

Рис. 3.3. Перелік створених ботів

3.2. Написання коду

Telegram-бот складається з двох файлів. Файл з кодом під назвою «*main.py*» та файл бази даних під назвою «*db_dump.sql*».

В самому початку файла «main.py» наведено модулі, що будуть задіяні в ході виконання проекту. Код підключення модулів поданий на рис. 3.4.

```
from datetime import timedelta, datetime
import os
import re

import telebot
import mysql.connector.connection as mysql
import xlsxwriter
```

Рис. 3.4. Підключення модулів, що використовуються в розробці чат-бота

Далі прописується інформація, що необхідна для створення бота та підключення БД (рис. 3.5). Для того, щоб під’єднати базу даних, нам необхідно вказати користувача, який буде мати доступ до БД, вказати назву та пароль до БД, а також хост, на якому буде знаходиться БД. Крім цього, ми задаємо токен отриманий від *BotFather* для з’єднання нашого коду з оболонкою *Telegram*.

```
# Параметри для підключення до БД
db_user = 'root'
db_password = 'root'
db_host = '127.0.0.1'
db_name = 'bot'

# Token бота в Telegram згенерований в @BotFather.
TOKEN = ''
bot = telebot.TeleBot(TOKEN)
```

Рис. 3.5. Підключення БД та під'єднання чат-бота до оболонки *Telegram*

Після налаштування всіх підєднань ми переходимо до створення клавіатури користувача, тобто кнопок, якими він буде користуватись в ході роботи з ботом (рис. 3.6). Цим кнопкам відповідає перелік заздалегідь описаних команд, які доступні по натисканню кнопок. Завдяки цьому користувачу немає необхідності кожен раз набирати всі команди вручну. Ці команди будуть доступні у вигляді головного меню після початку роботи з ботом.

```
# Клавіатура головного меню
main_menu_markup = telebot.types.ReplyKeyboardMarkup()
main_menu_markup.add('Виконати запис')
main_menu_markup.add('Переглянути розклад спеціаліста')
main_menu_markup.add('Відмінити запис')
main_menu_markup.add('Переглянути свої записи')
main_menu_markup.add('Допомога')
```

Рис. 3.6. Опис кнопок головного меню

Далі нам необхідно задіяти підключення до БД, внести зміни та зберегти їх (рис. 3.7).

```
# Відкриває підключення до БД. Повертає об'єкти MySQLConnection і MySQLCursor.
def open_db():
    db = mysql.MySQLConnection(user=db_user, password=db_password, host=db_host, database=db_name)
    db_curs = db.cursor(buffered=True)
    return db, db_curs


# Зберігає зміни в БД та закриває з'єднання.
def close_db(db, db_curs):
    db.commit()
    db_curs.close()
    db.close()
```

Рис. 3.7. Робота з базою даних

Після того як ми налаштували всі підключення, кнопки головного меню та внесли необхідні зміни в БД, необхідно описати початкову дію для користувача, що буде доступна після першого відкриття бота в *Telegram* або при введенні команди «/*start*» (див. Додаток А). При першому відкритті бота користувачем йому необхідно буде ввести свої дані – ПІБ та номер телефону. Ці дані записуються в БД та надають користувачу можливість подальшої роботи з чат-ботом. У випадку, якщо користувач вже зареєстрований в системі, йому буде одразу виводитись головне меню.

Після успішної реєстрації перед користувачем з'явиться вибір з п'яти кнопок для подальшого діалогу з ботом. Першим варіантом є виконання запису до необхідного спеціаліста (див. Додаток А). У разі вибору клієнтом цієї дії, йому пропонується обрати бажаного спеціаліста, потім обрати день в межах двох майбутніх тижнів для виконання запису. Після вибору необхідного дня запису перед користувачем з'являється набір кнопок з вільними для запису місцями. Це проміжки в 30 хвилин з 10 години ранку до 18:30. У випадку, якщо користувач вводить в поле повідомлення некоректні дані, йому виводиться повідомлення про помилку.

Також, в ході виконання запису програма перевіряє, чи нема у цього користувача вже існуючого запису до спеціаліста саме в цей день (див. Додаток А), оскільки у кожного клієнта є можливість записатись до кожного спеціаліста лише на один прийом на добу. У випадку, якщо користувач вже має запис до певного спеціаліста на певний день, йому буде виведено відповідне повідомлення. Якщо користувач буде вводити сторонні символи, в чат виведеться наступне повідомлення: «Помилка».

Після того, як користувач обрав необхідного спеціаліста, дату та час, його запис зберігається в БД та виводиться повідомлення «Запис виконано успішно». Також, якщо всі місця до обраного спеціаліста в обраний день зайнято, виводиться повідомлення «У спеціаліста немає вільних місць на цей день». Реалізація цього функціоналу зображена у Додатку А.

Крім можливості запису, після його виконання та просто у головному меню у користувача є можливість переглянути свої записи. Записи клієнта буде виведено у

наступному форматі: спеціаліст, дата, час (Додаток А). У випадку, якщо записи відсутні, виводиться повідомлення «У вас немає записів».

Якщо у клієнта змінились плани або він бажає змінити час запису, йому надано можливість відмінити існуючий запис (див. Додаток А). У цьому випадку необхідно в головному меню обрати кнопку «Відмінити запис». Після цього користувачу буде виведено системну клавіатуру з його записами та по натисканню на необхідний запис його буде прибрано з системи. Якщо у користувача ще немає записів, йому буде виведено повідомлення «У вас немає записів». Тут, як і у всіх попередніх функціях також проводиться перевірка на коректність введених даних. Також виконується видалення запису з БД (Додаток А).

Якщо користувач, лікар або консультант бажає переглянути розклад спеціаліста, саме для цього створено функцію «Переглянути розклад спеціаліста» (див. Додаток А). При обранні цієї функції користувачу виводиться системна клавіатура (кнопки) із спеціалістів на вибір. При обранні необхідного спеціаліста бот запитує день, за який необхідно переглянути розклад, також тут з’являється варіант отримання розкладу у вигляді *xlsx* таблиці (див. Додаток А). Коли користувач обере необхідний день, йому буде виведено повідомленням в чат вільні місця обраного спеціаліста. Якщо користувач обрав отримання розкладу *xlsx* файлом, чат-бот надасть файл з розкладом у вигляді таблиці, що буде заповнена помітками «Зайнято» або «Вільно». Виконується перевірка вводу.

У кінці коду прописаний обробник головного меню (рис. 3.8). Він відповідає за реакцію на дії користувача. Тут з’являється остання кнопка головного меню: «Допомога». У разі натискання на неї, в чат буде виведено таке повідомлення: «У разі виникнення труднощів або питань звертайтесь на гарячу лінію клініки»

```python
# Обробник повідомлення головного меню.
@bot.message_handler()
def main_menu_check(message):
    text = message.text
    if text == 'Виконати запис':
        menu_ap(message.chat.id)
    elif text == 'Переглянути розклад спеціаліста':
        check_shed(message.chat.id)
    elif text == 'Переглянути свої записи':
        menu_check_ap(message.chat.id)
    elif text == 'Відмінити запис':
        menu_cancel_ap(message.chat.id)
    elif text == 'Допомога':
        bot.send_message(message.chat.id, "У разі виникнення труднощів або питань звертайтесь на гарячу лінію клініки")
    else:
        pass


bot.polling(none_stop=True)
```

Рис. 3.8. Обробник повідомлення головного меню

3.3. Апробація чат-бота

Для перевірки працездатності бота знайдемо його в *Telegram* та почнемо роботу з ним. Після початку роботи з ботом, він запитає номер телефону, а потім ПІБ (рис. 3.9).

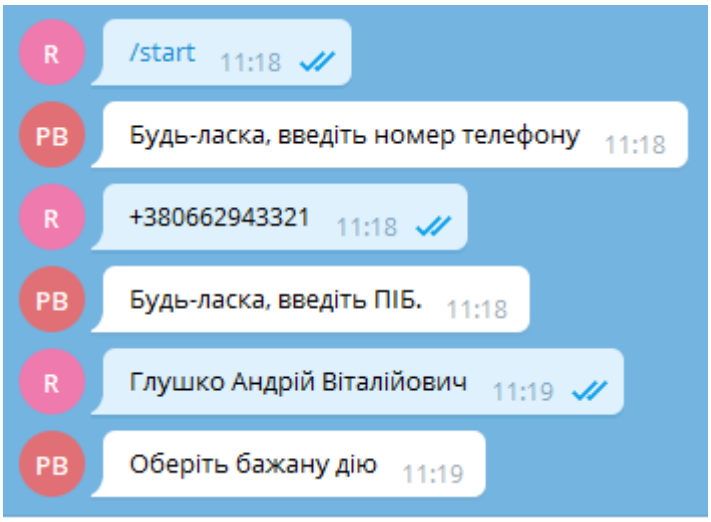

Рис. 3.9. Початок роботи з чат-ботом

Ввівши необхідну інформацію перед користувачем з’являється клавіатура (кнопки) головного меню у вигляді кнопок (рис. 3.10).

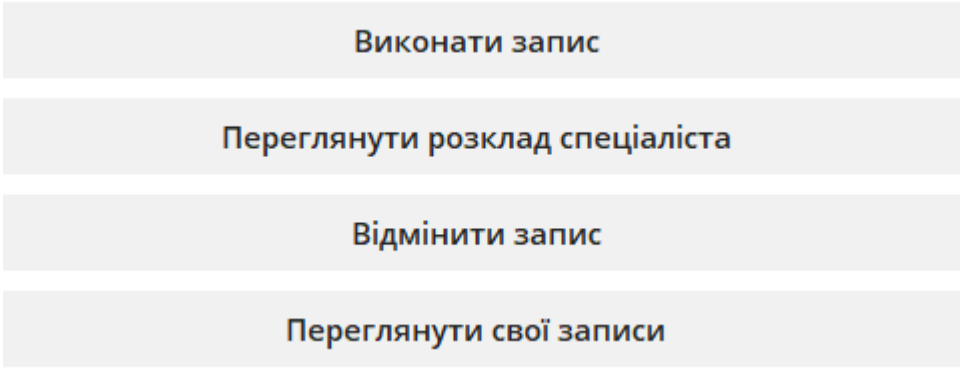

Рис. 3.10. Клавіатура (кнопки) головного меню

Спробуємо виконати запис до Хірурга. Для цього оберемо на клавіатурі кнопку «Виконати запис», після чого з’явиться системна клавіатура (кнопки) зі спеціалістами (рис. 3.11).

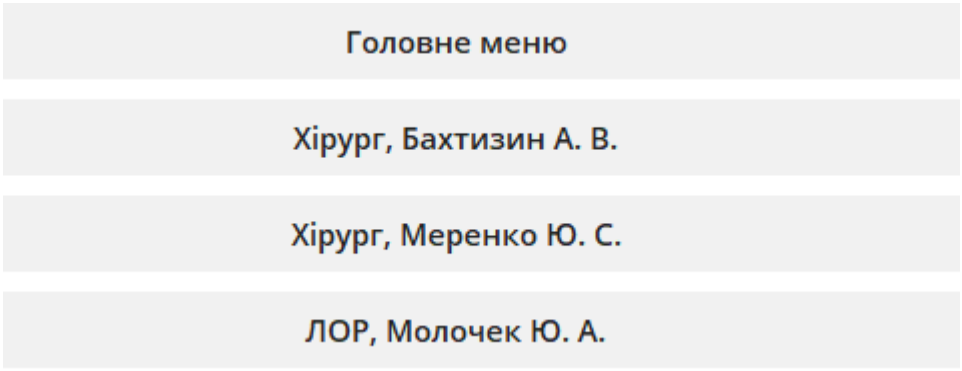

Рис. 3.11. Клавіатура (кнопки) вибору спеціаліста

Оберемо першого хірурга. Після вибору спеціаліста з’являється клавіатура (кнопки) з вибором дня та часу запису (рис. 3.12, рис. 3.13).

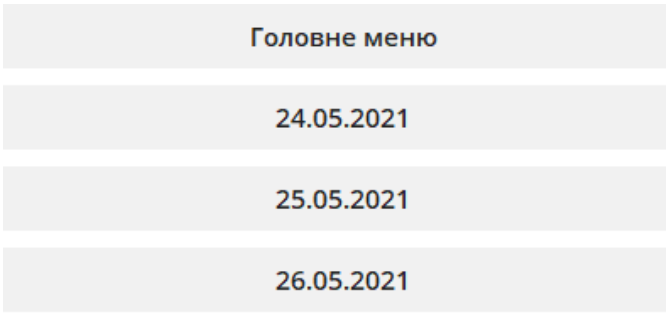

Рис. 3.12. Кнопки вибору дня запису

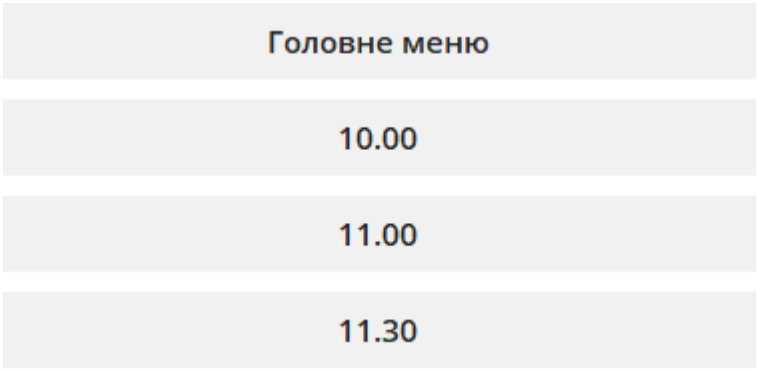

Рис. 3.13. Кнопки вибору часу запису

Після того, як було обрано дату та час, бот повідомив нас про те, що запис виконано успішно та автоматично повернув в головне меню (рис. 3.14).

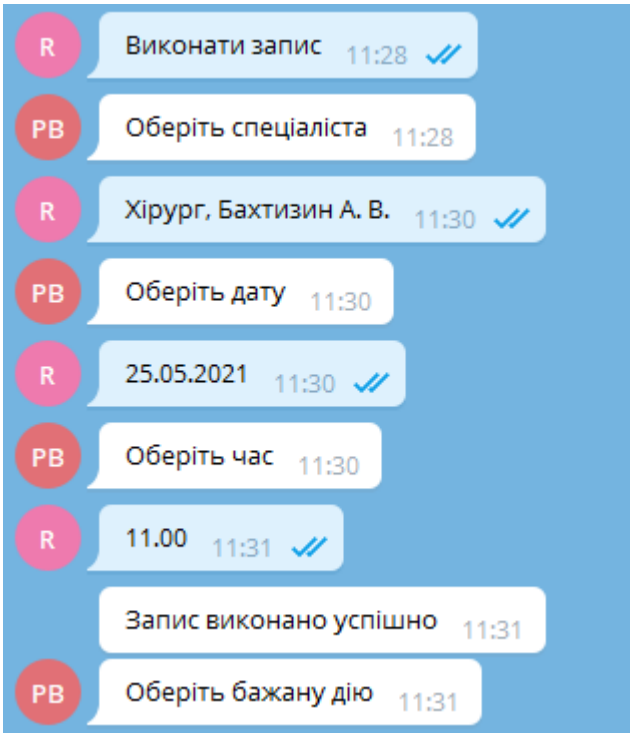

Рис. 3.14. Вибір спеціаліста

Якщо ми бажаємо відмінити запис, натиснемо у головному меню відповідну кнопку. Після цього на екрані з’явиться системна клавіатура (кнопки) з нашими записами (рис. 3.15). По натисканню на необхідний запис він видалиться (рис. 3.16).

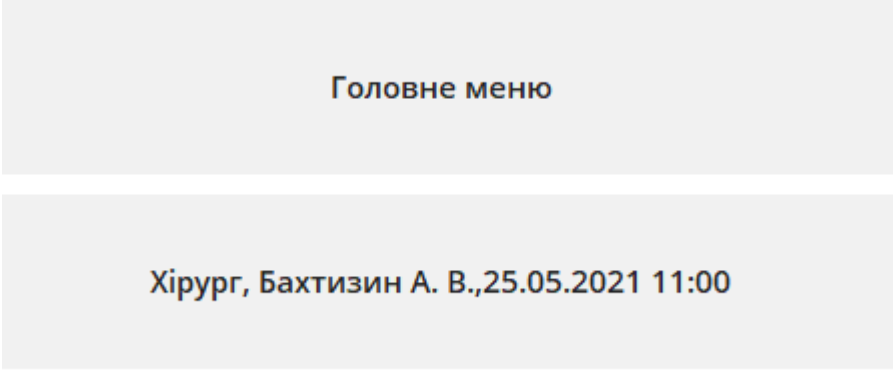

Рис. 3.15. Кнопки, що відповідають записам клієнта

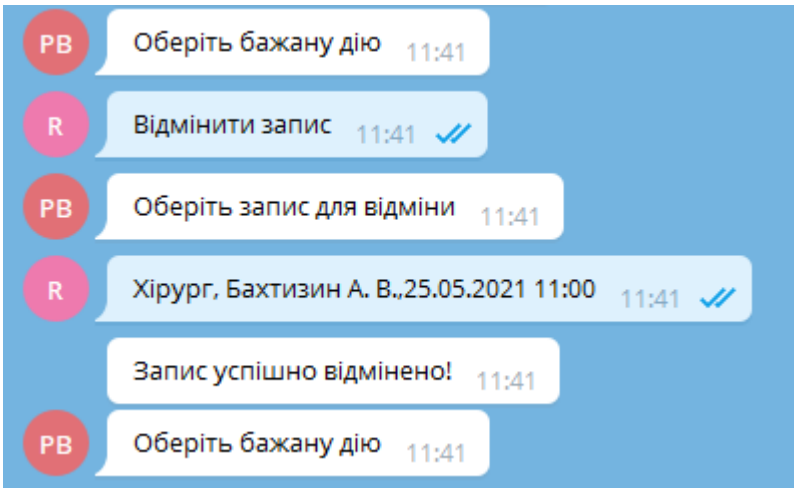

Рис. 3.16. Відміна запису

Крім видалення запису, також можна просто переглянути свої існуючі записи. Для перевірки цієї функції запишемося, наприклад, до терапевта та спробуємо переглянути свої записи. Натискаємо на кнопку «Переглянути свої записи» і отримуємо перелік записів у вигляді повідомлення у чаті (рис. 3.17).

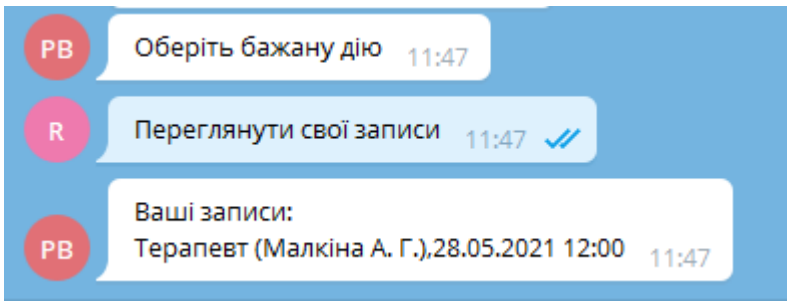

Рис. 3.17. Перегляд власних записів

Також, можна окремо переглянути розклад обраного спеціаліста. Для цього в головному меню натиснемо «Переглянути розклад спеціаліста». Спробуємо переглянути розклад кардіолога на 29.05.2021 (рис. 3.18).

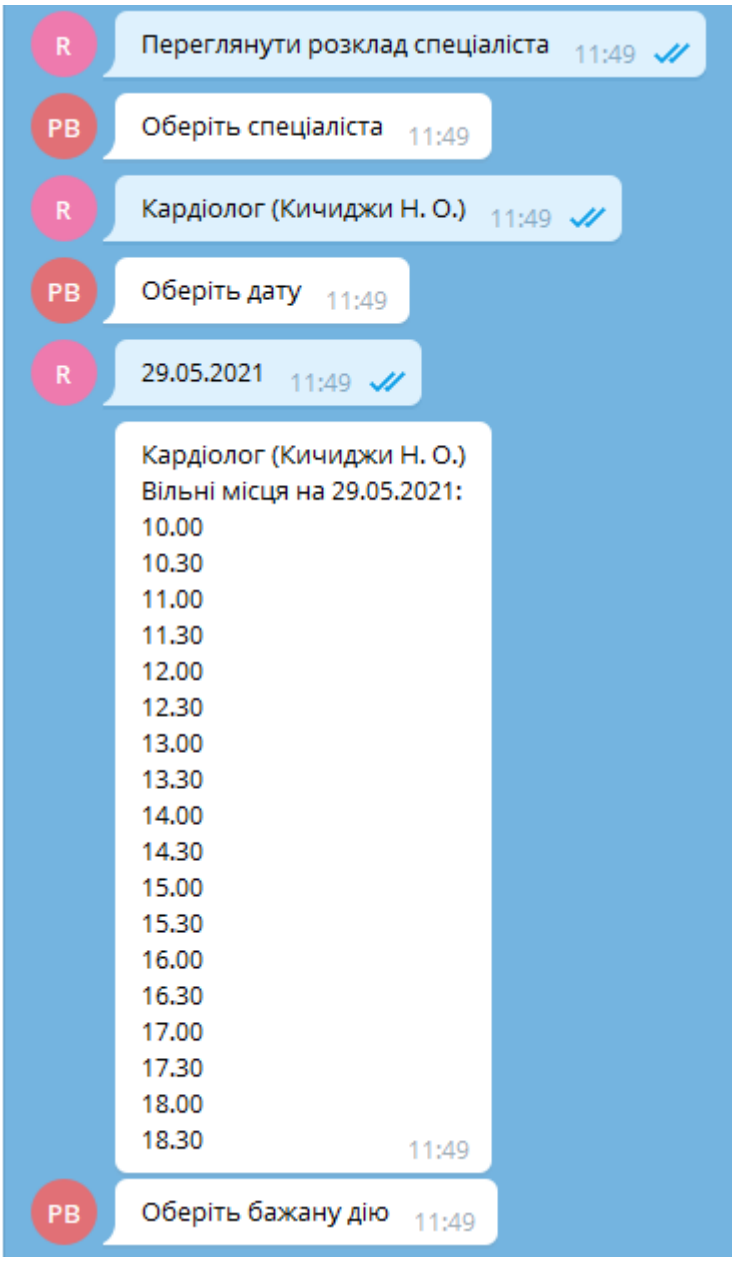

Рис. 3.18. Перегляд розкладу спеціаліста у вигляді повідомлення

Крім повідомлення, розклад спеціаліста можна згенерувати та переглянути у вигляді *xlsx* таблиці. Спробуємо записатись на сьогодні на 18:00 та переглянути розклад у вигляді таблиці. Бот генерує *xlsx*-файл, його можна завантажити та переглянути (рис. 3.19).

	A	B	C
1		24.05.2021	
2	10:00	Вільно	
3	10:30	Вільно	
4	11:00	Вільно	
5	11:30	Вільно	
6	12:00	Вільно	
7	12:30	Вільно	
8	13:00	Вільно	
9	13:30	Вільно	
10	14:00	Вільно	
11	14:30	Вільно	
12	15:00	Вільно	
13	15:30	Вільно	
14	16:00	Вільно	
15	16:30	Вільно	
16	17:00	Вільно	
17	17:30	Вільно	
18	18:00	Зайнято	
19	18:30	Вільно	

Рис. 3.19. *Xlsx* таблиця з розкладом спеціаліста

Як бачимо, в згенерованій таблиці час 18:00 відмічено поміткою «Зайнято».

При натисканні на кнопку «Допомога» користувачу буде виведено повідомлення зображене на рис. 3.20.

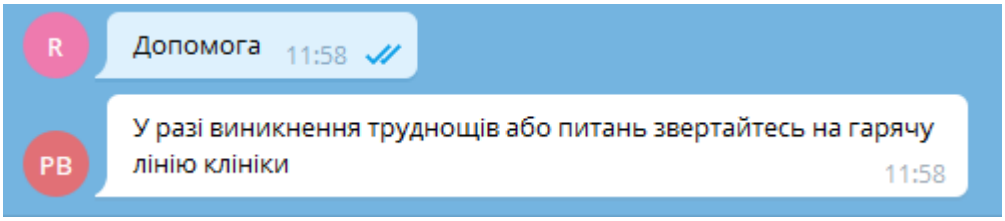

Рис. 3.20. Повідомлення, яке виводить бот після натискання користувачем кнопки «Допомога»

3.4. Висновки до розділу

В даному розділі було описано процес розробки чат-бота. Спершу було виконано реєстрацію бота та створення його оболонки в *Telegram* завдяки *BotFather*.

Після цього, створено код на мові програмування *Python* з детальним описом кожної структурної частини коду. Створений бот має такі функції: виконати запис, відмінити запис, переглянути існуючі записи, переглянути розклад спеціаліста у двох формах (повідомлення та таблиця), а також кнопку допомога.

В дипломній роботі наведена перспективна діаграма використання чат-бота з додатковим функціоналом для спеціалістів. Дана функція не була реалізована на поточному етапі розробки, оскільки вона є проєктованою. В майбутньому, при введенні чат-бота для обробки запитів у реальній поліклініці дана функція може бути перерозглянута для введення.

ВИСНОВКИ

Чат-боти – це сучасні засоби, що дозволяють за допомогою мови програмування автоматизувати дуже багато бізнес-процесів.

Функції чат-бота:

– автоматизація обробки запитів;

– пошук даних;

– відправка відповідей та підтверджень.

Актуальність теми обумовлена потребою поліклінік в автоматизації процесу обробки звернень клієнтів, для підвищення ефективності співробітників і скороченню ресурсних затрат, а також підвищення клієнтського сервісу поліклініки.

В дипломному проекті було представлено огляд та порівняльний аналіз технічних засобів для розробки чат-бота, а саме: розглянуто загальне поняття чат-бота і його переваги, область застосування, було представлено статистику і аналіз мов програмування, обрано мову програмування для розробки чат-бота.

У другому розділі було наведено необхідні модулі для реалізації чат-боту, розроблено діаграми, щоб наочно було видно логіку функціонування системи, розглянуто поняття *BotFather* та *Telegram API*.

В третьому розділі було описано практичну реалізацію системи: реєстрація бота в *Telegram*, далі було продемонстровано процес написання коду, апробування готової системи, показано фрагменти коду та інтерфейс.

В ході виконання дипломного проекту було розроблено чат-бот, що автоматизує процес запису на консультацію до спеціаліста в поліклініку, створеного на мові програмування *Python* та функціонуючого в мессенджері *Telegram*.

Підводячи підсумки дипломного проекту, можна зробити висновок, що поставлена ціль дослідження та розробки була досягнута. Тепер, завдяки можливості автоматичного запису клієнти можуть економити час, а якість роботи консультантів буде підвищена, завдяки вивільненню часу на інші, більш творчі задачі.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Топ мессенджерів в Україні та світі [Електронний ресурс]. – *URL*: *https://marketer.ua/ua/top-messengers-in-ukraine-and-the-world*

2. Чим відрізняються чат-боти в *Telegram, WhatsApp, Viber, Facebook* [Електронний ресурс]. – *URL*: *https://www.rpv-bot.ru/chem-otlichaetsya-chat-bot-v-telegram-whatsapp-vk-viber-facebook*

3. Поняття мови програмування [Електронний ресурс]. – *URL*: *https://ibrain.kz/informatika/ponyatie-yazyk-programmirovaniya*

4. Види мов програмування [Електронний ресурс]. – *URL*: *http://csaa.ru/vidy-jazykov-programmirovanija/*

5. Рейтинг мов програмування 2021 [Електронний ресурс]. – *URL*: *https://dou.ua/lenta/articles/language-rating-jan-2021/*

6. Мова програмування *Python* [Електронний ресурс]. – *URL*: *https://web-creator.ru/articles/python*

7. *JavaScript – Wikipedia* [Електронний ресурс]. – *URL*: *https://uk.wikipedia.org/wiki/JavaScript*

8. Текстовий редактор – *Wikipedia* [Електронний ресурс]. – *URL*: *https://uk.wikipedia.org/wiki/Текстовий_редактор*

9. Модуль *datetime* [Електронний ресурс]. – *URL*: *https://pythonworld.ru/moduli/modul-datetime.html*

10. *Telegram Bot API* [Електронний ресурс]. – *URL*: https://tlgrm.ru/docs/bots/api

11. *BotFather* [Електронний ресурс]. – *URL*: *https://ru.telegram-store.com/catalog/bots/botfather/*

12. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

13. ДСТУ ГОСТ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення».

ДОДАТОК А

Частини коду програми

Реалізація запису до спеціаліста:

```
# Починає процедуру запису на прийом.
# Запитує вибір спеціаліста та реєструє обробник наступного повідомлення.
    def menu_ap(chat_id):
        db, db_curs = open_db()
        db_curs.execute("SELECT * FROM specialists")
        specialists = db_curs.fetchall()
        close_db(db, db_curs)
        markup = telebot.types.ReplyKeyboardMarkup()
        markup.add('Головне меню')

        for row in specialists:
            markup.add(row[1])
        msg = bot.send_message(chat_id, "Оберіть спеціаліста", reply_markup=markup)
        bot.register_next_step_handler(msg, menu_ap_date)

# Генерує та показує користувачу клавіатуру з датами для запису на прийом.
    def menu_ap_date(message):
        if message.text == 'Головне меню':
            show_main_menu(message.chat.id)
            return

        db, db_curs = open_db()
        db_curs.execute(f"SELECT * FROM specialists WHERE name_specialists = '{message.text}'")
```

```
        specialist = db_curs.fetchone()
        close_db(db, db_curs)

# Перевірка на некоректне введення
        if specialist is None:
            bot.send_message(message.chat.id, 'Помилка!')
            show_main_menu(message.chat.id)
            return
        markup = telebot.types.ReplyKeyboardMarkup()
        markup.add('Головне меню')

        for day in range(14):
            markup.add(datetime.strftime(datetime.now() + timedelta(day), '%d.%m.%Y'))
        msg = bot.send_message(message.chat.id, "Оберіть дату",
reply_markup=markup)
        bot.register_next_step_handler(msg, menu_ap_time, specialist[0])
```

Вибір спеціаліста та часу запису, перевірка коректного введення:

```
# Перевіряє, чи нема у користувача запису до обраного спеціаліста на обрану дату.
# У випадку, якщо нема, генерує клавіатуру з доступним часом.
# У випадку, якщо є, видає відповідне повідомлення.
    def menu_ap_time(message, spec_id):
        if message.text == 'Головне меню':
            show_main_menu(message.chat.id)
            return

# Перевірка на коррекстніть вводу
        if not re.match(r'[0-3][0-9].[0-1][0-9].\d\d\d\d', message.text):
            bot.send_message(message.chat.id, 'Помилка!')
```

```
            show_main_menu(message.chat.id)
            return

        db, db_curs = open_db()
        db_curs.execute(f"SELECT ts_timestamps, id_users FROM timestamps WHERE id_specialists = {spec_id}")
        answer = []
        users = []
        for row in db_curs.fetchall():
            answer.append(row[0])
            users.append(row[1])
        db_curs.execute(f"SELECT * FROM users WHERE chatid_users='{message.chat.id}'")
        user_id = db_curs.fetchone()[0]
        close_db(db, db_curs)

        message_slc = message.text.split('.')
        time = datetime(int(message_slc[2]), int(message_slc[1]), int(message_slc[0]), hour=10)
# Перевірка чи є запис
        for idx, user in enumerate(users):
            if user == user_id and answer[idx][:10] == time.strftime('%d.%m.%Y'):
                msg = bot.send_message(message.chat.id, 'У вас вже є запис до цього спеціаліста на цей день')
                bot.register_next_step_handler(msg, menu_ap_time, spec_id)
                return

        markup = telebot.types.ReplyKeyboardMarkup()
        markup.add('Головне меню')
```

Продовження реалізації вибору часу та збереження запису в БД:

```
# Генерація клавіатури.
        while time.hour != 19:
            timestamp = message.text + '.' + time.strftime('%H.%M')
            if timestamp in answer or time <= datetime.now():  # Якщо час зайнято або якщо час пройшов.
                time += timedelta(minutes=30)
                continue
            markup.add(time.strftime('%H.%M'))
            time += timedelta(minutes=30)
            spec_has_time = True
        if spec_has_time:
            msg = bot.send_message(message.chat.id, "Оберіть час", reply_markup=markup)
            bot.register_next_step_handler(msg, menu_ap_proceed, spec_id, message.text)
        else:
            msg = bot.send_message(message.chat.id, "У спеціаліста немає вільних місць на цей день")
            bot.register_next_step_handler(msg, menu_ap_time, spec_id)
# Додає запис до спеціаліста в БД.
    def menu_ap_proceed(message, spec_id, date):
        if message.text == 'Головне меню':
            show_main_menu(message.chat.id)
            return

# Перевірка на коректність вводу.
        if not re.match(r'[0-1][0-8].[03]0', message.text):
            bot.send_message(message.chat.id, 'Помилка!')
            show_main_menu(message.chat.id)
```

```
            return

        timestamp = date + '.' + message.text

        db, db_curs = open_db()
        db_curs.execute(f"SELECT * FROM users WHERE chatid_users = '{message.chat.id}'")
        answer = db_curs.fetchone()[0]
        db_curs.execute(f"INSERT INTO timestamps (id_timestamps, id_specialists, id_users, ts_timestamps) "
                        f"VALUES(null, {spec_id}, {answer}, '{timestamp}')")
        close_db(db, db_curs)

        bot.send_message(message.chat.id, 'Запис виконано успішно')
        show_main_menu(message.chat.id)
```

Функція відміни запису:

```
# Початок процедури відміни запису.
# Генерує клавіатуру із записів користувача та показує її.
# Реєструє обробник наступного повідомлення.
    def menu_cancel_ap(chat_id):
        db, db_curs = open_db()
        db_curs.execute(f"SELECT * FROM users WHERE chatid_users = '{chat_id}'")
        user_id = db_curs.fetchone()[0]
        db_curs.execute(f"SELECT timestamps.ts_timestamps, specialists.name_specialists FROM timestamps "
                        f"INNER JOIN specialists ON timestamps.id_specialists = specialists.id_specialists "
                        f"WHERE id_users = {user_id}")
        timestamps = db_curs.fetchall()
```

```
        close_db(db, db_curs)

        markup = telebot.types.ReplyKeyboardMarkup()
        markup.add('Головне меню')

        user_has_aps = False

        for timestamp in timestamps:
            timestamp_spl = timestamp[0].split('.')
            if datetime(int(timestamp_spl[2]), int(timestamp_spl[1]),
int(timestamp_spl[0]), int(timestamp_spl[3]),
                    int(timestamp_spl[4])) > datetime.now():
                markup.add(timestamp[1] + ',' + timestamp_spl[0] + '.' + timestamp_spl[1]
+ '.' + timestamp_spl[2]
                        + ' ' + timestamp_spl[3] + ':' + timestamp_spl[4])
                user_has_aps = True

        if user_has_aps:
            msg = bot.send_message(chat_id, "Оберіть запис для відміни",
reply_markup=markup)
            bot.register_next_step_handler(msg, cancel_ap_proceed)
        else:
            bot.send_message(chat_id, "У вас немає записів")
```

Функція перегляду розкладу спеціаліста:

```
# Генерує клавіатуру з датами і кнопкою для генерації таблиці з розкладом.
    def check_shed_date(message):
        if message.text == 'Головне меню':
            show_main_menu(message.chat.id)
```

```
            return

        db, db_curs = open_db()
        db_curs.execute(f"SELECT * FROM specialists WHERE name_specialists = '{message.text}'")
        specialist = db_curs.fetchone()
        close_db(db, db_curs)

        if specialist is not None:
            markup = telebot.types.ReplyKeyboardMarkup()
            markup.add('Головне меню')
            markup.add('Отримати xlsx файл з розкладом')
            for day in range(14):
                markup.add(datetime.strftime(datetime.now() + timedelta(day), '%d.%m.%Y'))
            msg = bot.send_message(message.chat.id, "Оберіть дату", reply_markup=markup)
            bot.register_next_step_handler(msg, check_shed_proceed, specialist[0], message.text)
        else:
            bot.send_message(message.chat.id, 'Помилка!')
            show_main_menu(message.chat.id)
```

Реалізація вибору виведення розкладу повідомелнням або у вигляді xlsx таблиці:

```
# В залежності від вибору користувача на попредньому кроці:
# Генерує та відправляє таблицю з розкладом. Або
# Відправляє розклад спеціаліста за день повідомленням.
def check_shed_proceed(message, spec_id, spec_name):
    if message.text == 'Головне меню':
```

```
        show_main_menu(message.chat.id)
        return

    db, db_curs = open_db()
    db_curs.execute(f"SELECT ts_timestamps FROM timestamps WHERE id_specialists = {spec_id}")
    answer = []
    for row in db_curs.fetchall():
        answer.append(row[0])
    close_db(db, db_curs)

    if message.text == 'Отримати xlsx файл з розкладом':
        xlsx = xlsxwriter.Workbook(f'{message.chat.id}.xlsx')
        sheet = xlsx.add_worksheet()
        now = datetime.now()

        for day in range(1):
            date = datetime(now.year, now.month, now.day + day, hour=10)
            sheet.write(0, day+1, date.strftime('%d.%m.%Y'))
            for time in range(18):
                sheet.write(time+1, 0, date.strftime('%H:%M'))
                timestamp = date.strftime('%d.%m.%Y.%H.%M')
                if timestamp in answer:
                    sheet.write(time+1, day+1, 'Зайнято')
                else:
                    sheet.write(time+1, day+1, 'Вільно')
                date += timedelta(minutes=30)
        xlsx.close()

        xlsx = open(f'{message.chat.id}.xlsx', 'rb')
```

```
        bot.send_document(message.chat.id, xlsx)
        os.remove(f'{message.chat.id}.xlsx')
    else:
        # Перевірка на коректність вводу.
        if not re.match(r'[0-3][0-9].[0-1][0-9].\d\d\d\d', message.text):
            bot.send_message(message.chat.id, 'Помилка!')
            show_main_menu(message.chat.id)
            return

        message_str = f"{spec_name}\nВільні місця на {message.text}:\n"
        message_slc = message.text.split('.')
        time = datetime(int(message_slc[2]), int(message_slc[1]), int(message_slc[0]),
hour=10)
        spec_has_time = False

        while time.hour != 19:
            timestamp = message.text + '.' + time.strftime('%H.%M')
            if timestamp in answer:
                time += timedelta(minutes=30)
                continue

            message_str += time.strftime('%H.%M') + '\n'
            time += timedelta(minutes=30)
            spec_has_time = True
        if spec_has_time:
            bot.send_message(message.chat.id, message_str)
        else:
            bot.send_message(message.chat.id, "У спеціаліста немає вільних місць в цей
день")
    show_main_menu(message.chat.id)
```

Реалізація команди «/start» та подальша реєстрація в системі:

```
# Обробляє команду /start, починає процедуру реєстрації.
# У випадку, якщо користувач зареєстрований, показує головне меню.
@bot.message_handler(commands=['start'])
def start_message(message):
    db, db_curs = open_db()
    db_curs.execute(f"SELECT * FROM users WHERE chatid_users = '{message.chat.id}'")
    answer = db_curs.fetchone()
    close_db(db, db_curs)

    if answer is None:
        # Запит номеру телефона і реєстрація обробника наступного повідомлення.
        msg = bot.send_message(message.chat.id, "Будь-ласка, введіть номер телефону")
        bot.register_next_step_handler(msg, number_step)
    else:
        show_main_menu(message.chat.id)

# Запитує ПІБ і реєструє обробник наступного повідомлення.
def number_step(message):
    msg = bot.send_message(message.chat.id, "Будь-ласка, введіть ПІБ.")
    bot.register_next_step_handler(msg, name_step, message.text)

# Записує дані про користувача в БД і показує головне меню.
def name_step(message, user_number):
    db, db_curs = open_db()
    db_curs.execute(f"INSERT INTO users (id_users, number_users, name_users, chatid_users) "
                    f"VALUES(null, '{user_number}', '{message.text}', '{message.chat.id}')")
```

```
    close_db(db, db_curs)
    show_main_menu(message.chat.id)

# Показує головне меню.
def show_main_menu(chat_id):
    bot.send_message(chat_id, "Оберіть бажану дію",
reply_markup=main_menu_markup)
```

Функція відображення власних записів:

```
# Показує користувачу його записи.
# У випадку якщо записів немає, виводить відповідне повідомлення.
def menu_check_ap(chat_id):
    db, db_curs = open_db()
    db_curs.execute(f"SELECT * FROM users WHERE chatid_users = '{chat_id}'")
    user_id = db_curs.fetchone()[0]
    db_curs.execute(f"SELECT timestamps.ts_timestamps, specialists.name_specialists
FROM timestamps "
                    f"INNER JOIN specialists ON timestamps.id_specialists =
specialists.id_specialists "
                    f"WHERE id_users = {user_id}")
    timestamps = db_curs.fetchall()
    close_db(db, db_curs)

    if timestamps is not None:
        message_str = "Ваші записи:\n"
        for timestamp in timestamps:
            timestamp_spl = timestamp[0].split('.')
            if datetime(int(timestamp_spl[2]), int(timestamp_spl[1]), int(timestamp_spl[0]),
int(timestamp_spl[3]),
```

```
                int(timestamp_spl[4])) > datetime.now():
            message_str += timestamp[1] + ',' + timestamp_spl[0] + '.' + timestamp_spl[1]
+ '.' \
                        + timestamp_spl[2] + ' ' + timestamp_spl[3] + ':' + timestamp_spl[4]
+ '\n'
        bot.send_message(chat_id, message_str)
    else:
        bot.send_message(chat_id, "У вас немає записів")
```

Видалення запису з БД та перевірка вводу:

```
# Видаляє з БД запи до спеціаліста.
def cancel_ap_proceed(message):
    if message.text == 'Головне меню':
        show_main_menu(message.chat.id)
        return

# Обробка виключення у випадку некоректності вводу даних.
    try:
        spec_name = message.text.split(',')[0]
        timestamp = message.text.split(',')[1].replace(' ', '.').replace(':', '.')

        db, db_curs = open_db()
        db_curs.execute(f"SELECT * FROM specialists WHERE name_specialists =
'{spec_name}'")
        spec_id = db_curs.fetchone()[0]
        db_curs.execute(f"DELETE FROM timestamps WHERE id_specialists = {spec_id}
AND ts_timestamps = '{timestamp}'")
        close_db(db, db_curs)
```

```
        bot.send_message(message.chat.id, "Запис успішно відмінено!")
        show_main_menu(message.chat.id)
    except:
        bot.send_message(message.chat.id, 'Помилка')
        show_main_menu(message.chat.id)
        return
```