**Лабораторна робота: Розробка Telegram бота для збору відгуків про перебування в готелі**

**Мета роботи**

Метою виконання лабораторної роботи є розробка Telegram бота для збору відгуків користувачів про їхнє перебування в готелі та враження від сервісу. Основна увага зосереджена на порівнянні звичайної та асинхронної версій бота з точки зору паралельного мультипоточного програмування.

**Завдання**

* Розробити Telegram бота для збору відгуків від клієнтів сайту для бронювання кімнат у готелі. Інтегрувати в Backend-частину проєкту.
* Порівняти звичайну та асинхронну версії бота.
* Дослідити переваги та недоліки паралельного мультипоточного програмування у контексті розробки Telegram бота.

**Опис виконаної роботи та загальні факти**

**Telegram бот**

Telegram бот, що використовується для цієї лабораторної роботи, написаний за допомогою мови програмування Python та бібліотеки pyTelegramBotApi. Основна функція бота – збір відгуків від користувачів готелю про їхнє перебування та враження від сервісу. Бот приймає особливий 6-значний код, що складається з великих латинських літер, який користувачі можуть побачити на своїй веб-сторінці після авторизації. Цей код використовується для підтвердження особи, щоб ускладнити спроби недобросовісного накручування фейкових відгуків. Після підтвердження бот зберігає текстові повідомлення від користувача, обробляє їх та зберігає відгуки до бази даних сайту для подальшого аналізу.

**Backend-частина сайту**

Оскільки Telegram бот лише доповнює основних функціонал більшого Spring Boot 3 Java + React.js веб-проєкту, виявилося очевидно необхідним налагодити взаємодію бота з бекендом. Таким чином, було реалізовано комунікацію через POST-запити, створено відповідну інфраструктуру на сервері: додано нові контролери, сервіси, репозиторії, таблиці, тощо; також імплементувано механізм створення та верифікації спеціального 6-значого коду, що згадувався раніше

**Звичайна версія бота** *(bot\_default.py)*

У звичайній версії бота, всі запити обробляються послідовно. Кожне повідомлення від користувача обробляється в окремому потоці, але ці потоки працюють синхронно, що може призвести до затримок у відповідях при великій кількості запитів.

**Асинхронна версія бота** *(bot\_async.py)*

Асинхронна версія бота використовує асинхронні функції та корутини для обробки запитів. Це дозволяє обробляти кілька запитів одночасно без блокування основного потоку виконання програми. Асинхронність досягається за допомогою бібліотеки asyncio, яка дозволяє створювати неблокуючі виклики та забезпечує більш ефективне управління ресурсами, використовуючи ключові слова async і await.

**Паралельне мультипоточне програмування**

**Переваги**

* **Підвищена продуктивність**: Асинхронні боти можуть обробляти більше запитів одночасно завдяки неблокуючим викликам, що значно підвищує продуктивність системи.
* **Зменшення затримок**: Використання асинхронного програмування дозволяє зменшити затримки у відповідях на запити користувачів, оскільки час очікування на I/O операції (наприклад, доступ до бази даних або мережевих ресурсів) використовується для обробки інших запитів.
* **Краща масштабованість**: Асинхронні програми легше масштабувати для обробки великої кількості одночасних з'єднань, що є критичним для сервісів з високим навантаженням.

**Недоліки**

* **Складність розробки**: Асинхронні програми складніші у розробці та налагодженні порівняно з синхронними. Необхідно враховувати безліч аспектів паралельного виконання, таких як синхронізація даних та обробка виключень.
* **Відладка та тестування**: Відладка та тестування асинхронних програм може бути складнішим завданням через труднощі у відстеженні потоку виконання та можливість виникнення race condition.
* **Обмежена підтримка бібліотек**: Деякі бібліотеки можуть не підтримувати асинхронність, що обмежує можливості використання асинхронного підходу у певних випадках.

**Порівняння звичайної та асинхронної версій бота**

**Звичайна версія**

* **Синхронна обробка**: Усі запити обробляються послідовно, що може призводити до затримок при великій кількості запитів.
* **Менша продуктивність**: Звичайна версія бота менш ефективна при обробці великої кількості одночасних запитів.

**Асинхронна версія**

* **Асинхронна обробка**: Можливість обробки кількох запитів одночасно без блокування основного потоку виконання.
* **Вища продуктивність**: Асинхронна версія бота більш ефективна та здатна обробляти більшу кількість одночасних запитів, що знижує затримки у відповідях.

**Висновок**

Розробка асинхронного Telegram бота для збору відгуків користувачів дозволяє значно підвищити продуктивність та ефективність обробки запитів. Хоча реалізація асинхронного бота є дещо громоздкішою та подекуди складнішою, переваги у вигляді зменшення затримок та кращої масштабованості роблять цей підхід більш привабливим для Telegram ботів із високим навантаженням. У випадку з сайтом для бронювання кімнат у готелі, асинхронність коду не сильно покращила б ефективність роботи бота, оскільки велика кількість користувачів не передбачається. Тим не менш, можна побачити, що написання коду одразу з використанням асинхронних викликів може зекономити час у майбутньому, якщо проєкт масштабуватиметься.

P.s. Оскільки проєкт не завершений і перебуває в процесі доопрацювання, будуть вноситися відповідні зміни до коду та звітів.