УДК 004.41

А.О. Юрченко
О.В. Романюк

РОЗРОБКА TELEGRAM-БОТА ДЛЯ СИСТЕМИ УПРАВЛІННЯ РОБОТОЮ СЛУЖБИ ТАКСІ

Вінницький національний технічний університет

Анотація
Ця робота присвячена розробці Telegram-бота для системи управління роботою служби таксі. У роботі наведено процес розробки та загальний алгоритм роботи системи.
Ключові слова: Telegram-бот, UML, обробка команд, маршрут, замовлення

Abstract:
This work is dedicated to the development of a Telegram bot for a taxi service management system. The paper presents the development process and the general algorithm of the system.
Keywords: Telegram bot, UML, command processing, route, order

Вступ

Таксі є одним із найпопулярніших та найзручніших засобів транспорту в містах. Популярність таксі зумовлена необхідністю швидкого та комфортного переміщення, особливо в умовах трафіку та зайнятості сучасних людей. Використання таксі є частиною повсякденного життя для багатьох мешканців міст.

В сучасному світі технології месенджерів стають все більш популярними серед користувачів мобільних пристроїв [1].

З розвитком технологій та зростанням популярності месенджерів, таких як Telegram, месенджери стають все більш популярними та важливими засобами комунікації. Вони дозволяють не лише обмінюватися повідомленнями, а й використовувати різноманітні сервіси та застосунки, включаючи інтеграцію з різними сервісами [2].

Таким чином, об'єднання популярності таксі як засобу транспорту та зростаючої популярності месенджерів, зокрема Telegram, відкриває шлях до новаторських рішень у сфері обслуговування пасажирів. Розробка Telegram-бота для управління таксі відповідає сучасним тенденціям у розвитку інформаційних технологій та може відкрити нові перспективи у сфері транспортних послуг [3].

Метою даної роботи є покращення процесу замовлення таксі шляхом розроблення спеціалізованого Telegram-бота.

Основна частина

Для більш глибокого розуміння роботи програмних модулів у Telegram-боті для системи управління роботи служби таксі було вирішено створити модель системи за допомогою діаграми діяльності мовою моделювання UML (рис. 1). UML діаграми — це візуальні моделі, які використовуються для специфікації, проектування, документування та розуміння системи, що розробляється. UML є стандартом для об'єктно-орієнтованого моделювання, який забезпечує широкий набір діаграм для представлення різних аспектів системи. UML діаграми допомагають візуалізувати та документувати складні системи, полегшуючи їх розуміння та проектування. Вони сприяють ефективній комунікації між членами команди, аналізу вимог, плануванню тестування і виявленню помилок на ранніх етапах розробки. Завдяки цьому UML діаграми забезпечують кращу організацію та управління проектами, підтримують повторне використання компонентів і слугують важливою частиною технічної документації, необхідної для підтримки та розширення систем [4].

Створення діаграми діяльності в мові UML дозволяє візуально представити потік робіт або управління в системі, відображаючи послідовність дій та їх взаємозв'язки. Це допомагає зрозуміти логіку процесів, визначити можливості для паралельного виконання завдань, виявити точки ухвалення рішень та взаємодію між різними етапами діяльності. Діаграми діяльності сприяють аналізу та оптимізації бізнес-процесів, алгоритмів і робочих потоків, полегшують комунікацію між членами

команди і зацікавленими сторонами, та допомагають виявити потенційні вузькі місця для подальшого покращення процесів.

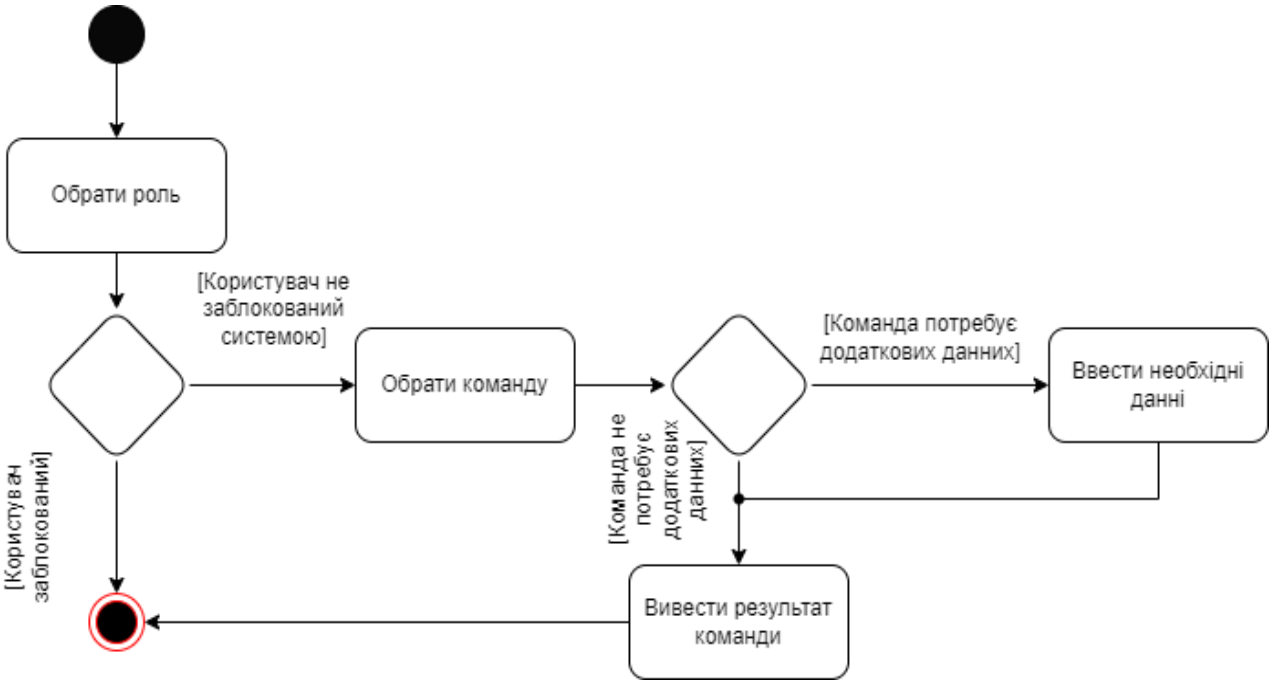

Рисунок 1 – UML-діаграма роботи системи

Згідно з цією діаграмою діяльності, користувач спочатку обирає свою роль, що визначає доступні команди та функції в системі. Далі, користувач має вибрати одну з доступних команд, яка відповідає обраній ролі. Після вибору команди користувач вводить необхідні дані, що дозволяє системі виконати відповідну операцію. Результат виконання команди виводиться на екран для користувача.

Ця послідовність дій забезпечує структурований та зручний процес взаємодії користувача із системою. Вибір ролі на початку дозволяє адаптувати інтерфейс та функціонал під конкретні потреби користувача, що підвищує ефективність та зручність використання системи.

Розробка такої системи потребує створення загального алгоритму її роботи [5] (рис. 2), який буде керувати всіма етапами взаємодії.

Нижче представлено пояснення кроків виконання загального алгоритму роботи:

Крок 1. Початок.

Крок 2. Виведення доступних ролей користувачу, після перевірки чи існує користувач в базі і має доступні ролі для використання.

Крок 3. Вибір ролі користувача.

Крок 4. Перевірка з якою роллю входить користувач.

Крок 5. Очікування вибору команди від користувача.

Крок 6. Перевірка чи потрібно вводити данні користувача для конкретної команди.

Крок 7. Якщо команда потребує даних від користувача, перевірка чи користувач обрав переписку з іншим користувачем.

Крок 8. Встановлення стану переписки користувачу, для того щоб його наступні повідомлення дублювалися іншому користувачу.

Крок 9. Перевірка чи користувач в стані спілкування.

Крок 10. Очікуємо повідомлення від користувача.

Крок 11. Перенаправлення повідомлення.

Крок 12. Встановлення необхідного статусу дії користувачу для коректної обробки даних.

Крок 13. Очікування даних.

Крок 14. Виводимо результат.

Крок 15. Кінець.

Ця діаграма діяльності дозволяє візуалізувати процес взаємодії користувача із системою, забезпечуючи чітке розуміння логіки роботи програмних модулів та їх взаємодії між собою.

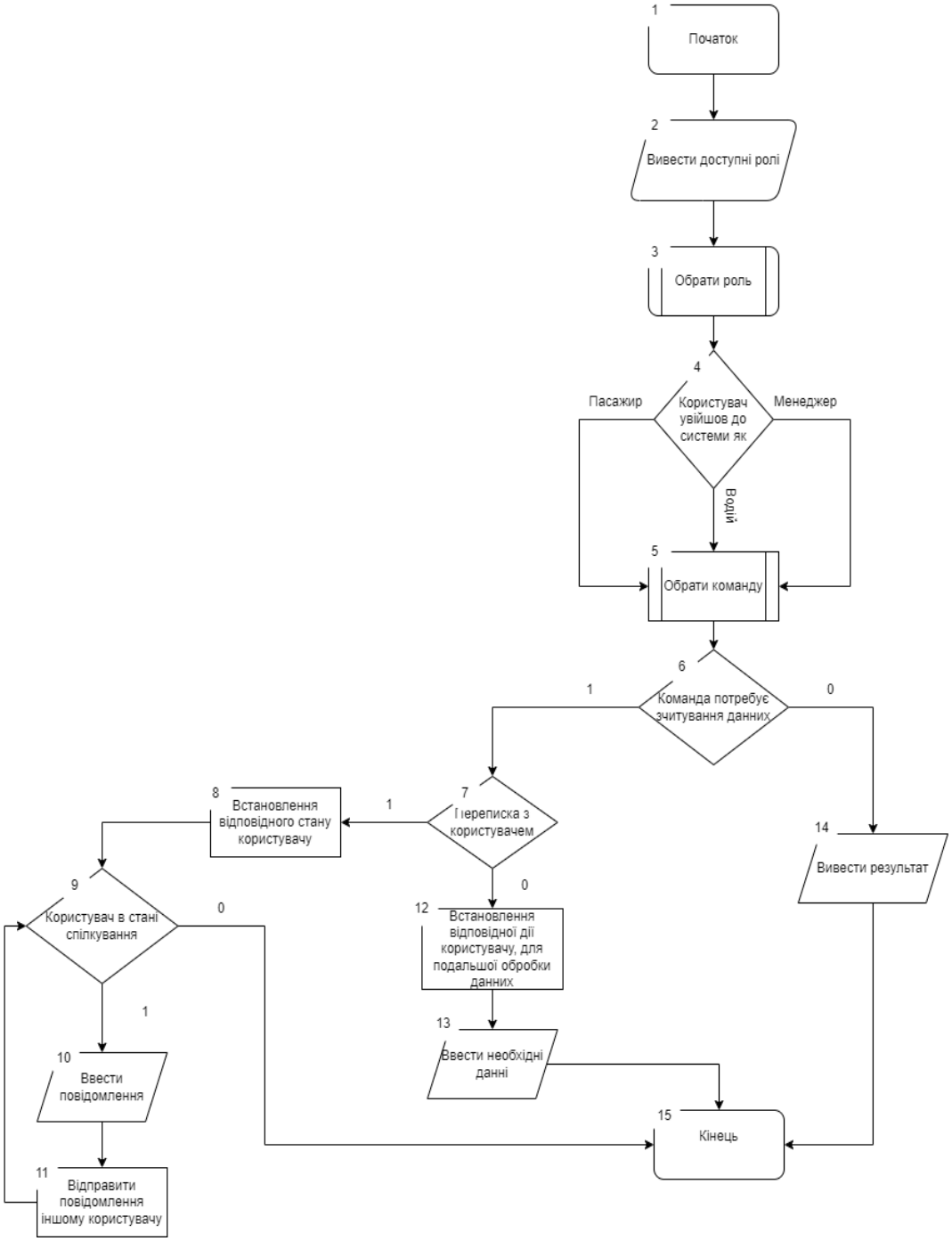

Рисунок 2 – Блок-схема алгоритму роботи системи

Такий алгоритм забезпечує зручний та ефективний спосіб використання системи для користувачів.

Головною функціональністю програмного продукту є коректне опрацювання команд користувача. Для досягнення цієї мети необхідно розділити команди по пріоритетності та додати відповідні стани. Це дозволить програмі коректо обробляти вхідні дані користувача.

Після отримання повідомлення від користувача необхідно правильно його обробити та звернути увагу чи має користувач встановлений стан на обробку даних в системі або ж потрібно виконати команду без обробки додаткових даних.

Програмну реалізацію процесу обробки команд користувача зображено на рисунку 3.

```
private static final List<String> firstPriorityCommand = List.of("Відмінити замовлення",
        "Назад ←\uFE0F",
        "Закрити тікет",
        "Закінчити замовлення",
        "Закінчити розмову з водієм");

public void processTextCommand(Update update) {
    Long chatId = update.getMessage().getChatId();
    UserAction userAction = userService.getUserAction(chatId);
    UserState userState = userService.getUserState(chatId);

    if (update.getMessage().getText() !=null && firstPriorityCommand.contains(update.getMessage().getText())){
        commandControllerForText.processCommand(update.getMessage().getText(), chatId, update);
    }
    else if (!UserAction.DEFAULT.equals(userAction)){
        userActionController.processUserActionCommand(userAction, chatId, update);
    } else if (!UserState.DEFAULT.equals(userState)){
        userStateController.processUserStateCommand(userState, chatId, update);
    } else commandControllerForText.processCommand(update.getMessage().getText(), chatId, update);
}
```

Рисунок 3 – Програмний код методу обробки команд користувача

Також одна з головних функцій – це коректна обробка початкової та кінцевої точки маршруту користувача з визначенням ціни поїздки та приблизним її часом. Після отримання початкової та кінцевої точки маршруту користувача потрібно отримати довжину маршруту, приблизний час поїздки та порахувати ціну поїздки (рис. 4).

```
public JSONObject getRouteInfo(double startLon, double startLat, double endLon, double endLat) {
    String apiUrl = String.format(ROUTE_INFO_URL, startLon, startLat, endLon, endLat);

    try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
        HttpGet request = new HttpGet(apiUrl);
        try (CloseableHttpResponse response = httpClient.execute(request)) {
            String responseBody = EntityUtils.toString(response.getEntity());
            JSONObject json = new JSONObject(responseBody);
            JSONArray routesArray = json.getJSONArray( key: "routes");
            return routesArray.getJSONObject( index: 0);
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

public String getStreetName(double latitude, double longitude) {
    String apiUrl = String.format(URL_FOR_STREET_NAME, latitude, longitude);
    try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
        HttpGet request = new HttpGet(apiUrl);
        try (CloseableHttpResponse response = httpClient.execute(request)) {
            String responseBody = EntityUtils.toString(response.getEntity());
            JSONObject json = new JSONObject(responseBody);
            String houseNumber = getHouseNumber(json);
            return json.getJSONObject( key: "address").get("road").toString() + " " + houseNumber;
        }
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
}


private static String getHouseNumber (JSONObject jsonObject) {
    try {
        return jsonObject.getJSONObject( key: "address").get("house_number").toString();
    } catch (Exception e){
        return "";
    }
}
```

Рисунок 4 – Програмний код для отримання даних про маршрут

Після успішного отримання даних про маршрут, необхідно відобразити користувачеві інформацію про його замовлення (рис. 5).

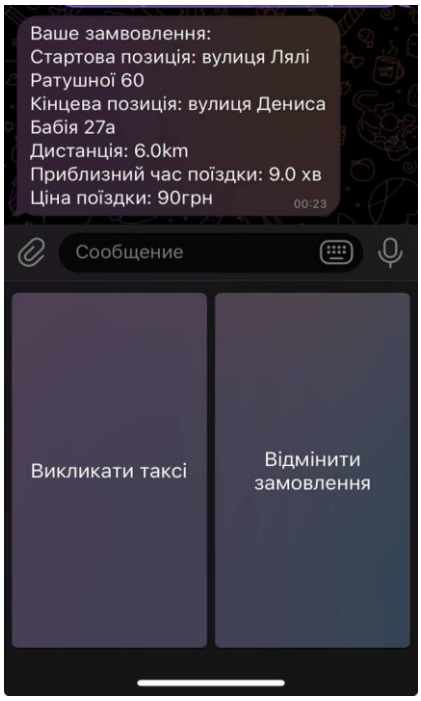

Рисунок 5 – Приклад відображення інформації про замовлення

Висновок

В ході роботи було розглянуто процес розробки Telegram-бота для системи управління роботою служби таксі . Також була розроблена модель системи за допомогою UML-діаграм, яка візуалізує її структуру та взаємодії між компонентами. Для наочного пояснення принципу роботи був розроблений загальний алгоритм, а також програмний модуль для коректної обробки команд користувача та показаний метод обробки замовлення.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Популярність месенджерів [Електронний ресурс]____-____Режим доступу до ресурсу: https://gerabot.com/article/populyarnist_mesendzheriv
2. І.А. Роїк О.В. Романюк. Особливості розробки модуля формування графічних звітів telegram-додатку для контролю особистого бюджету // Матеріали молодіжної науково-практичної інтернет-конференції студентів аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2020)» : збірник матеріалів. – Вінниця: ВНТУ, 2021. – 2 с. –. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/viewFile/13251/11112.
3. Юрченко А. О. Переваги та недоліки телеграм боту порівняно з мобільним застосунком / А. О. Юрченко, О. В. Романюк // Матеріали LIII Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20453/16974
4. What is UML Diagram? URL: https://miro.com/diagramming/what-is-a-uml-diagram/ (date of access 09.05.2024).
5. What is programming algorithm. URL: https://www.indicative.com/resource/programming-algorithm (date of access 09.05.2024).

Юрченко Антон Олександрович – студент групи 1ПІ-20б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: yurchenko.anton@gmail.com

Романюк Оксана Володимирівна – к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: romaniukoksanav@gmail.com

Anton Yurchenko – student of group 1PI-20b, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: yurchenko.anton@gmail.com

Oksana Romaniuk – Candidate of Technical Sciences, Associate Professor of the Software Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: romaniukoksanav@gmail.com