МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Центр заочної, дистанційної та вечірньої форм навчання

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна система моніторингу якості умов життя на основі опитувань «CityMonitor»»

здобувача групи ІН - 91 Денисенка Дмитра Ігоровича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Дмитро ДЕНИСЕНКО

(підпис)

Керівник,
старший викладач, кандидат
технічних наук

Борис КУЗІКОВ _____

(підпис)

Суми – 2023

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-91 Денисенка Дмитра Ігоровича

1. Тема роботи: «Інформаційна система моніторингу якості умов життя на основі опитувань «CityMonitor»»
затверджую наказом по СумДУ від _____
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз предметної області. 2) Постановка завдання для розробки. 3) Огляд архітектури та вибір методів реалізації. 4) Практична реалізація поставленого завдання. 5) Висновки.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз предметної області*		
2	*Постановка завдання для розробки*		
3	*Огляд архітектури та вибір методів реалізації*		
4	*Практична реалізація поставленого завдання*		
5	*Висновки*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 52 стр., 31 рис., 1 додаток, 9 використаних джерел.

Обґрунтування актуальності теми роботи — тема кваліфікаційної роботи є актуальною, оскільки присвячена розробці системи, яка дозволяє ефективно взаємодіяти жителям населеного пункту та його адміністрації.

Об’єкт дослідження — інформаційна система моніторингу якості умов життя на основі опитувань.

Мета роботи — проєктування та розробка інформаційної системи моніторингу якості умов життя на основі опитувань.

Методи дослідження — алгоритми аналізу і порівняння об’єктів, синтез власних рішень.

Результати — проведено аналіз літератури, що дозволило детально розглянути поняття інформаційної системи, їх класифікації та принципи проєктування. Розглянуто якості, якими повинні володіти інтерфейси взаємодії системи з користувачами. Також був проведений аналіз існуючих рішень та різних підходів до реалізації. На основі цього аналізу було сформовано постановку задачі та обрано власний алгоритм її вирішення. В процесі вибору різноманітних інструментів та підходів імплементації було обрано та обґрунтовано вибір інтерфейсів взаємодії, СУБД, мов програмування, фреймворку. Розроблено систему, яка дозволяє проводити опитування жителів, формувати статистику, приймати звернення жителів.

ІНФОРМАЦІЙНІ СИСТЕМИ, SPRING FRAMEWORK, JAVA, POSTGRESQL, JAVASCRIPT, THYMELEAF, BOOTSTRAP, TELEGRAM API, GOOGLE MAPS API.

ЗМІСТ

ВСТУП 5

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7

1.1 Інформаційні системи 7

1.2 Інтерфейси взаємодії з користувачами 9

1.3 Огляд існуючих рішень 11

1.4 Постановка задачі 14

2 ОГЛЯД АРХІТЕКТУРИ ТА ВИБІР МЕТОДІВ РЕАЛІЗАЦІЇ 16

2.1 Підбір інтерфейсів взаємодії 16

2.2 Архітектура системи 17

2.3 Інструменти для розробки 18

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ 26

3.1 Проєктування додатків 26

3.2 Проєктування БД 31

3.3 Розробка системи 33

3.4 Користувацький інтерфейс 37

ВИСНОВКИ 44

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 46

Додаток 1. Скрипт реалізації БД 47

ВСТУП

З розвитком технологій, інформаційні системи знаходять все ширше застосування в повсякденному житті людини і тим самим значно полегшують його. Основною причиною цього є те, що сам розвиток зумовлений обміном великими обсягами інформації – міграцією знань між окремими особами, колективами чи навіть поколіннями.

Проте, разом зі збільшенням обсягів інформації, яку ми отримуємо, також збільшується проблема неупорядкованості цих даних. Часто нам потрібно працювати з великою кількістю непов'язаних між собою даних, які можуть бути розпорошені по різних джерелах. Це може призвести до труднощів у пошуку, обробці та аналізі цих даних, а також до втрати часу та збитків. Тому, важливо розробляти ефективні методи збору, організації та управління даними, щоб забезпечити їхню ефективну обробку та використання.

Будь-яка організація, що намагається йти в ногу з часом та прагне досягти успіху в сфері надання послуг, повинна використовувати систему, яка забезпечує автоматизацію внутрішніх процесів. Від її функціонування залежить не тільки продуктивність, а й оцінка цільової аудиторії закладу в цілому, його популярність.

Активний контакт є запорукою продуктивності будь-яких взаємовідносин, а особливо між мешканцями населеного пункту і його адміністрацією. В часи, коли майже кожен у своєму повсякденному житті активно користується соціальними мережами, месенджерами, такі способи взаємодії, як паперові прохання, дзвінки на гарячу лінію, а тим паче передача інформації через третіх осіб, є неактуальними та малоефективними. Люди повинні мати можливість швидко та зручно інформувати представників адміністрації про їхні проблеми, зауваження чи побажання, а ті ж, в свою чергу, вчасно та відповідно на них реагувати.

Актуальність розробки інформаційної системи за цією тематикою пояснюється тим, що вона дасть змогу адміністрації населеного пункту проводити регулярні опитування жителів на будь-яку тему, отримувати актуальну інформацію прямо від споживача послуг, аналізувати її в зручному вигляді, зберігати результати опитувань для обрахунку статистики, а також обробляти окремі запити. Простота інтерфейсу та його мінімалістичний дизайн дозволить впевнено приймати участь в житті населеного пункту людям будь-якого покоління.

Розробка такої системи є важливою, оскільки вона сприятиме більш ефективному розподілу ресурсів, що, в свою чергу, допоможе підвищити рівень задоволеності жителів послугами, які надає населений пункт.

Для успішної реалізації поставленої мети, необхідно виконати такі етапи:

- проаналізувати предметну область;
- визначити функціональні вимоги;
- виконати моделювання системи;
- обрати інструменти для розробки;
- розробити систему для моніторингу.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інформаційні системи

Інформаційна система - організований набір елементів, що збирає, обробляє, передає, зберігає та надає дані. Вона складається із людей, обладнання, процесів, процедур, даних та операцій [1].

Інформаційна система є не лише інструментом відображення функціонування об'єкта управління, але й має активний вплив на нього через органи управління. Вона складається з комплексу процесів збирання, оброблення, збереження та передачі інформації для задоволення потреб різних рівнів у процесі прийняття управлінських рішень. Основною метою інформаційної системи є створення та надання інформації для використання управлінським апаратом. Вона відповідає за збір, оброблення, збереження, узагальнення та конкретизацію інформації, яка використовується у процесі управління об'єктом (див. рис.1.1).

Рисунок 1.1 – Загальне зображення основних процесів в інформаційній системі [2]

Інформаційні системи можна класифікувати за різними ознаками:

- сферою діяльності: державні, територіальні, підприємств, галузеві, технологічних процесів;
- рівнем автоматизації: інформаційно-пошукові та довідкові, інформаційно-керуючі, системи підтримки прийняття рішень;
- ступенем централізації обробки інформації: централізовані, децентралізовані, колективного використання;
- ступенем інтеграції функцій: багаторівневі з інтеграцією за рівнями управління, багаторівневі з інтеграцією за функціями управління.

До основних принципів проєктування інформаційних систем входять:

- аналіз вимог: перед початком розробки системи необхідно зрозуміти потреби та вимоги користувачів. Це допоможе забезпечити, що система буде ефективно використовуватися та задовольняти потреби користувачів.
- модульна архітектура: система повинна бути створена з окремих модулів, які можуть бути розроблені та тестовані окремо. Це дозволяє забезпечити більшу гнучкість та можливість легко змінювати окремі частини системи.
- забезпечення безпеки: система повинна бути захищена від несанкціонованого доступу до даних та інформації. Це може бути досягнуто за допомогою використання різних методів, таких як аутентифікація, авторизація та шифрування даних.
- стандартизація: розробка системи повинна відбуватися з використанням стандартів та рекомендацій, що допоможе забезпечити сумісність та інтеграцію з іншими системами.
- тестування: перед введенням системи в експлуатацію необхідно провести комплексне тестування, яке дозволить виявити та виправити помилки та недоліки.

- зручність використання: система повинна бути зручною у використанні та мати зрозумілий інтерфейс користувача, що дозволяє мінімізувати час та зусилля, необхідні для навчання користувачів.
- масштабованість: система повинна бути готовою до масштабування та збільшення ємності, якщо потрібно. Це дозволяє забезпечити ефективне використання системи в майбутньому та уникнути необхідності в розробці нової системи з нуля.
- підтримка: система повинна мати можливість підтримки та забезпечення необхідних оновлень та покращень. Це дозволяє забезпечити ефективну та безперебійну роботу системи протягом її життєвого циклу.
- ефективність: система повинна бути ефективною виконувати свої функції та завдання з використанням мінімальних ресурсів, таких як час та пам'ять.
- функціональність: система повинна мати можливість виконувати необхідні функції та завдання користувачів з високою якістю та точністю.
- розширюваність: система повинна мати можливість додавати нові функції та можливості у майбутньому без необхідності в розробці нової системи з нуля.
- інтегрованість: система повинна мати можливість інтегруватися з іншими системами та програмами, що дозволяє забезпечити її більшу ефективність та функціональність.
- інтероперабельність: система повинна мати можливість взаємодіяти з різними платформами та пристроями, що дозволяє забезпечити її більш широку доступність та корисність.

1.2 Інтерфейси взаємодії з користувачами

Важливе місце в проєктуванні системи займає підбір інтерфейсів взаємодії. З огляду на те, що користувач може працювати з системою по

кілька годин, він має бути зручним, швидким, зрозумілим. Для забезпечення цих якостей, інтерфейс повинен мати такі властивості як:

- адаптованість:
    - задовольняти всі потреби і можливості користувача;
    - забезпечувати простий перехід між виконанням різних функцій;
    - надавати можливість користувачу мати відчуття повного контролю над ситуацією, тобто він повинен бути впевненим, що саме він розв’язує поставлену задачу;
    - мати змогу відображати інформацію у різному вигляді, підлаштувавшись під рівень користувача;
    - забезпечувати користувача взаємно-доповнюючими формами представлення результатів дивлячись на тип запиту або характер отриманого рішення.
- гнучкість:
    - мати можливість адаптації до розв’язання конкретної задачі. У випадку, коли ця задача є дуже складною, то інтерфейс має полегшувати виконання запитів і представляти їх результат у вигляді, який швидко і легко сприймається користувачем.
- достатність:
    - для користувачів всіх рівнів та прикладних задач всіх типів допустимі запити повинні бути чіткими і однозначними;
    - реакція системи на будь-який запит має бути зрозумілою, однозначною.
- дружність:

- бути максимально простим для використання та в повній мірі задовольняти його запити, при розв’язанні визначеного типу задач.

1.3 Огляд існуючих рішень

Для огляду подібних рішень візьмемо до уваги лише продукти призначені для налагодження взаємодії місцевих органів влади з громадою та залучення її до прийняття рішень. До таких можна віднести наступні сервіси:

1) SeeClickFix - це онлайн-платформа, яка дозволяє жителям звітувати про проблеми у своїй місцевості, такі як пошкоджені дороги, затоплення вулиць або проблеми зі сміттям. Після звітування, інформація автоматично передається до місцевих органів влади, які мають відповісти на звернення. Крім того, платформа дозволяє жителям обговорювати проблеми та вносити пропозиції щодо їх вирішення.

2) CitizenLab - це онлайн-платформа, яка надає можливість місцевим владам спілкуватися з громадянами та збирати їхні пропозиції та ідеї щодо розвитку міста. Громадяни можуть подавати пропозиції та голосувати за ініціативи, які вони вважають важливими. CitizenLab дозволяє місцевим органам влади збирати дані та аналізувати їх, щоб приймати кращі рішення та забезпечувати більшу участь громадян у прийнятті рішень.

3) FixMyStreet - це веб-платформа, яка дозволяє громадянам повідомляти про проблеми з інфраструктурою, такі як ями на дорозі, відсутність освітлення чи розбиті ліхтарі. Громадяни можуть подавати заявки та відстежувати їхній статус до того, як вони будуть вирішені місцевими органами влади. Однією з особливостей FixMyStreet є те, що ця платформа може інтегруватися з іншими системами, наприклад, з системами Open311 або SeeClickFix. FixMyStreet використовується в багатьох країнах, в тому числі у Великій Британії.

У порівняльній характеристиці будуть враховані та оцінені за шкалою від 0 до 10 такі параметри:

- дизайн
- зручність використання
- безпека
- функціонал
- ліцензування

Таблиця 1.1 Порівняльна характеристика існуючих систем

	SeeClickFix	CitizenLab	FixMyStreet
Дизайн	8 – сучасний, простий, зрозумілий інтерфейс	7 – також має дружній до користувача, сучасний інтерфейс, але він має більш розширені функції, що може забрати більше часу на їх вивчення	6 – застарілий, проте простий та зрозумілий інтерфейс, але може бути менш зручним для користувачів, які не знайомі з британськими термінами
Зручність використання	6 – сайт не має адаптивного дизайну, відсутня можливість встановити українську локалізацію, наявні мобільні додатки з для IOS та Android, проте вони не є зручними в деяких задачах, наприклад при роботі з картами	6 - сайт має адаптивний дизайн, відсутній мобільний додаток, немає можливості встановити українську локалізацію	4 – має мобільний додаток з обмеженим функціоналом (в порівнянні з веб-версією) тільки для IOS платформи, відсутня можливість встановити українську локалізацію

Продовження табл. 1.1

	SeeClickFix	CitizenLab	FixMyStreet
Безпека	9 – відповідає найвищим стандартам безпеки	10 – відповідає найвищим стандартам безпеки, а також забезпечує додаткову безпеку для електронних голосувань та інших дій, що потребують високого рівня захисту.	9 – відповідає найвищим стандартам безпеки
Функціонал	5 - основна функція платформи – збір звернень. Надає можливість додавати до них фото, відео та документи, позначати місце на карті, відстежувати статус звернення та спілкуватися з місцевими посадовцями	9 - додатково до функціоналу, наявного в SeeClickFix, пропонує проведення опитування та сесії мозкового штурму	4 - забезпечує тільки збір звернень та пересилання їх до місцевих органів влади
Ліцензування	8 – пропонує два тарифні плани: Basic та Pro. Basic план є безкоштовним, але обмежується базовими функціями, такими як створення та відстеження запитів. Pro план коштує від $40 на місяць і надає додаткові функції, такі як аналітика та налаштування	6 – пропонує декілька версій: безкоштовна – Basic, комерційні – Standard та Premium. Ціноутворення залежить від багатьох речей, таких як кількість громадян, кількість опцій/функцій, наданих на платформі, і періоду підтримки. В середньому від 5 до 30 тисяч євро на рік.	10 – є безкоштовним ресурсом, що фінансується від держави та пожертвувань громадян

Загальний висновок з порівняння існуючих рішень полягає в тому, що кожна з платформ має свої переваги та недоліки. Наприклад, SeeClickFix та FixMyStreet забезпечують користувачам можливість повідомляти про проблеми в конкретному місці, тоді як CitizenLab має ширший функціонал та дозволяє збирати відгуки від громади та пропозиції про різні ідеї та проєкти, проводити опитування. В той же час, комерційна ліцензія CitizenLab має досить високу вартість. Тільки SeeClickFix має мобільний додаток для обох найрозповсюдженіших платформ – IOS та Android. Жодне з рішень не підтримує українську локалізацію.

Тому можна сказати, що жодна з платформ не відповідає повністю всім критеріям, які були визначені для порівняння та зробити висновок, що створення власної інформаційної системи моніторингу якості умов життя є необхідним.

Така система може бути розроблена з урахуванням специфіки місцевості та потреб громади. Вона може бути більш безпечною, доступною, зручною у використанні та додатково мати особливий функціонал. Також, створення власної системи дасть можливість збирати дані та статистику, що дозволить визначити пріоритети для подальшого розвитку та покращення умов життя громади.

1.4 Постановка задачі

Основною метою цієї роботи є розробка інформаційної системи, що дозволятиме підтримувати активний контакт між жителями та адміністрацією, а саме відповідатиме наступним функціональним вимогам:

- система дозволятиме проводити опитування населення стосовно того чи іншого питання на виділених ділянках місцевості.
- система дозволятиме вказувати час початку та завершення опитування, тим самим автоматизуватиме його життєвий цикл.

- система буде представляти результати опитування у вигляді кругової діаграми з загальними результатами, теплової карти де буде можливість порівняти результати на окремих виділених ділянках місцевості та кругової діаграми для кожної з них.
- система дозволятиме переглядати результати проведених раніше опитувань, що дасть змогу бачити тенденцію вирішення проблем;
- система дозволятиме завчасно завершувати опитування.
- система повинна мати три рівні ролей користувачів: звичайний користувач – може переглядати результати опитувань, макети районування, звернення жителів; адміністратор – може створювати опитування, керувати зверненнями, макетами районування; системний адміністратор – може створювати та керувати профілями користувачів.
- жителі матимуть можливість брати участь в опитуваннях, вибравши одну з декількох відповідей.
- жителі матимуть можливість звертатися до адміністрації з окремих питань та отримувати відповіді на них.
- звернення буде містити опис, опціонально файли (фото, документи, відео) та геолокацію.

Для успішної реалізації поставленої мети, необхідно виконати такі етапи:

- проаналізувати доступні інтерфейси взаємодії та обрати оптимальні;
- розробити архітектуру системи;
- обрати інструменти для розробки;
- спроєктувати елементи системи;
- розробити інформаційну систему.

2 ОГЛЯД АРХІТЕКТУРИ ТА ВИБІР МЕТОДІВ РЕАЛІЗАЦІЇ

2.1 Підбір інтерфейсів взаємодії

Дивлячись на вирішувану проблему, можемо з впевненістю сказати, що всі функції системи які визначено для жителя населеного пункту можна забезпечити за допомогою звичайного чат-боту. Особливої популярності досягли саме Telegram боти, завдяки їх зрозумілому для користувача інтерфейсу та простоту розробки. Про це свідчить статистика на рисунку 2.1, де видно, що цим месенджером користуються 85.7% людей, серед тих, у яких є смартфон. Отже, взаємодія з майбутньою системою буде швидкою, зрозумілою, зручною.

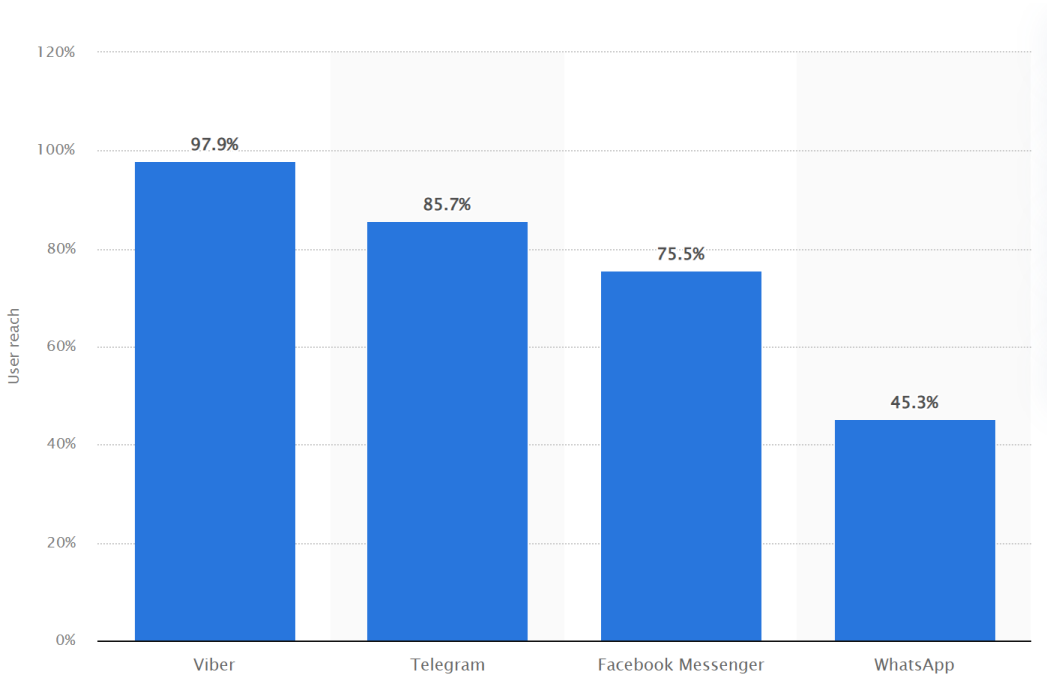

Рисунок 2.1 – Рейтинг мессенджерів [3]

Основними перевагами цього рішення є відсутність необхідності встановлювати додаткове програмне забезпечення, мінімальна потреба в ресурсах, доступність на всіх пристроях. А також, завдяки концепції

Telegram, в якій кожен аккаунт прив’язаний до номеру телефона, зникає потреба в додатковій аутентифікації та авторизації.

Адмін-частина розрахована на складніший функціонал: робота з динамічними картами, показ значного обсягу даних в структурованому вигляді, велика кількість форм. Це все потребує спеціального інструментарію для розробки, який наявний в галузі Web-розробки.

Взаємодія з системою через Web-інтерфейс обумовлює собою використання браузера для доступу до необхідних сторінок. Це рішення здобуло своєї популярності в час активного розвитку сайтів і використовується до сьогодні. Однією з його переваг є простота доступу – будь-яка людина, маючи телефон чи комп’ютер з підключенням до мережі Інтернет, може з ним взаємодіяти. Не має потреби нічого встановлювати локально на пристрій та підтримувати актуальну версію. Однак існують і такі недоліки як помітна затримка при перезавантаженні сторінок та відсутність підтримки стану сеансу роботи.

2.2 Архітектура системи

Розглянемо змодельовану багаторівневу архітектуру системи (рис.2.2).

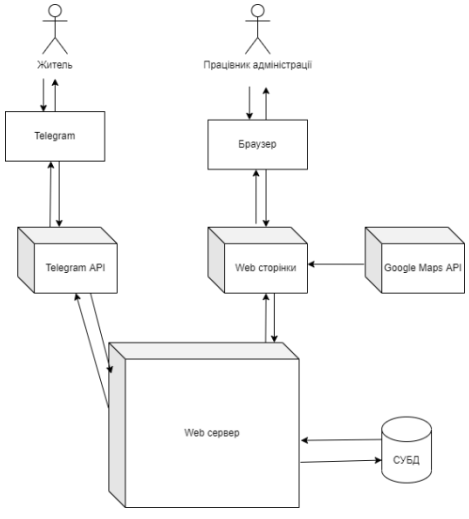

Рисунок 2.2 – Архітектура ІС

Виділеними акторами нашої інформаційної системи є житель та представник адміністрації населеного пункту. Вони взаємодіють з основним компонентом – Web-сервером за допомогою Telegram месенджера і браузера відповідно. Для можливості додавати та взаємодіяти з Google мапами на веб сторінках використовується Google Maps API. Функція зберігання даних покладена на СУБД.

Web-сервер – це апаратне та програмне забезпечення, що за допомогою HTTP(протокол передачі гіпертексту) або інших протоколів відповідає на запити клієнтів. Головним його завданням є відображення змісту Web-сайтів, шляхом зберігання обробки та доставки Web-сторінок користувачам.

Google Maps API – інтерфейс взаємодії з мапами Google.

Telegram API – інтерфейс для взаємодії з сервером Telegram.

Система управління базами даних (СУБД) - це комплексне програмне забезпечення, призначене для організації ефективного управління даними в базах даних. СУБД забезпечує можливість створення, зберігання, редагування та пошуку інформації в базах даних, а також контролює доступ до цих даних. СУБД забезпечує можливість одночасної роботи з базою даних для декількох користувачів, що робить його важливим посередником між даними, що зберігаються в базі та користувачем [4].

2.3 Інструменти для розробки

Найкращим підходом при виборі інструментів є спостереження за ефективністю їх використання в реальних проєктах.

При виборі СУБД для інформаційної системи, який базується на таких критеріях, як можливість роботи з геоданими, сумісність багатьма мовами та швидкість роботи, PostgreSQL є одним з кращих варіантів.

За даними DB-Engines Ranking - сервіс який надає рейтинг баз даних за їх популярністю, а саме: фактами частоти пошуку Google Trends, кількістю

вакансій, згадками в пошукових системах та інших даних (рис.2.3), PostgreSQL займає четверте місце серед найбільш популярних СУБД у світі.

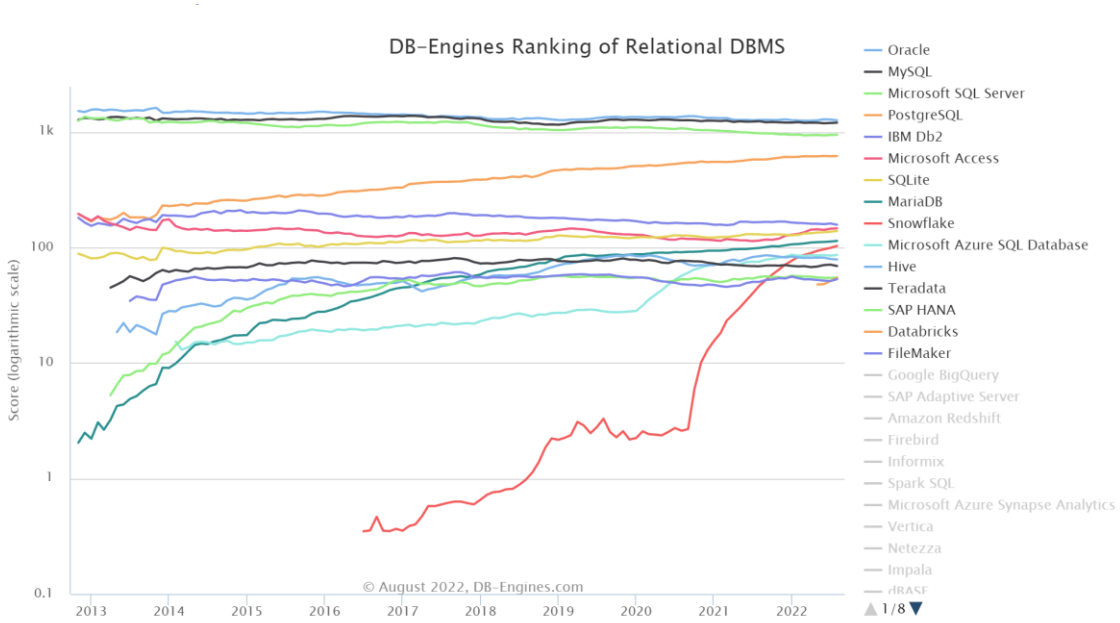

Рисунок 2.3 – DB-Engines Ranking [5]

Це свідчить про широке застосування та підтримку спільнотою. Більш того, PostgreSQL підтримує багато мов програмування, таких як C, C++, Java, Perl, Python, Ruby та багато інших. Він також є сумісним з багатьма операційними системами, включаючи Linux, Windows та macOS.

Однією з головних переваг PostgreSQL є підтримка геоданих, зокрема розширення PostGIS. Воно дозволяє зберігати, опрацьовувати та запитувати геодані в базі даних. Це робить його ідеальним вибором для розробки систем, пов'язаних з геоданими.

Щодо швидкості, PostgreSQL є достатньо швидким для більшості додатків та має ряд функцій, що дозволяють оптимізувати роботу з базою даних. Він також має можливість кешування запитів, що дозволяє зменшити час відповіді системи.

Одним з обмежень PostgreSQL є те, що він може бути складним у встановленні та налаштуванні. Однак, для більшості додатків, стандартні налаштування достатньо ефективні.

Взагалі, PostgreSQL є потужною та гнучкою СУБД, яка підходить для більшості застосувань, зокрема для розробки інформаційної системи на нашу тематику.

Під час вибору мови програмування для back-end, необхідно звернути увагу на її сучасність, інтегрованість з обраною СУБД та front-end технологіями. На рисунку нижче наведено рейтинг мов програмування у сфері back-end, проведений спільнотою DOU.ua у 2022 році (рис.2.4).

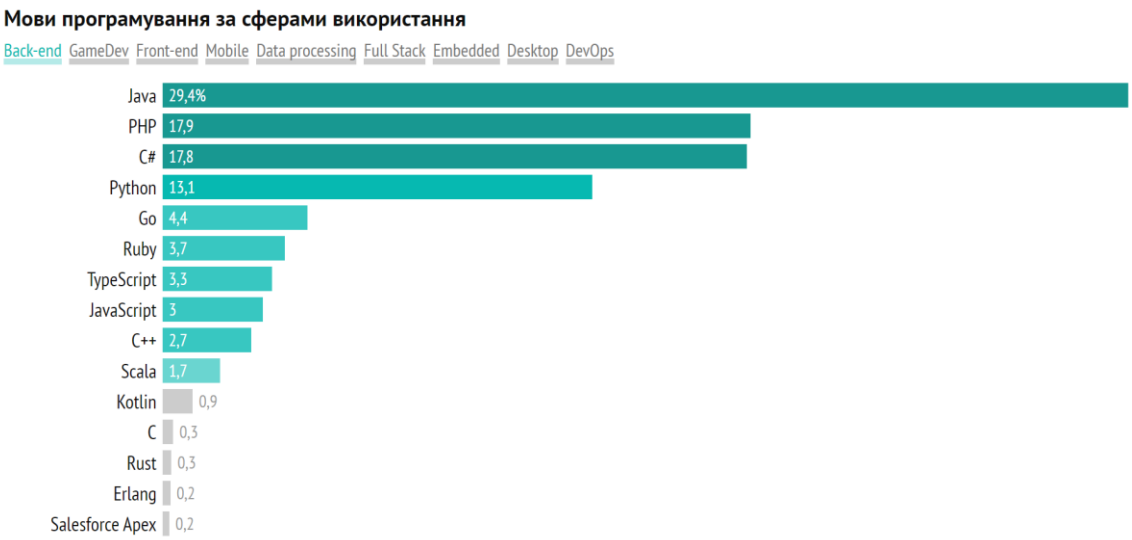

Рисунок 2.4 – Рейтинг мов програмування для Back-end [6]

Хоч і бере свої початки Java у далекому 1995-ому, вона залишається однією з найпопулярніших мов у світі. Причиною створення Java була необхідність в програмному забезпеченні для електронних приладів, що не залежить від платформи, а вже потім, з розвитком мережі Internet, вона знайшла своє застосування в Web-розробці. Так звана кроссплатформовість була досягнута за допомогою використання проміжного байт-коду, згенерованого Java компілятором. Для запуску програми на мові Java

необхідне спеціальне середовище Java Runtime Environment (JRE), а також Java Virtual Machine (JVM). На сьогодні Java використовується не тільки в Web-системах, а і у сфері Android розробки, десктоп додатках.

Як і будь-яка інша мова програмування, вона має свої переваги та недоліки. До головних переваг Java можна віднести:

- Вищезгадана платформонезалежність – значуща перевага над багатьма іншими мовами.
- Об’єктно-орієнтований підхід (ООП) – сучасний підхід програмування побудований на оперуванні об’єктами та їх властивостями, дає перевагу зручності побудови архітектури додатку та підтримки його функціонування, в можливості повторного використання коду.
- Простота використання – високорівнева мова програмування, що запозичила всім знайомий С-синтаксис позбувшись значної кількості його недоліків, полегшено процес розробки об’єктно-орієнтованих програм.
- Велика кількість бібліотек та фреймворків, що дозволяють використовувати Java для більшості задач.
- Безпека – виконання програми повністю обмежується віртуальною машиною JVM, відсутні засоби та механізми для прямої роботи з апаратним забезпеченням комп’ютера.
- Надійність – виявлення основних логічних помилок на етапі компіляції, строга типізація даних, контроль виняткових ситуацій, відсутність механізмів, які потенційно можуть призвести до помилок: неявне перетворення типів з втратою точності, арифметика покажчиків і подібне.
- Наявність збирача сміття, який автоматично звільняє пам’ять від об’єктів, що більше не використовуються в програмі, звільняє програміста від додаткових зобов’язань, тим самим роблячи його код надійнішим.

Нижче перераховано недоліки:

- Низька швидкість роботи – одна з проблем всіх високорівневих мов програмування, Java не виняток. В цьому випадку використання

проміжного коду, який виконується JVM є навпаки недоліком, проте за останні часи розробники значно пришвидшили роботу JVM і програми на Java вже не так відстають від аналогів на тому ж C.

- Java власність компанії Oracle та є платною для використання в комерційних цілях.

У випадку, коли користувач хоче бачити щось більше ніж статичну вебсторінку виникає необхідність використання деякої мови програмування для front-end частини. JavaScript можна впевнено назвати лідером цього ринку, дивлячись на нижченаведений рейтинг від спільноти DOU.ua 2022 року (рис.2.5).

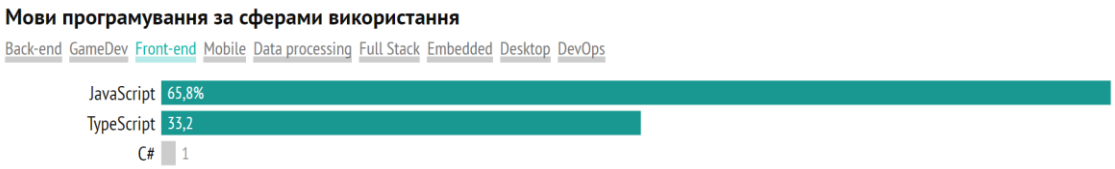

Рисунок 2.5 - Рейтинг мов програмування для Front-end [6]

JavaScript – сучасна, легка у використання, мультипарадигмова мова програмування, яка дає змогу впроваджувати складні функції на вебсторінках. Основна її функція використання на стороні клієнта надавати сторінкам інтерактивності та динамічності, що дозволяє реагувати на події, перевіряти дані, відображати спеціальні ефекти, редагувати DOM-дерево, маніпулювати cookie, робити запити на сервер, визначати браузер користувача тощо.

Популярність JavaScript є не випадковою, а завдяки безперечним перевагам:

- Продуктивність та швидкість роботи – мова дозволяє частково обробляти вебсторінки на стороні клієнта без запитів на сервер, що заощаджує час, трафік та знижує навантаження на сервер.

- Незамінність в веброзробці - підтримка усіма популярними браузерами, повна інтеграція зі сторінками та серверною частиною.
- Раціональність та простота застосування – прості завдання можна виконати за допомогою вже готових рішень, в іншому випадку підібрати найкраще та адаптувати його.
- Потужна екосистема – неймовірна кількість готових рішень у відкритому доступі, що додає зручності використання JavaScript та його фреймворків.

Також є і недоліки:

- Безпека – оскільки виконання коду відбувається на стороні клієнта, помилки та прогалини іноді можуть використовуватись для особистих цілей зловмисників. Прикладом цього є всім відомі XSS-атаки.
- Відсутність підтримки віддаленого доступу – неможливо писати додатки де необхідна роботи з мережею.
- Підтримка браузерів – між виконанням одного і того ж скрипту різними браузерами можуть виникнути відмінності. Не дивлячись на те, що в наш час це трапляється набагато рідше, потреба в тестуванні коду в різних середовищах не зникла.

Для спрощення процесу розробки Web-сервісу, доцільно використати якийсь з фреймворків – каркас програмного рішення, що задає певні правила написання коду та впливає на архітектуру. Найчастіше складається з постійної частини і окремих модулів, бібліотек, які можна додати в разі потреби.

Нижче наведено рейтинг фреймворків для серверної частини за показником популярності використання в межах сервісу GitHub (рис.2.6).

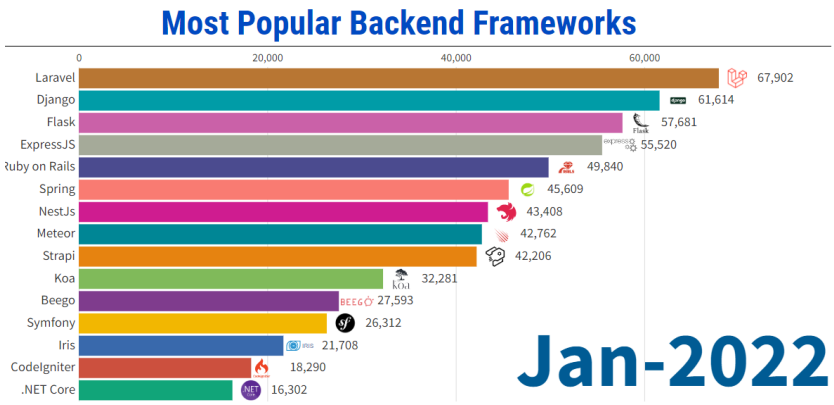

Рисунок 2.6 – Рейтинг фреймворків для Back-end [7]

З огляду на те, що на попередньому кроці мовою для backend було обрано Java, спектр фреймворків значно звужується. Найбільш популярним з них є Spring Framework. Основною метою його розробки була не заміна більш старої моделі Enterprise Java Beans (EJB), а створення альтернативи або навіть її доповнення.

Spring є одним з найбільш універсальних фреймворків з відкритим кодом для платформи Java. Використання його модулів забезпечує вирішення багатьох задач, з якими стикаються Java-розробники. Найбільшого застосування він знайшов в розробці складних додатків корпоративного рівня.

На схемі нижче (рис.2.7) можна побачити основні модулі Spring Framework.

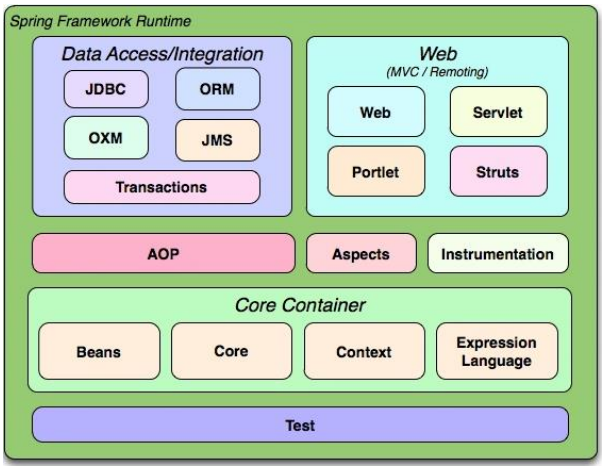

Рисунок 2.7 – Модулі Spring Framework [8]

Перелік основних можливостей Spring:

- Інверсія контролю – Spring-контейнер може повністю керувати життєвим циклом об’єкта відповідно до його конфігурації.
- Впровадження залежності – автоматичне створення зв’язку між об’єктами на основі конфігурацій контейнера.
- Можливість створення RESTful web-сервісів та web-додатків за допомогою Spring MVC.
- Прості способи роботи з базою даних за використанням модулів Spring Data JPA або Spring JDBC.
- Функціонал Test, JMS, Transactions, AOP та інших модулів.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Проєктування додатків

Для розуміння того, як повинна працювати система, було проаналізовано функціональні вимоги та побудовано діаграму варіантів використання в нотації UML (рис.3.1, рис.3.2). Такий вид діаграм описує функціонал розроблюваної системи для кожного виду користувачів.

Рисунок 3.1 – UML діаграма варіантів використання чат боту

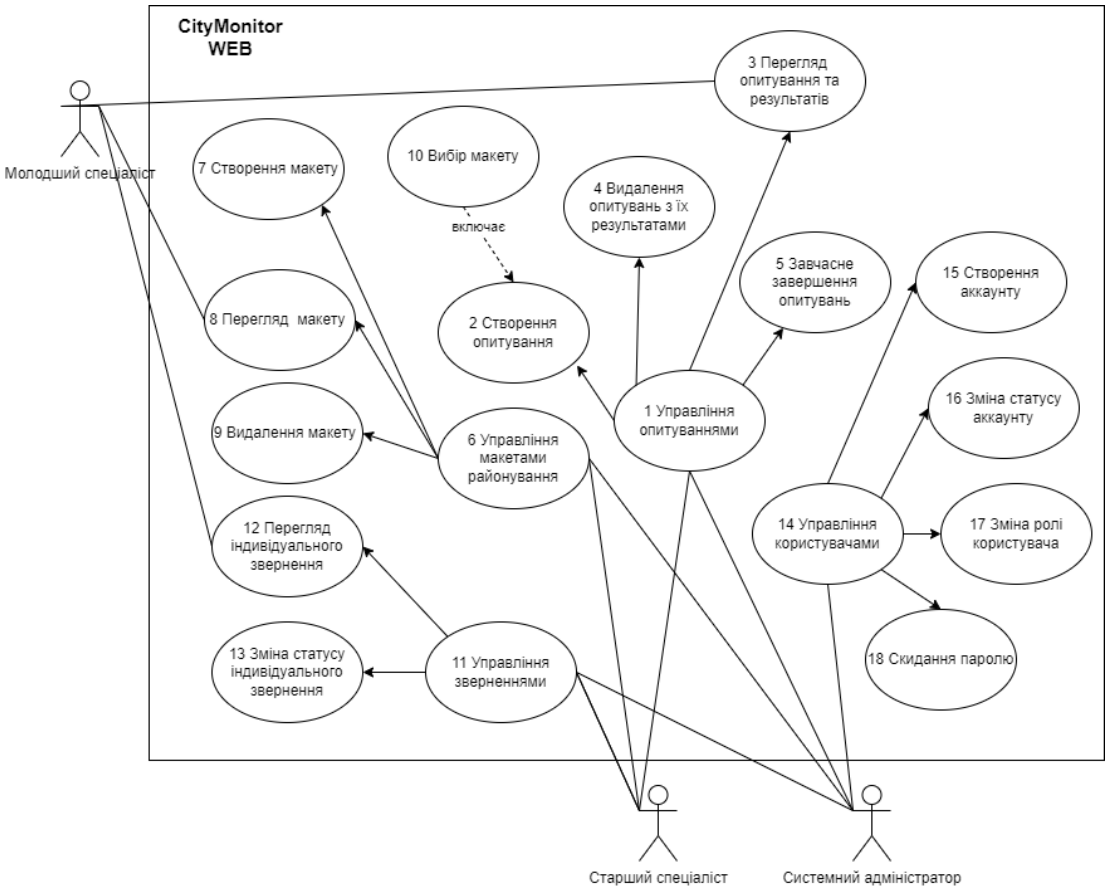

Рисунок 3.2 – UML діаграма варіантів використання веб додатку

Макет районування – карта місцевості, що господарюється цим підприємством розбита на окремі ділянки – райони. Сенс поділу на райони може мати різний характер: за відповідальними особами, за адміністративними межами тощо.

Для інформаційної системи було виділено чотири типи акторів:

- Житель –людина, яка хоче приймати активну участь в житті свого населеного пункту.
- Системний адміністратор – працівник адміністрації, який відповідає за роботу інформаційної системи та має всі права доступу.
- Старший спеціаліст – працівник адміністрації населеного пункту, який відповідає за проведення опитувань, реагування на індивідуальні звернення.

- Молодший спеціаліст – працівник адміністрації населеного пункту, який відповідає за аналіз результатів опитування та індивідуальних звернень, збирання статистики.

Варіанти використання для жителя:

- Реєстрація – вказання ім’я та координат місця проживання
- Зміна особистих даних
- Проходження опитувань
- Написання звернень

Варіанти використання для системного адміністратора:

- Управління макетами районування: створення, перегляд, видалення
- Управління зверненнями: перегляд, зміна статусу
- Управління опитуваннями: створення, перегляд результатів, видалення, завчасне звершення
- Управління користувачами: створення аккаунтів, зміна статусу аккаунту, зміна ролі користувача, скидання паролю

Варіанти використання для старшого спеціаліста:

- Управління макетами районування: створення, перегляд, видалення
- Управління зверненнями: перегляд, зміна статусу
- Управління опитуваннями: створення, перегляд результатів, видалення, завчасне звершення

Варіанти використання для молодшого спеціаліста:

- Перегляд результатів опитувань
- Перегляд індивідуальних звернень
- Перегляд макетів районування

Нижче наведено побудовані діаграми активності, які відображають логіку процесу взаємодії жителя з системою через Telegram (рис.3.3) та процесу управління опитуваннями працівника адміністрації з використанням

Web-інтерфейсу (рис.3.4). З їх використанням, є змога відстежити послідовність можливих дій користувача та процесів в системі, які за ними слідують. Також добре видно взаємодію окремих компонент інформаційної системи.

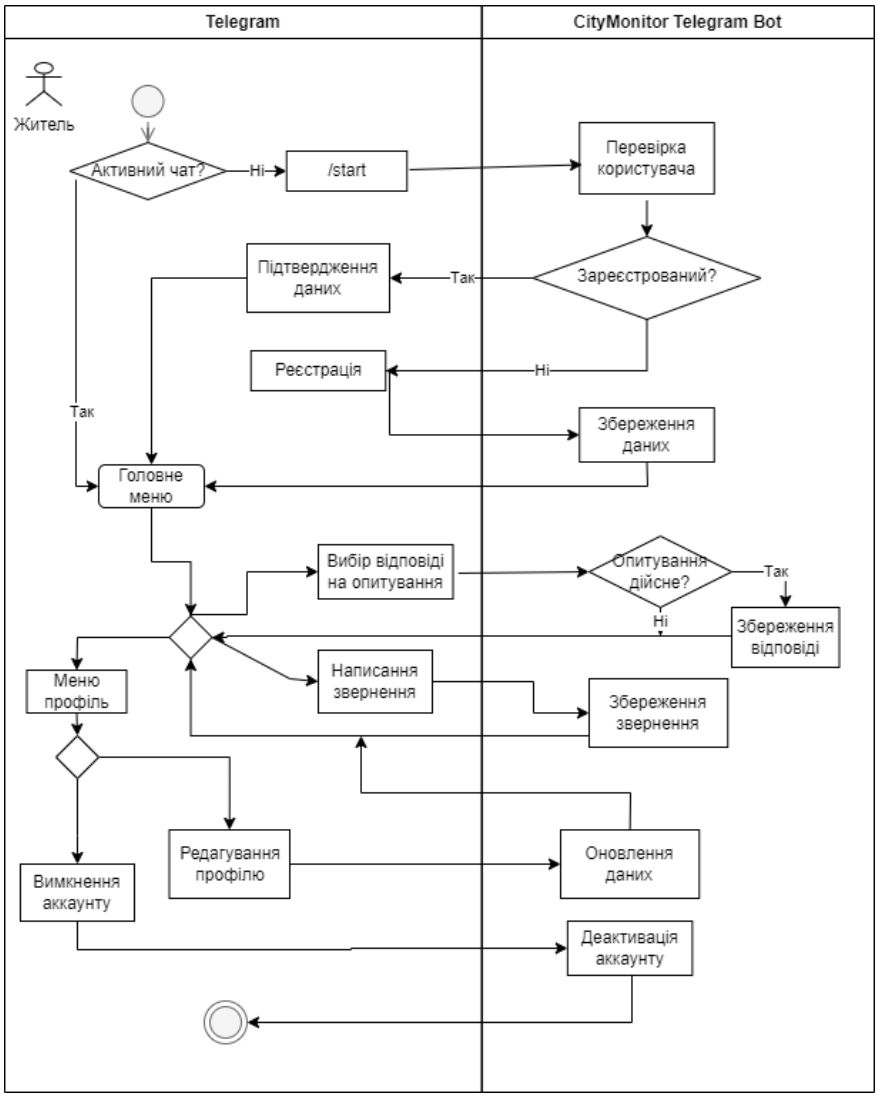

Рисунок 3.3 – Діаграма активності UML для процесу взаємодії жителя

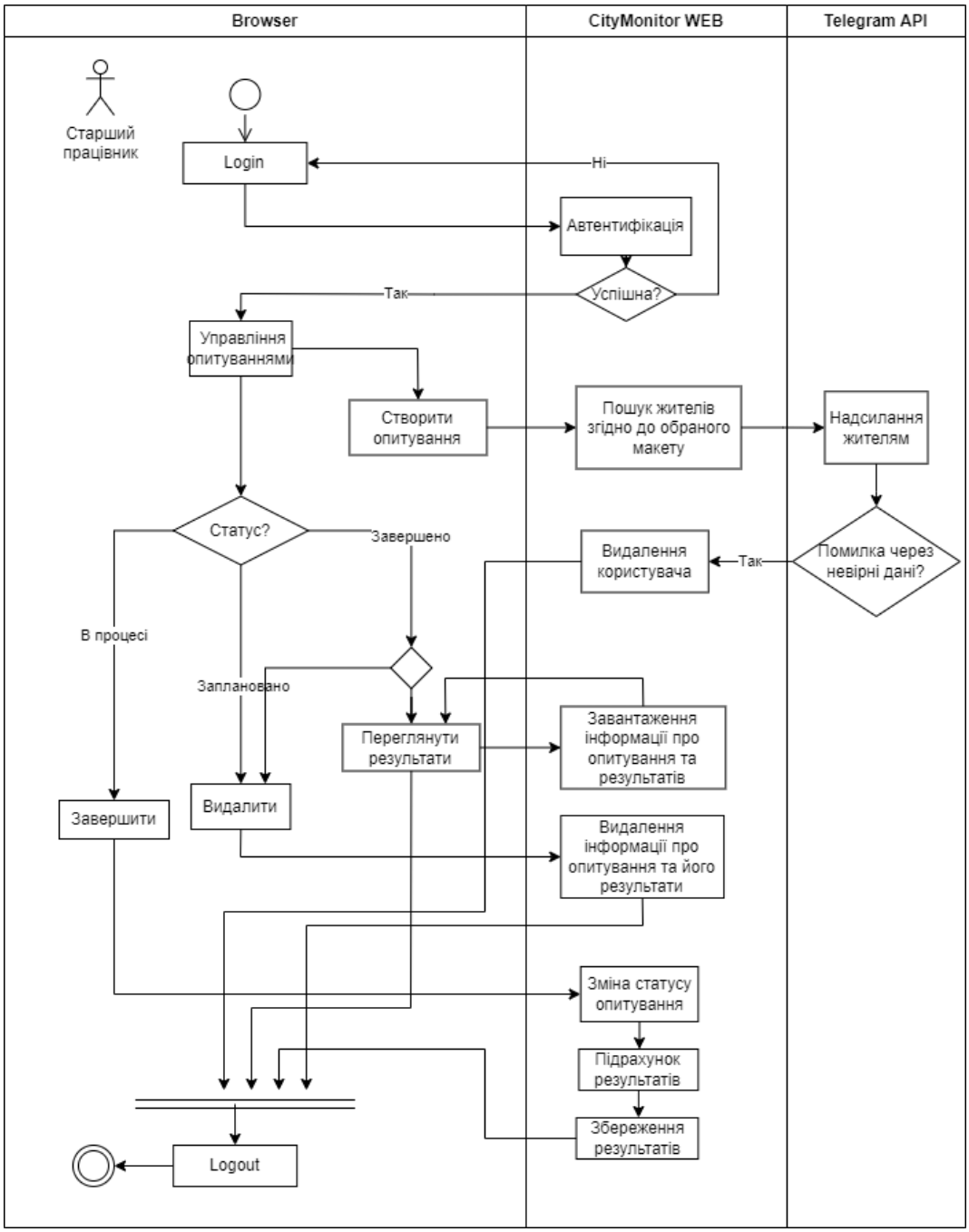

Рисунок 3.4 – Діаграма активності UML процесу управління опитуваннями

Результати опитування будуть підраховуватись після його автоматичного або примусового завершення. Для кожної ділянки місцевості виділеній на макеті районування та в цілому ми зможемо побачити статистику. Для визначення чи перебуває опитана людина в якомусь з

виділених районів буде застосований один з алгоритмів вирішення задачі належності точки многокутнику – метод трасування променів на основі обліку кількості перетинів.

Метод, що використовується для визначення належності точки простому многокутнику, полягає в тому, що випускається промінь з даної точки у довільному напрямку, а потім рахується, скільки разів промінь перетинає ребра многокутника. Якщо кількість перетинів непарна, то точка лежить всередині многокутника, якщо парна - ззовні. Цей алгоритм працює за час O(N) для N-кутника. В методі можуть виникнути ускладнення в випадку, коли промінь перетинає вершину многокутника, але їх можна подолати, вважаючи, що такі вершини лежать на нескінченно малу відстань вище (або нижче) променя, тоді будуть відсутні перетини в вершині. Існує спосіб визначення належності точки опуклому або зірчастому N-кутнику за допомогою двійкового пошуку за час O(log N), але це вимагає витрати O(N) пам'яті та O(N) часу на попередню обробку [9].

3.2 Проєктування БД

Для зображення сутностей та їх відносин в базі даних було побудовано ERD-діаграму (рис.3.5). Також на ній видно атрибути сутностей, їх основні обмеження та властивості.

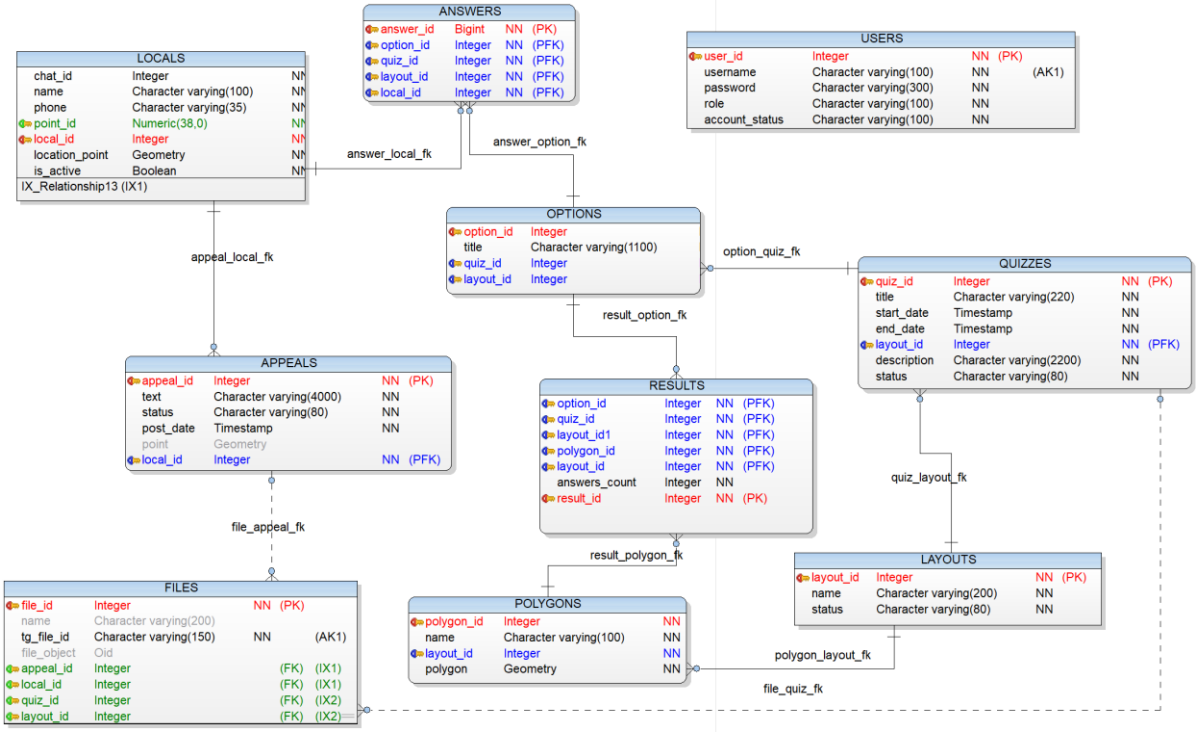

Рисунок 3.5 – ERD інформаційної системи CityMonitor

Опис виділених сутностей:

- LOCAL – метадані жителя, що взаємодіє з чат-ботом Telegram;
- USER – метадані працівника адміністрації;
- APPEAL – звернення від жителя;
- QUIZ – інформація про опитування;
- OPTION – варіант відповіді;
- ANSWER – відповідь жителя, на питання;
- LAYOUT – макет районування;
- POLYGON – многокутник виділений на певному макеті;
- RESULT – кількість обрання варіанту відповіді, на конкретне питання, для конкретного многокутника на макеті;
- FILE – дані про файл, який може бути прикріплений до опитування чи звернення;

3.3 Розробка системи

Процес розробки системи можна поділити на декілька основних етапів:

- Розробка бази даних
- Розробка Telegram-боту
- Розробка Web-додатку

Зважаючи на попередньо змодельовану структуру бази даних, було написано скрипт для її ініціалізації, його наведено в додатку 1.

Розробка Telegram-боту починається з його реєстрації в месенджері за допомогою BotFather. Після успішної реєстрації, було отримано токен, який надає доступ до взаємодії з новоствореним ботом через Telegram API. Основним етапом є написання серверної частини боту, яка відповідає за обробку запитів користувачів та генерацію відповідей. Для цього, з використанням діаграми активності (рис.3.3), було побудовано механізм відстеження станів чату з користувачем.

```
package com.denysenko.citymonitorbot.enums;

public enum BotStates {
    MAIN_MENU( title: "main"),
    EDITING_PROFILE_NAME( title: "profile:entering_name"),
    EDITING_PROFILE_PHONE( title: "profile:entering_phone"),
    EDITING_PROFILE_LOCATION( title: "profile:entering_location"),
    PROFILE_MENU( title: "profile"),
    APPEAL_ENTERING_DESCRIPTION( title: "appeal:entering_description"),
    APPEAL_ATTACHING_FILES( title: "appeal:attaching_files"),
    APPEAL_ENTERING_LOCATION( title: "appeal:entering_location");

    private String title;

    BotStates(String title) { this.title = title; }

    public String getTitle() { return title; }
}
```

Рисунок 3.6 – Скріншот переліку станів боту

Таким чином, додаток, пам’ятаючи в якому стані знаходиться інтерфейс користувача, має змогу відповідно реагувати, слідуючи логіці чату. Для прикладу, обробка текстових повідомлень відбувається наступним чином (рис.3.7).

```java
@Log4j
@Component
public class TextHandler implements Handler {

    @Autowired
    private BotUserService botUserService;
    @Autowired
    private ProfileEnterNameCommand profileEnterNameCommand;
    @Autowired
    private ProfileEnterPhoneNumberCommand profileEnterPhoneNumberCommand;
    @Autowired
    private AppealEnterDescriptionCommand appealEnterDescriptionCommand;

    @Override
    public boolean isApplicable(Update update) {
        if(!update.hasMessage() || !update.getMessage().hasText()) return false;
        Message message = update.getMessage();
        Long chatId = message.getChatId();
        Optional<BotStates> botUserState = botUserService.findBotStateByChatId(chatId);
        if(botUserState.isPresent()){
            BotStates botState = botUserState.get();
            return botState.equals(BotStates.EDITING_PROFILE_NAME) || botState.equals(BotStates.EDITING_PROFILE_PHONE)
                    || botState.equals(BotStates.APPEAL_ENTERING_DESCRIPTION);
        } else return false;
    }

    @Override
    public void handle(Update update) {
        Message message = update.getMessage();
        Long chatId = message.getChatId();
        log.info("Update handled by TextHandler: updateId = " + update.getUpdateId() + ", chatId = " + chatId.toString());
        BotStates botState = botUserService.findBotStateByChatId(chatId).get();

        if(botState.equals(BotStates.EDITING_PROFILE_NAME)){
            profileEnterNameCommand.saveUserName(chatId, message.getText());
        }else if(botState.equals(BotStates.EDITING_PROFILE_PHONE)){
            profileEnterPhoneNumberCommand.savePhoneNumber(chatId, message.getText());
        }else if(botState.equals(BotStates.APPEAL_ENTERING_DESCRIPTION)){
            appealEnterDescriptionCommand.saveDescription(chatId, message.getText());
        }
    }
}
```

Рисунок 3.7 – Скріншот реалізації обробника текстових повідомлень

Взаємодія з базою даних реалізована за допомогою модулю Spring JPA. Приклад співставлення сутності додатку з атрибутами таблиці в БД показано на рисунку 3.8.

```java
@Getter
@Setter
@NoArgsConstructor
@Entity
@Table(name = "ANSWERS")
public class Answer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "answer_id")
    private Long id;
    @Column(name = "option_id")
    private Long optionId;
    @Column(name = "local_id")
    private Long localId;
    @ManyToOne(optional = false)
    @JoinColumn(name = "quiz_id")
    private Quiz quiz;

    public Answer(Long localId, Long optionId, Quiz quiz){
        this.localId = localId;
        this.optionId = optionId;
        this.quiz = quiz;
    }

}
```

Рисунок 3.8 – Скріншот сутності “Answer”

Розробка Web-додатку системи CityMonitor в свою чергу поділилась на окремі частини: front-end та back-end. В якості механізму поєднання був використаний стандартний підхід в Spring –шаблонізатор Thymeleaf (рис.3.9).

```java
@GetMapping("/{id}")
@PreAuthorize("hasAnyAuthority('quizzes:read')")
public String quizPage(Model model, @PathVariable("id") Long id) throws JsonProcessingException {
    log.info("Getting quiz page with parameters: id = " + id);
    Quiz quiz;
    try {
        quiz = quizService.getById(id);
    }catch (EntityNotFoundException e){
        throw new InputValidationException(e.getMessage(), e);
    }

    QuizDTO quizDTO = quizConverter.convertEntityToDTO(quiz);
    model.addAttribute( attributeName: "quiz", quizDTO);

    if(quiz.getStatus().equals(QuizStatus.FINISHED)){
        model.addAttribute( attributeName: "mapCenterLat", mapCenterLat);
        model.addAttribute( attributeName: "mapCenterLng", mapCenterLng);
        model.addAttribute( attributeName: "mapZoom", mapZoom);
        model.addAttribute( attributeName: "googlemaps_apikey", GOOGLE_MAPS_API_KEY);
        List<Result> results = resultService.findResultByQuizId(id);
        List<ResultDTO> resultDTOs = resultConverter.convertListsEntityToDTO(results);
        ObjectWriter ow = new ObjectMapper().writer().withDefaultPrettyPrinter();
        String resultsJSON = ow.writeValueAsString(resultDTOs);
        model.addAttribute( attributeName: "resultsJSON", resultsJSON);
    }
    log.info("Returning template 'quizzes/quiz' with model");
    return "quizzes/quiz";
}
```

Рисунок 3.9 – Скріншот прикладу передачі даних на front-end

Приклад інтеграції переданих даних в HTML показано на рисунку 3.10

```
<table class="table table-hover table-bordered table-condensed align-middle">
    <thead>
    <tr>
        <th>Тема</th>
        <th>Назва макету</th>
        <th>Дата початку</th>
        <th>Дата закінчення</th>
        <th>Статус</th>
        <th></th>
    </tr>
    </thead>
    <tbody>
    <tr class="clickable-row" th:id="${'tr_' + quiz.id}" th:data-href="${'/quizzes/' + quiz.id}" th:each="quiz : ${quizzes.page}">
        <td><t th:text="${quiz.title}"></t></td>
        <td><t th:text="${quiz.layoutDTO.name}"></t></td>
        <td><t th:text="${#temporals.format(quiz.startDate, 'dd-MM-yyyy HH:mm:ss')}"></t></td>
        <td><t th:text="${#temporals.format(quiz.endDate, 'dd-MM-yyyy HH:mm:ss')}"></t></td>
        <td><t class="statusValue" th:text="${quiz.status}"></t></td>
        <td th:if="${quiz.status == 'В процесі'}" sec:authorize="hasAuthority('quizzes:write')">
            <button class="btn btn-primary submit-btn finishQuizBtn" th:data-id="${quiz.id}">Завершити</button>
        </td>
    </tr>
    </tbody>
</table>
```

Рисунок 3.10 – Скріншот прикладу інтеграції даних в HTML

Для імплементації взаємодії з динамічними картами був використаний Google Maps API.

```
let mapElement = $('#map')[0];
let mapCenterLat = Number($('#mapCenterLat').val());
let mapCenterLng = Number($('#mapCenterLng').val());
let mapZoom = Number($('#mapZoom').val());
openedInfoWindow = new google.maps.InfoWindow();

let mapOptions = {
    zoom : mapZoom,
    center : {lat: mapCenterLat, lng : mapCenterLng}
}
map = new google.maps.Map(mapElement, mapOptions);
drawingManager = new google.maps.drawing.DrawingManager({
    drawingMode: google.maps.drawing.OverlayType.POLYGON,
    drawingControl: true,
    drawingControlOptions: {
        position: google.maps.ControlPosition.TOP_CENTER,
        drawingModes: [google.maps.drawing.OverlayType.POLYGON]
    },
    polygonOptions: {
        clickable: true,
        fillOpacity: 0.6,
        strokeWeight: 2
    }
});
drawingManager.setMap(map);
```

Рисунок 3.11 – Скріншот прикладу використання Google Maps API

3.4 Користувацький інтерфейс

Один з найважливіших елементів будь-якого додатку для користувача - його інтерфейс, який визначає враження від використання програми. Перш за все, перед тим як приступати до програмної реалізації, було створено прототип, який дозволив оптимізувати використання ресурсів і відкинути хибні ідеї, що могли негативно впливати на користувацький досвід використання додатку. Дизайн взаємодії з користувачем (UX) грав ключову роль у проєктуванні інтерфейсу. Взагалі, дизайн повинен бути сучасним, лаконічним та не використовувати застарілі методи взаємодії з користувачем.

Для прикладу, пройдемо повний цикл реєстрації в Telegram боті.

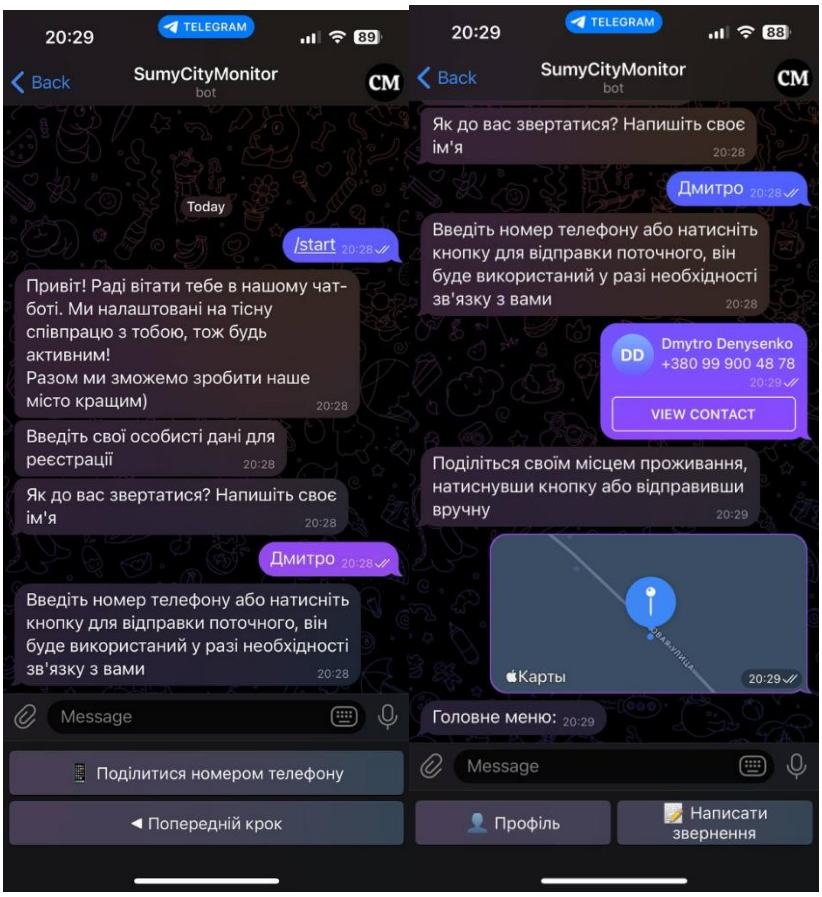

Рисунок 3.12 – Скріншот прикладу реєстрації в чат боті

Після успішної реєстрації, користувач готовий проходити опитування і потрапляє в головне меню звідки може перейти в налаштування профілю (рис.3.13) або приступити до написання звернення (рис.3.14).

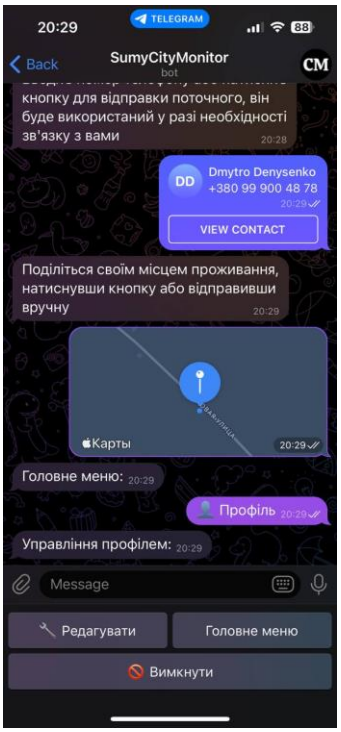

Рисунок 3.12 – Скріншот прикладу меню управління профілем

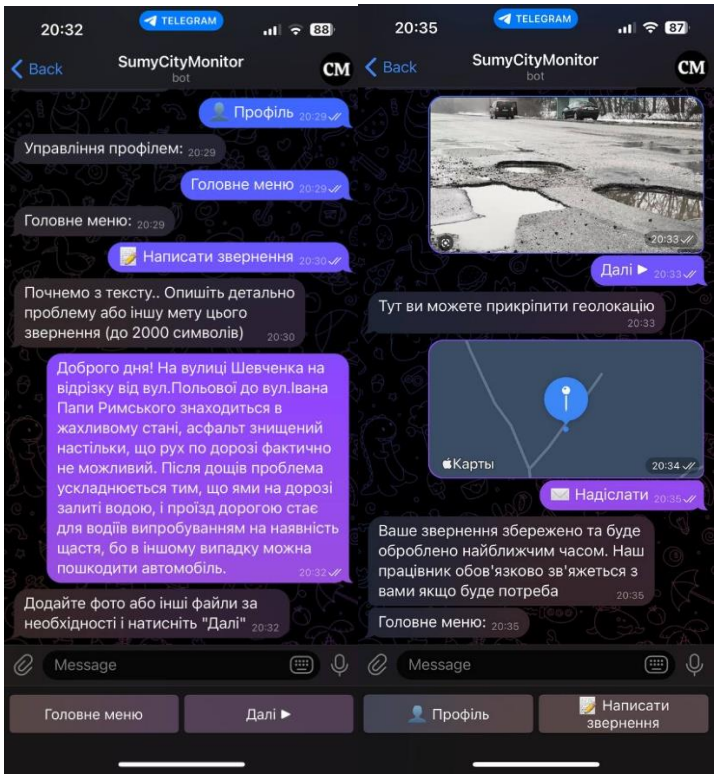

Рисунок 3.13 – Скріншот прикладу написання звернення

У разі, якщо користувачеві доступне опитування, він може прийняти в ньому участь, натиснувши на один з варіантів відповідей (рис.3.14).

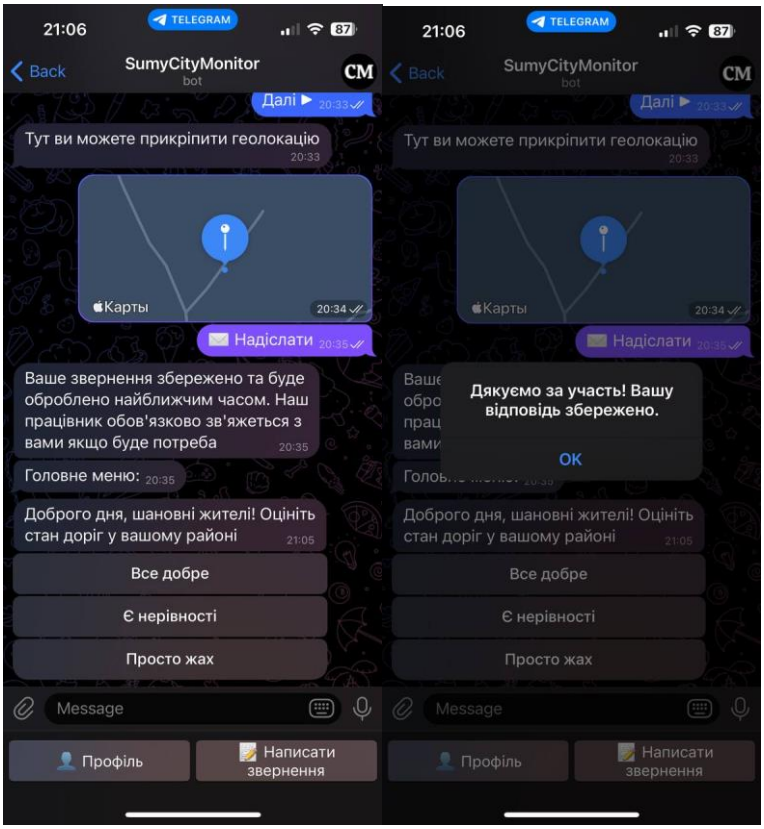

Рисунок 3.14 – Скріншот прикладу написання звернення

Для користувача Web-додатку відображення буде відбуватись наступним чином. Перша сторінка яку бачить користувач – сторінка входу (рис.3.15).

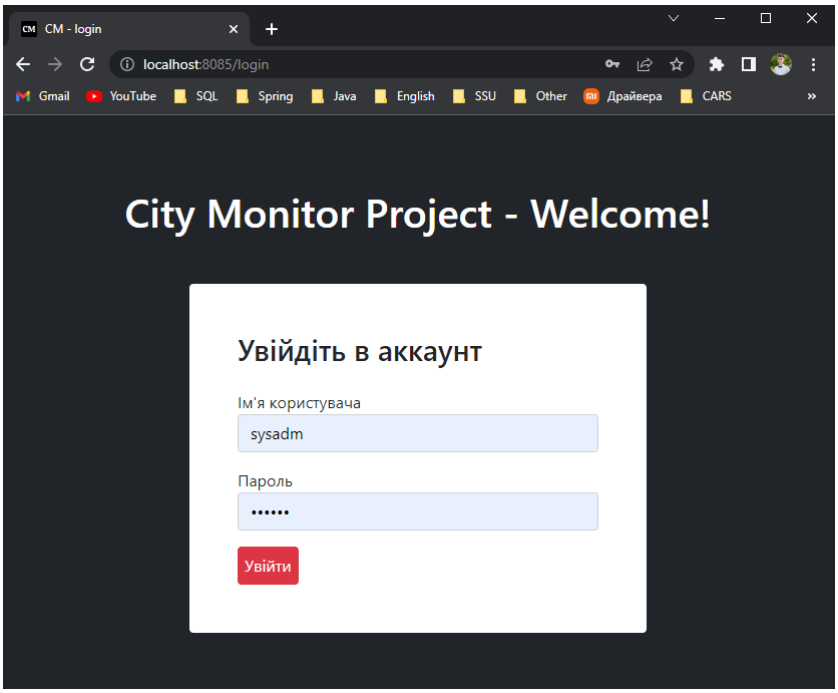

Рисунок 3.15 – Скріншот сторінки входу

У разі успішної аутентифікації, користувач потрапляє на головну сторінку, де може побачити перелік останніх опитувань (рис.3.16).

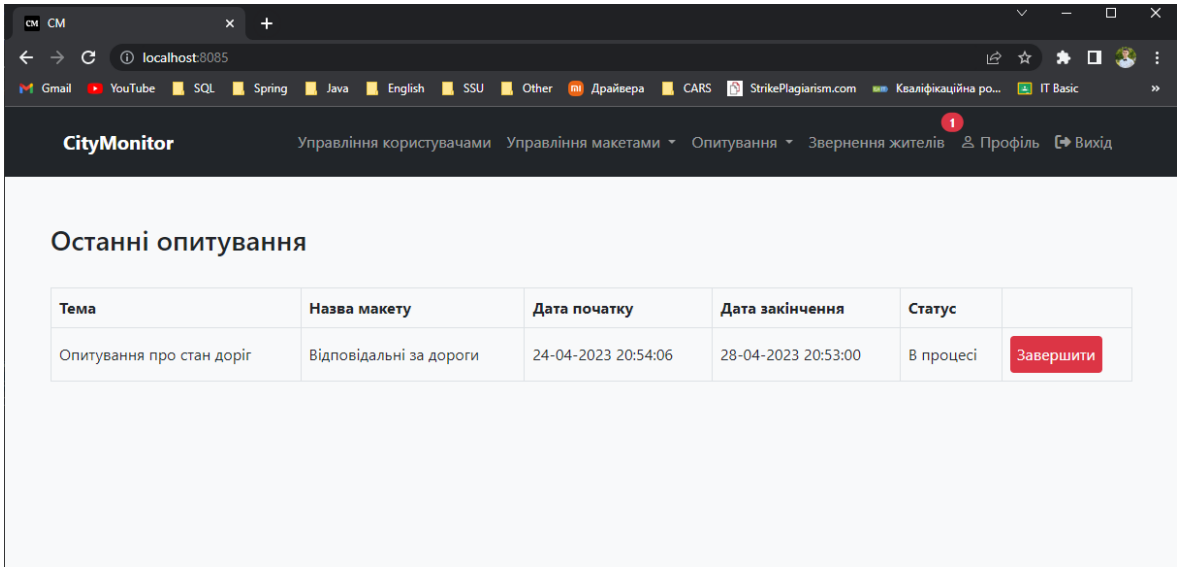

Рисунок 3.16 – Скріншот головної сторінки

Для створення опитування, спочатку необхідно підготувати макет районування (рис.3.17).

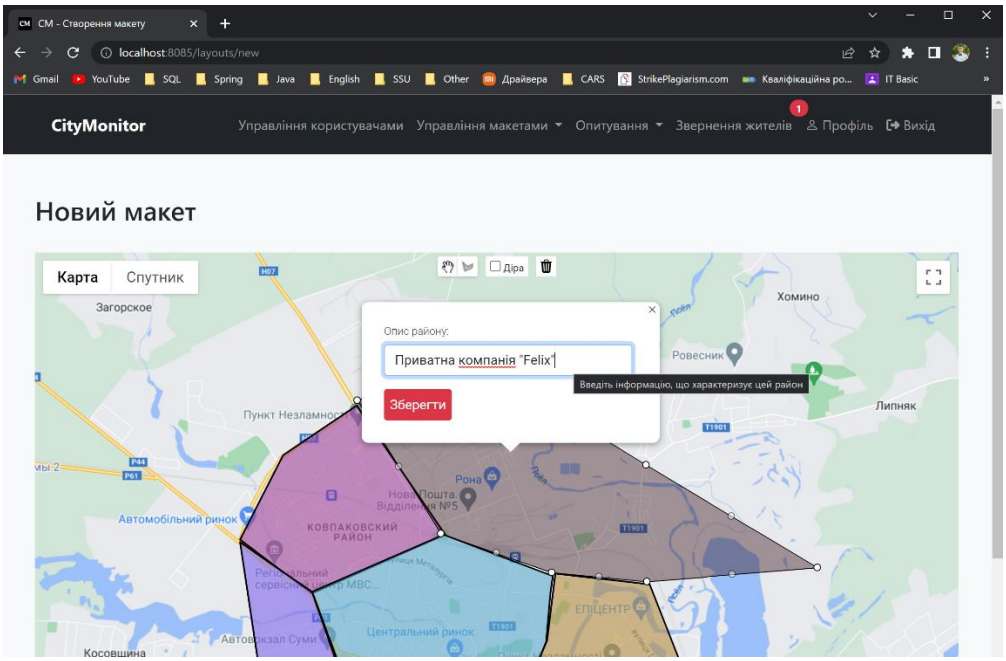

Рисунок 3.17 – Скріншот сторінки створення макету

На наступному рисунку показано приклад створення опитування (рис.3.18).

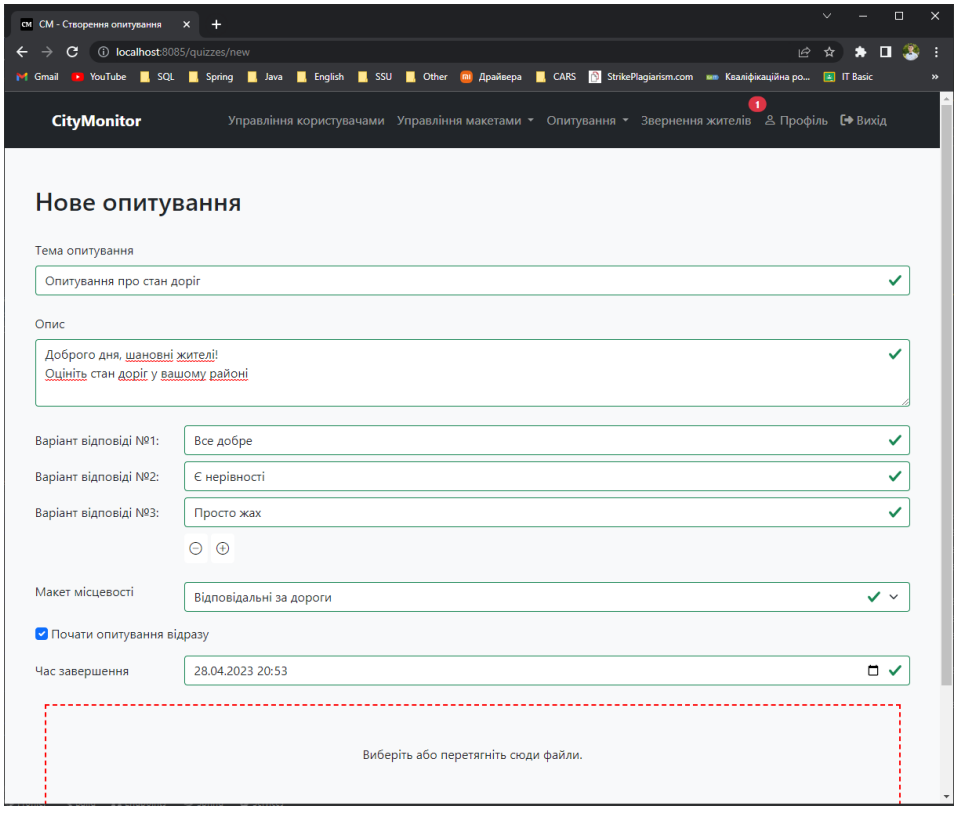

Рисунок 3.18 – Скріншот сторінки створення опитування

Після закінчення терміну опитування або примусового завершення користувач зможе побачити результати у такому вигляді (рис.3.19-21).

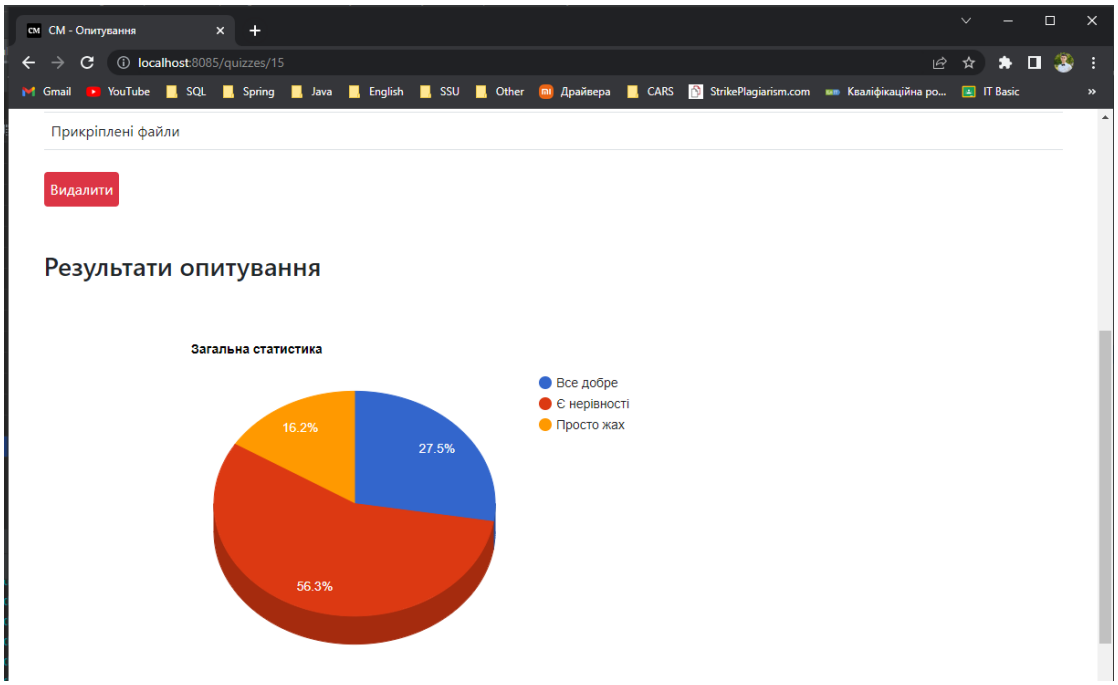

Рисунок 3.19 – Скріншот з загальними результатами

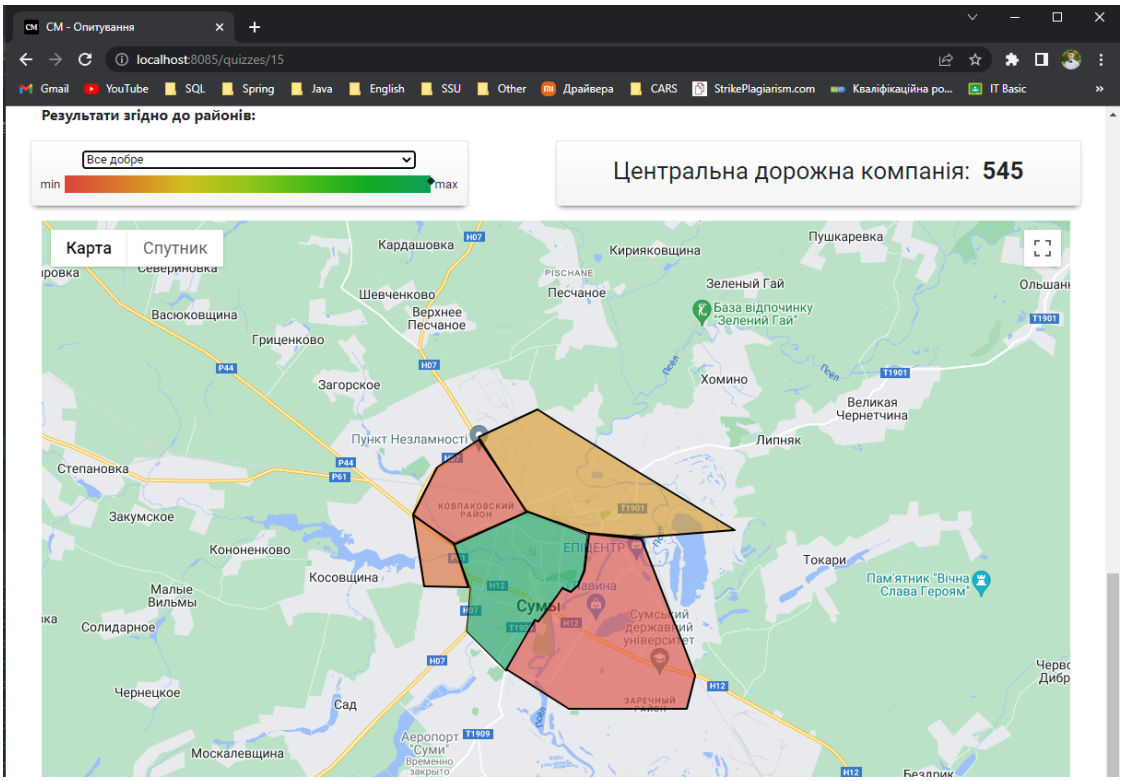

Рисунок 3.20 – Скріншот результатів на основі теплової карти

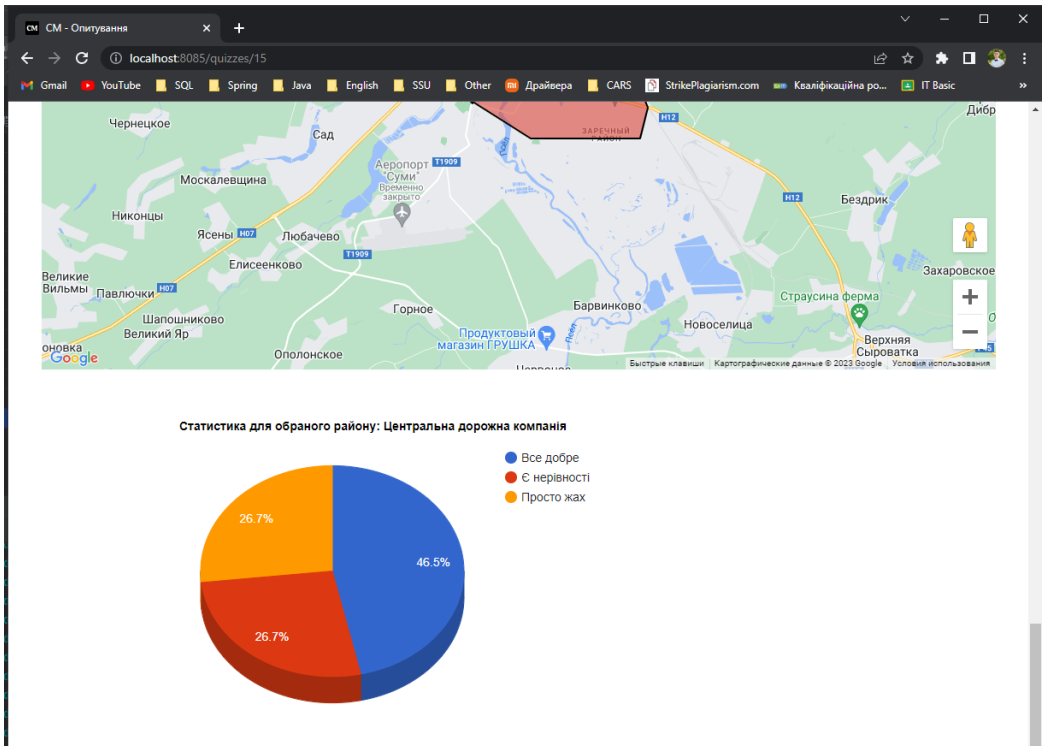

Рисунок 3.21 – Скріншот з результатами для обраної ділянки

На рисунку 3.22 наведено вигляд сторінки звернення.

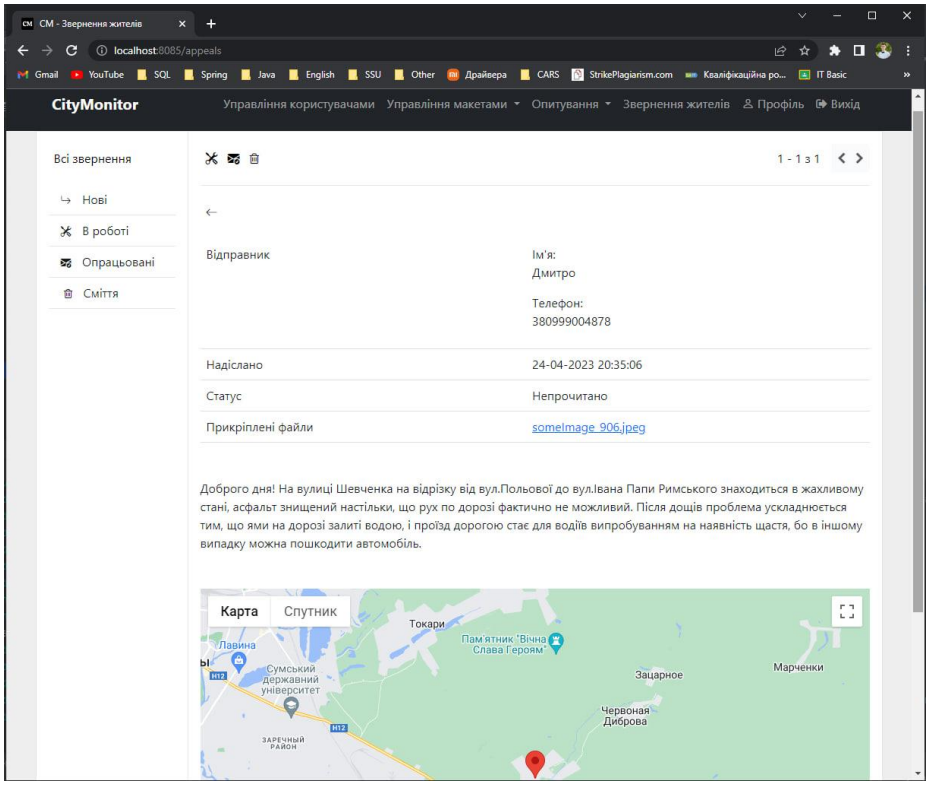

Рисунок 3.22 – Скріншот прикладу звернення

ВИСНОВКИ

В процесі виконання бакалаврської роботи було проаналізовано теоретичний матеріал за тематикою побудови інформаційних систем. Розглянуто такі існуючі рішення як SeeClickFix, CitizenLab та FixMyStreet, які частково відповідають вимогам даної роботи, але в той же час і мають деякі недоліки, які переважно полягають в зручності використання, через відсутність інтерфейсів взаємодії для мобільних пристроїв та української локалізації, недостачі функціоналу проведення опитувань у випадку SeeClickFix та FixMyStreet. Згідно проаналізованого матеріалу сформовано функціональні вимоги до системи. Ключовими є такі:

- система дозволятиме проводити опитування населення стосовно того чи іншого питання на виділених ділянках місцевості.
- жителі матимуть можливість брати участь в опитуваннях, вибравши одну з декількох відповідей.
- жителі матимуть можливість звертатися до адміністрації з окремих питань та отримувати відповіді на них.

Проаналізовано такі популярні інтерфейси взаємодії з системами як Web-інтерфейс, десктоп та мобільні додатки, чат боти. В результаті обрано ті, що найкраще підходять для вирішення поставлених задач: для працівників адміністрації - Web-інтерфейс, для жителів - Telegram-бот.

Побудовано схему багаторівневої архітектури майбутньої системи, виділеними частинами якої є: рівень доступу до даних – СУБД, рівень бізнес-логіки – Web-сервер, презентаційний рівень – Web-сторінки, Telegram.

Проаналізовано сучасні інструменти для розробки інформаційних систем. В результаті, мовою для back-end обрано Java, front-end – JavaScript. В якості системи управління базою даних буде використана СУБД PostgreSQL. Для полегшення розробки серверної частини додатку, прийнято рішення застосувати функціонал Spring Framework.

На основі сформованого переліку функціональних вимог, побудовано діаграму варіантів використання в нотації UML. Розроблено алгоритм роботи системи і продемонстровано його головні процеси взаємодії з нею за допомогою діаграм активності. Змодельована структура відношення сутностей на рівні БД та представлена у вигляді ERD діаграми.

Розроблена система CityMonitor служитиме основним інструментом збору думок населення з питань благоустрою населеного пункту, які в результаті будуть головним чинником прийняття рішень.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Інформаційні системи та їх роль в управлінні економікою [Електронний ресурс] // Ужгородський національний університет – Режим доступу до ресурсу: https://www.uzhnu.edu.ua/uk/infocentre/get/6742.

2. Інформаційні системи, їх види; апаратне та програмне забезпечення інформаційної системи. [Електронний ресурс] Режим доступу: http://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/95/95.html

3. Reach of mobile messenger apps among Android users in Ukraine [Електронний ресурс] // Statista. – 2022. – Режим доступу до ресурсу: https://www.statista.com/statistics/1188579/most-popular-messengers-in-ukraine/.

4. Database management systems: definition, types and benefits [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://uk.indeed.com/career-advice/career-development/database-management-systems.

5. DB-Engines Ranking of Relational DBMS [Електронний ресурс] – Режим доступу: https://db-engines.com/en/ranking/relational+dbms.

6. Рейтинг мов програмування 2022 [Електронний ресурс] // DOU.ua. – 2022. – Режим доступу до ресурсу: https://dou.ua/lenta/articles/language-rating-2022/.

7. Most Popular Backend Frameworks – 2012/2022 [Електронний ресурс] – Режим доступу до ресурсу: https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2022/.

8. Spring Framework Reference Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-framework/docs/3.2.4.RELEASE/spring-framework-reference/htmlsingle/.

9. Препарата, Франко; Шеймос, Майкл (1985). Computational Geometry – An Introduction. Springer-Verlag. ISBN 0-387-96131-3. 1ша редакція: ISBN 0-387-96131-3; 2ге видання, виправлене і доповнене (Розділ 2.2.1).

Додаток 1. Скрипт реалізації БД

```
-- Create tables section -------------------------------------------------

-- Table USERS

CREATE TABLE USERS(
   user_id Integer GENERATED ALWAYS AS IDENTITY,
   username VARCHAR(100) UNIQUE NOT NULL,
   password VARCHAR(300) NOT NULL,
   role VARCHAR(100) NOT NULL CHECK ( role IN ('SUPER_ADMIN', 'ADMIN', 'VIEWER')),
   account_status VARCHAR(100) NOT NULL CHECK ( account_status IN ('ACTIVE', 'NOT_ACTIVE'))
);

-- Table LOCALS

CREATE TABLE LOCALS(
                local_id Integer GENERATED ALWAYS AS IDENTITY,
                chat_id Integer UNIQUE NOT NULL,
                name Varchar(100) NOT NULL,
                phone Varchar(35) NOT NULL UNIQUE,
                location_point GEOMETRY NOT NULL,
                is_active BOOLEAN NOT NULL,
                PRIMARY KEY (local_id)
);

-- Table LAYOUTS
```

```
CREATE TABLE LAYOUTS(
        layout_id Integer GENERATED ALWAYS AS IDENTITY,
        name Varchar(200) NOT NULL,
        status VARCHAR(80) NOT NULL,
        PRIMARY KEY (layout_id),
        CHECK (status IN ('IN_USE', 'AVAILABLE', 'DEPRECATED'))
);

-- Table POLYGONS

CREATE TABLE POLYGONS(
        polygon_id Integer GENERATED ALWAYS AS IDENTITY,
        name Varchar(200) NOT NULL,
        polygon GEOMETRY NOT NULL,
        layout_id Integer NOT NULL,
        PRIMARY KEY (polygon_id)
);

-- Table QUIZZES

CREATE TABLE QUIZZES(
        quiz_id Integer GENERATED ALWAYS AS IDENTITY,
        title Varchar(220) NOT NULL, --JUST FOR ADMIN PANEL
        description VARCHAR(2200) NOT NULL,
        status VARCHAR(80) NOT NULL,
```

```
        start_date TIMESTAMP NOT NULL,
        end_date TIMESTAMP NOT NULL,
        layout_id Integer NOT NULL,
        PRIMARY KEY (quiz_id),
        CHECK (status IN ('PLANNED', 'IN_PROGRESS', 'FINISHED'))
);

-- Table OPTIONS

CREATE TABLE OPTIONS(
        option_id Integer GENERATED ALWAYS AS IDENTITY,
        title Varchar(1100) NOT NULL,
        quiz_id Integer NOT NULL,
        PRIMARY KEY (option_id)
);

-- Table ANSWERS

CREATE TABLE ANSWERS(
        answer_id BIGINT GENERATED ALWAYS AS IDENTITY,
        option_id Integer NOT NULL,
        local_id Integer NOT NULL,
        quiz_id Integer NOT NULL,
        PRIMARY KEY (answer_id)
);

-- Table APPEALS
```

```
CREATE TABLE APPEALS(
        appeal_id Integer GENERATED ALWAYS AS IDENTITY,
        text Varchar(4000),
        status Varchar(80) NOT NULL,
        post_date TIMESTAMP NOT NULL,
        point GEOMETRY,
        local_id Integer NOT NULL,
        PRIMARY KEY (appeal_id),
        CHECK ( status IN ('UNREAD', 'VIEWED', 'PROCESSED',
'IN_PROGRESS', 'TRASH'))
);

-- Table RESULTS

CREATE TABLE RESULTS(
        option_id Integer NOT NULL,
        answers_count Integer NOT NULL,
        result_id Integer GENERATED ALWAYS AS IDENTITY,
        polygon_id Integer,
        PRIMARY KEY (option_id,result_id)
);

--Table FILES

CREATE TABLE FILES(
        file_id Integer GENERATED ALWAYS AS IDENTITY,
        name VARCHAR(200),
        tg_file_id VARCHAR(150) UNIQUE,
```

```
        file_object OID,
        appeal_id INTEGER,
        quiz_id INTEGER,
        PRIMARY KEY (file_id)
);
-- Create foreign keys (relationships) section ------------------------------------
----------
```

ALTER TABLE FILES ADD CONSTRAINT file_appeal_fk FOREIGN KEY (appeal_id) REFERENCES APPEALS (appeal_id) ON DELETE CASCADE;

ALTER TABLE POLYGONS ADD CONSTRAINT polygon_layout_fk FOREIGN KEY (layout_id) REFERENCES LAYOUTS (layout_id) ON DELETE CASCADE;

ALTER TABLE QUIZZES ADD CONSTRAINT quiz_layout_fk FOREIGN KEY (layout_id) REFERENCES LAYOUTS (layout_id);

ALTER TABLE OPTIONS ADD CONSTRAINT option_quiz_fk FOREIGN KEY (quiz_id) REFERENCES QUIZZES (quiz_id) ON DELETE CASCADE;

ALTER TABLE ANSWERS ADD CONSTRAINT answer_option_fk FOREIGN KEY (option_id) REFERENCES OPTIONS (option_id);

ALTER TABLE ANSWERS ADD CONSTRAINT answer_local_fk FOREIGN KEY (local_id) REFERENCES LOCALS (local_id);

```
ALTER TABLE APPEALS ADD CONSTRAINT appeal_local_fk FOREIGN KEY (local_id) REFERENCES LOCALS (local_id);

ALTER TABLE RESULTS ADD CONSTRAINT result_option_fk FOREIGN KEY (option_id) REFERENCES OPTIONS (option_id) ON DELETE CASCADE;

ALTER TABLE RESULTS ADD CONSTRAINT result_polygon_fk FOREIGN KEY (polygon_id) REFERENCES POLYGONS (polygon_id);

ALTER TABLE FILES ADD CONSTRAINT file_quiz_fk FOREIGN KEY (quiz_id) REFERENCES QUIZZES ON DELETE CASCADE;

ALTER TABLE ANSWERS ADD CONSTRAINT  answer_quiz_fk FOREIGN KEY (quiz_id) REFERENCES QUIZZES ON DELETE CASCADE;

INSERT INTO USERS VALUES (DEFAULT, 'sysadm', '$2y$12$fa1H8AwmY42d0B3nBAusLOvOdk7O1GGPOKBhkmonqvLCMAM8IszAy', 'SUPER_ADMIN', 'ACTIVE');
```