Національний технічний університет України
«Київський Політехнічний Інститут»
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

Розрахунково-графічна робота
З предмету «Інтеграційні програмні системи»

Виконали:

Студенти
групи ІО-42
Марич В.
Висоцький Н.
Коваленко В.

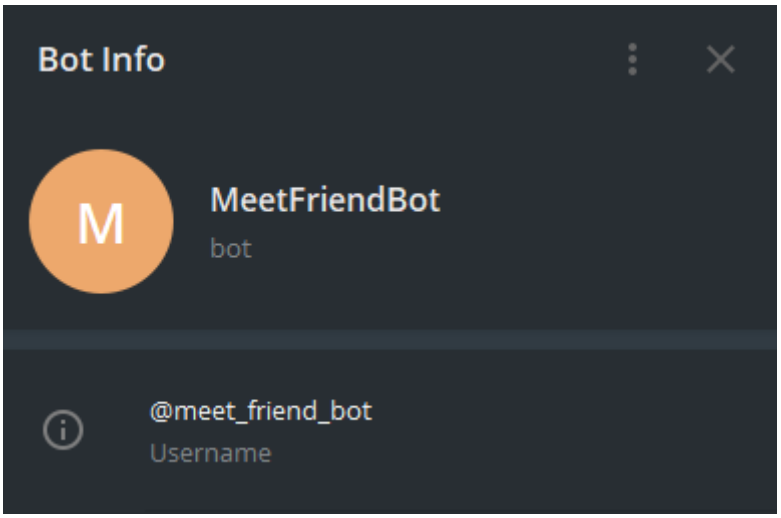
Київ-2017

Короткий опис проекту

У розрахунково-графічній роботі було реалізовано чат-бота для месенджера Telegram.

Знайти бота можна за посиланням:

https://t.me/meet_friend_bot



Призначення бота

Чат-бот дозволяє користувачеві знаходити нові контакти та знайомства у месенджері Telegram.

Користувач реєструється у системі пошуку бота, вказує дані та своє місцезнаходження. Бот додає користувача до бази даних.
При регістрації користувач також вказує термін його доступності для пошуку.

На основі пошукових параметрів користувач отримує вибірку найближчих до нього та найбільш відповідних користувачів. При цьому він сам може потрапити у вибірку для інших користувачів.

Подальші рішення для встановлення контакту користистувач приймає самостійно.

Стек технологій та реалізація

Чат-бот представляє собою систему, що складається з двох компонентів: скрипта, що обробляє дані, що приходять з Телеграм від співрозмовників бота, реалізує логіку його поведінки та відправляє запити на сервер щоб отримати або передати дані та серверу.

Бот написаний на мові Python3 з використанням бібліотеки pyTelegramBotApi, яка являє собою обгортку над запитами до Telegram API. Пакет bot складається з трьох скриптів:

1) Utils.py – містить всі константи
2) run.py – запускає бот, та встановлює хендлери на команди та повідомлення
3) handler – містить всі хенделери, серед яких:
    1. /start – надсилає повідомлення з інформацією про бота та пропонує можливі розвитки подій – або користувач реєструє(/login) себе, щоб його змогли побачити інші користувачі, або ж користувач шукає(/find) людей по вказаним параметрам.
    2. /login – надсилає повідомлення про заповнення необхідних полів та проставляє “state” в стан – початку реєстрації
    3. /find – надсилає повідомлення про заповнення необхідних полів та проставляє “state” в стан – початку пошуку
    4. Хенделер на прикріплення місцерозташування. В бібліотеці це вважається не як звичайне повідомлення, адже має іншу структуру.
    5. Хендлер на текстові повідомлення. В цій функції оброблюються текстові повідомлееня і згідно алгоритму змінюють стани, або ж вказують на помилки при введенні інформації.

Взагалі, алгоритм для бота – це кінцевий автомат зі станами. Тобто в користувача є лише завчасно встановлені шляхи, якими він повинен рухатися, немаючи змоги пропускати якісь стани.

Для реєстрації користувач повинен ввести наступні дані:

- Вік
- Стать
- Місцерозташування
- Час, який користувач буде активний

Для пошуку необхідно ввести всі ті ж дані, окрім часу.

Сервер реалізовано на мові Python3 з використанням фреймворку aiohttp (https://aiohttp.readthedocs.io/en/stable/).

Даний фреймворк є дуже перспективним, оскільки він побудований на основі бібліотеки asyncio, що реалізує асинхронність у мові Python. Його перевага у тому, що він дозволяє досягати кращої швидкодії та продуктивності серверу.

У якості бази даних було обрано реляційну PostgreSQL, оскільки вона є традиційною для використання у веб-додатках на python.

У якості ORM було використано SQLAlchemy. Це набір інструментів, що являє собою прошарок між базою даних та python-застосунком. Вона інкапсулює запити в базу даних, сворення та опис моделей, містить інструменти для використання атомік транзакцій, що в свою чергу використовують такі можливості python, як контекстні менеджери тощо.

Опис вихідного коду серверу:

- run.py
  Файл, що слугує вхідною точкою і запускає сервер.
  Функція init отримує змінні, що містять конфіграцію серверу, сворює об’єкт app та додає дві функції – callback`и, що будуть викликані подіями on_startup та on_cleanup. Відповідно, перша відкриє підключення до бази даних, друга закриє його.
  Функція main отримує об’єкт app та викликає метод web.run_app, після чого сервер готовий для роботи.

- routes.py
  Додає до об’єкту app роути (віповідності запитів функціям – хендлерам, що будуть викликані при певному запиті на сервер).

- settings.py
  Завантажує змінні оточення з файлу .env і повертає об’єкт з конфігурацією серверу.

- models.py
  Описує таблиці бази даних та константи, що пов’язані з моделями.

- db.py
  Містить функції для ініціалізіції бази даних, створення об’єкту engine, через який в подальшому буде відбуватися взаємодія з нею.
  Також містить функції-обгортки над запитами у базу даних:
  - get_all_users (повертає всі записи з таблиці users)
  - get_users_with_params (повертає записи з таблиці users за параметрами)
  - insert_user (додає новий запис до таблиці users)
  - get_state_by_chat_id (отримує запис з таблиці state за chat_id)
  - insert_init_state_for_chat_id (додає пустий запис state)
  - insert_state (додає state)
  - update_state_by_chat_id (оновлює state)

- views.py

Містить хендлери запитів на сервер.
Сервер має такі роути:

- POST /users/
  Handler: add_user.
  Приймає об’єкт user у форматі json, додає його в бд.

- GET /users/find?age={age}&sex={sex}&location={}
  Handler: find_users.
  Функція пошуку користувачів із заданими параметрами (стать, вік, координати).
  За допомогою бібліотеки geopy з бази обирається 3 найближчі користувачі, що задовольняють іншим параметрам.
  Повертає масив імен користувачів у форматі json.

- GET /states/{chat_id}/
  Handler: get_state.
  Повертає об’єкт state. Він потрібен для процесу регістрації користувача. Оскільки телеграм бот не надає можливості заповнити для регістрації традиційну форму, чат-бот реалізує алгоритм покрокової регістрації. Таблиця states зберігає в собі поточний крок регістрації, на якому знаходится користувач.

- PUT /states/
  Handler: update_state.
  Оновлює запис state коли користувач переходить на наступний крок. Коли користувач закінчить процес регістрації, буде створений запис user, що міститиме його дані та інтервал часу, протягом якого цей користувач буде доступний для пошуку.

- tests.py
  Містить unit-тести функцій-хендлерів, описаних вище.

Опис того, яка система збірки використовується у проекті

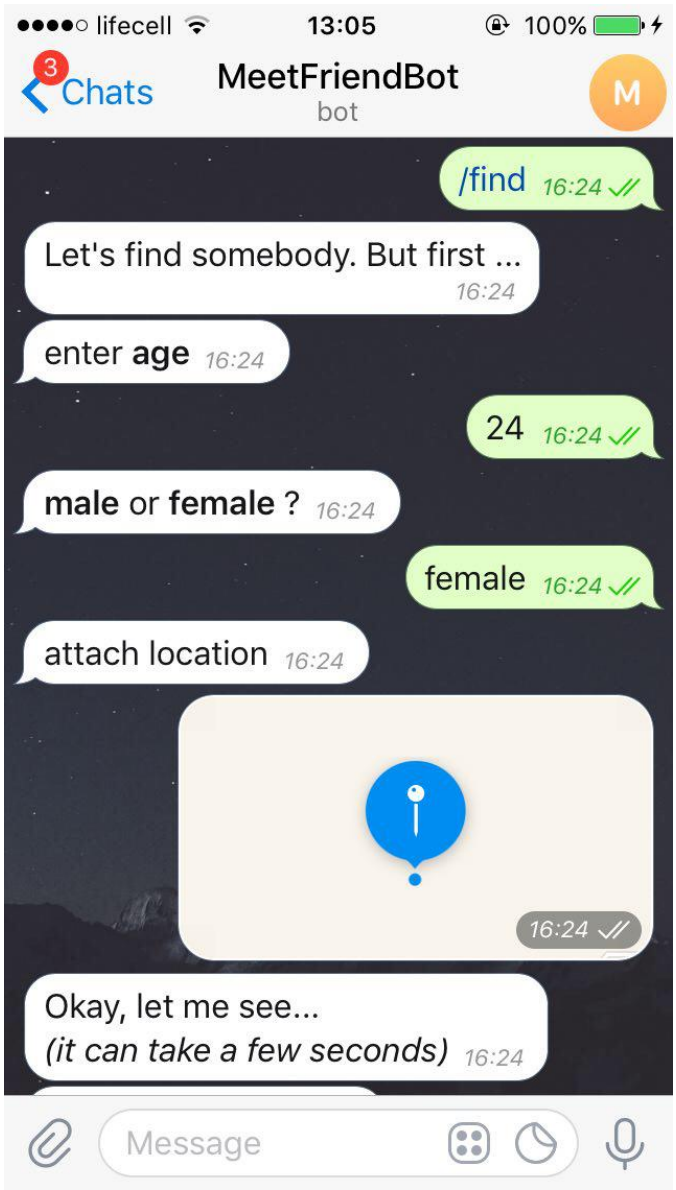
Python – мова, що інтерпретується, проект не потребує ніяких маніпуляцій з вихідним кодом для забезпечення його працездатності, тому системи збірки не використовуються.

Перелік та опис задач, які виконуються на сервері безперервної інтеграції

Система безперервної інтергації (в даному випадку Travis-CI) виконує такі задачі:

- Встановлює потрібну версію python
- Клонує git-репозиторій
- Додає до оточення необхідні змінні конфігурації
- Активує віртуальне оточення python
- Запускає postgresql
- Створює тестову базу даних
- Запускає sql скрипт, що створює таблиці в бд
- Встановлює необхідні пакети python, що перераховані у файлі requirements.txt
- Запускає unit-тести

Приклад функціонування програми:



Графік, який ілюструє вибрані інтервали для повтору спроб при експоненціальній витримці