НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Індивідуальний дослідницький проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Вебзастосунок логістичної допомоги постраждалим від війни в Україні»

Виконав:
студент IV курсу, групи ІА-83
Нечитайло Дмитро Юрійович _____
Керівник:
Старший викладач
Моргаль Олег Михайлович _____

Засвідчую, що у цьому проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАВДАННЯ

на індивідуальний дослідницький проєкт студенту

Нечитайло Дмитру Юрійовичу

1. Тема проєкту «Вебзастосунок логістичної допомоги постраждалим від війни в Україні», керівник проєкту Моргаль Олег Михайлович старший викладач.

2. Термін подання студентом проєкту: 15 червня 2022 року

3. Вихідні дані до проєкту:

Мова програмування Java, фреймворк Spring Boot, середовище програмування IntelliJ IDEA 2022.1.1, СУБД MySQL, веб представлення Telegram, розподілена система керування версіями файлів GIT.

4. Зміст пояснювальної записки:

Постановка завдання на дослідницький проєкт, дослідження предметної області, вибір технологій та засобів розробки, проєктування та реалізація застосунку, тестування системи.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

Інфологічна модель предметної області, даталогічна модель бази даних, діаграма варіантів використання, діаграма діяльності.

6. Дата видачі завдання 1 грудня 2021 року

Календарний план

№ з/п	Назва етапів виконання проєкту	Термін виконання етапів проєкту	Примітка
1	Аналіз предметної області	02.05.2022	Виконано
2	Огляд та аналіз методів розв’язання поставленої задачі	09.05.2022	Виконано
3	Проектування та розроблення схеми бази даних	16.05.2022	Виконано
4	Розроблення програмного додатку	23.05.2022	Виконано
5	Тестування	30.06.2022	Виконано
6	Оформлення дослідницької роботи	06.06.2022	Виконано

Студент Дмитро НЕЧИТАЙЛО

Керівник Олег МОРГАЛЬ

АНОТАЦІЯ

Нечитайло Д. Ю. Вебзастосунок логістичної допомоги постраждалим від війни в Україні. КПІ ім. Ігоря Сікорського, Київ, 2022.

Пояснювальна записка містить 72 с. тексту, 29 рисунки, 1 додаток та 17 літературних джерел.

Ключові слова: завдання, здібність, замовник (реципієнт), виконавець (донор), логістика, логіст, бот, стан.

Об’єктом розробки є веб-сервіс (телеграм-бот) де потребуючий та людина, яка має змогу та бажання допомогти знаходять один одного за системою фільтрів та співставлень.

Мета розробки – створення застосунку, який надасть можливість кожному потребуючому в короткі терміни знайти людину чи декілька, які мають змогу допомогти, а останнім, в свою чергу, зробити внесок в країну та її народ.

При проєктуванні та розробці дослідницького проєкту було створено застосунок на базі Telegram API з використанням технології Spring Boot та середовища розробки IntelliJ IDEA 2022.1.1. Проведено аналіз предметної області та визначено ключові сутності, щодо яких відбувалась подальша розробка. Застосунок містить увесь необхідний функціонал для простого та зрозумілого надання та пошуку заявок, проста та зрозуміла логіка користування та стилістичне і текстове обрамлення застосунку.

Отримані результати можуть бути корисними для всіх українців та не тільки за кордоном та в межах України.

SUMMARY

Nechitaylo D. Y. Web application of logistical assistance to war victims in Ukraine. KPI them. Igor Sikorsky, Kyiv, 2022.

The explanatory note contains 72 pages. text, 29 figures, 1 appendix and 17 literature sources.

Key words: task, ability, customer (recipient), executor (donor), logistics, logistician, bot, state.

The object of development is a web service (telegram bot) where the needy and the person who has the ability and desire to help find each other through a system of filters and comparisons.

The purpose of the development is to create an application that will enable everyone in need to find a person or several who can help in a short time, and the latter, in turn, to contribute to the country and its people.

During the design and development of the diploma project, an application based on the Telegram API was created using Spring Boot technology and the IntelliJ IDEA 2022.1.1 development environment. The analysis of the subject area is carried out and the key essences concerning which further development took place are defined. The application contains all the necessary functionality for simple and clear submission and search of applications, simple and clear logic of use and stylistic and textual framing of the application.

The obtained results can be useful for all Ukrainians and not only abroad and within Ukraine.

Номер рядка	Формат	Позначення	Найменування	Кільк. листів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА83.220БАК.003 ПЗ	Пояснювальна записка	72		
6	А3	ІА83.220БАК.003 Д1	Вебзастосунок логістичної	1		
7			допомоги постраждалим			
8			від війни в Україні.			
9			Інфологічна модель			
10	А3	ІА83.220БАК.003 Д2	Вебзастосунок логістичної	1		
11			допомоги постраждалим			
12			від війни в Україні.			
13			Даталогічна модель			
14	А3	ІА83.220БАК.003 Д3	Вебзастосунок логістичної	1		
15			допомоги постраждалим			
16			від війни в Україні.			
17			Діаграма варіантів			
18			використання			
19	А3	ІА83.220БАК.003 Д4	Вебзастосунок логістичної	1		
20			допомоги постраждалим			
21			від війни в Україні.			
22			Діаграма діяльності			
23						
24						
25						
26						
27						
28						

Зм.	Лист	№ докум.	Підпис	Дата	ІА83.220БАК.003 ТП		
Розроб.		Нечитайло Д.Ю.			Вебзастосунок логістичної допомоги постраждалим від війни в Україні Відомість проєкту	Літ.	Аркуш / Аркушів
Перевір.		Моргаль О.М.					1
						КПІ ім. І. Сікорського ФІОТ Група ІА-83	
Затв.							

Пояснювальна записка

до індивідуального дослідницького проєкту

на тему: «Вебзастосунок логістичної допомоги постраждалим від війни в Україні»

Київ – 2022 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

СУБД – система управління базами даних

API – Application Programming Interface

DAO – Decentralized Autonomous Organisation

GoF – Gang of four. Назва шаблонів проєктування

HTTP – HyperText Transfer Protocol

IDE – Integrated development environment

IoC – Inversion of Control

IP – Internet Protocol

JDBC – Java DataBase Connectivity

JDK – Java Development Kit

JVM – Java Virtual Machine

JSON – JavaScript Object Notation

ORM – Object-Relational Mapping

OS – Operating system

SQL – Structured Query Language

TCP –Transmission Control Protocol

UML –Unified Modeling Language

URL – Uniform Resource Locator

XML – eXtensible Markup Language

ВСТУП

Протягом останніх місяців, через військові дії, в нашій країні утворилося безліч людей, які з тих чи інших причин не мають можливості отримати необхідні послуги, або отримання яких викликає деякі труднощі, через відсутність необхідної інформації.

У сучасній ситуації порушено логістику, багато людей втратили транспорт, перебувають у регіонах, де ведуться бойові дії та заблокований доступ до гуманітарних ресурсів, або були змушені екстрено залишити місце проживання, рятуючи свої життя, тимчасово переїхавши у інші регіони нашої країні чи за кордон [1]. Саме цим людям потрібна допомога та сервіс, який зможе забезпечити постраждалих джерелом цієї допомоги.

В даний момент кількість громадян нашої країни, які потребують допомоги і кількість волонтерських організацій та окремих волонтерів зростає, і стає все складніше швидко організувати передачу допомоги, і щоб всі у максимально короткі терміни отримували те, що так їм необхідно.

Сучасний світ не стоїть на місці і галузь інформаційних технологій безпосередньо впливає на швидкоплинність його розвитку. Якщо донедавна для вирішення основної маси питань використовувалися комп'ютерні програми, додатки або веб-сайти, то зараз на першість претендують чат-боти, які мають широкі перспективи у різних сферах нашого життя.

Завдяки швидкому росту компанії та вимогам щодо конфіденційності та безпеки Telegram став важливою платформою для бізнесу. Завдяки розширеній базі користувачів Telegram став важливою платформою, на якій можна зосередитися.

Оскільки щодня з’являються нові варіанти для використання ботів [2], боти стають все більш популярними для роботи в середовищі Telegram. Це допомагає підприємствам досить довго утримувати користувачів, збирати необхідні дані тощо.

Враховуючи глобальні тенденції щодо випадків злому, Telegram вважається досить безпечним, головним чином, тому що повідомлення надсилаються через

платформу в зашифрованому вигляді. Це є досить вагомою причиною для використання його в такій особистій сфері, як прохання та волонтерство, оскільки він захистить всі дані клієнтів. Це надасть користувачам безпечне та сприятливе місце для взаємодії щодо продуктів і послуг.

У цей нелегкий час щодня сотні тисяч людей стикаються з необхідністю отримати ту чи іншу послугу або допомогу, яку можуть надати люди, що знаходяться поруч, але не знають про існування один одного. Сучасна логістика з надання підтримки таким людям, у більшості випадків, не дає можливості волонтерам або просто охочим допомогти людям, і людям, які опинилися в критичній ситуації миттєво знайти один одного. Саме тому метою даного проєкту, стало створення сервісу, який буде збирати всю необхідну інформацію від користувачів, які запитують послугу, та у користувачів, які можуть допомогти, маючи ту чи іншу професію або ресурси, накопичувати її, та за системою фільтрів та співставлень надавати можливість знаходити їм один одного.

Метою проєкту є створення телеграм-бота, який зможе реєструвати та зберігати дані користувачів, згідно їх місцезнаходження, потреб та можливостей, за запитом, надавати зіставлення, найбільш продуктивні, згідно поточного стану бази даних.

А також інформувати користувачів про всі зміни станів заявок та корисну інформацію, згідно історії взаємодії із ботом.

На основі мети було визначено наступні завдання:

- дослідження предметної області;
- огляд та аналіз існуючих рішень;
- вибір технологій та засобів для розробки;
- проєктування та реалізація застосунку з визначеними функціями.

1 ПОСТАНОВКА ЗАВДАННЯ НА ДОСЛІДНИЦІКИЙ ПРОЄКТ

У проєкті необхідно розробити сервіс логістичної допомоги постраждалим від війни в Україні. Який дозволить автоматизувати організацію розподілу гуманітарної допомоги, а також – створювати зв’язки між донорами та реципієнтами тієї чи іншої послуги.

У боті повинні бути реалізовані наступні функції:

- реєстрація користувача;
- функціонал розміщення заявок на допомогу;
- функція пошуку заявок та відгуків виконавців на них;
- функція пошуку заявок на логістику та відгуків на них;
- функція сповіщення при зміні статусу заявки;
- корегування профілю та заявок.

Щоб уникнути ситуацій, коли користувач опиняється в незрозумілій або безвихідній ситуації необхідно створити логічний, простий інтерфейс. Проаналізувати сценарії дій користувача в системі та обробити всі можливі дії. Обробити введення користувачем невірної команди чи помилки перевірки вхідних даних. У разі отримання сервером неочікуваних даних, інформувати користувача відповідним повідомленням.

Необхідно розробити меню управління функціями програми, яке забезпечувало б зручну роботу користувача. Впровадити можливість змінювати збережені дані, які відносяться до користувача, таку інформацію, як: дані профілю, список особистих здібностей та список створених завдань.

2 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Об’єктом дослідження даної дослідницької роботи є наявність потреби в отриманні допомоги у людей, які зазнали складнощів з початком війни в Україні [3]. Потреби розглядаються як відсутність можливості скористатися деякими послугами, або відсутність тих чи інших ресурсів.

На даний час мешканці територій, на яких або поряд з якими проводяться бойові дії, у великій кількості випадків не мають доступу до продовольчих магазинів, аптек, лікарень тощо. У багатьох містах порушено рух міського транспорту, а також логістика тих чи інших продуктів [4].

В системі, що проєктується, зроблено акцент на знаходження допомоги від людей, які знаходяться поряд, та мають можливість, бажання та кваліфікацію щоб надати підтримку [5].

Підхід до вирішення даної проблеми можна поділити на кілька етапів:

- збір та обробка особистих даних;
- створення заявки на допомогу;
- створення заявки на транспортне перевезення;
- надання передбачуваним виконавцям найбільш доречних завдань;
- надання логістам найбільш доречних заявок на транспортне перевезення;
- моніторинг станів заявок.

Відповідно до цього, можна виділити кілька основних ролей: реципієнти, тобто користувачі, які створюють завдання, та донори – які їх виконують. В свою чергу донорів можна поділити на виконавців завдань, та на логістів.

На практиці і донором і реципієнтом може бути один й той самий користувач, який має нагальні потреби і при цьому може допомогти комусь ще.

Усі завдання теж можна поділити на кілька типів:

- допомога руками;
- дім / притулок;
- допомога речами;

- харчування;
- техніка;
- ліки;
- лікарі / медсестри;
- освіта;
- допомога із дітками;
- психологія;
- юриспруденція / документи;
- інше.

Створена заявка на допомогу зберігає інформацію про статус цієї заявки, індивідуальний номер, категорію, опис, замовника, виконавця, якщо є потреба, то логіста, місце реалізації та дату та час створення.

Також необхідно врахувати можливість не реалізації завдання, наприклад через те, що не знайшли виконавця вчасно. Та реалізувати механізм автоматичного виключення прострочених завдань з загального списку, що надається користувачу системи.

2.1 Огляд існуючих рішень

2.1.1 Вебзастосунок Helplist

Вебзастосунок Helplist [6] є сайтом, який було створено відносно недавно, за допомогою якого користувачі мають можливість розмістити запит на отримання чи надання допомоги а також на знаходження близьких.

Додаток містить в собі модуль для волонтерів або фондів, який складається з критеріїв фільтру запитів, після уточнення яких змінюється список наданих запитів.

Зразок користувацького інтерфейсу з усіма функціями системи зображений на рисунку 2.1

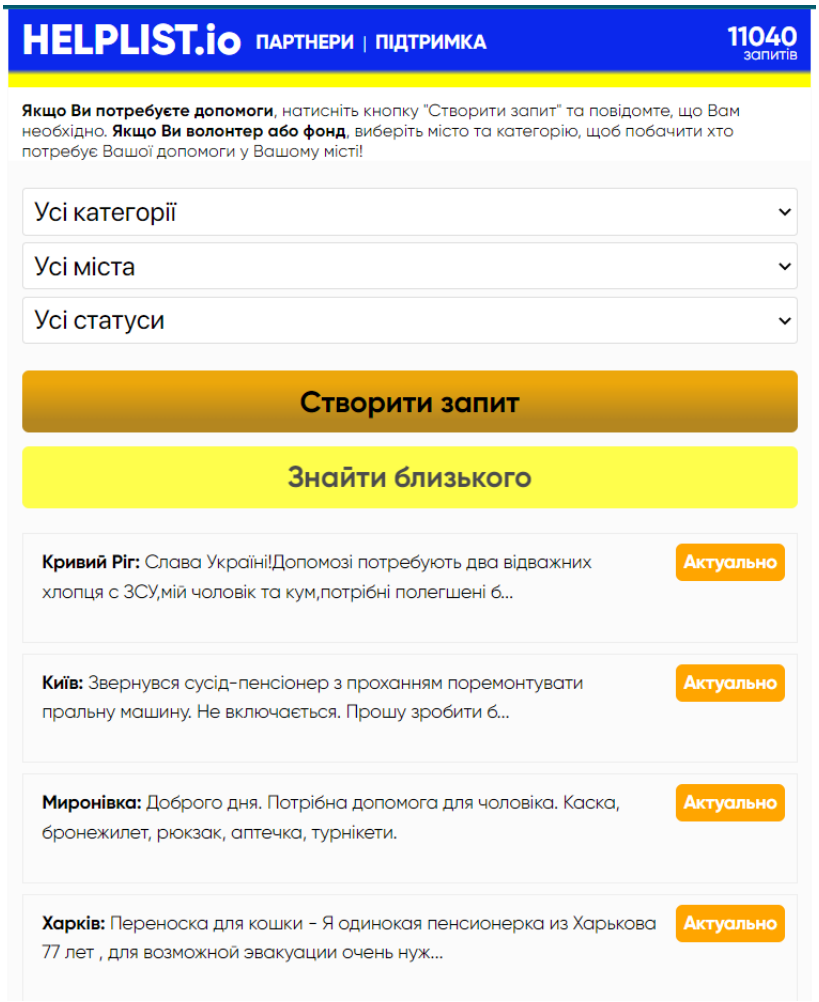

Рисунок 2.1 – Зразок користувацького інтерфейсу

Основні переваги:

- миттєва готовність до виконання головної функції без надмірностей;
- простий інтерфейс;
- наявність підтримки.

Основні недоліки:

- відсутність авторизації, через це: відсутність зручної можливості відстежувати та змінювати користувачем статус запиту, відсутність можливості відгородити людей, яким дійсно потрібна допомого від несумлінних користувачів;
- не скрізь логічний дизайн взаємодії з користувачем;
- не до всіх мобільних пристроях адаптується інтерфейс користувача;
- відносно складне для сприйняття представлення списку заявок (все в

одному списку, і оголошення про бажання допомогти, і заявки на прохання про допомогу).

2.1.2 Додаток UA HELP NOW

Телеграм бот UA HELP NOW [7]— це бот, який знаходиться у системі обміну повідомленнями Telegram, головна мета якого полягає у тому, щоб допомогти тим, хто шукає допомогу, знайти тих, хто її надає. Для більш вузько спрямованого створення заявки бот має різноманітні категорії, які охоплюють значну частину ситуацій. Система повідомлень нагадує про можливо забуті дії, та пропонує завершити той, чи інший етап.

Дана система має величезну базу корисної інформації, яка допоможе тим, хто не знає як поводитись в тій чи інше ситуації, допоможе знайти прихисток чи інший волонтерський центр. Система має зручний та зрозумілий інтерфейс, стабільно та швидко працює.

Користувацький інтерфейс додатку зображений на рисунку 2.2

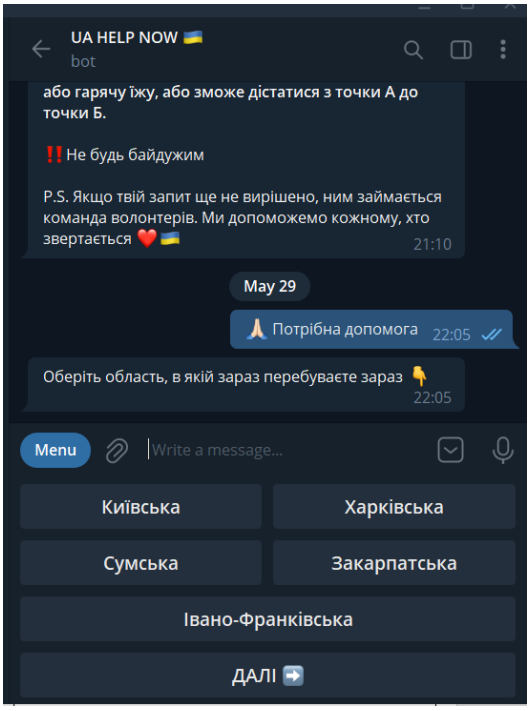

Рисунок 2.2 – Зразок користувацького інтерфейсу

Найбільшим і ключовим недоліком цього бота є те, що всю обробку додатків і логістику виконують адміністратори. Цей недолік значно знижує ефективність роботи системи, крім того, неможливо переглянути всі оголошення та обрати на свою думку найкращій.

2.1.3 Viber бот Україна допомога

Viber бот Україна допомога [8]— це бот, який знаходиться у системі обміну повідомленнями Viber, головна мета якого полягає у тому, щоб допомогти тим, хто шукає допомогу, знайти тих, хто її надає.

Користувацький інтерфейс додатку зображений на рисунку 2.3

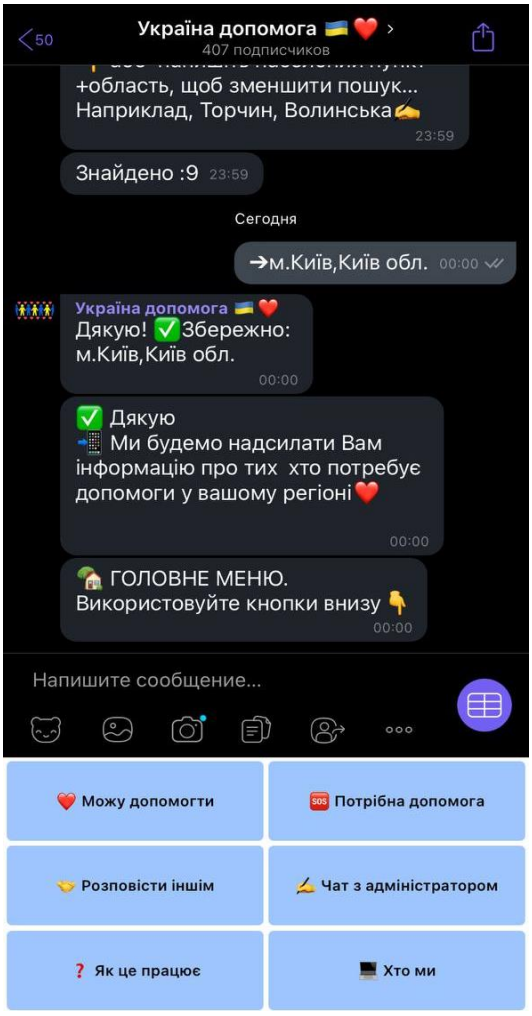

Рисунок 2.3 – Зразок користувацького інтерфейсу

Бот має зрозумілий, дружній інтерфейс, перехід між станами логічний.

Метою проєкту є об’єднання людей в одному боті, де будь-хто може допомогти або попросити допомогу.

Основні переваги:

- доступність;

- простий та зрозумілий інтерфейс;

- можливість розмістити в системі завдання, яке буде автоматично оброблене.

Основні недоліки:

- відсутність можливості переглянути список всіх активних завдань в системі;

- наявність загальмувань в відповідях;

- відсутність збереження інформації про місцезнаходження.

2.2 Інфологічна модель

Проведення дослідження предметної області включає в себе розробку інфологічної моделі.

Інфологічний аспект мається на увазі при розгляді питань, пов’язаних із семантичним змістом даних, незалежно від того, як система представлена в пам’яті. На етапі проєктування інфологічної моделі були вирішені такі питання:

- які предмети чи явища потрібні для накопичення чи використання інформації;

- які їх основні характеристики та взаємозв’язки слід враховувати.

Таким чином, на цьому етапі здійснюється відбір та опис частини реального світу, яка має бути представлена в інформаційній системі, тобто визначається предметна область (програмне забезпечення) проєктованої бази даних.

Інфологічна модель предметної області даних зображений на кресленику ІА83.220БАК.003 Д1.

На моделі зображені основні сутності бази даних, а саме:

- користувач, який має інформацію про своє місцезнаходження, ПІБ, номер телефону, ім’я користувача, яке зареєстроване в телеграм, інформацію про час та дату маніпуляцій з системою та особистий ідентифікаційний номер користувача;

- місцезнаходження, яке містить інформацію про країну, регіон та місто, рідний регіон, довготу та широту і номер місцезнаходження, що характеризує його унікальність;

- країна, яка містить особисту назву на порядковий номер;

- регіон, яке містить особисту назву на порядковий номер;

- місто, яке містить особисту назву на порядковий номер;

- завдання, яке містить інформацію про користувача, який створив його (реціпієнт), про виконавця (донор), про логіста та опис завдання, статус і номер завдання;

- вміння, яке містить інформацію про користувача, опис, номер та дату створення.

Основним вузлом зв’язку в системі виступає користувач, який має місцезнаходження, створює завдання, та додає свої вміння.

Висновки до розділу 2

У даному розділі було розглянуто та досліджено деякі існуючи рішення програмних застосунків для організації допомоги постраждалим від війни в Україні.

Основні недоліки готових рішень:

- недостатня автоматизація процесів;

- відносно складні для користувача засоби представлення цільових оголошень;

- відсутність можливості відстежувати статус оголошення;

- відсутність збереження інформації, що часто вводиться.

Відповідно до цих недоліків можна сформувати перелік вимог до функціоналу сервісу, що розробляється в межах дослідницького проєкту.

Основною вимогою є впровадження програмного компоненту з інтерфейсом, використовуючи функціонал якого, користувач мав би можливість без зусиль зробити те, за чим він прийшов на цей застосунок.

Важливим критерієм є забезпечення належних та безпечних умов роботи застосунку на усіх операційних системах. Важливим аспектом є простота на початку користування сервісом.

У розділі також була проаналізована предметна область та визначені основні сутності проєктованої бази даних, а також атрибути цих сутностей та ролі користувачів.

На основі виявлених даних побудовано інфологічну модель предметної області, яка відображає сутності та зв’язки в нотації Чена. Ця модель відповідає вимогам до проєктованої системи, тому її буде доцільно використовувати для подальшого проєктування сутностей, створення даталогічної моделі бази даних та програмної реалізації.

3 ВИБІР ТЕХНОЛОГІЙ ТА ЗАСОБІВ ДЛЯ РОЗРОБКИ

3.1 Мова програмування

Для написання додатку з клієнтською стороною у вигляді Телеграм боту існує чимало мов програмування. З лідерів можна виділити Python та Java.

Python і Java є двома найпопулярнішими та надійними мовами програмування. Java, як правило, швидша та ефективніша, ніж Python, оскільки це скомпільована мова. Як інтерпретована мова, Python має простіший, більш стислий синтаксис, ніж Java. Він може виконувати ту ж функцію, що й Java, у меншій кількості рядків коду.

Java залишається однією з найпоширеніших мов програмування з моменту її створення. Java надає можливість вирішувати різні типи задач, вона була розроблена як універсальна мова програмування. За весь час було опубліковано багато різних версій.

Поточна версія — Java 18, випущена 22 березня 2022 року. Java перетворилася з універсальної мови на повноцінну екосистему, яка поєднує в собі різноманітні методики та технології, які можна застосувати майже в усіх сферах: настільні застосунки, вебзастосунки, великі дані, мобільна розробка тощо.

При порівнянні мов по критерію виявлення помилок було візначено, що у Python будь-які помилки, внесені програмістом, не будуть знайдені, доки не буде запущено цей рядок коду. Це може призвести до збою в роботі та подовжити час виконання. Хоча Python залишає об'єкти вразливими до мутацій, в Java мутації об'єктів неможливі. Це веде до безпечної розробки програмного забезпечення.

Головною перевагою мови Java є транслювання коду в окремий незалежний від будь-якої платформи байт-код. Після цього байт-код виконується віртуальною машиною JVM (Java Virtual Machine). Саме в цьому Java має відмінність від стандартних інтерпретованих мов, наприклад як PHP, код яких виконується інтерпретатором негайно. При цьому Java не є чисто скомпільованою мовою, як C або C ++.

Дана архітектура забезпечує кросплатформність програм, написаних на Java, завдяки чому такі програми можуть без перекомпіляції безпроблемно працювати на різних платформах – Linux, Windows, Mac OS тощо. Кожна платформа може мати власну реалізацію JVM, але кожна буде виконувати один і той же самий код.

Ще одна вагома особливість Java полягає в тому, що вона підтримує автоматичне збирання сміття, не потрібно вручну звільняти пам’ять від раніше використовуваних об’єктів, як у C ++, оскільки збирач сміття робить це автоматично.

Java є об'єктно-орієнтованою мовою. Ця методологія заснована на використанні в програмі об'єктів і класів. Об’єктно-орієнтований підхід дозволяє створювати великі, але в той же час гнучкі, масштабовані і розширювані додатки.

Oracle JDK і OpenJDK

Для розробки на мові програмування Java потрібен спеціальний набір інструментів – JDK. Дві найпопулярніші реалізації JDK, – Oracle JDK і OpenJDK. У чому їх відмінність?

Oracle JDK був повністю розроблений Oracle. OpenJDK розробляється Oracle та багатьма іншими компаніями разом.

Найбільші відмінності полягають у ліцензуванні. За ліцензією Oracle JDK можливо використовувати безкоштовно для особистого користування, а також для розробки, тестування та демонстрації додатків. В інших випадках (наприклад, для отримання підтримки) потрібна комерційна ліцензія у вигляді підписки. А OpenJDK абсолютно безкоштовний.

З точки зору функціональності, набір функцій Oracle JDK і OpenJDK повинен бути практично нерозрізненим. Але з точки зору продуктивності, помічено, що Oracle JDK трохи швидше OpenJDK. Крім того, деякі розробники помічають, що Oracle JDK більш стабільний.

Після аналізу функцій, архітектури, лексики та поширеності мов програмування було прийнято рішення використати саме Java для реалізації дослідницького проєкту.

3.2 Фреймворк

Для основи додатку, як інфраструктуру програмних рішень

Spring – один із найпопулярніших Java-фреймворків. Йому віддають перевагу більшість розробників цієї мови, зокрема завдяки можливості впровадження залежностей.

Незважаючи на те, що Spring Framework є потужним і всеосяжним, він все одно потребує значного часу та знань для налаштування, налаштування та розгортання додатків Spring, Spring Boot [9] пом'якшує ці зусилля.

Обравши Spring Boot з’являється можливість робити розробку вебзастосунків за допомогою Spring Framework швидшою та простішою, використовуючи автоналаштування та можливості створювати окремі програми. Ці функції працюють разом, щоб надати інструмент, який дозволяє налаштувати програму на основі Spring з мінімальною конфігурацією.

Spring Framework пропонує функцію ін’єкції залежностей, яка дозволяє об’єктам визначати власні залежності, які пізніше впроваджує в них контейнер Spring. Це дозволяє розробникам створювати модульні програми, що складаються з слабо пов’язаних компонентів, які ідеально підходять для мікросервісів і розподілених мережевих додатків.

Spring Framework також пропонує вбудовану підтримку типових завдань, які має виконувати програма, таких як прив’язка даних, перетворення типів, перевірка, обробка винятків, управління ресурсами та подіями, інтернаціоналізація тощо. Він інтегрується з різними технологіями Java EE, такими як RMI (віддалений виклик методу), AMQP (розширений протокол черги повідомлень), веб-служби Java та інші. Загалом, Spring Framework надає розробникам усі інструменти та особливості, необхідні для створення слабо пов’язаних міжплатформних програм Java EE, які працюють у будь-якому середовищі.

Ключовою відмінністю або ключовою особливістю Spring є впровадження залежностей, а для Spring Boot це автоналаштування, за допомогою Spring Boot

Framework розробники можуть скоротити час розробки, зусилля розробника та підвищити продуктивність.

Наразі добре зрозуміла різниця між Spring та SpringBoot, хоча вони йдуть рука об руку. Загалом, Spring Boot містить всю функціональність стандартного фреймворку Spring, а також значно полегшує розробку додатків. У порівнянні зі Spring, є можливість запустити програму за значно менше часу, оскільки всі атрибути Spring Boot налаштовуються автоматично.

Тому найбільш підходящим фреймворком для розроблюваного додатку буде Spring Boot.

3.3 Spring Data та Hibernate

Spring Data — це проєкт SpringSource високого рівня, метою якого є уніфікація та полегшення доступу до різних типів сховищ, як до систем реляційних баз даних, так і до сховищ даних NoSQL.

Основною метою шаблону Spring Data (і всіх інших шаблонів Spring) є розподіл ресурсів і переклад винятків.

У нашому випадку ресурсом є сховище даних, доступ до якого часто здійснюється віддалено через TCP/IP-з'єднання.

JPA представив стандарт для відображення O/R (тобто відображення графіків об'єктів у таблиці реляційної бази даних).

Hibernate, мабуть, найпоширеніший маппер O/R, який реалізує специфікацію JPA.

Spring Data дає можливість повністю видалити реалізації DAO. Інтерфейс DAO тепер є єдиним артефактом, який нам потрібно чітко визначити.

Рівень DAO зазвичай складається з великої кількості шаблонного коду, який можна і потрібно спростити. Переваг такого спрощення багато: зменшення кількості артефактів, які потрібно визначити та підтримувати, узгодженість шаблонів доступу до даних та узгодженість конфігурації.

3.4 Telegram Bot API

Telegram — це програма для обміну повідомленнями.

Враховуючи глобальні тенденції щодо випадків злому на кожній платформі соціальних мереж, Telegram вважається досить безпечним, головним чином, тому що повідомлення надсилаються через платформу в зашифрованому вигляді. Telegram швидкий, одержувач отримує повідомлення так само швидко, як буде натиснуто кнопку надсилання.

Месенджер telegram присутній майже на всіх платформах: телефони Android, iOS, Windows, Mac, Linux. Крім того, він також має веб-версію, яка дозволяє націлити потенційних клієнтів у дуже широкому масштабі.

Телеграмний бот може обслуговувати компанії чи бренди з багатьма функціями, такими як надсилання інформації, нагадування, відтворення мелодій, замовлення тощо.

Для розширення функціональності Telegram доцільно використовувати Bot API [10] — інтерфейс на основі HTTP.

Боти — це сторонні програми, які працюють усередині Telegram. Користувачі можуть взаємодіяти з ботами за допомогою надсилання повідомлень, команд та вбудованих запитів. Розробник керуєте ботами, використовуючи HTTPS-запити до Telegram Bot API.

Для навігації по застосунку, кожного разу, коли бот надсилає повеления, є можливість відправляти специальную клавиатуру, з попередньо заданою інформацією. Натискання будь-якой з кнопок негайно надсилає відповідну команду. Таким чином, значно спрощується взаємодія користувача із ботом.

Отримання даних від Telegram можна реалізувати двома способами:

Webhook [11] – під час роботи через вебхуки про всі зміни в чатах Telegram відразу надсилає в http-запитах структури даних, які називаються апдейтами (Updates) – серіалізований JSON. Потрібна підтримка https з'єднання та сертифікат SSL початкового рівня (право володіння доменом).

Long polling – бот повинен сам періодично вимагати оновлення у Telegram, використовуючи метод getUpdates. Опитування має відбуватися не рідше 1 разу на добу, оскільки довше Telegram не зберігає оновлення.

3.5 Інтегроване середовище розробки

Найпопулярнішими IDE для Java є Intellij IDEA та Eclipse.

IntelliJ Idea є інтегрованим середовищем для розробки мови Java, і IntelliJ визнаний одним з найкращих інструментів розробки Java в галузі, особливо в розумних помічників коду, підказках автоматизації коду, рефакторингу, підтримці Java EE, Ant, JUnit, CVS консолідація, перегляд коду, інноваційний дизайн графічного інтерфейсу та інші аспекти функції можна назвати надзвичайними.

З переваг можна виділити розумний відбір. У багатьох випадках потрібно обрати метод або цикл або зробити крок від змінної до всього класу, дане середовище передбачає це на основі синтаксису вибору. Багата модель навігації.

IDEA забезпечує розширений режим перегляду навігації, за допомогою однієї з комбінацій клавіш можна переглянути нещодавно відкриті файли, за допомогою іншої – вікно пошуку назви класу тощо. Однією з її головних переваг є покращена підтримка рефакторингу. Idea є першою в усіх IDE, яка підтримує рефакторинг.

Також в цьому середовищі розробки ідеальна підтримка XML. А також простий і зручний у використанні графічний інтерфейс.

Eclipse надає базову робочу область, систему розширюваних плагінів, спеціальний компілятор та підтримку інструментів. Це дозволяє розробникам редагувати, переробляти, налагоджувати та налаштовувати програми за допомогою настільної або хмарної версії платформи (через веб-браузер, використовуючи Eclipse Che).

Є багато причин, чому Eclipse залишається однією з найпопулярніших середовищ IDE для Java. Eclipse підтримує безліч плагінів. Користувачі навіть

можуть створити власне середовище розробки плагінів.

Eclipse призначений для роботи з великими проєктами розробки. Отже, якщо розробник чи команда працює над великим проєктом, можливо, буде розумно використовувати Eclipse. Але Eclipse має тенденцію працювати повільніше, ніж Intellij IDEA. Для цього є багато причин, наприклад, недостатнє виділення пам’яті.

Підводячи плючи та мінуси було прийнято рішення використовувати Intellij IDEA в якості IDE.

3.6 СУБД

MySQL є популярною і широко використовуваною системою СУБД, яка працює переважно на моделі реляційної бази даних . Це робить адміністрування баз даних легшим і гнучкішим.

Postgre — це об'єктно-реляційна система управління базами даних (ORDBMS), яка пропонує підтримку SQL і NoSQL. MySQL відомий у всьому світі як найбезпечніша та найнадійніша система керування базами даних.

Підтримка транзакцій, яку має MySQL буде корисним для застосунку, а захист та безпека даних збереже всію конфіденційну інформацію, яка була надана користувачем.

MySQL [12] має чітку структуру системи зберігання даних, яка полегшує системним адміністраторам конфігурацію сервера баз даних MySQL для

бездоганної роботи.

Обидві бази даних підтримують керування користувачами та групами та надання привілеїв SQL ролям. PostgreSQL підтримує фільтрацію та аутентифікацію на основі IP-клієнтів за допомогою PAM і Kerberos, а MySQL підтримує PAM, власні служби Windows і LDAP для аутентифікації користувачів. З точки зору безпеки, дві бази даних мають порівнянні варіанти.

MySQL [13] є одним з найнадійніших механізмів транзакційних баз даних, в порівнянні за аналогами. Повну цілісність даних в СУБД характеризує

ізольованість, атомарність, довговічність підтримки транзакцій, необмежене блокування на рівні рядків, багатоверсійність та послідовність транзакцій в системі.

Особливості MySQL:

- дозволяє реплікацію SSL на основі журналів і тригерів;
- об’єктно-орієнтований і сумісний з ANSI-SQL2008;
- пропонує підтримку багатоверсійного контролю паралельності;
- багатошаровий дизайн з незалежними модулями;
- сумісний з різними платформами, які використовують усі основні мови та проміжне програмне забезпечення;
- пропонує вбудовані інструменти для аналізу запитів та аналізу простору;
- повністю багатопотоковий з використанням потоків ядра.

Особливості PostgreSQL:

- працює на всіх основних платформах ОС;
- розширена індексація для високопродуктивної звітності;
- об’єднання та перегляди таблиць для гнучкого пошуку даних.

Недоліки використання MySQL:

- відсутній модуль аутентифікації плагіна;
- немає підтримки ролей;
- таблиці, які використовуються для процедури або тригера, завжди попередньо заблоковані.

Недоліки використання PostgreSQL:

- немає можливості оновлення для основних випусків;
- дані потрібно експортувати або реплікувати в нову версію;
- під час оновлення потрібне подвійне сховище;
- індекси не можна використовувати для прямого повернення результатів запиту;
- операції масового завантаження можуть бути пов’язані з процесором;
- підтримка незалежних постачальників програмного забезпечення.

Для розроблювальної системи більш доречним буде використання СУБД MySQL, через її швидкість та продуктивність для такого рівня додатків.

3.7 GitHub

GitHub — це веб-сайт і хмарний сервіс, який допомагає розробникам зберігати код і керувати ним, а також відстежувати та контролювати зміни коду.

Контроль версій допомагає розробникам відстежувати та керувати змінами в коді програмного проєкту. У міру розвитку програмного проєкту контроль версій стає важливим.

Контроль версій дозволяє розробникам безпечно працювати через розгалуження та злиття.

За допомогою розгалуження розробник дублює частину вихідного коду — репозиторій. Потім розробник може безпечно вносити зміни до цієї частини коду, не впливаючи на решту проєкту.

Потім, як тільки розробник запустить свою частину коду належним чином, він чи вона може об’єднати цей код назад у основний вихідний код, щоб зробити його офіційним.

Усі ці зміни потім відстежуються та можуть бути скасовані, якщо це необхідно.

Git — це специфічна система контролю версій з відкритим вихідним кодом. А саме, Git – це розподілена система контролю версій, що означає, що вся кодова база та історія доступні на кожному комп’ютері розробника, що дозволяє легко розгалужуватися та об’єднуватися.

3.8 Технологія отримання даних від телеграм

Є два способи, як бот може отримувати повідомлення з серверів Telegram. Вони називаються long pooling та webhook.

У цьому розділі описано, що насправді таке long pooling та webhook, і, у свою чергу, окреслено деякі переваги та недоліки використання того чи іншого методу розробки.

Long pooling означає, що сервер проактивно надсилає запит до Telegram із проханням про нові оновлення (повідомлення). Якщо жодних повідомлень немає, Telegram поверне порожній список, який вказує, що жодних нових повідомлень не було надіслано боту з моменту останнього запиту.

Коли сервер надсилає запит до Telegram, при цьому нові повідомлення будуть надіслані боту, Telegram поверне їх у вигляді масиву до 100 об’єктів оновлення.

Схема принципу роботи long pooling наведена на рисунку 3.1

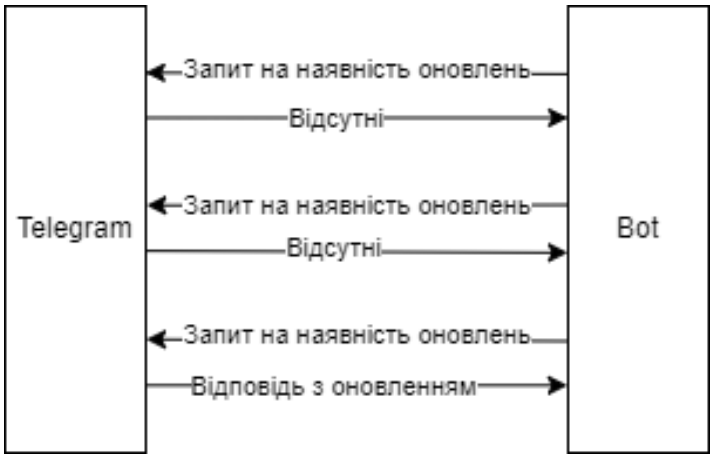

Рисунок 3.1 – Принцип роботи боту з використанням способу розробки long pooling

Використання webhook означає, що кожного разу, коли боту буде надіслано нове повідомлення, Telegram (а не сервер) візьме на себе ініціативу і надішле запит з об’єктом оновлення на сервер.

Основна перевага long pooling перед веб-хуками полягає в тому, що він простіший. Доволі менше налаштувань.

Long pooling використовує модель зв’язку, коли система витягує інформацію з іншої системи, тоді як webhook використовує модель push, передаючи

інформацію з вихідної програми до програми призначення.

Запити на long pooling виконуються клієнтом, а запити на webhook – сервером. webhook також автоматично запускається, коли відбувається подія, тоді як long pooling налаштовано на запуск через фіксовані інтервали та виконується незалежно від того, є нова подія чи ні.

Схема прикладу принципу роботи long pooling наведена на рисунку 3.2

Рисунок 3.2 – Принцип роботи боту з використанням способу розробки webhook

Long pooling може бути ресурсомістким, і необхідно вирішити чи будуть ці зусилля плідними чи ні. Це не стосується запитів webhook, які виконуються лише тоді, коли є нова інформація.

Підбиваючи підсумки, можна зробити висновок що для розробляємого програмного додатку більш доречним буде використання бота, побудованого на системі webhook.

Додаток має непостійне з’єднання з користувачем, немає необхідності періодично оновлювати інтерфейс, спілкування з користувачем відбувається один на один, поток даних не є великим, тому, без втрат в ресурсах можна використовувати модель push, щоб негайно надсилати оновлення, тому більш продуктивним рішенням буде використання саме webhook підходу.

3.9 Система автоматизації збірки

Система автоматизація збірки – це процес автоматизації створення збірки програмного забезпечення та пов’язаних з ним процесів, включаючи компіляцію вихідного коду комп’ютера в двійковий код, упаковку двійкового коду та виконання автоматизованих тестів.

Утиліти автоматизації збірки виконують завдання створення артефактів збірки. Maven і Gradle належать до цієї категорії інструментів автоматизації збірки.

І Gradle, і Maven забезпечують узгодження щодо конфігурації. Однак Maven надає дуже жорстку модель, яка робить настройку більш об’емною, а іноді й неможливою. Хоча це може полегшити розуміння будь-якої збірки Maven, якщо у немає особливих вимог, це також робить її непридатною для багатьох завдань автоматизації. Gradle, з іншого боку, створений з урахуванням уповноважених для цього та відповідальних користувачів.

Покращення часу збірки є одним із найбільш прямих способів пришвидшити доставку. І Gradle, і Maven використовують певну форму паралельної збірки проєкту та паралельного зіставлення залежностей.

Інкрементальність — Gradle уникає зайвої роботи, відстежуючи введення та вихід завдань і виконуючи лише те, що необхідно, і обробляючи лише файли, які змінилися, коли це можливо.

Кеш збірки — повторно використовує вихідні дані будь-якої іншої збірки Gradle з тими самими вхідними даними, у тому числі між машинами.

Gradle Daemon — довготривалий процес, який зберігає інформацію про збірку оновеною в пам'яті.

Ці та інші функції роблять Gradle принаймні вдвічі продуктивнішим майже для кожного сценарію в порівнянні з Maven.

Більш тривалий термін використання Maven означає, що його підтримка через IDE є кращою для багатьох користувачів. Проте підтримка IDE Gradle продовжує швидко покращуватися. Хоча IDE є важливими, велика кількість

користувачів вважають за краще виконувати операції збирання через інтерфейс командного рядка. Gradle надає сучасний інтерфейс командної команди, який має такі функції, як «завдання gradle», а також покращене ведення журналів і завершення командного рядка.

Обидві системи збірки забезпечують вбудовану можливість вирішення залежностей із репозиторіїв для конфігурування. Обидва можуть локально кешувати залежності та завантажувати їх паралельно.

За допомогою Maven є модливість легко визначити метадані та

залежності проєкту, але створення строго індивідуального складання може стати кошмаром для користувачів. Файл POM може мати величезний розмір, у міру зростання проєкту і пізніше може стати нечитаним XML-файлом.

Як споживач бібліотеки, Maven дозволяє перевизначати залежність, але тільки за версією. Gradle надає настроювані правила вибору залежностей і заміни, які можна оголосити один раз і обробляти небажані залежності в усьому проєкті. Цей механізм заміни дозволяє Gradle створювати кілька вихідних проєктів разом для створення складених збірок.

Розв’язання конфліктів залежностей Maven працює за найкоротшим шляхом, на який впливає впорядкування оголошень. Gradle виконує повне вирішення конфліктів, вибираючи найвищу версію залежності, знайдену на графіку. Крім того, за допомогою Gradle можливо оголошувати версії як strictly, що дозволяє їм мати перевагу над перехідними версіями, дозволяючи зменшити залежність.

Як виробник бібліотек, Gradle дозволяє виробникам оголошувати залежності `api` та `implementation`, щоб запобігти витоку небажаних бібліотек на шляхи класів споживачів. Maven дозволяє видавцям надавати метадані через додаткові залежності, але лише як документацію. Gradle повністю підтримує варіанти функцій і додаткові залежності.

З усього вищепереліченого можна зробити висновок, що в додатку, що розробляється, більш доцільно буде використовувати систему автоматизації збірок gradle. Він уникає зайвої роботи, відстежуючи вхідні та вихідні завдання і запускає

лише ті завдання, які були змінені. Тому це забезпечує більш швидку продуктивність а також уникає компіляції java, що пришвидшує процес збірки, який безпосередньо впливає на час розробки, а також на розгортання додатку на сервері.

Висновки до розділу 3

У розділі було проведено аналіз та вибір технологій та засобів для розробки застосунку. Всі обрані технології та засоби є сучасними, активно використову.ться розробниками та є найбільш підходящими або зручними для реалізації проєкту.

В якості мови програмування було обрано Java за її швидкість, ефективність та велику кількість зовнішніх засобів, які відкривають широкі можливості розробки.

Додаток було вирішено побудувати на фреймворці Spring Boot, за його рішення реалізації інверсії контролю, що значно пришвидшить розробку.

Для зберігання даних була обрана СУБД MySQL через її гнучкість та продуктивність. Для більш спрощених запитів та роботи з базою буде використовуватися ORM Hibernate.

В якості клієнтської частини, для реалізації застосунку обрано чат бота додатку для обміну повідомленнями Telegram та для роботи з ним Telegram Bot API. Для отримання повідомлень від телеграму технологію Webhook.

Для збірки програми в один модуль, була обрана система автоматизації збірки gradle, за його передові якості швидкості, а також через його популярність, що означає наявність всіх передових технологій.

Для контролю версій програми буде використано систему контролю версій GitHub.

4 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

4.1 Опис сутностей БД

Після виконання аналізу предметної області було визначено основні сутності та види зв’язків між ними [14].

Нижче представлено таблиці бази даних, що представляють ці сутності та зв’язки у структурованому вигляді.

Опис сутності користувач показано в таблиці 4.1.

Таблиця 4.1 – Опис сутності користувач.

Поле	Тип	Опис
id	bigint	Унікальний ідентифікатор, що є первинним ключем таблиці
username	varchar(255)	Нік в telegram
first_name	varchar(255)	Ім'я користувача
last_name	varchar(255)	Прізвище користувача
phone_number	bigint	Номер телефону
location_id	bigint	Зовнішній ідентифікатор, що посилається на таблицю Location, значення відповідають первинному ключу таблиці Location. Характеризує місцезнаходження користувача.
change_dateime	datetime(6)	Дата та час останьої зміни

		профіля
creation_dateime	datetime(6)	Дата та час створення користувача

Під час нормалізації бази даних, для приведення в третю нормальну форму, поля, що зберігають дані про країну, регіон та місто було витягнуто в окремі сутності.

Опис сутності місто показано в таблиці 4.2.

Таблиця 4.2 – Опис сутності місто

Поле	Тип	Опис
id	bigint	Унікальний ідентифікатор, що є первинним ключем таблиці
name	varchar(64)	Найменування міста

Опис сутності місто показано в таблиці 4.3.

Таблиця 4.3 – Опис сутності країна

Поле	Тип	Опис
id	bigint	Унікальний ідентифікатор, що є первинним ключем таблиці
name	varchar(64)	Найменування країни

Для зберігання сутності регіону буде створена особлива таблиця з полем main, таблиця має строки за замовчуванням, в яких атрибут main буде «1», що буде означати можливість вибору його як рідного регіону.

Опис сутності регіон показано в таблиці 4.4.

Таблиця 4.4 – Опис сутності регіон

Поле	Тип	Опис
id	bigint	Унікальний ідентифікатор, що є первинним ключем таблиці
name	varchar(64)	Найменування регіону
main	bit(1)	Логічне значення, яке відповідає за можливість обрати регіон як рідний

Опис сутності місцезнаходження показано в таблиці 4.5.

Таблиця 4.5 – Опис сутності місцезнаходження

Поле	Тип	Опис
id	bigint	Унікальний ідентифікатор, що є первинним ключем таблиці
latitude	double	Атрибут, що показує широту координат місцезнаходження.
longitude	double	Атрибут, що показує довготу координат місцезнаходження.
city_id	int	Зовнішній ідентифікатор, що посилається на таблицю City, значення відповідають первинному ключу таблиці City. Зв’язок «Один до Багатьох». Характеризує місто місцезнаходження користувача.

country_id	int	Зовнішній ідентифікатор, що посилається на таблицю Country, значення відповідають первинному ключу таблиці Country. Зв’язок «Один до Багатьох». Характеризує країну місцезнаходження користувача.
region_id	int	Зовнішній ідентифікатор, що посилається на таблицю Region, значення відповідають первинному ключу таблиці Region. Зв’язок «Один до Багатьох». Характеризує регіон місцезнаходження користувача.
native_region_id	int	Зовнішній ідентифікатор, що посилається на таблицю Region, значення відповідають первинному ключу таблиці Region. Характеризує рідний регіон місцезнаходження користувача.

Опис сутності завдання показано в таблиці 4.6.

Таблиця 4.6 – Опис сутності завдання

Поле	Тип	Опис
id	bigint	Унікальний ідентифікатор, що є первинним ключем таблиці
creation_datetime	datetime(6)	Дата та час створення користувача
description	varchar(255)	Атрибут, який зберігає опис заявки
recipient_id	bigint	Зовнішній ідентифікатор, що посилається на таблицю User, значення відповідають первинному ключу таблиці User. Характеризує особу, яка створила завдання – надала запит (реципієнт).
donor_id	bigint	Зовнішній ідентифікатор, що посилається на таблицю User, значення відповідають первинному ключу таблиці User. Характеризує особу, яка відгукнулася на завдання (донор), та виконує його.
logistician_id	bigint	Зовнішній ідентифікатор, що посилається на таблицю User, значення відповідають первинному ключу таблиці User. Характеризує особу, яка надає логістичні послуги.

status_id	smallint	Зовнішній ідентифікатор, що посилається на таблицю Task_status, значення відповідають первинному ключу таблиці Task_status. Характеризує статус завдання.

Опис сутності здібність показано в таблиці 4.7.

Таблиця 4.7 – Опис сутності стан

Поле	Тип	Опис
id	smallint	Унікальний ідентифікатор, що є первинним ключем таблиці.
name	varchar(64)	Назва стану, що характеризує її значення.

Опис сутності здібність показано в таблиці 4.8.

Таблиця 4.8 – Опис сутності здібність

Поле	Тип	Опис
id	bigint	Унікальний ідентифікатор, що є первинним ключем таблиці.
creation_datetime	datetime(6)	Дата та час створення користувача.
description	varchar(255)	Атрибут, який зберігає опис можливостей вирогідних виконавців завдань.
user_id	bigint	Зовнішній ідентифікатор, що посилається на таблицю User,

		значення відповідають первинному ключу таблиці User.

4.2 Даталогічна модель бази даних

На цьому етапі проєктування будується логічна структура бази даних бота. При цьому вихідна інфологічна модель перетворюється в модель даних, яка підтримується конкретною СУБД. Після цього перевіряється адекватність даталогічної моделі предметної області, що відображається. Кінцевим результатом даталогічного проєктування є опис структури бази даних мовою опису даних конкретної СУБД.

Процес нормалізація також був використаний під час моделювання.

Даталогічна модель являє собою модель, що відображає логічні зв'язки між елементами даних, незалежно від їх змісту та фізичної організації. Кожна конкретна СУБД має ряд обмежень на побудову логічної моделі даних, тому насамперед необхідно врахувати специфіку конкретної СУБД, а також виявити всі фактори, які можуть вплинути на логічну модель БД.

Усі сутності бази даних приведені у вигляді таблиць, атрибути – у вигляді рядків таблиць з відповідними типами даних, що зберігаються в цих рядках. Даталогічна модель орієнтована на СУБД MySQL та використовує типи даних, властиві саме цій СУБД

Даталогічна модель зображений на кресленику ІА83.220БАК.003 Д2.

4.3 Діаграма варіантів використання

За допомогою діаграми варіантів використання було описано відносини між діючими особами та варіантами використання в системі.

Діаграма варіантів використання є однією з діаграм поведінки уніфікованої

мови моделювання, яку можна використовувати для опису цілей користувачів та інших систем, які взаємодіють із системою, що моделюється.

Була використана для опису функціональних вимог системи, і представлено картину того, як система буде використовуватися. Діаграми варіантів використання підказують, що повинна робити система, не вказуючи використаних технічних методів.

Варіант використання описує функцію, яку виконує система для досягнення мети користувача. Випадок використання повинен давати спостережуваний результат, який є цінним для користувача системи. Варіанти використання малюються за допомогою овалів.

Актор представляє роль користувача, який взаємодіє з системою, яка моделюється. Користувач може бути користувачем-людиною, організацією, машиною або іншою зовнішньою системою. Щоб показати на схемі актора, потрібно намалювати людинку.

У моделях UML підсистеми є типом стереотипних компонентів, які представляють незалежні, поведінкові одиниці в системі. Підсистеми використовуються в діаграмах класів, компонентів і варіантів використання для представлення великомасштабних компонентів у системі, яка моделюється.

В UML зв’язок — це зв’язок між елементами моделі. Для зв’язків між випадками використання використовуйте стрілки з позначкою «використовує» або «розширює». Відношення «використання» вказує на те, що один варіант використання потрібен іншому для виконання завдання. Відношення «розширює» вказує на альтернативні варіанти для певного випадку використання. У діаграмі варіантів використання є довільний семантичний зв’язок між окремими елементами моделі.

- асоціації;
- включення;
- розширення;
- узагальнення.

Кожен варіант використання відноситься мінімум до одного діючого актора, має ініціатора, та призводить до відповідного результату.

Варіанти використання також взаємодіять з іншими варіантами використання.

В рамках дослідницького проєкту була розроблена діаграма варіантів використання, що наведено на кресленику ІА83.220БАК.003 Д3.

Було визначено загальні межі проєктованої системи в контексті змодельованої предметної області, уточнено вимоги до функціональної поведінки проєктованої системи у вигляді варіантів використання, логічно проаналізовано вихідну модель системи для її деталізації у формі логічних і фізичних моделей.

Першим етапом побудови моделі варіантів використання було визначення дійових осіб. Основними ролями системи виступають пересічний користувач.

Після цього було визначено варіанти використання для кожної ролі.

Від системи вимагається надавати послуги по створенню та відстеженню завдань на отримання допомоги. Для цього користувач повинен мати можливість створити, переглянути та змінити своє завдання, для цього потрібно мати меню, яке містить функції створення нового завдання, а також меню, яке містить функції відстеження та редагування активних та архівних завдань.

Користувач повинен мати можливість змінити інформацію про себе, саме для цього необхідно додати власний кабінет користувача, який містить дані про користувача, та функції їх зміни.

Для задоволення потреб користувачів, які розмістили завдання, необхідно мати функції їх знаходження волонтерами, на відгуку на них, для цього була проаналізована та змодельована систему фільтрації та сортування найбільш підходящих завдань.

Меню для волонтерів, які мать змогу допомогти, та для логістів, які зберігають в собі функції пошуку завдань за встановленими критеріями, швидкого пошуку за найімовірнішими критеріями.

4.4 Діаграма діяльності

Для розроблюваної системи також розроблена діаграма діяльності, яку зображений на кресленику ІА83.220БАК.003 Д4, вона відображає як набір варіантів використання координується для представлення робочих процесів бізнес-логіці.

Діаграма діяльності є ще однією важливою діаграмою поведінки для опису динамічних аспектів системи. Діаграма діяльності є розширеною версією діаграми варіантів використання, яка моделює перехід від однієї діяльності до іншої, та характеризує сценарії користування ботом.

На основі моделі варіантів використання було визначено безпосередньо варіанти використання системи, були визначені передмови та постумови (контекст) для всіх випадків, проаналізовано процеси між або в межах варіантів використання та змодельований загальний складний робочий процес.

На початку роботи з ботом, після його активації, першим чином користувач перевіряється чи не заблокований він в системі, якщо ні, то перевіряється наявність його облікового запису, якщо запис відсутній, користувач повинен пройти через процедуру реєстрації, ввести особисті данні.

Одразу після реєстрації користувачеві надається можливість створити нову заявку на розміщення оголошення про необхідну допомогу, та пропонується ввести дані про професію, або будь-що інше, чим людина може бути корисна спільноті. Ці кроки можна пропустити, та ввести ці данні пізніше, під час редагування особистого профілю користувача.

Після цього користувач потрапляє на головне меню, за допомогою якого, користувач може змінити інформацію про себе, створити нове оголошення про допомогу, переглянути список найбільш йому відповідних оголошень, відгукнутися на один чи кілька з них, перевірити або змінити стан його завдань, що знаходяться в роботі, переглянути загальну корисну інформацію.

4.5 Телеграм боти

Перед тим як розробляти основний функціонал, треба проаналізувати, як працюють Telegram боти.

Повідомлення, команди та запити, надіслані користувачами, через анонімний проксі сервер Telegram шифруються та передаються в програмне забезпечення, яке працює на сервері. Телеграм забезпечує зв'язок між програмною частиною застосунка та користувачем.

Алгоритм роботи між ботом та користувачем, має вигляд: до бота надходить команда від користувача –> команда направляється ботом на сервер –> написана програма обробляє запит –> від сервера приходить відповідь –> відповідь відображається користувачеві.

Цей цикл повторюється кожен раз, коли користувач натискає кнопки або пише команди, які починаються з символу "/". У кожного бота може бути унікальний набір команд, це все залежить від розробника. Існує декілька базових команд для ботів, по типу “start”, ця команда розпочинає роботу бота при першому запуску та “help” ця команда визиває текст в якому прописана інструкція користування ботом, та перелік усіх команд з описом.

Для взаємодії з месенджером найкраще за все використовувати Bot API. Це інтерфейс для розробників які створюють ботів у Telegram, в основі лежить HTTP. За допомогою API можна отримати доступ до більшості функцій, які можна оброблять в програмному застосунку. За допомогою бібліотеки TelegramBots, яка створена для Java, використовується Telegram bot API.

4.6 Створення бота

Мессенджер Telegram один з самих популярних платформа для створення ботів. Telegram має дуже зручне API та зрозумілу документацію. Телеграм абсолютно безкоштовний, також розширення його функціоналу за рахунок чат-

ботів безкоштовне та для використання API непотрібна жодна ліцензія чи дозвіл.

Для початку створення боту необхідно його зареєструвати в Telegram та отримати унікальний токен. Цей токен, може отримати кожний користувач месенджеру, для цього потрібно знайти у відповідний за їх створення Telegram бот, доступний за посиланням https://t.me/BotFather та дотримуючись інструкцій створити бота. На рисунку 4.1 продемонстровано створення боту.

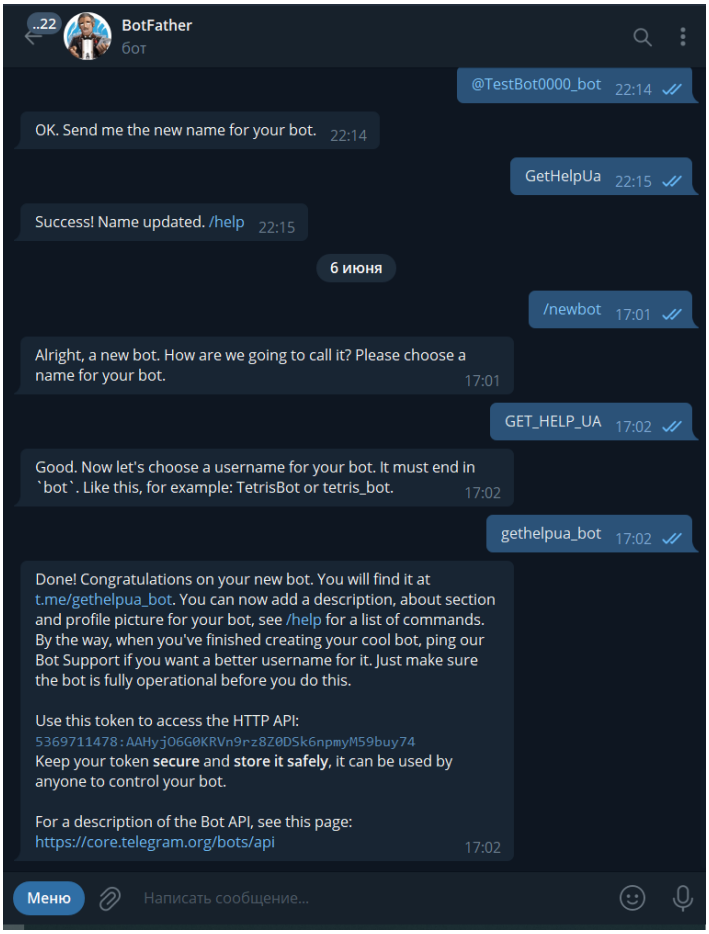

Рисунок 4.1 – Реєстрація бота

Було визначено унікальна мнемонічна назва боту, після чого було отримано токен – «ключем» для доступа до створеного боту. Токен використовується для підключення програмної частини до бота.

Наступним кроком було задання назви бота. Після цього обрана назва буде доступна для пошуку.

На рисунку 4.2 представлений бот для пошуку допомогу українцям GET HELP UA, на його прикладі розглянемо з чого складаються ітерфейс та базові налаштування боту.

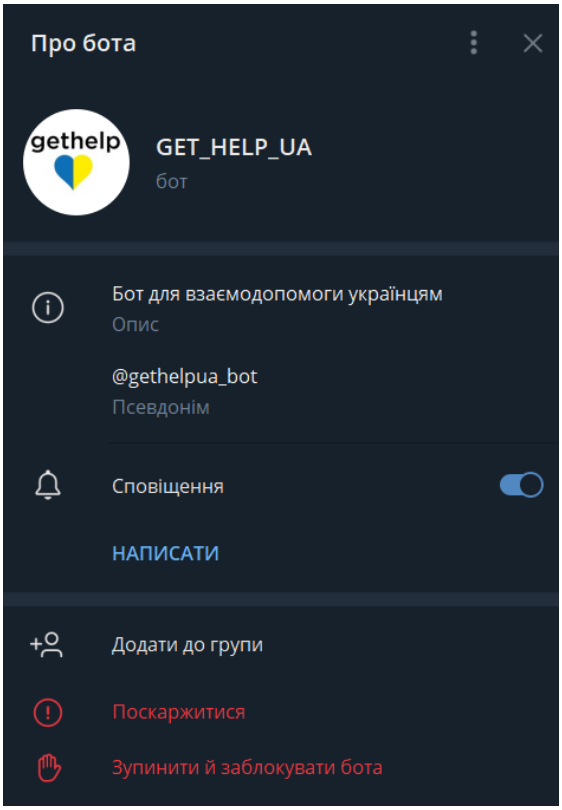

Рисунок 4.2 – Інтерфейс налаштувань створеного бота

Власно створений бот можна налаштувати через BotFather: меню /mybots →Edit Bot.

В налаштуваннях можна змінити:

- ім'я створюваного бота;

- опис (Description) – це привітний текст, який користувачі бачать на початку діалогу з ботом під заголовком "Для чого цей бот?";

- інформація (About) – це текст, який буде видно в профілі бота;

- графічне представлення боту. Це картинка, яку бачать користувачі коли шукають бота або відкривають профіль;

- команди – тут маються на увазі підказки команд в боті.

4.5 Робота з ботом

Окрім команд, заданих розробником, бот повинен надавати форму, після якої користувачу потрібно буде ввести якусь інформацію, наприклад ім’я. Але найпопулярнішим видом взаємодії є кнопки. Telegram бот використовує декілька їх видів.

- звичайні кнопки, які з’являються замість клавіатури;
- кнопки під повідомленнями (inline keyboards або inline buttons).

Користувач при натисканні кнопки відправляє боту текст, який на ній написано.

Для обробки повідомлення від користувача система використовує методи API, наведені в таблиці 4.8.

Таблиця 4.8 – Методи API для робота з повідомленнями Telegram

Назва методу	Опис
messages.getMessages	Повертає список повідомлень за їхніми ідентифікаторами.
messages.sendMessage	Надсилає повідомлення в чат
messages.editMessage	Використовується для редагувати повідомлення.
messages.deleteMessages	Видаляє повідомлення за їх ідентифікаторами.

4.6 Налаштування залежностей gradle

Для початку роботи над додатком необхідно створити проєкт в IntelliJ IDEA та настроїти всі залежності gradle. На рисунку 4.3 наведено фрагмент коду файлу build.gradle.

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'mysql:mysql-connector-java'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    implementation group: 'org.telegram', name: 'telegrambots-spring-boot-starter', version: '5.7.1'
}
```

Рисунок 4.3 – Фрагмент коду файлу build.gradle

Таблиця 4.9 – Опис залежностей build.gradle

Залежність	Опис
org.springframework.boot:spring-boot-starter-data-jpa	Для того щоб Spring Boot мав можливість налаштовувати та комунікувати з базою даних за допомогою технології ORM [15], була додана залежність org.springframework.boot:spring-boot-starter-data-jpa. Spring Boot використовує залежність spring-boot-starter-data-jpa для ефективного з'єднання програми Spring з реляційною базою даних.
org.springframework.boot:spring-boot-starter-validation	Підтримує перевірку введених користувачем даних. Перевірка компонентів працює, визначаючи обмеження для полів класу.
org.springframework.boot:spring-boot-starter-web	Стартер Spring Web використовує Spring MVC, REST і Tomcat як вбудований сервер за замовчуванням. Одна залежність spring-boot-starter-web транзитивно залучає всі залежності, пов'язані з веб-розробкою.

org.projectlombok:lombok	Додає lombok — плагін для скорочення коду в класах та розширення функціональності мови Java. Lombok перетворює інструкції у вихідному коді Java-оператори до того, як компілятор їх обробить.
mysql:mysql-connector-java	Додає конектор MySQL — міст між сервером MySQL і програмою, написану мовою програмування Java. Коннектор — це частина програмного забезпечення, яка забезпечує реалізації API та пропонує інтерфейс для виконання запиту MySQL на сервері.
org.springframework.boot:spring-boot-starter-test	Імпортує тестові модулі Spring Boot, а також має JUnit, AssertJ, Hamcrest та ряд інших корисних бібліотек.
telegrambots-spring-boot-starter	Надає Spring Boot адаптацію Telegram Bot API. Бібліотека Java для створення ботів за допомогою Telegram Bots API.

4.8 Ініціалізація застосунку

Застосунок побудований на програмній платформі Spring, що визначає структуру програмної системи. Spring допомагає розробникам створювати високопродуктивні програми за допомогою.

Налаштувати, конфігурувати програму, за допомогою Spring можна декількома способами:

- XML конфігурація;
- Java конфігурація;
- Конфігурація за допомогою анотацій.

В даному програмному додатку було використано конфігурацію за допомогою Java та анотацій, тому що це є найбільш сучасним способом та спрощує візуальне розуміння коду.

Щоб Spring Framework створював екземпляри об’єктів і заповнював залежності, необхідно вказати Spring, якими об’єктами керувати і які залежності є для кожного класу. Для цього використовуються анотації. Основну частину анотацій описано в таблиці 1.1.

Таблиця 4.10 – Опис анотацій Spring

Анотація	Опис
@SpringBootApplication	Використовується в класі програми під час налаштування проєкту Spring Boot. Клас, анотований @SpringBootApplication, повинен зберігатися в базовому пакеті. Анотація сканує компоненти.
@Configuration	Використовується для класів, які визначають bean. Клас Java, анотований @Configuration, повинен використовуватись для створення екземплярів та налаштування залежностей.
@Component	Використовується в класах для позначення компонента Spring. Анотація @Component позначає клас Java як компонент bean або say, щоб механізм сканування компонентів Spring міг додатися до контексту програми.

@RestController	Позначає клас як контролер, де кожен метод повертає об’єкт домену замість представлення.
@Service	Позначає клас Java, який виконує певні служби, наприклад, виконує бізнес-логіку, виконує обчислення та викликає зовнішні API.
@Repository	Використовується для класів Java, які мають прямий доступ до бази даних. Використовується як маркер для будь-якого класу, який виконує роль репозиторію або об’єкта доступу до даних.
@Bean	Використовується на рівні методу. Працює з @Configuration для створення Spring bean. @Configuration матиме методи для створення екземплярів та налаштування залежностей. Такі методи будуть анотовані @Bean та будуть створювати та повертати bean.
@Autowired	Застосовується до полів, методів встановлення та конструкторів. Анотація @Autowired неявно впроваджує залежність об’єкта.
@Value	Вказує вираз значення за замовчуванням для поля або параметра для ініціалізації властивості.

@RequestMapping	Використовується як на рівні класу, так і на рівні методу. Анотація @RequestMapping використовується для відображення веб-запитів на певні класи і методи обробників.
@Query	Використовується для визначення SQL для виконання методу репозиторію Spring Data, @Query — його атрибут значення містить JPQL або SQL для виконання.

Першим етапом створення програмного застосунку, використовуючи Spring Boot є створення класу

GetHelpUaBotApplication. Даний клас має метод main, який є точкою входу у застосунок. Побачити код класу можна на рисунку 4.4.

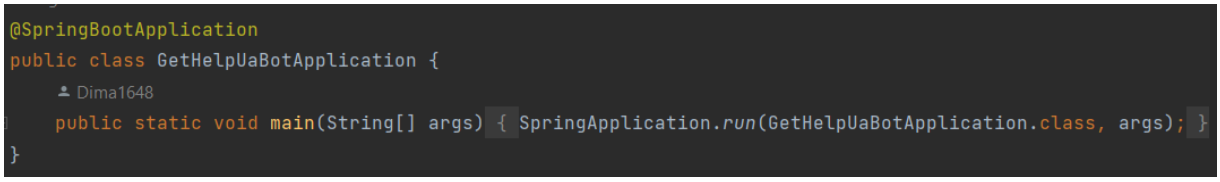

Рисунок 4.4 – Код класу GetHelpUaBotApplication

Анотація SpringBootApplication вказує, що програма буде запускатися як Spring Boot додаток. Вона також показує спрингу, що для всього що є у файлі залежностей pom.xml необхідно підключити автоматичну конфігурацію.

Наступний етап це створення класу конфігурації. Spring автоматично запускає всі класи, створені з анотацією Configuration, після старту застосунку, та автоматично створює об’єкти, так звані Beans, які було описано в класі конфігурації. Побачити код класу можна на рисунку 4.5.

У Spring об’єкти, які утворюють основу додатка і якими керує контейнер

Spring IoC, називаються bean-компонентами. Бін — це об’єкт, який створюється, збирається та іншим чином керується контейнером Spring IoC.

```java
@Configuration
public class AppConfig {
    4 usages
    private BotConfig botConfig;

    Dima1648
    public AppConfig(BotConfig botConfig) { this.botConfig = botConfig; }

    Dima1648
    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) { return builder.build(); }

    Dima1648
    @Bean
    public MessageSource messageSource() {
        ReloadableResourceBundleMessageSource messageSource
                = new ReloadableResourceBundleMessageSource();
        messageSource.setBasename("classpath:messages");
        messageSource.setDefaultEncoding("UTF-8");
        return messageSource;
    }

    Dima1648
    @Bean
    public Bot getHelpUaBot(TelegramFacade telegramFacade){
        Bot bot = new Bot(telegramFacade);
        bot.setWebHookPath(botConfig.getWebHookPath());
        bot.setBotUsername(botConfig.getUsername());
        bot.setBotToken(botConfig.getToken());
        return bot;
    }
}
```

Рисунок 4.5 – Код класу AppConfig

Для отримання HTTPS запитів від Telegram використовується спосіб зворотних запитів – webhook [16].

Для реалізації цього метода в файлі конфігурації ініціалізований синхронний клієнт для виконання запитів HTTP – RestTemplate.

Для взаємодії з REST необхідно створити екземпляр клієнта та об'єкт запиту, виконати запит, інтерпретувати відповідь, зіставити відповідь з об'єктами домену,

а також опрацювати помилки. Для середовища Spring характерне як створення API, так і використання внутрішніх чи зовнішніх API-интерфейсов застосунку. Ця перевага також допомагає у розробці мікросервісів.

Щоб уникнути шаблонного коду, Spring надає зручний спосіб використання REST API через «RestTemplate». Використання REST API представлено на рисунку 4.6.

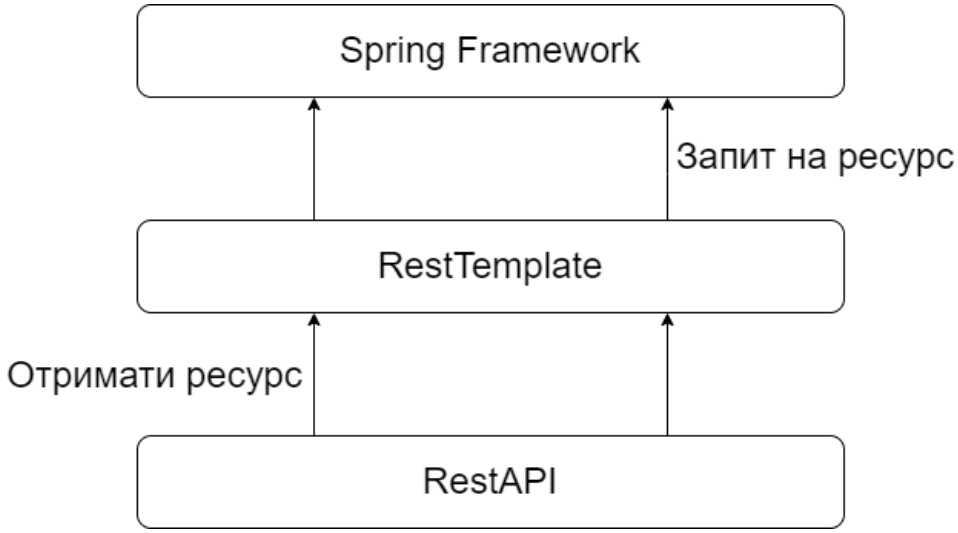

Рисунок 4.6 – Схема Використання REST API

Завершальним етапом створення конфігурації буде опис біна безпосередньо бота, де указано його назва, токен, які було введено та отримано за допомогою BotFather. Вхідним параметром буде об’єкт TelegramFacade, який буде грати роль контексту.

4.9 Обробка HTTPS запиту

У підході Spring до створення веб-сервісів REST, HTTP-запити обробляються контролером. За обробку запитів, що надходять з TelegramAPI, відповідає клас WebHookController. Код цього класу представлено на рисунку 4.7.

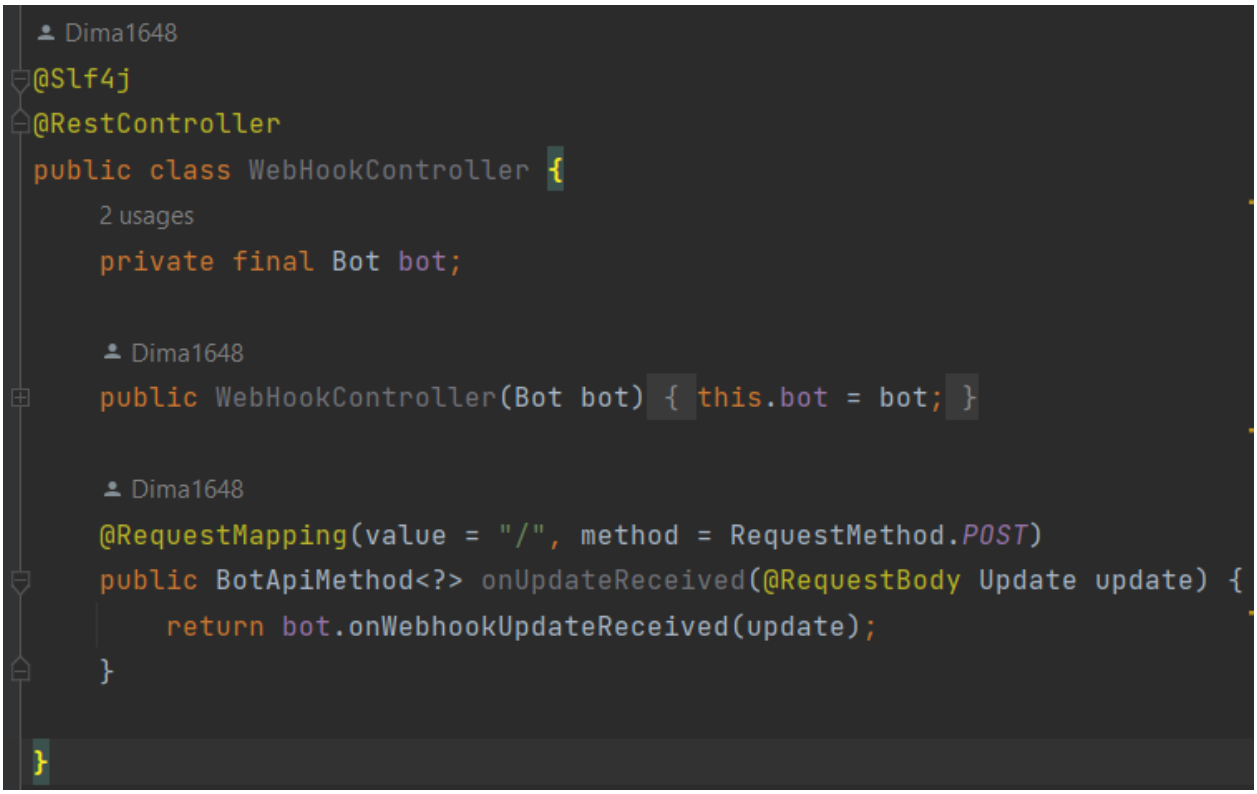

Рисунок 4.7 – Код класу WebHookController

Даний клас містить анотацію @RestController

@RestController є зручною анотацією для створення контролерів Restful. Це специфікація @Component та автоматично виявляється за допомогою сканування шляху до класу. Ця анотація додає анотації @Controller та @ResponseBody, а також перетворює відповідь у JSON. @RestController не працює з технологією представлення, тому методи не можуть повернути ModelAndView. Саме тому використовується в поєднанні з анотованим методом обробника, з @RequestMapping анотацією.

Анотація @RequestBody створена для методів-обробників в контролерах Spring. Ця анотація вказує, що Spring має десеріалізувати тіло запиту в об’єкт. Цей об’єкт передається як параметр методу обробника.

Об’єкт оновлення оброблюється в класі бота, фрагмент коду класу представлено на рисунку 4.8.

```java
public class Bot extends TelegramWebhookBot {
    1 usage
    String webHookPath;
    String botUsername;
    String botToken;

    2 usages
    private TelegramFacade telegramFacade;

    1 usage  Dima1648
    public Bot(TelegramFacade telegramFacade) { this.telegramFacade = telegramFacade; }

    2 usages  Dima1648
    @Override
        public BotApiMethod<?> onWebhookUpdateReceived(Update update) {
        telegramFacade.handleUpdate(update);
        return null;
    }
```

Рисунок 4.8 – Код класа Bot

В методі onWebhookUpdateReceived відбувається обробка об’єкту оновлення. Для цього використовується клас TelegramFacade. На рисунку 4.9 наведено фрагмент коду класу, який відповідає за обробку.

```java
1 usage  Dima1648 *
public void handleUpdate(Update update) {
    try {
        Long userid = -1L;
        if (update.hasCallbackQuery()) {
            log.info("New callbackQuery from User: {} with data: {}",
                    update.getCallbackQuery().getFrom().getUserName(),
                    update.getCallbackQuery().getData());
            userid = getUserIdFromCallback(update);
            userDataCache.setCurrentUserId(userid);
            callbackQueryFacade.processCallbackQuery(update.getCallbackQuery());
        }
        Message message = update.getMessage();
        if (message != null && message.hasText()) {
            log.info("New message from User:{}, chatId: {},  with text: {}",
                    message.getFrom().getUserName(), message.getChatId(), message.getText());
            userid = getUserIdFromMessage(update);
            userDataCache.setCurrentUserId(userid);
            handleInputMessage(message);
        }
        if (userid == -1) {
            throw new ApplicationStartException("User Id could not be obtained while receiving the update");
        }
        userDataCache.setCurrentUserId(userid);
    } catch (Exception e) {
        log.error(e.getMessage());
    }
}
```

Рисунок 4.9 – метод класу TelegramFacade, який оброблює об’єкт оновлення

Метод handleUpdate, класу TelegramFacade приймає оновлення та, в залежності від його типу, змінює стан користувача, активує процес пошуку відповіді та зміну ситуації в меню бота.

Для реалізації можливості користувача, мати особистий досвід взаємодії з ботом, щоб система мала індивідуальний алгоритм дій для кожного користувача, бізнес логіку організовано за допомогою принципу «Кінцевий автомат».

4.10 Моделі

Для зберігання інформації про користувачів та про завдання в цьому проєкті використовується СУБД MySQL.

Для створення бази даних, таблиць та для роботи з нею використовується технологія ORM (англ. Object-relational mapping, об'єктно-реляційного відображення).

ORM в даному програмному застосунку пов'язує таблиці бази даних з сутностями, представленими за допомогою концепції ООП. В ООП представлені об'єкти, які відповідають реально існуючим, ORM використовується для перетворення цих об'єктів у форму, доцільну для збереження в базі даних, а також для подальшого вилучення за допомогою програмної частини з бази даних. За допомогою ORM буде замінено часто повторювані запити до БД на класи Java, що дасть можливість зосередитись на ООП. Але можливість написання SQL щапитів залишається.

Для реалізації ORM було використано:

- API, що реалізує базові операції (СТВОРЕННЯ, ЧИТАННЯ, ЗМІНА, ВИДАЛЕННЯ) об'єктів-моделей;

- засоби налаштування метаданих зв'язування;

- техніку взаємодії з транзакціями, що дозволяє реалізувати такі функції, як dirty checking, lazy association fetching тощо.

Hibernate одна з найбільш популярних відкритих реалізації останньої версії

специфікації (JPA 2.1). Навіть сама популярна, майже стандарт де-факто. Якщо JPA тільки описує правила та API, Hibernate реалізує ці описи, щоб у Hibernate (як і в багатьох інших реалізації JPA) є додаткові можливості, які не описані в JPA (і не переносяться на інші реалізації JPA).

Для початку потрібно визначити сутності, поля та первинні ключі в базі даних.

Щоб визначити сутність, необхідно створити клас, який анотується анотацією @Entity. Анотація @Entity – це анотація-маркер, яка використовується для визначення постійних об'єктів. Наприклад, якщо потрібно створити сутність користувача, необхідно призначити її таким чином, як зображений на рисунку 4.10

```
@Entity
public class User {
    @Id
    private Long id;
    private String username;
    private String firstName;
    private String lastName;
    private long phoneNumber;
    @OneToMany(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id",
            referencedColumnName = "id",
            nullable = false)
    private List<Location> locations;
    @OneToMany(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", referencedColumnName = "id")
    private List<Possibility> possibilities;
    @OneToMany(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", referencedColumnName = "id")
    private List<Task> tasks;
    private Instant creationDateime;
    private Instant changeDateime;
}
```

Рисунок 4.10 – Код класу сутності користувача

За замовчуванням ця сутність буде зіставлена з таблицею User, як визначається ім’ям класу.

Коли сутність відображено в таблиці, наступне завдання – визначити її поля. Поля визначаються як змінні-члени класу, причому ім’я кожного поля співставляється з назвою стовпця в таблиці. Для зміни відображення за замовчуванням, використовується анотація @Column.

Однією з вимог до таблиці реляційної бази даних є те, що вона повинна містити первинний ключ або ключ, який однозначно ідентифікує певний рядок у базі даних. У JPA для визначення поля як первинного ключа таблиці використовується анотація @Id. Первинний ключ має бути примітивним типом Java, примітивною обгорткою, наприклад Integer або Long, String, Date, BigInteger або BigDecimal.

Для користувачів, сутностей місцезнаходження, завдань та вмінь використовується тип bigint, для більш обмежених по кількості сутностей – smallint.

Тепер, коли було визначено сутність, розглядаються зв’язки між сутностями. JPA визначає чотири анотації для визначення сутностей:

@OneToOne

@OneToMany

@ManyToOne

@ManyToMany

Анотації @OneToMany та @ManyToOne полегшують обидві сторони одних тих самих відносин. Розглянемо, користувач може мати лише одне місцезнаходження, але одне й те саме місцезнаходження може відноситись до багатьох користувачів. Або користувач може мати багато здібностей для допомоги а також багато завдань.

Для зберігання інформації про місцезнаходження користувача було створено сутність місцезнаходження. Даний клас містить інформацію про країну, регіон та місто користувача, рідний регіон, а також його точні координати. Побачити

реалізацію сутності місцезнаходження можна в класі Location, на рисунку 4.11

```
@Entity
public class Location{
    @Id
    @Column(name = "id", nullable = false)
    private Long id;
    @ManyToOne(fetch = FetchType.EAGER, cascade=CascadeType.ALL)
    @JoinColumn(name="country_id", nullable = false)
    private Country country;
    @ManyToOne(fetch = FetchType.EAGER, cascade=CascadeType.ALL)
    @JoinColumn(name="region_id", nullable = false)
    private Region region;
    @ManyToOne(fetch = FetchType.EAGER, cascade=CascadeType.ALL)
    @JoinColumn(name="city_id", nullable = false)
    private City city;
    @ManyToOne(fetch = FetchType.EAGER, cascade=CascadeType.ALL)
    @JoinColumn(name="native_region_id", nullable = false)
    private Region nativeRegion;
    private Double longitude;
    private Double latitude;

}
```

Рисунок 4.11 – Код класу сутності місцезнаходження

Даний клас наочно показує використання зв’язку багато до одного.

4.11 Шаблон проєктування «Стан»

Для реалізації шаблону «Стан» використовуємо методику кінцевий автомат – математичну абстракцію, що використовується для розробки алгоритмів.

Кінцевий автомат зчитуватиме ряд вхідних даних. Коли він зчитує введення, він переходить в інший стан. Кожен стан вказує, який стан слід переключитися для цього входу.

Для реалізації кінцевого автомату використовується шаблон проєктування Стан [17].

Основна ідея шаблону полягає в тому, щоб дозволити об'єкту змінювати свою поведінку, не змінюючи його клас. Крім того, завдяки його реалізації код повинен залишатися чистішим без багатьох операторів якщо/інакше.

Об’єкти можуть перебувати лише в одному стані в даний момент часу. Кожен стан є статусом системи, який змінюється на інший стан. Ці зміни стану називаються переходами.

Шаблон стану є одним із добре відомих двадцяти трьох шаблонів проєктування GoF. Цей шаблон запозичує концепцію з моделі в математиці. Завдяки шаблону проєктування State можливо інкапсулювати логіку у виділені класи, застосовувати принцип єдиної відповідальності та принцип відкритого/закритого, мати чистіший та зручніший код. Є можливість запрограмувати перехід між станами і пізніше визначити окремі стани.

На рисунку 4.12 видно, що клас Context має асоційований стан, який буде змінюватися під час виконання програми.

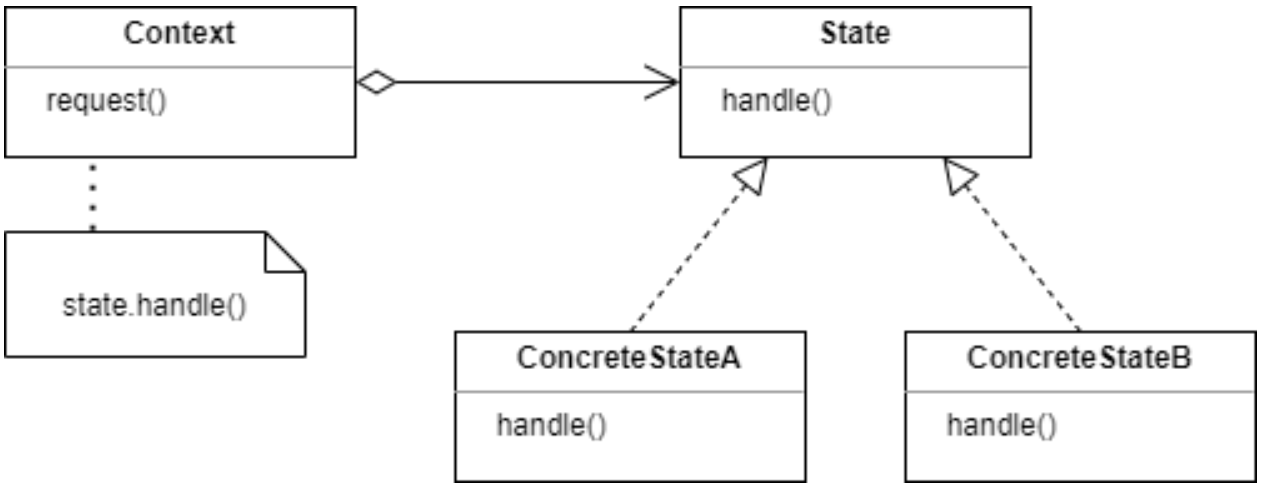

Рисунок 4.12 – UML діаграма шаблону State

Контекст буде делегувати поведінку реалізації стану. Тобто всі вхідні запити будуть оброблятися конкретною реалізацією стану.

Логіка розділена, а додавання нових станів просте та зводиться до додавання іншої реалізації стану, якщо це необхідно.

Першим етапом реалізації шаблону State буде визначення контексту, це буде клас TelegramFacade. На рисунку 4.13 представлено метод, handleInputMassage, який приймає атрибут Massage, який, свою чергу, несе в собі запит користувача, це може бути звичайне текстове повідомлення, інформація про це повідомлення: дата відправки, ідентифікаційний номер відправника, ідентифікаційний номер чату, тощо, або фото, відео, аудіофайл, голосове повідомлення, тощо. Цей метод обробляє стан користувача, що зберігається в класі UserDataCache, інтерфейс показаний на рисунку 4.14. В кожному стані користувач робить той чи інше вибір, надана інформація якого оброблюється в конкретній реалізації стану, змінює стан та записує його в кеш, і при наступному кроці, наступне повідомлення буде оброблюватись вже в іншому стані.

```
71      private void handleInputMessage(Message message) {
72          String inputMsg = message.getText();
73          Long userId = message.getFrom().getId();
74          BotState botState;
75
76          switch (inputMsg) {
77              case "/start":
78                  botState = BotState.SIGN_UP;
79                  break;
80              default:
81                  botState = userDataCache.getUserCurrentBotState(userId);
82                  break;
83          }
84
85          userDataCache.setUserCurrentBotState(userId, botState);
86
87          botStateContext.processInputMessage(botState, message);
88      }
```

Рисунок 4.13 – метод класу TelegramFacade, який керує станами користувача

```
public interface DataCache {

    5 usages  1 implementation  Dima1648
    void setUserCurrentBotState(Long userId, BotState botState);

    1 usage  1 implementation  Dima1648
    BotState getUserCurrentBotState(Long userId);

    3 usages  1 implementation  Dima1648
    Long getCurrentUserId();

    3 usages  1 implementation  Dima1648
    void setCurrentUserId(Long userid);

}
```

Рисунок 4.14 – Інтерфейс класу UserDataCache

Інтерфейс, який описує поведінку станів представлений на рисунку 4.15

```
public interface InputMessageHandler {
    1 usage  3 implementations  Dima1648
    void handle(Message message);

    1 usage  3 implementations  Dima1648
    BotState getHandlerName();
}
```

Рисунок 4.15 – Інтерфейс стану

Кожен стан відповідає поточній ситуації в чаті, набору кнопок, та чекає від користувача тієї чи іншої інформації. Якщо користувач надає інформацію не за регламентом, сервер бачить це, та видає коригувальне повідомлення. На прикладі початкового стану додатку розглянемо структуру реалізації інтерфейсу стану. На рисунку 4.16 наведено реалізацію початкового стану реєстрації.

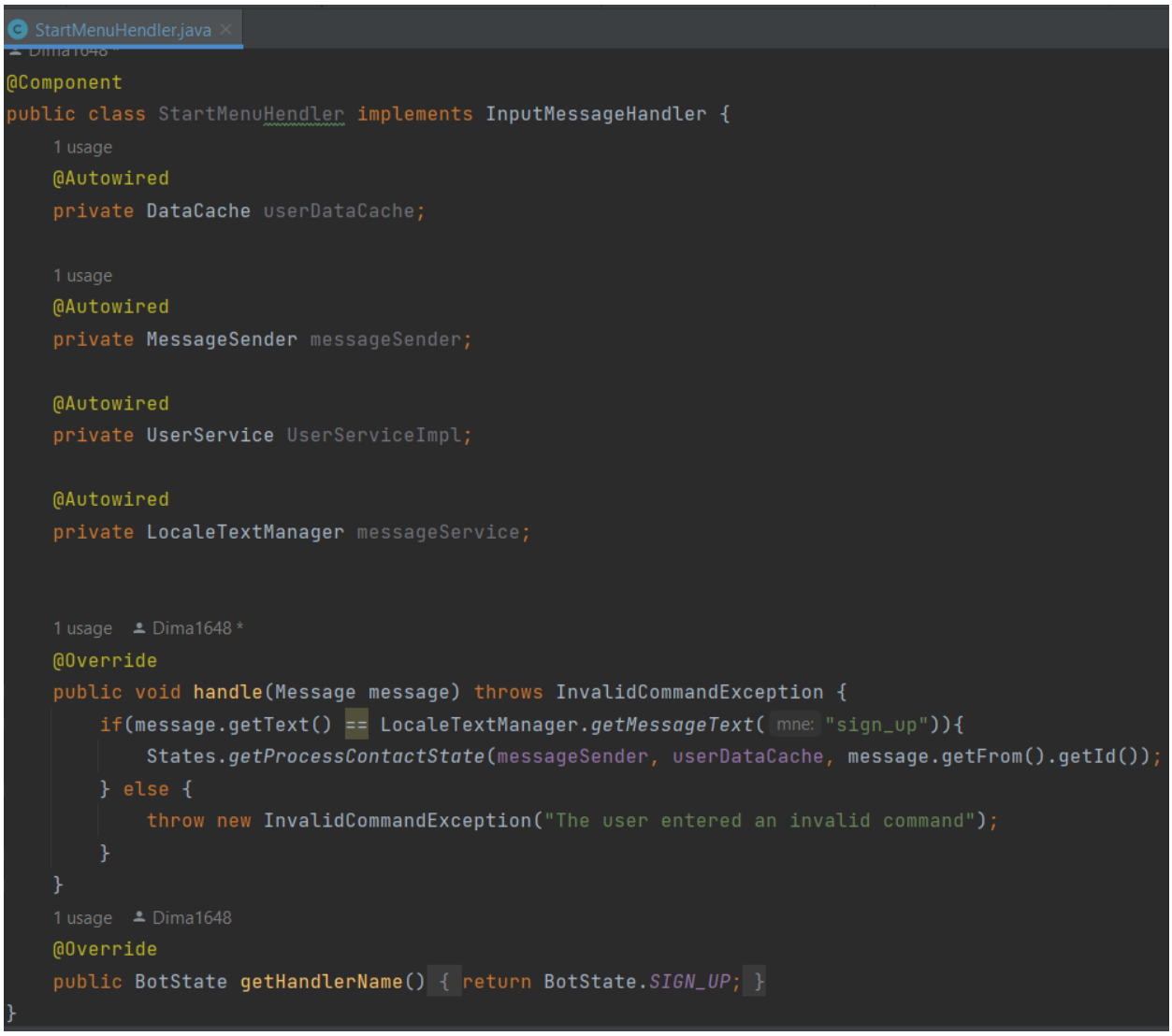

Рисунок 4.16 – Реалізація стану реєстрації

Реалізація методу handle як параметр приймає об’єкт повідомлення, та оброблює його. На даному етапі користувач має лише одну кнопку «Зареєструватися», при натисканні на цю кнопку, генерується зовнішній вигляд наступного стану, користувачу відправляється відповідне повідомлення та формуються кнопки, які будуть оброблені в наступному стані.

Для ідентифікації стану було створено метод getHandleName, за допомогою якого контекст ідентифікує цей стан. Наявність цього метода дає свободу в масштабуванні та зміні програмного додатку, тому що стан не має жорсткої прив’язки до класу.

Повний код модулів прийому та обробки повідомлень системою, та обробки, керування станами користувача без бізнес логіки наведений в додатку А.

Висновки до розділу 4

В розділі було описано проєктування та реалізацію застосунку по зазначених у завданні критеріях. Було описано ключові підходи і шаблони розробки вебзастосунку.

Проведено моделювання бази даних, побудовано даталогічну модель, на основі якої, за допомогою ORM технології було побудовано базу даних в СУБД MySQL. За допомогою ORM було архітектурно розділено області відповідальності об’єктів та класів додатка. За допомогою Hibernate було зв’язано базу даних з концепцією об’єктно орієнтованого простору Java.

Обравши Java в якості мови програмування було отримано зручний та повноцінний комплект для розробки JDK, середовище виконання Java, яке містить інструменти IntelliJ IDEA, призначені для редагування, віртуальну машину Java та бібліотеку класів Java, які дуже корисні при поширенні програмного забезпечення.

За допомогою Java та реалізації ORM у вигляді Hibernate, було створено сутності програми.

За допомогою Spring було реалізовано шаблон проєктування ін’єкція залежностей, шаблон, який дозволив створити більш роз’єднані архітектури, що дає можливість у майбутньому масштабувати систему.

Для прийняття http запитів від телеграму було використано механізм оповіщення системи про події webhook.

Для реалізації бізнес логіки було використано шаблон проєктування «Стан».

Для його реалізації було створено контекст, який керує станами користувача в системі, клас кешу, який зберігає поточну інформацію та надає контексту інформацію про стан, а також класи станів, в яких реалізована бізнес логіка додатку.

5 ТЕСТУВАННЯ СИСТЕМИ

Для тестування було обрано ручне тестування додатку.

Ручне тестування – це пряма взаємодія з програмою. У процесі можна отримати зворотний зв'язок про продукт.

Наприкінці кожного етапу розробки аналізуються та пишуться тест-кейси та на їх основі проводиться тестування працездатності.

Критерії оцінки якості бота:

- коректність переходів між станами в залежності від команди, введеної чи вибраної користувачем;
- збереження історії переходів при використанні команди «Назад», та коректне повернення до кожного пункту меню в ланцюзі переходів;
- в усіх станах при невірно введеній команді отримання користувачем повідомлення про невірно введену команду;
- після будь-якої дії користувача присутній відгук бота;
- швидкість надання відповіді після вводу команди;
- коректна та своєчасна робота системи повідомлень;
- відсутність помилок правопису та форматування відповідей бота.

Тестування боту проводилося на мобільному пристрою операційної системи iOS, з версією Telegram 8.7.1

На початку користування ботом, після його активації, користувач може розпочати роботи з ботом використавши команду /start. Після відправки цієї команди, користувач отримує перше повідомлення, яке містить інформацію про бота (рисунок 5.1).

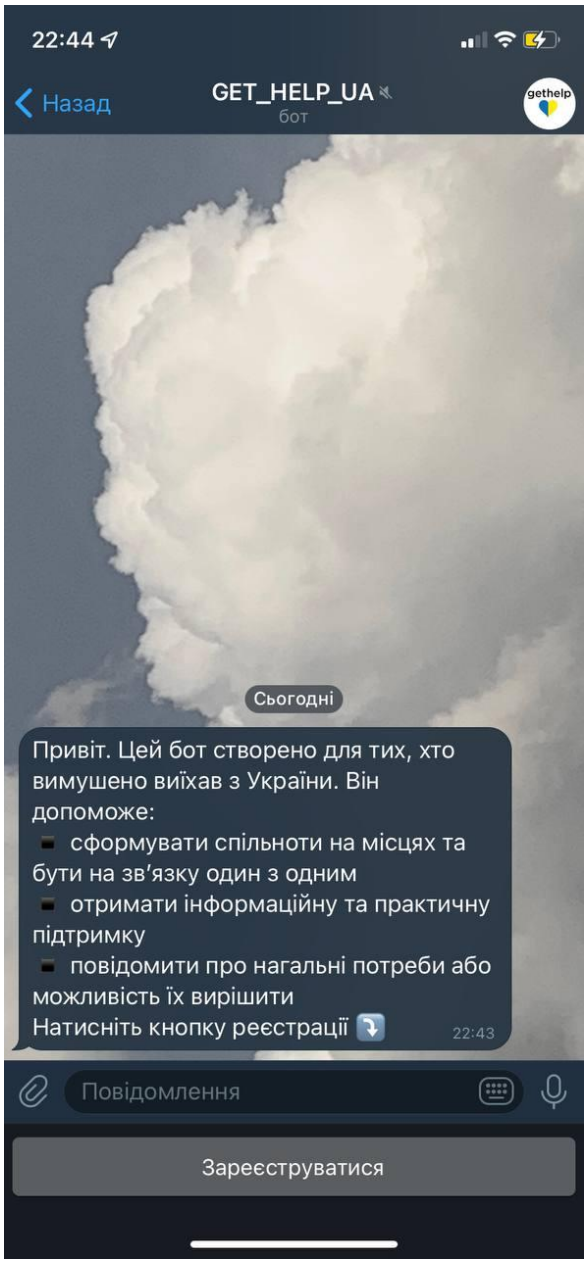

Рисунок 5.1 – Початок роботи з ботом

Після успішної активації бота повинна з'явитися кнопка, за допомогою якої можливо надіслати данні свого контакту, натиснувши кнопку «Поділитися контактом». З міркувань безпеки бот просить дозволу на отримання контактної інформації користувача (рисунок 5.2).

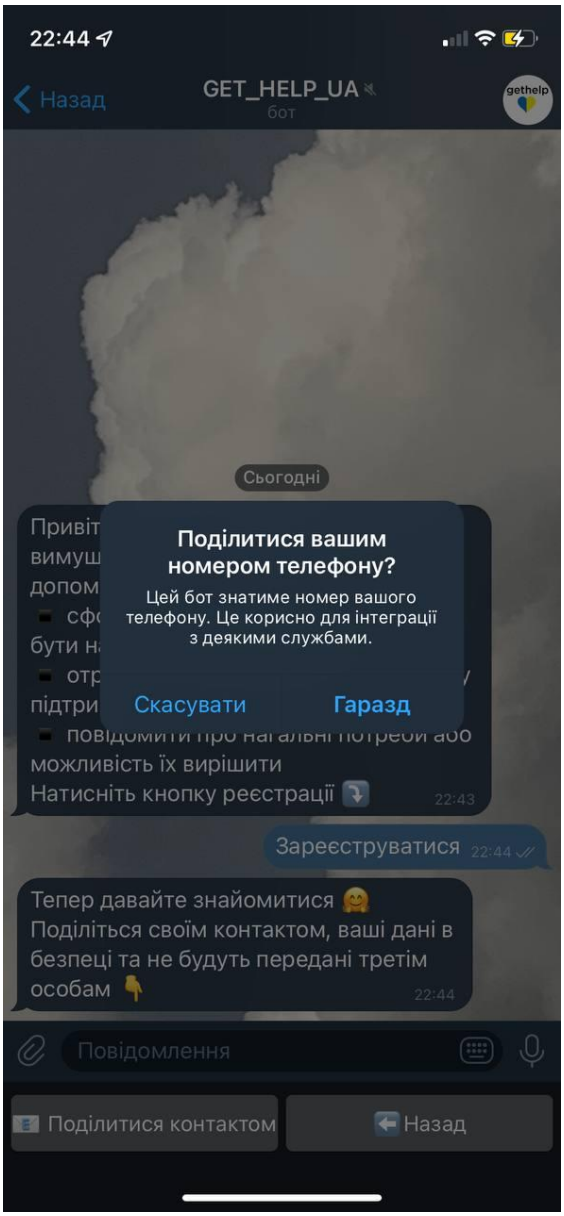

Рисунок 5.2 – Запит на отримання доступу до контакту

Відповіддю повинно бути частково заповнений профіль користувача, що буде означати успішну обробку контакту за запис в базу даних, також повинна з’явитися можливість надіслати своє місцезнаходження, користувач може сформувати запит на надіслання свого поточного місцезнаходження за допомогою кнопки «Поділитися локацією». Бот повинен запитати дозволу на отримання інформації про місцезнаходження користувача. Коректну роботу бота представлено на рисунку 5.3.

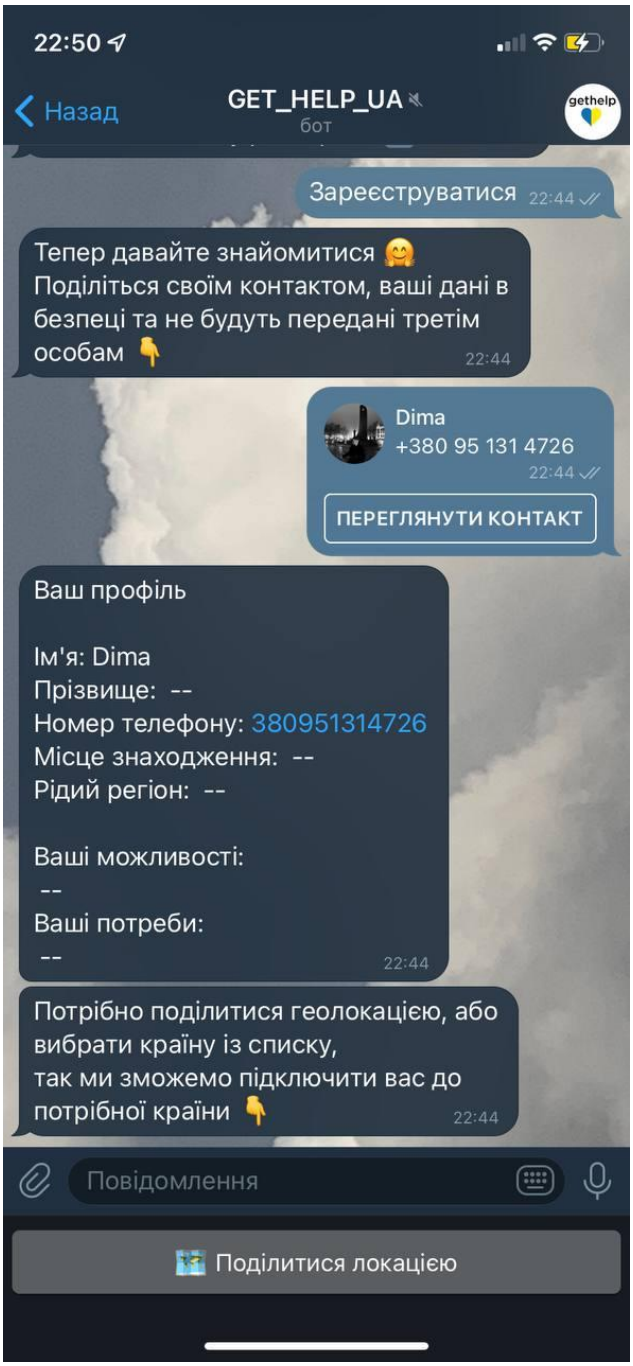

Рисунок 5.3 – Відповідь на наданий користувачем контакт

Після надання системі даних про геолокацію, система повинна спитати рідний регіон користувача, та надати список регіонів України. Побачити коректну роботи застосунку можна на рисунку 5.4.

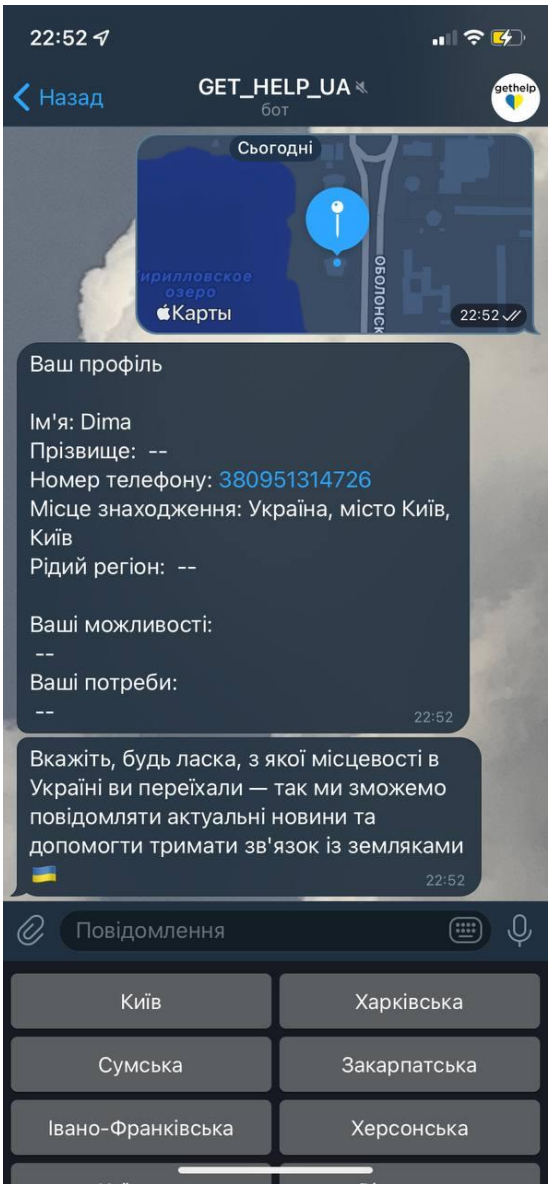

Рисунок 5.4 – Відповідь на надані користувачем дані геолокації

Для перевірки правильності переміщення між станами перевіряється команда «Назад», бот повинен повернути користувача в стан надання геолокації, і при необхідності змінити дані його профілю. На рисунку 5.5 наведено коректну обробку команди «Назад» та перезапис місцезнаходження користувача.

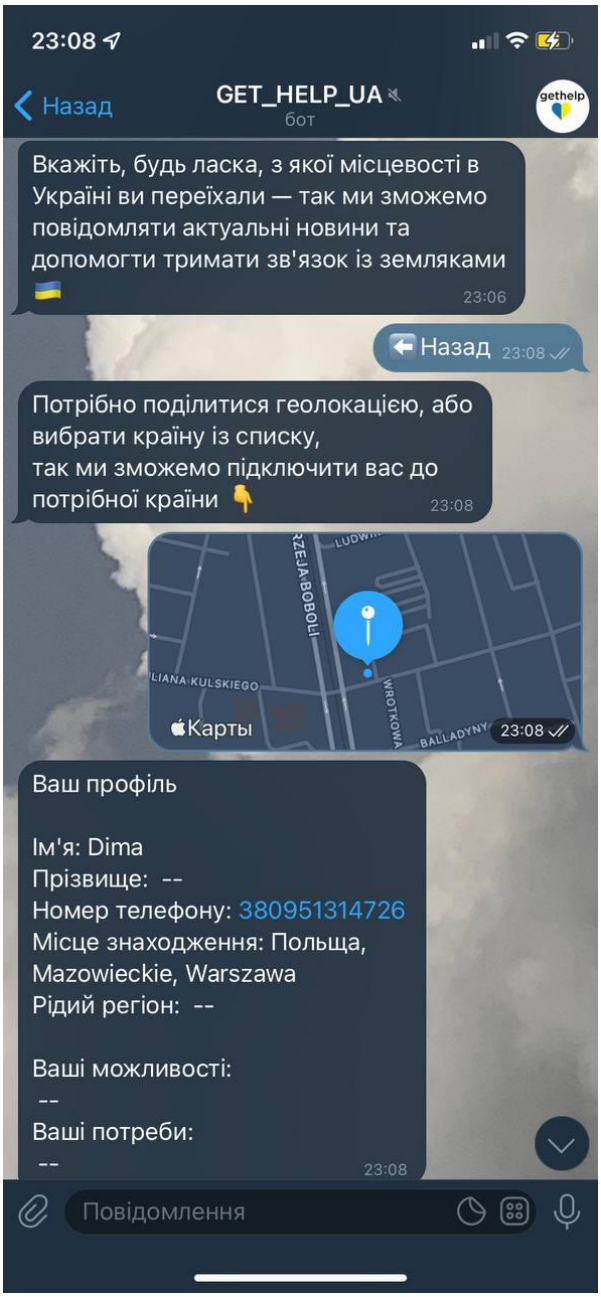

Рисунок 5.5 – Використання команди «Назад» та перевибір геолокації

Для перевірки обробки ситуації введення користувачем команди не за сценарієм, вводить некоректна команда, очікується реакція системи у вигляді попереджувального повідомлення. Коректну роботу системи представлено на рисунку 5.6

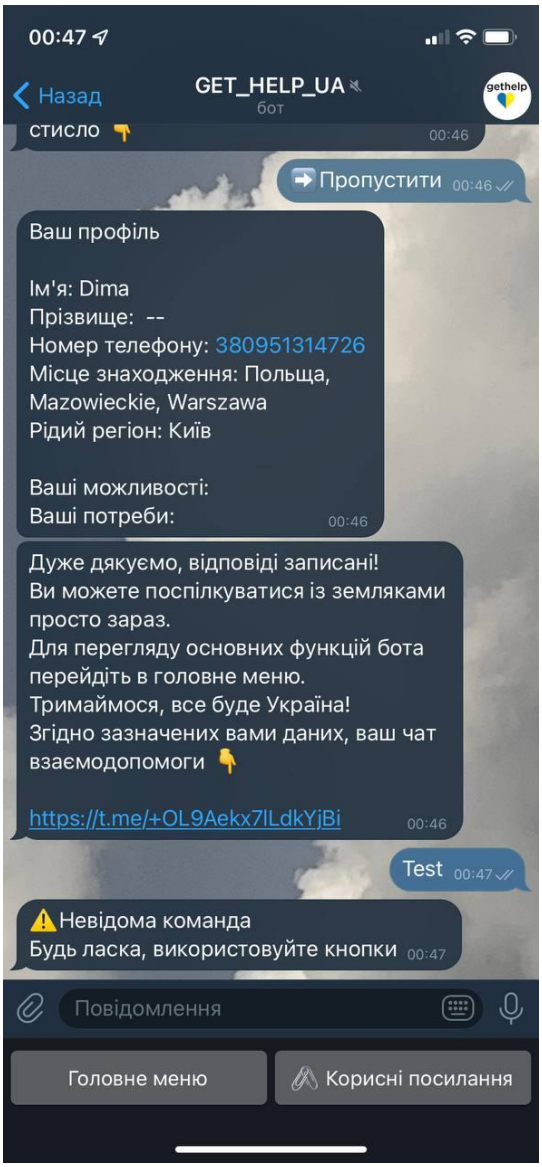

Рисунок 5.6 – Реакція системи на невірно введену команду

Тестування швидкості відповіді системи показало, що затримки, помітні людському оку, відсутні, система працює швидко і коректно. Граматичні та стилістичні помилки відсутні

Висновки до розділу 5

В розділі був описаний процес тестування. Головною метою тестування була перевірка коректності роботи всіх сторін системи та контроль якості виготовленого продукту.

Перевірка коректності переходів між станами в залежності від команди,

введеної чи вибраної користувачем показала що всі складові програми працюють справно та безперебійно виконують свої функції. При переході зі стану в стан, а для користувача з переходом з меню в меню, або на наступний крок сценарію, не було виявлено жодного некоректного відображення чи неправильно вибраного стану системи, програма здійснює обробку команд користувача справно, та забезпечує безперебійну роботу бота.

При зворотному русі по меню жодних помилок виявлено не було, користувач повертається саме в ту частину сценарію, з якої здійснював перехід в той чи інший стан.

При невірно введеній команді користувач отримає попереджувальне повідомлення, програма оброблює всі можливі дії користувача вірно.

Під час перевірки швидкісних показників застосунку було зроблено висновок що користувач отримує відповідь миттєво, система працює продуктивно, запити до бази даних оброблюються миттєво, будь-які зависання відсутні.

Також було перевірено зовнішній вигляд застосунку, зокрема текстова складова. В текстах відсутні помилки чи порушення формату.

Створена система пройшла всі етапи тестування вдало.

ВИСНОВКИ

В ході виконання дослідницького проєкту був розроблений веб сервіс для логістичної допомоги постраждалим від війни в Україні. Було розроблено систему зберігання та видання інформації за допомогою алгоритму сортування та системі фільтрів та співставлень. Для додатку було обрано Telegram в якості представлення. Це дає змогу без труднощів підключитися до системи з будь-якого пристрою, а також, без зайвих витрат часу на встановлення чи освоєння додатку, отримати очікувану послугу.

На перших етапах було проведення дослідження та аналіз предметної області, після якого було визначено актуальність і потрібність створеного додатку у спільноті.

Огляд та аналіз існуючих рішень показав деякі переваги та недоліки аналогів. Аналіз виявив сильні та слабкі місця кожного із оглянутих застосунків та сформував загальні вимоги.

На основі дослідження предметної області було виділено всі сутності системи а також визначено ролі користувачів та спроєктовано базу даних.

Вибір технологій і середовища розробки сформував стек технологій, необхідних для реалізації застосунку. Програмну складову додатку створено за допомогою сучасних засобів розробки, архітектур та методик проєктування, що створює можливість без проблемного масштабування системи.

На етапі тестування було визначено безперебійність робот застосунку.

ПЕРЕЛІК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ І ПОСИЛАННЯ

1. Біженці з україни за кордоном. URL: https://tvoemisto.tv/exclusive/bizhenets_chy_osoba_yaka_potrebuie_zahystu_mozhlyvosti_dlya_ukraintsiv_u_yevropi_ta_ssha_131184.html (дата звернення: 06.05.2022)

2. Чому чат-боти це must-have для бізнесу у 2022 році?. URL: https://ain.ua/2022/02/18/chomu-chat-boty-cze-must-have-dlya-biznesu-u-2022-roczi/ (дата звернення: 12.05.2022)

3. Рекомендації громадам щодо організації надання соціальних послуг в умовах воєнного стану. URL: https://auc.org.ua/novyna/rekomendaciyi-gromadam-shchodo-organizaciyi-nadannya-socialnyh-poslug-v-umovah-voyennogo (дата звернення: 08.05.2022)

4. Как изменилась логистика в Украине во время войны. URL: https://mind.ua/ru/openmind/20241674-sohranit-i-obespechit-kak-izmenilas-logistika-v-ukraine-vo-vremya-vojny (дата звернення: 05.05.2022)

5. Помощь Украине и украинским беженцам. Подробное руководство для тех, кому нужна помощь. URL: http://trans.info (дата звернення: 05.05.2022)

6. Helplist (ХЕЛПЛІСТ) – Український сервіс що об'єднує волонтерів і тих, хто потребує допомоги. URL: https://helplist.io/ (дата звернення: 04.05.2022)

7. Telegram bot UA HELP NOW. URL: http://t.me/uahelpnow_bot (дата звернення: 04.05.2022)

8. Viber Бот Україна допомога. URL: https://bit.ly/UkraineHelpViber (дата звернення: 04.05.2022)

9. Spring Boot URL: https://spring.io/projects/spring-boot. (дата звернення: 12.05.2022)

10. Telegram Bot API. URL: https://core.telegram.org/bots/api (дата звернення: 12.05.2022)

11. Marvin's Marvellous Guide to All Things Webhook. URL: https://core.telegram.org/bots/webhooks (дата звернення: 12.05.2022)

12. Why MySQL is best choice among the databases?. URL: https://www.vindaloosofttech.com/blog/why-mysql-is-best-choice-among-the-databases/ (дата звернення: 11.05.2022).

13. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 12.05.2022)

14. Проектування баз даних: етапи та основи. URL: https://presa.com.ua/navchannia/proektuvannya-baz-danikh-etapi-ta-osnovi.html (дата звернення: 17.05.2022)

15. ORM object-relational mapping (объектно-реляционное отображение) — Варианты реализации. URL: https://intellect.icu/orm-object-relational-mapping-obektno-relyatsionnoe-otobrazhenie-varianty-realizatsii-active-record-i-data-mapper-i-alternativy-4701 (дата звернення: 12.05.2022)

16. Webhooks done right. URL: https://medium.com/prospa-engineering/webhooks-done-right-676d4e74578a (дата звернення: 14.05.2022)

17. Эрик Гамма Design Patterns: Elements of Reusable Object-Oriented Software. Массачусетс: Addison–Wesley, 1995. 395 с.

ДОДАТОК А

Лістинг коду програми прийняття та обробки повідомлень

Код програми розміщено на зовнішньому ресурсі для зберігання коду GitHub за посиланням: https://github.com/DimaNechitaylo/GetHelpUaBot