МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

Кваліфікаційна робота магістра

ГЕОІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОБРОБЛЕННЯ ЗАПИТІВ НАСЕЛЕННЯ В УМОВАХ НАДЗВИЧАЙНИХ СИТУАЦІЙ ВІЙСЬКОВОГО ХАРАКТЕРУ

Здобувач освіти гр. ІН.м-12ан/2у Поліна ПЕРЕХРЕСТЮК

Науковий керівник,
доцент, к.т.н. Наталія БАРЧЕНКО

В.о. завідувача кафедри,
доцент, к.т.н. Ігор ШЕЛЕХОВ

Суми 2022

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«122 - Комп'ютерні науки»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Перехрестюк Поліні Олегівні
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Геоінформаційна технологія оброблення запитів населення в умовах надзвичайних ситуацій військового характеру*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи)_____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Огляд технологій, що дозволять реалізувати поставлену задачу; 2) Постановка завдання й формування завдання дослідження; 3) Огляд технологій, що використовуються під час розробки додатків на ОС Windows, Python, Django; 4) Моделювання та проектування; 5) Програмна реалізація; 6) Аналіз результатів.

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 _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
	Огляд наявних рішень		
	Постановка задачі та формування завдань дослідження.		
	Проектування та моделювання		
	Програмна реалізація		
	Оформлення пояснювальної записки до кваліфікаційної магістерської роботи		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 64 стор., 45 рис., 2 таблиці, 1 додаток, 15 літературних джерел.

Об’єкт дослідження — алгоритм та методи обробки аналітичних даних про руйнування та пошкодження власних та державних будівель під час військових дій.

Мета роботи — розробка веб-сервісу для громадян України з інтеграцією API мап для публікування запитів та надання допомоги після військових дій.

Результати — розроблено метод та веб-сервіс для збору та обробки аналітичних даних про ділянки руйнування завданих агресором під час військових дій. Також для публікування запитів від місцевого населення та надання допомоги під час або після війни.

ВЕБ-СЕРВІС, АНАЛІЗ СТАТИСТИЧНИХ ДАНИХ, ВБУДОВАНЕ API МАП, ПУБЛІКАЦІЯ ЗАПИТІВ ПРО ДОПОМОГУ, PYTHON, DJANGO, LEAFLET, SQL, INTERACTIVE WEB MAP.

ЗМІСТ

ВСТУП

Чи може людина у двадцять першому сторіччі знати напевно що її чекає завтра? Чи може вона із впевненістю сказати, що день, який на неї чекає, буде кращим або принаймні не гіршим? Мабуть, що в переважній більшості, так. Але для українців від сходу до заходу у 2022 році така впевненість знизилася до мінімуму. Насправді, ніхто навіть і не уявляв, але реальність з її вразливою геополітикою змушує пристосовуватись до цих обставин і оперативно вирішувати проблеми сьогодення на побутовому рівні задля виживання.

З початком широкомасштабних воєнних дій на території України невпинно працюють різні комунальні служби та інші працівники різних сфер з відновлення інфраструктури та житлових будівель, але найшвидше люди отримують допомогу саме один від одного, адже іноді влада просто не в змозі охопити одразу весь масштаб запитів.

Сумський регіон – у трійці лідерів з руйнувань, спричинених військовою агресією рф [1]. За програмою "fast recovery" [2], на реалізацію якої Сумщина отримала 250 млн грн, наразі відновлено 24 багатоквартирних будинка, 143 інфраструктурних об’єкта, 9 медичних закладів, 35 закладів освіти, 13 об’єктів життєзабезпечення, 4 адмінбудівлі та 7 інших будівель. Роботи досі тривають. Не дивлячись на це, досі є велика кількість запитів від населення про відновлення домівок і про звичайні потреби в медикаментах, продуктах для різних вікових груп, а також перевезеннях.

Тому ця робота присвячена розробці веб-сервісу, за допомогою якого користувачі з будь-яких областей України матимуть змогу без особливих зусиль самостійно створювати запити про допомогу різного виду: від матеріальної, до гуманітарної і, можливо, просто моральної. У користувачів буде можливість позначити місце руйнування, а ці данні будуть відображатися на карті, аби в подальшому використовувати ці дані не тільки для надання допомоги, а і для проведення аналізу масштабів завданих руйнувань та збитків. Це також може привернути увагу закордонних і українських

інвесторів, які хотіли б зробити внесок у відновлення країни, адже руйнувань, на жаль, зазнали не тільки особисте і державне майно, а і культурна спадщина регіонів.

Новизна та основна перевага даного веб-сервісу полягатиме у інтеграції з картою місцевості задля більшої наочності та ефективнішому аналізу обстрілів. Можливо, деякі деталі про руйнування, такі як близькість до кордону, матеріали будівель, види озброєнь, тощо стануть в нагоді для подальшого безпечнішого будівництва або відновлення наявних споруд.

У перспективі дана робота допоможе дати відповіді на два важливих запитання:

1. Які регіони України зазнали найбільших руйнувань від російської агресії?
2. На скільки ефективно можна допомагати населенню кооперуючи зусилля влади і небайдужих громадян?

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Літературний огляд

1.1.1 Чому саме веб-застосунок ?

Щоб зрозуміти чому саме веб-застосунок, потрібно подумати про те, які альтернативи взагалі доступні. Залежно від конкретного випадку використання, це може бути будь-яке з наступного:

- Використання веб-сайту.
- Створення нативної програми.
- Використання локального програмного пакета.

Маючи на увазі усі ці альтернативи, можна перевірити деякі конкретні переваги розробки веб-додатків, особливо щодо локального програмного забезпечення або власних програм.

Швидке розгортання.

Створити та запустити веб-програму неймовірно легко. Загалом, існує дуже багато труднощів, які потрібно подолати, але все ж також набагато більше інструментів і фреймворків, які надають гнучкості при розробці, які можна використовувати. Наприклад, у випадку створення нативної програми для мобільних пристроїв і відправлення її в Apple App Store або Google Play Store, розробка буде набагато більше обмежена власними фреймворками, такими як Xamarin і PlayKit, для програм Android і IOS відповідно. У випадку Apple також доведеться пройти тривалий процес перевірки наданої заявки.

Подібним чином, створення локальної настільної програми, потрібно буде фактично розгорнути її для своїх користувачів і встановити на їхніх машинах. У великій організації це може зайняти дні або навіть тижні.

Простий доступ.

Веб-програми дозволяють розгорнути набагато простіше і швидше. Щоб зробити додаток доступним користувачам, просто потрібно надіслати їм URL-адресу. Так само розробка веб-додатків є розумним вибором, якщо потрібно полегшити користувачам пошук і використання ваших інструментів. Сьогодні

більшість користувачів роблять усе зі своїх веб-браузерів – навіть у професійному контексті.

Зручність веб-додатків полягає в тому, що вони можуть отримати доступ до інструментів користувача з будь-якого веб-браузера. Якщо, звичайно, не було вживано конкретних заходів, щоб обмежити це. Подібним чином клієнти можуть насолоджуватися однаковими враженнями незалежно від того, користуються вони телефоном, комп'ютером чи планшетом.

Об'єднуючи це, розробка веб-додатків дає змогу забезпечити високий рівень зручності для користувачів. Коли смартфони були ще в зародковому стані, більшість брендів прагнули розробляти власні мобільні програми. Єдина проблема в тому, що більшість людей цього не хочуть. Завантаження нових програм дратує. Так, це веде до засмічення головного екрану смартфону та скорочення часу автономної роботи. Здебільшого клієнти не збираються завантажувати програму, якщо це не те, чим вони будуть користуватися щодня. *Сьогодні більшість людей віддають перевагу веб-додаткам.* Принаймні, якщо немає вагомих причин для використання вбудованої програми.

Менші витрати на розробку.

Зазвичай розробка веб-додатків дешевша та швидша, ніж створення власних додатків або програм для настільних ПК. Значною мірою це пояснюється тим, що не потрібно виділяти додаткові ресурси для вивчення власних фреймворків, проходження процесів перевірки або розробки локальних інсталяційних пакетів.

Окрім цього, розробка веб-додатків зазвичай вимагає значно менше індивідуальної роботи, ніж інші види програмного забезпечення. Частково це відбувається завдяки різноманіттю фреймворків, інтерфейсних бібліотек та інших інструментів, які прискорюють розробку. Більше того, багато розробників веб-додатків все більше переходять на інструменти з простішим кодом, щоб ще більше прискорити розробку.

Чи є у веб-додатків якісь недоліки?

Звичайно, бувають ситуації, коли ви немає можливості вибирати інструмент на основі браузера. Дійсно, є кілька обмежень, про які потрібно знати, перш ніж ви зможете прийняти зважене рішення. Основна річ - залежність від доступу до мережі Інтернет. Загалом, хоча й не завжди, веб-програми вимагатимуть від користувачів стабільного підключення до Інтернету. Щонайменше, їм зазвичай потрібно бути онлайн, щоб отримати повну функціональність. Або, якщо є можливість розгорнути свої інструменти на локальних серверах, у цьому випадку користувачам потрібно буде бути у спільній з розробником мережі, щоб отримати доступ. Це не означає, що все це обов'язково є проблемою.

Обмеження функціональності.

За інших рівних умов веб-програми також матимуть певні функціональні обмеження, особливо щодо апаратного забезпечення та інших власних функцій на певних пристроях. Класичним прикладом цього може бути використання камери чи мікрофона на певних пристроях. Швидше за все, можна побачити обмеження щодо здатності вашої програми взаємодіяти з певними аспектами вашої ОС і конфігурації. Наприклад, такі речі, як push-сповіщення, які працюють по-різному на різних платформах. У більшості випадків досягти певної функціональності буде неможливо. Швидше за все, це буде просто складніше, і можете з'явитися необхідність прийняти менш елегантне рішення.

Утримання користувачів.

У контексті B2C (англ. Business-to-Consumer) також потрібно думати про те, як розробка веб-застосунків вплине на утримання користувачів. Гарним варіантом буде взяти до уваги термін CRO. CRO – conversion rate optimization, іншими словами – оптимізація коефіцієнта конверсії. Даний коефіцієнт потрібен для підвищення ймовірності конверсії на сайті, збільшення загального обсягу конверсії і відсотку конвертованих користувачів. Раніше було зазначено, що сучасні користувачі не хочуть, щоб на їхніх пристроях зберігалася велика кількість програм.

1.1.2 Шляхи створення веб-застосунків

Розробка веб-застосунків представляє новий підхід до створення динамічних веб-сторінок. Саме сучасний підхід до створення ефективних діджитал рішень зробив веб-додатки сьогодні більш інтерактивними, приємними до ока та ефективними. Більш того, з появою сучасних фреймворків створення веб-додатків стало значно легшим і швидшим.

Різниця між архітектурою програмного забезпечення та дизайном програмного забезпечення.

Багато хто вважає, що архітектура програмного забезпечення та дизайн програмного забезпечення взаємозамінні речі, але це не так. Архітектура програмного забезпечення — це скелет і всі високорівневі компоненти системи та те, як вони взаємодіють один з одним. Відповідає на запитання «Де?» та «Як?». З архітектурою програмного забезпечення легше побачити загальну картину. Основна мета цього — визначити функціональні вимоги та вимоги до якості та впоратися з ними для покращення загальної якості програми. Отже, в цілому, за допомогою архітектури програмного забезпечення можна контролювати продуктивність, можливість масштабування і надійність.

Говорячи про дизайн програмного забезпечення, то він більше стосується дизайну на рівні коду, і він відповідає за функціональність кожного модуля та його цілі. Це «як» процесу розробки програмного забезпечення. Після того як був пройдений етап архітектури, настає час для розробника програмного забезпечення подумати про функції, класи, інтерфейси та інші деталі програми. Дизайн програмного забезпечення – це рівень деталей або компонентів. Наприклад, створення API. Рівень розробки програмного забезпечення – це саме той момент, коли пишеться специфікація API. Тому, коли справа доходить до етапів кодування, розробники зовнішнього та внутрішнього програмного забезпечення можуть працювати відповідно до цієї специфікації. Таким чином, за допомогою дизайну програмісти мають

ефективну спільну мову, щоб знаходити рішення повторюваних проблем і концептуалізувати їх. Це мінімізує обсяг роботи, яку вони виконують, оскільки немає потреби винаходити колесо.

Схема процесу користувач-сервер може пояснити суть архітектури веб-застосунку:

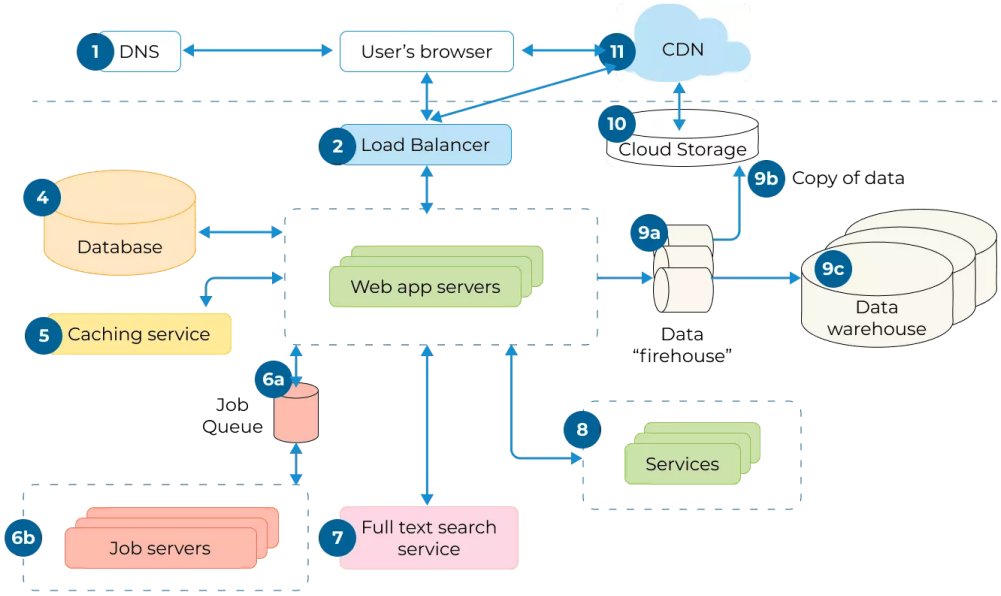

Рис. 1.1 – Схема архітектури веб-застосунку

1.1.3 Фреймворки розробки веб-застосунків

Мета фреймворків — зробити розробку веб-застосунків легшою та швидшою, ніж кодування їх з нуля.

Фреймворки веб-застосунків є самовпевненими, і кожен має свою філософію та переваги. Вони бувають двох видів: бекенд і інтерфейсні. Наведені нижче зовнішні фреймворки насправді взагалі не є; вони представляють лише рівень перегляду веб-програми. Але для простоти ми будемо називати їх фреймворками.

Бекенд фреймворки:

- Rails. Rails описує себе як «фреймворк веб-застосунків, який містить усе необхідне для створення веб-застосунків із базою даних відповідно до шаблону Model-View-Controller (MVC)». Rails є чудовою структурою для метапрограмування (де комп'ютерна програма може розглядати інші програми як свої дані) та веб-

програмування, орієнтованого на бази даних. Rails є ідеальним фреймворком для невеликих проектів.

- Django. Django описує себе як «веб-фреймворк Python високого рівня, який заохочує швидку розробку та чистий, прагматичний дизайн». Даний фреймворк гарно підходить для тих, хто займається науковим програмуванням або маніпулюванням даними. Також, він є одним із найпростіших фреймфорків для вивчення і застосування на практиці. Вміщує в собі велику кількість інструментів для управління сервером, наприклад зручна адміністративна панель.
- Laravel описує себе як «фреймворк веб-програми з виразним, елегантним синтаксисом». Laravel написаний на PHP - мові програмування. Laravel слідує архітектурному шаблону «модель-перегляд-контролер». Laravel має багато інструментів, які роблять його доступним і простим у використанні. Він добре підходить для різних типів додатків.

Фронтернд фреймворки:

Усі наступні інтерфейс фреймворки написані на JavaScript.

- React. React просто описує себе як «бібліотеку JavaScript для створення інтерфейсу користувача». Це дуже простий, але влучний опис React. Це потужна інтерфейс бібліотека, створена та підтримувана Facebook. З усіх перерахованих інтерфейсних фреймворків React є найпопулярнішим і найпотужнішим. Вона добре підходить для масштабних веб-проектів. Знову ж таки, для малих і середніх проектів варто обрати інший фреймворк.
- Vue. Vue описує себе як «прогресивний фреймворк JavaScript». Vue менший за розміром і легший для вивчення, ніж React, і підходить для більшості проектів. Також, його з легкістю можна впровадити у проект, що є перевагою.

- Svelte. Svelte описує себе як «веб-застосунок, покращений кібернетичною системою». Svelte — це новачок у світі, який є компілятором, а не фреймворком. Це означає відсутність віртуального DOM, фреймворків поверх фреймворків і фреймворків для завантаження під час виконання, що робить неймовірно продуктивні веб-програми. Синтаксис Svelte робить фреймворк найлегшим для вивчення серед згаданих інтерфейс фреймворків і ідеально підходить для малих і середніх веб-застосунків. Але, на жаль, це не доведено для великих веб-застосунків. Спільнота та екосистема менші, ніж React і Vue, але вона досі розвивається.

1.1.4 Огляд технологій інтеграції карт

Перш за все треба чітко розуміти – це вид програмного продукту, який розробляється, для чого він і для кого. Це може бути відмітка певної локації на карті, прокладання маршруту від точки А до точки Б, або ж це пошук за координатами. Проблема у тому, що не всі сервіси підтримують роботу в офлайн режимі, якщо це необхідно. Не всі карти мають можливість наносити на них користувацькі дані. Також, буває варіант використанні растрових, або ж векторних карт, які мають потужний потенціал до масштабування.

Ззовні векторні та растрові електронні карти можуть виглядати однаково. Ті самі контури та лінії, назви об'єктів та залиті кольором простори суші та акваторій. Але варто суттєво змінити масштаб зображень, як картина радикально зміниться. На растрової карті з’являється розмитість ліній та інші дефекти. Що ж відбувається при масштабуванні на векторній карті? По-перше, з'являються нові деталі та об'єкти нанесені на електронні карти, яких за дрібного масштабу не було, тому що вони знаходяться на іншому інформаційному шарі карти. При цьому всі лінії та контури залишаються. При наведенні курсору на об’єкт з’являється вікно з чіткою назвою об’єкта та його характеристиками. Цей процес, який отримав назву ідентифікації або

розпізнавання об’єкта, дуже важливий для навігаційного використання карти. Кожен об’єкт векторної карти та її атрибути мають певні коди, з якими відповідним чином має реагувати навігаційна система під час її використання.

Основна перевага векторних карт над растровими полягає у можливості розвантаження карти від зображень деяких фізичних об’єктів, що не впливають на безпеку руху. Завдяки чому карта стає наочною і добре зчитуваною. Растрова ж карта є більшою мірою просто «картинкою». Безумовно, вона є корисною для спостереження навколишнього оточення, але цим усі її переваги та можливості вичерпуються. Діапазон її масштабування дуже вузький. Звичайно, растрову карту можна використовувати для навігації, помітивши на ній місце розташування. Але ряд важливих навігаційних завдань, таких як розробка попереджень при наближенні до небезпек, вирішити в системах, які використовують растрову карту, неможливо. В них також дещо ускладнене стикування растрових карт різних масштабів та перетворення проекцій, тощо. У них зберігаються суттєві обмеження щодо налаштування відображення на екрані. Більш того, у них неможливо змінити інформаційне навантаження карти та отримати прийнятну з ергономічної точки зору денну та вечірню палітру кольорів.

Векторна карта зі зміною масштабу у більш ширших межах дозволяє реалізувати динамічну електронну навігацію, яка може показати користувачеві безліч поточних автоматичних попереджень, наприклад, під час наближення до небезпеки або досягнення пункту призначення. У системах з векторними картами є унікальна функція попередження водія про небезпечний курс та обчислення безпечних курсових секторів. Саме завдяки широким функціональним можливостям векторних карт у даний час всі сучасні навігаційні системи використовують тільки їх. Але, за нові можливості векторних електронних карток доводиться розплачуватись складністю логічної структури їх даних та збільшеною трудомісткістю створення векторних карток.

Другим важливим пунктом про що треба подумати – це чи буде використовуватись мобільна адаптація продукту або навіть чи буде створений окремий мобільний додаток. Тоді варто звернути увагу на те, що кожний телефон має власну потужність і у пересічних громадян немає потужного смартфону з багатоядерним процесором. Отже, треба звернути увагу на використання простих бібліотек, таких як LeafLet, натомість відмовитись від дещо потужних гігантів типу Google Maps, бо вони, наприклад, можуть набагато довше завантажуватись на смартфоні із невеликою пам’яттю.

Третє, на що необхідно звернути увагу – це регіони використання. Цілком ймовірно, що географія використання впирається не в країни, а в міста або навіть селища міського типу. Тоді варто порівняти ту саму точку з точки зору різних картографів. Може таке трапитись, що у когось у селі Мокіївці прописані вулиці, а хтось грішить сірими областями, але ж у вас додаток для фермерського господарства, яке спеціалізується на зборі молока у місцевого населення.

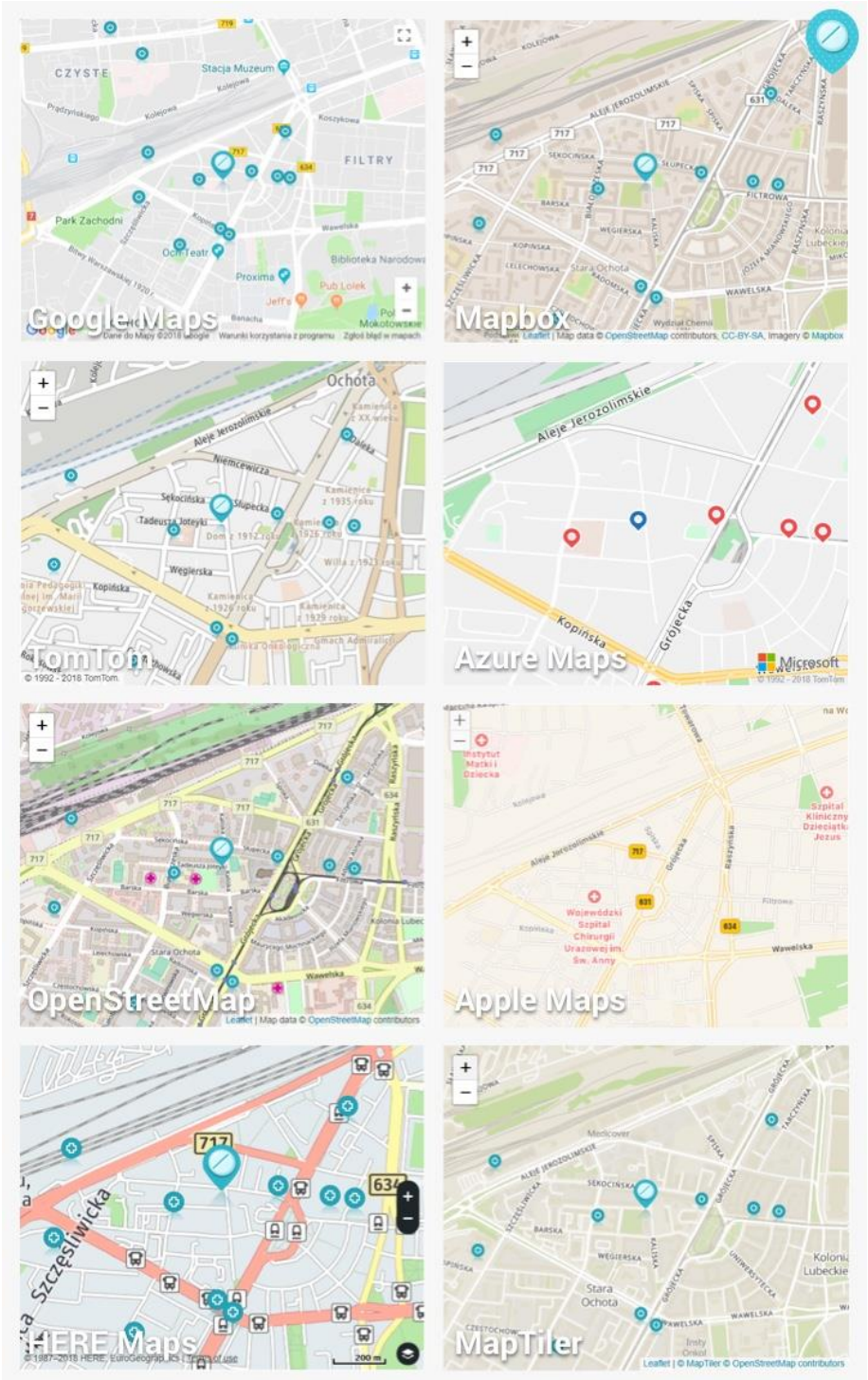

Рис. 1.2 – Як приклад аптека на картах у польському містечку [**Error! Reference source not found.**]

Тож, варто перевірити яким саме картам віддають перевагу місцеве населення у потрібних ареалах. Можливо, можна зробити чудову маршрутизацію, додати різнокольорові шари з купою інформації, а користувачам буде просто незвично використовувати її, вони будуть

плутатись. І ось через це можуть бути потенційні відмови від користування веб-сервісом.

Важливою складовою при виборі картографічного інструменту є запланований бюджет. Треба зважати на те, що не всі карти є однаково безкоштовними. До прикладу польський стартап, у якому йдеться якраз про гроші: у їхньому випадку тариф 0,5$ для комерційного використання збільшився до $7 (у 14 разів). Це, звісно, істотно збільшило витрати на обслуговування продукту. Відіграла роль і нова система оплати, pay-as-you-go: при постоплаті GdziePolek втратили можливість планувати витрати. Втім, тут є один делікатний момент: якщо у веб-сервісу багато завантажень, кількість користувачів зростає не щодня, а щогодини, а грошей на оплату карток немає – то, можливо, треба замислитись нас стратегією масштабування?

Якщо розроблений сервіс є у вільному доступі і будь-хто може в ньому зареєструватися – то скоріш за все вбудовані карти отримані безкоштовно від будь-якого провайдера, навіть від провідних гравців. В іншому випадку (наприклад, якщо це платний додаток для бухгалтерів) доведеться використовувати комерційну ліцензію. І тут варто чітко розуміти, яку частку функціональності займають карти і скільки звернень до них з боку користувача буде на місяць.

1.2 Огляд наявних рішень для інтеграції мап

Спираючись на попередній розділ можна виділити декілька провайдерів мап, які можна було б використати під час створення проекту.

Варто почати, мабуть, з найпопулярнішого продукту на ринку – **Google Maps** [3]. Мапи Google є об’єктивно найкращим продуктом на своєму ринку, який за останні роки випередив конкурентів. Джастін О'Бірн (австралійський журналіст) чудово проаналізував, наскільки просунуті мапи Google порівняно з іншими рішеннями. Інтеграція API Google Maps доволі проста, документація повна і зрозуміла. Істотний мінус – ціна. Більш того, дані мапи насправді не

служать ні для чого іншого, крім ефективного показу розташування певних міток на мапі. Інші розширені функції, такі як супутникові знімки, маршрутизація та інші просто не будуть використовуватись с огляду на ціль веб-сервісу, та і швидше за все, їх буде важко замінити іншим рішенням.

Інтерфейс прикладного програмування (API) — це набір функцій і процедур, які дозволяють створювати програми для доступу до функцій або даних операційної системи, програми чи іншої служби. API є механізмом для дослідження взаємозв'язку різного програмного забезпечення та обміну форматами, такими як XML і JSON. Google Maps (GM) використовує веб-сервіс для обміну даними з API. GM API — це набір додатків експонування за протоколом HTTP, що дозволяє здійснювати запити геоданих сховища даних [4]. З різних варіантів можна отримати інформацію про вихідні та пункти призначення у даних із геоприв'язкою або десятковими градусними координатами. Відстань і час у дорозі також надає Google Maps Distance Matrix API. Цю інформацію можна отримати в режимі реального часу або зробити прогноз на основі даних середнього трафіку. Щоб гарантувати якість геоданих, пов'язаних із запитом про відстань і час подорожі, необхідно, щоб інформація, яка надсилається до GM API, мала мати правильні посилання та кутові координати (широта та довгота). Пізніше алгоритм виконує запит «Distance MatrixAPI», беручи точки відправлення та призначення для оцінки в матриці відстані. API повертає значення запиту в потрібному форматі (XML або JSON). Потім дані автоматично обробляються за допомогою циклів на обраній мові програмування. Нарешті, інформація зберігається в геобазі даних. На рисунку 2 показано процес, прийнятий для отримання інформації з карт Google.

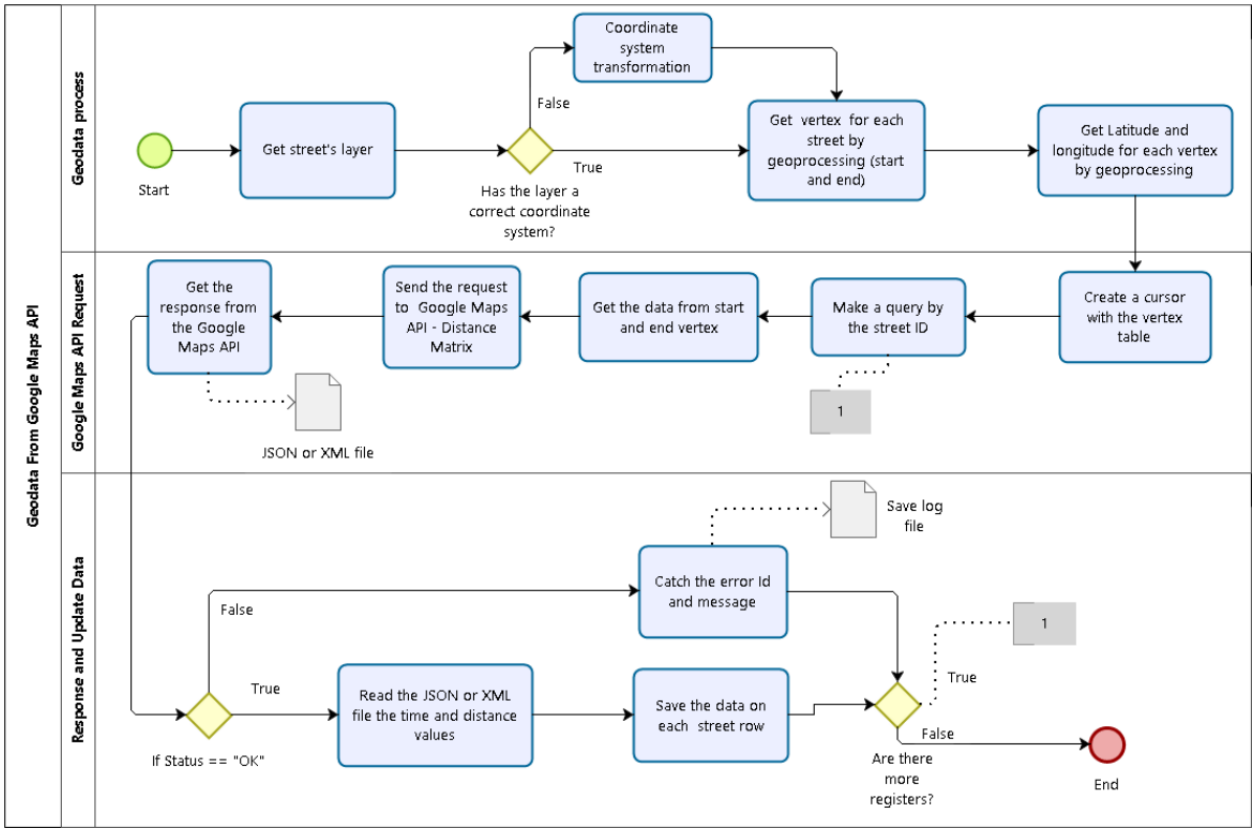

Рис. 1.3 - Зображує три фази процесу отримання геоданих: обробка геоданих, запити API Google і оновлення геоданих.

На жаль, Google Maps має обмеження щодо використання API матриці відстані. Веб-застосунок дозволяє лише 2500 запитів на день у рамках безкоштовної підписки, що запобігає перегляду великих обсягів інформації.

OpenStreetMap [5] – сервіс з відкритим кодом, що набирає все більшої популярності. Дані створюються та вносяться безпосередньо учасниками спільноти, що допомогло створити так звану «Вікіпедію» для мап. Даний сервіс набув широкого розголосу в ЗМІ як ефективний варіант кризової мапи, коли в 2010 році після землетрусу на Гаїті тисячі користувачів підхопили процес актуалізації мапи регіону з урахуванням рельєфу, що змінився, і нанесенням точок, де постраждалі могли отримати допомогу. Основна мета OpenStreetMap – це не тільки малювання мапи, а й створення бази даних на її основі і наповнення точок інформацією. Тому на базі OpenStreetMap можна створити безліч цікавих карт: велосипедні маршрути, мапи курортів з визначними пам'ятками тощо. Ці мапи безкоштовні, але є свої тонкощі

технічного характеру (наприклад, обмеження кількості запитів геокодування в секунду до сервера OpenStreetMap). Плюс незручність можуть доставити всі додаткові підключення до мережевого проекту.

Mapbox [6] – ще один цікавий сервіс, який можна назвати своєрідною «обгорткою» над OpenStreetMap, що дає відмінну деталізацію мап. До того ж Mapbox в новинних зведеннях часто називають картографічним сервісом, що найбільш енергійно розвивається. Вартість використання, як і в більшості конкурентів, формується виходячи з кількості запитів до карти за певний період. Є можливість використання безкоштовно, якщо за місяць у вас буде до 50 тисяч: активних користувачів, запитів на геокодування, запитів на створення напрямків. Якщо картою через ваш додаток скористалося більше людей та сформовано більше запитів, то доведеться заплатити 0,5$ за кожних 500 користувачів та тисячу запитів за системою pay-as-you-go. CEO Mapbox Ерік Гундерсен в одному з інтерв'ю [0] сказав: «Якщо робиш технології для машин, то спілкуйся з автомобільними інженерами. <...> По суті ми робимо карти не для людей, а для роботів. Вони спроектовані з урахуванням можливостей автомобільних сенсорів, радарів та інших систем.»

HERE WeGo Maps [7] – геолокаційна платформа, заснована Nokia (зараз належить консорціуму, до якого входять компанії Audi, BMW, Daimler AG, Intel, Navinfo, NVIDIA, Pioneer, Bosh, Continental). Покриття задовільне, зокрема в Україні. Але карти в нашій країні, однак, не страждають на подробиці: ви можете прокласти маршрут, побачити номери будинків, проте POI (point of interest, пам'ятки, магазини, офіси, заправки і так далі) часто виявляються неактуальними. Тому якщо програма буде орієнтована на пішоходів або пошук організацій (особливо в невеликих містах), Here в Україні – це не найкращий провайдер. Безумовним плюсом Here можна назвати пробний період у 90 днів. Розробники особливо звертають увагу, що ключ на 90 днів дається без відправки своїх платіжних даних в Here (такий легкий кивок у бік Google з їхньою вимогою показати кредитку, навіть якщо ви розраховуєте ніколи не виходити за межі ліміту $200).

LeafLet [8] та **OpenLayers** [9]. Дані бібліотеки можуть надати повну свободу, є можливість розгорнути власну мапу на базі OSM на власному сервері з потрібними опціями. Обидві бібліотеки мають відкритий вихідний код JavaScript. Мапи можуть бути кастомізовані за будь-яким бажанням. Немає залежностей від маркетингових рішень вендора типу Google. Leaflet вдало зайняв порожню нішу у правильний момент, ставши фактично стандартом індустрії. Тепер його використовують такі компанії, як Facebook, GitHub, NASA, Європейська комісія, Reuters, Washington Post тощо. Leaflet не потребує від програміста досвіду роботи з сервісами мап, значно спрощує вбудову мапи на html-сторінки або веб-застосунки. Бібліотека дозволяє працювати з різними шарами, в якості джерел мапи використовувати будь-якій загальнодоступний веб-сервіс порізаних зображень мапи, так званих тайлів. Також є можливість завантажувати дані та накладати їх з GeoJSON файлів, змінювати стилі, додавати інтерактивні маркери. Більш того, варто зазначити, що Leaflet був створений українцем Володимиром Агафонкіним, який живе в Києві, тож варто підтримати вітчизняних і таких талановитих розробників.

Рис. 1.4 – Логотип бібліотеки мапи

Отже, підводячи підсумки можна вказати, що LeafLet є найбільш якісним і у той же час найбільш збалансованим саме для цілей даного веб-сервісу. Безоплатна основа використання, детальність шарів, чудова інтеграція у код, детальна документація та постійні оновлення неодмінно вказують на якість продукту.

Під час аналізу наявних веб-сервісів вирішення даної проблеми не було знайдено, особливо, для наших регіонів. Саме тому вирішено використати API найбільш підходящої мапи і адаптувати її під умови проекту.

2 ВИБІР МЕТОДІВ РЕЛІЗАЦІЇ

2.1. Постановка задачі та визначення мети

Метою даної роботи є створення веб-сервісу для громадського об’єднання з інтеграцією карт. Даний веб-сервіс стане у нагоді під час та після бойових дій на території України.

Головна особливість сервісу – відображання запитів на карті у вигляді міток, натискаючи (наводячи) на які можна отримати детальну інформацію (укр. або англ. мовами) про руйнування або переглянути запит. Запит може містити фото, інформацію про власників (зашифровано для початкового перегляду), місце де є пошкодження (за бажанням). Таким чином можна побачити найбільш уражені райони в області або допомогти співгромадянам з їх проблемою. Люди самостійно або адміністратори будуть викладати запити на допомогу. Види допомоги можуть бути такими: гуманітарна допомога, кошти на відбудову, корм для тварин, допомога матеріалами для відбудови помешкання.

Про руйнування критичної інфраструктури не буде вказуватись із міркування безпеки, можливо, тільки після оприлюднення військовою адміністрацією.

У майбутньому є ідея інтегрувати повідомлення про нові запити з допомогою для користувачів або лише для адміністраторів для того, аби якомога швидше відгукнутися за запит.

Для досягнення поставленої мети треба вирішити наступні задачі:

- провести аналіз предметної області;
- обрати засоби та методи для вирішення задачі додавання запитів про допомогу від місцевого населення;
- виконати проектування системи (створення макету, проектування бази даних, а саме ERD);
- здійснити програмну реалізацію спроектованої системи.

2.2 Вибір методів реалізації

Мова програмування.

На сьогоднішній день існує безліч мов програмування та інструментів для створення різного типів застосунків, вибір яких повністю залежить від цілі застосунку та його типу. Саме тому необхідно провести аналіз та обрати найкращий варіант відповідно до конкретного проекту та його задач.

Список мов програмування настільки величезний, що люди цілком зрозуміло мають різні думки щодо того, яка з них є найкращою. Очевидно, що як і було зазначено вище, все залежить від цільового проекту, тому твердження, що певна мова програмування краща за всі інші, є досить контроверсійним. Однак це не означає, що немає мов програмування, які б користувалися загальною пошаною та прихильністю. Серед таких можна виділити Python, який вважається однією з найпопулярніших мов.

Python безсумнівно вважається найкращою мовою програмування на тому ж рівні, що й JavaScript, наприклад, і це одна з найбільш використовуваних мов компаніями та підприємствами. Незважаючи на те, що цій мові вже майже 30 років, Python все ще актуальний, враховуючи його простоту використання, його активну спільноту та багато програм. Цих характеристик має бути достатньо, щоб підтвердити заяву про одну найпопулярніших мов програмування.

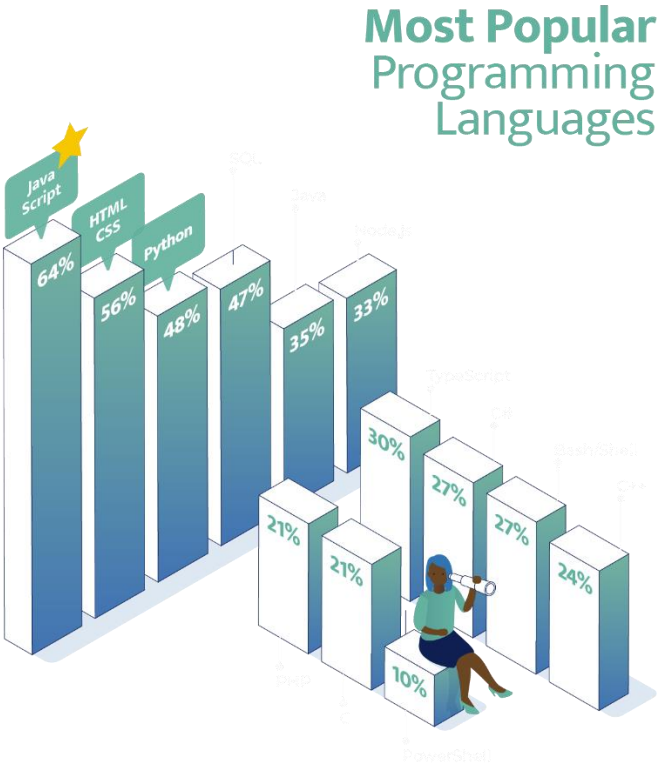

Рис. 2.1 – Візуальне відображення популярності мов програмування

Тож що ж таке Python ?

Згідно з його офіційним веб-сайтом, Python є «інтерпретованою, об’єктно-орієнтованою мовою програмування високого рівня з динамічною семантикою». Він має зрозумілий і простий синтаксис, який робить Python дуже легким для вивчення, що робить його ідеальним для початківців у програмуванні. Додає зручності й те, що мова підкреслює читабельність і робить кодування дуже легким.

Python також посідає одне с перших місць по найшвидшому розвитку серед інших мов у світі. Його високорівнева, інтерпретована та об’єктно-орієнтована архітектура робить його ідеальним для всіх типів програмних рішень. Більше того, акцент мови на читабельності синтаксису, модульності програми та можливості повторного використання коду значно збільшує швидкість розробки та зменшує вартість обслуговування.

Іншою головною особливістю Python є підтримка різноманітних модулів і пакетів. Вони полегшують розробникам на Python повторне використання

коду в різних проектах. Це відмінний спосіб підвищити продуктивність і заощадити час і сили при програмуванні на Python.

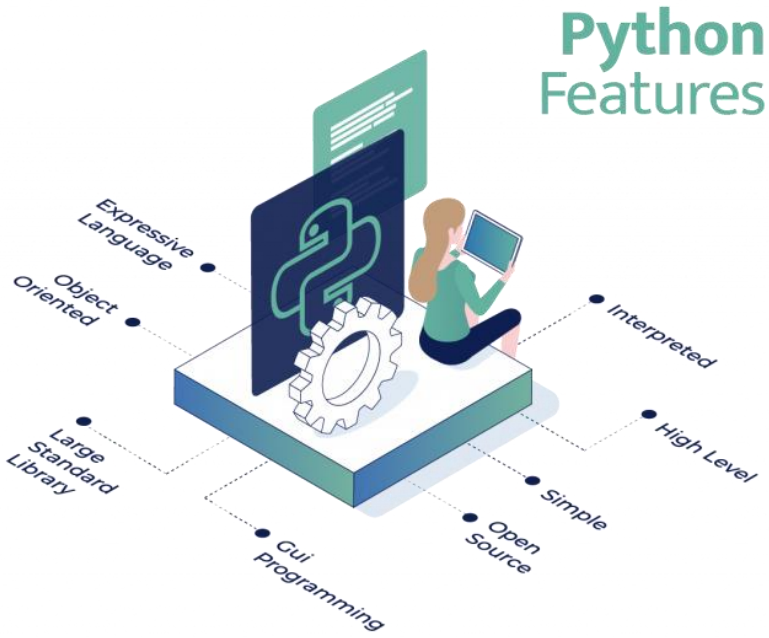

Рис. 2.2 – Основні переваги Python

Бізнес-програми на Python охоплюють надзвичайно широкий спектр пропозицій. Здається, що програми Python з кожним днем стають кращими та потужнішими, починаючи від веб-розробки та програмування з графічним інтерфейсом до великих даних і індивідуальних корпоративних рішень. Завдяки потужним фреймворкам, які постійно розвиваються, як-от Django, CherryPy, Tornado та Flask, є ще багато можливостей для розвитку.

Facebook, Google, Dropbox, Spotify, Quora, Wikipedia, Netflix, Yahoo!, NASA та багато інших компаній використовують Python завдяки численним перевагам, які він надає. Це одна з найбільш гнучких, надійних і потужних мов програмування у світі, і вона відіграє важливу роль у більшості типів проектів розробки програмного забезпечення.

Простіше кажучи, програми на Python універсальні. Компанії в усіх галузях використовують код Python для створення всього: від простих легких компонентів до повноцінних корпоративних програм. У поєднанні з належною підтримкою коду практично немає обмежень для того, що може досягти розробка програмного забезпечення на Python.

Незважаючи на те, що багато хто міг би сказати, що Python є найкращою мовою програмування у світі, але це досить голослівна заява. Очевидно, що Python не може бути рішенням для усіх цифрових проблем. Інші мови такі ж потужні, як Python, наприклад, такі як Java, JavaScript і C++, або відносно новачки, такі як Go і R. Тому варто вказати, що Python не є ідеальною мовою для будь-якого рішення. Найпопулярніші мови – це ті, які охоплюють можливі варіанти використання і водночас є ефективними, але немає єдиної мови, яка могла б зробити все.

Усі найпопулярніші мови програмування без сумніву мають переваги та недоліки, тому здається, що це скоріше питання вибору залежно від вимог проекту. Зважаючи на це, можна сміливо казати, що Python – це одна з найкращих мов завдяки її простоті використання, бібліотекам, спільнотам і безлічі використань.

До речі, за рейтингом TIOBE та PYPL Python посідає перше місце у 2022 році! Індекс TIOBE Programming Community є показником популярності мов програмування. Індекс оновлюється раз на місяць. Рейтинги базуються на кількості кваліфікованих інженерів у всьому світі, курсах і сторонніх постачальниках [10]. Натомість, індекс популярності PYPL створюється шляхом аналізу того, як часто в Google шукають підручники або туторіали з мови певної мови програмування [11].

Dec 2022	Dec 2021	Change	Programming Language	Ratings	Change
1	**1**		**Python**	**16.66%**	**+3.76%**
2	2		C	16.56%	+4.77%
3	4	▲	C++	11.94%	+4.21%
4	3	▼	Java	11.82%	+1.70%
5	5		C#	4.92%	-1.48%
6	6		Visual Basic	3.94%	-1.46%
7	7		JavaScript	3.19%	+0.90%

Рис. 2.3 – Рейтинг TIOBE мов програмування

Фреймворк.

Варто почати з того, що не зважаючи на те, що Python є доволі потужною та загалом чудовою мовою програмування, у нього все ще є свої обмеження. Однак вони в основному компенсуються широким розмаїттям фреймворків і середовищами розробки Python. Тому фреймворки – це невід'ємна частина будь-якої мови програмування, а Python вони роблять ще більш функціональним.

Django часто вважають одним з найкращих фреймворків Python. Його навіть включили до списку найулюбленіших фреймворків в опитуванні розробників Stack Overflow у 2018 році [12].

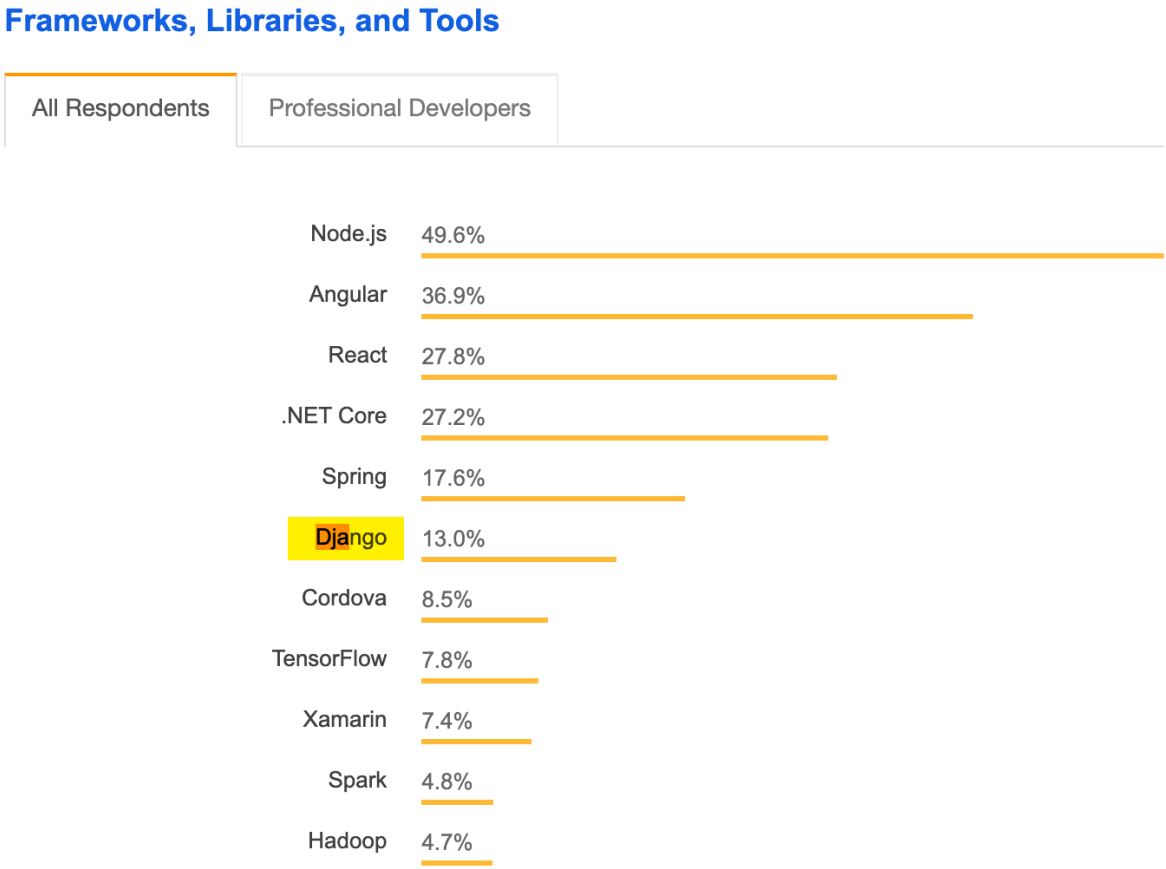

Рис. 2.4 – Рейтинг фреймворків і бібліотек за версією Stack Overflow

Перш за все, Django написано на Python, тому це дозволяє розробникам легко використовувати синтаксис Python, який наразі добре підтримується, що дуже важливо для створення складних веб-застосунків. Більш того, фреймворк додає низку функцій, що робить розробку ще кращою [13].

Фреймворк Django має все необхідне для розробки повноцінної програми. Django пропонує вбудовані шаблони HTML, URL-маршрутизацію, об'єктно-реляційне відображення та керування сеансами, допомагаючи розробникам уникнути набридливого пошуку інструментів сторонніх розробників. Різноманіття пакетів Django допомагають швидко створювати будь-що: від чат-ботів до складних рішень із підтримкою GPS.

Django майже вдвічі скорочує час розробки. Розробники Django стверджують, що він є «веб-фреймворком для перфекціоністів із дедлайнами» і це варте того. Розробники можуть використовувати вбудовані функції Django для створення веб-застосунків для будь-якої галузі. Використовуючи філософію «Don't repeat yourself» (DRY), на якій побудований Django, мотивує розробників повторно використовувати код, який вони написали для інших проектів, ще більше скорочуючи час, необхідний для розробки іншого продукту.

Django з легкістю розширюється та масштабується. Компоненти Django відокремлені, тобто їх можна додавати або видаляти за потреби. Залежно від певних конкретних вимог до продукту, розробку можна збільшувати або зменшувати, змінюючи кількість і складність компонентів Django [14].

Django забезпечує досить надійну безпеку. Django за замовчуванням захищає програми, створені з його допомогою. Він унеможливлює деякі з найпоширеніших безпекових помилок, а також захищає програми від підробки запитів і впровадження SQL ін'єкцій.

Django працює з більшістю основних баз даних. Об'єктно-реляційне відображення (ORM) Django сумісне з багатьма популярними базами даних, але його ключовою особливістю є те, що воно дозволяє розробникам працювати з кількома базами даних одночасно. Крім того, Django дає змогу переходити з однієї бази даних в іншу та виконувати звичайні операції без необхідності писати додатковий код. Під час виконання даного проекту було обрано MySQL базу даних, яка є доволі простою у запитах і встановлена за замовченням у Django.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Моделювання системи оброблення запитів населення в умовах надзвичайних ситуацій військового характеру

3.1.1 Визначення інформаційних потреб користувачів

Нижче представлений перелік основних сутностей та можливих дій при використанні застосунку.

Користувачі:

- адміністратор;
- користувач, який потребує допомогу;
- користувач, який надає допомогу.

Інформація, яку надає веб-додаток:

- переглянути карту руйнувань / статистику;
- створити запит на допомогу;
- допомогти.

Потреби, які можуть викладати користувачі:

- матеріальна допомога;
- волонтерська допомога (фізично щось зробити, транспортування);
- інформаційна допомога (знайти когось).

3.1.2 Use Case діаграма

Загальний опис системи можна описати за допомогою Use Case діаграми. Use Case використовується для опису сценаріїв взаємодії учасників (як правило – користувача та системи). Кількість учасників може бути від двох та більше. Користувачем може бути як людина, так і інша система. Алістер Кокберн у своїй книзі описав це так: «Варіант використання фіксує угоду між учасниками системи щодо її поведінки. Варіант використання описує поведінку системи при її відповідях на запит одного з учасників, що називається основною дійовою особою, у різних умовах» [15].

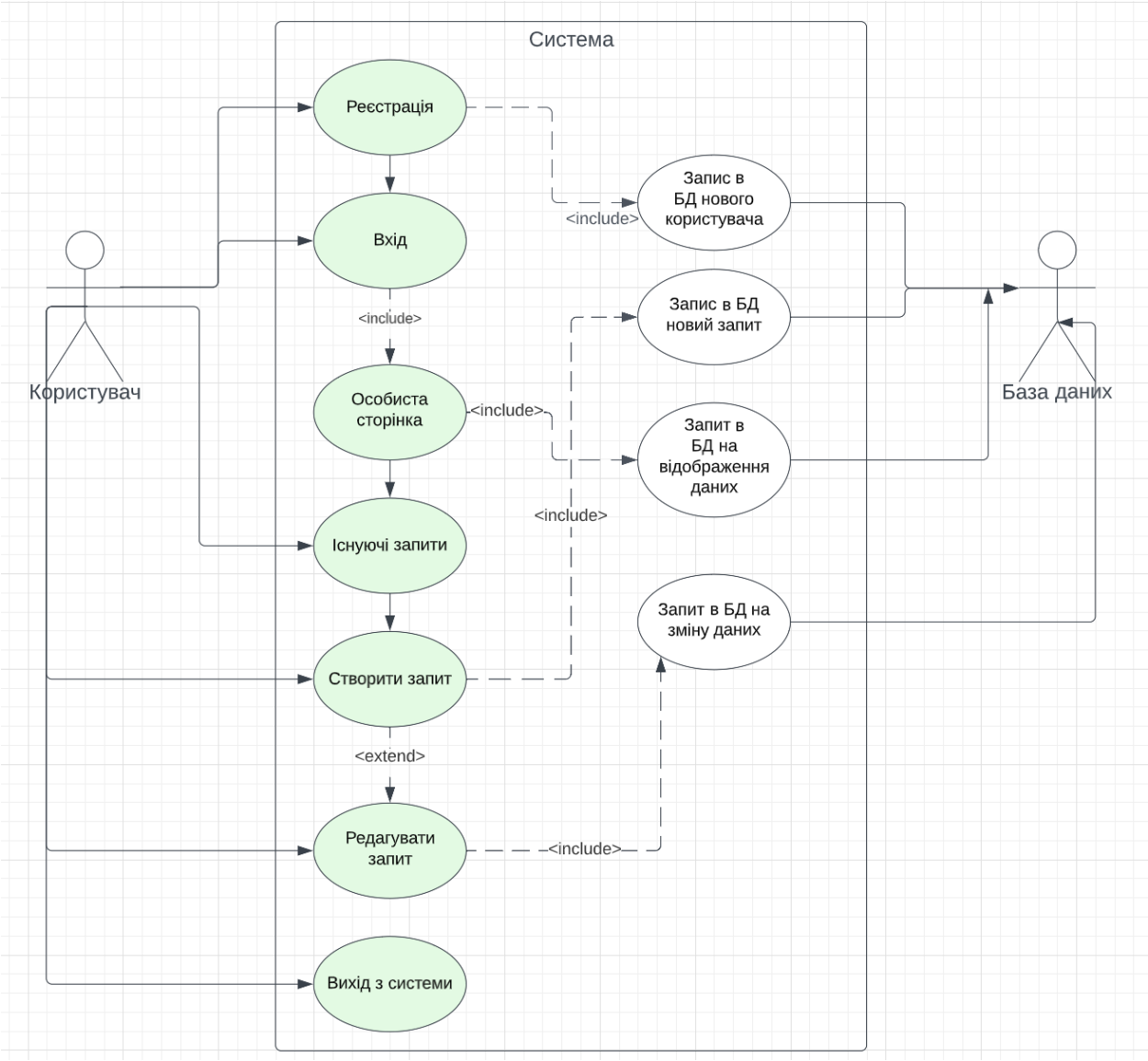

Рис. 3.1 – Use case діаграма

3.1.3 ERD діаграма

Для правильного проектування бази даних необхідно створити ER-діаграму. Проектування дозволить якомога точніше визначити предметну область, визначити основні сутності та встановити зв’язки між ними. На рисунку 3.3.1 зображена ER-діаграма, яка відповідає потребам системи. Для уникнення потенційних суперечностей між даними модель було приведено до третьої нормальної форми.

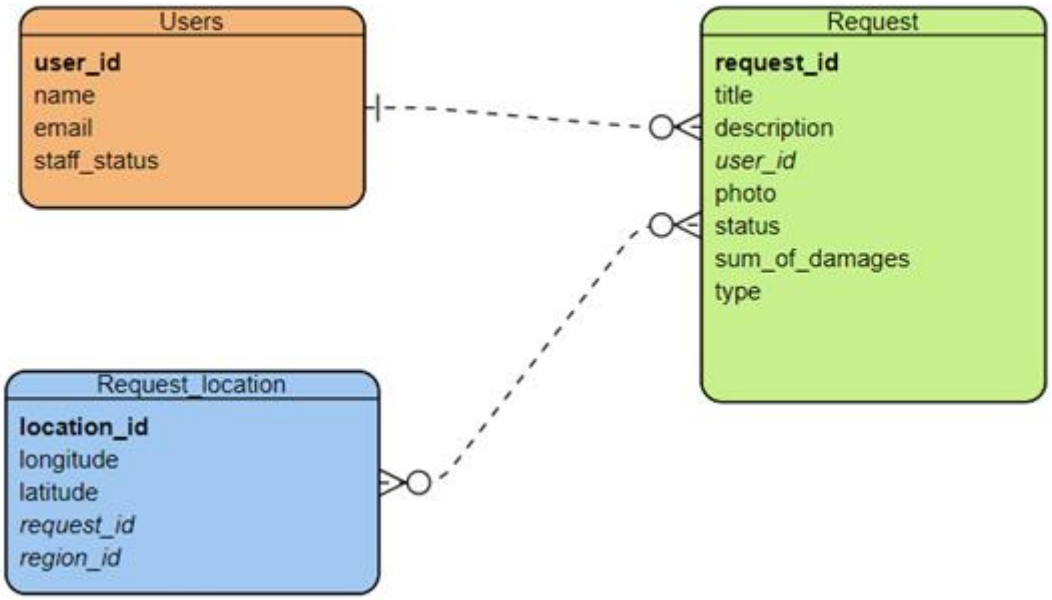

Рис. 3.2 – ERD проекту

Таблиця 3.1 – Опис полів таблиці Users

Column name	**Value**
user_id	not null, Primary Key
name	not null
email	not null
role	not null

Таблиця 3.2 – Опис полів таблиці Request

Column name	**Value**
request_id	not null, Primary Key
user_id	not null, Foreign Key
title	not null
description	not null
photo	null
status	not null

3.1.4 Описова модель предметної області

На основі методології системного аналізу розроблено комплекс описових моделей, які можна подати у вигляді загальної формули:

IMAP = <Users, Request, Request_location, Region>,

де *Users* - множина користувачів, які проводять певні операції з веб-ресурсом,

Request - множина запитів про допомогу від користувачів (*Users*),

Request_location - локація від користувача, який створює запит (не обов'язкове значення);

Region - множина регіонів (міст, областей), де будуть зберігатися локації від користувачів.

При цьому Users:

$Users = <\{x_i, Vx_i\}\ i\ (1,4)>$, де

x_i - множина даних про користувача;

Vx_i - допустимі значення даних про користувача.

До множини даних про користувача належать наступні значення:

x_1 - user_id (унікальний ідентифікатор користувача),

x_2 - name (ім'я користувача),

x_3 - email (електронна адреса користувача),

x_4 - role (статус користувача, від якого залежать права).

$Request = <\{x_i, Vx_i\}\ i\ (1,7)>$, де

x_i - множина даних про користувача;

Vx_i - допустимі значення даних про користувача.

До множини даних про користувача належать наступні значення:

x_1 - request_id (унікальний ідентифікатор запиту),

x_2 - user_id (унікальний ідентифікатор сутності "*Users* "),

x_3 - title (заголовок запиту),

x_4 - description (тіло запиту),

x_5 - photo (фото, не обов'язково),

x_6 - status (статус запиту для відслідковування етапів),

x_7 - sum_of_damages (сума збитків).

Request_location = $<\{x_i, Vx_i\}\ i\ (1,5)>$, де

x_i - множина даних про користувача;

Vx_i - допустимі значення даних про користувача.

До множини даних про користувача належать наступні значення:

x_1 - location_id (унікальний ідентифікатор локації запиту),

x_2 - longitude (координата довготи),

x_3 - latitude (координата широти),

x_4 - request_id (унікальний ідентифікатор сутності “*Request_location*”),

x_5 - region_id (унікальний ідентифікатор сутності “*Region* ”).

Region = $<\{x_i, Vx_i\ i\ (1,4)\}>$, де

x_i - множина даних про користувача;

Vx_i - допустимі значення даних про користувача.

До множини даних про користувача належать наступні значення:

x_1 - region_id,

x_2 - longitude (координата довготи),

x_3 - latitude (координата широти),

x_4 - name (назва регіону).

3.2 Опис програмної реалізації

3.2.1 Створення макету

Початок роботи зазвичай починається з основного – проектування. Перед безпосереднім написанням коду необхідно створити макет майбутнього веб-сайту. Це чудово допомагає узагальнено оцінити план роботи, додати за необхідності деталі, або навпаки прибрати зайве. Макети були створені у Figma – зручний сервіс для проектування, має всі необхідні інструменти та є відносно безкоштовним. Далі будуть показані зроблені макети сторінок у Figma.

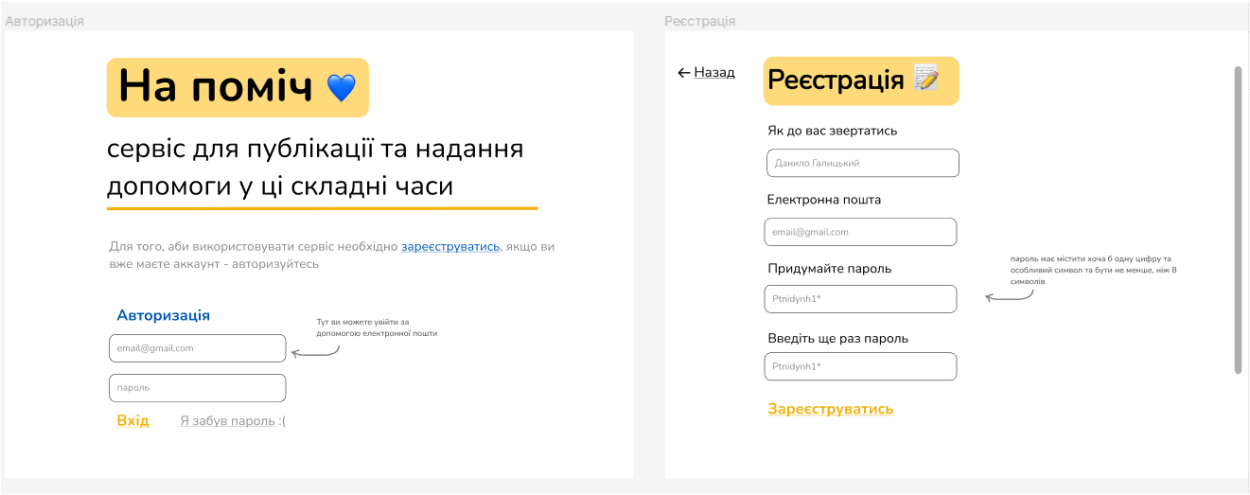

Рис. 3.3 – Приклад макету для сторінок Авторизації та Реєстрації

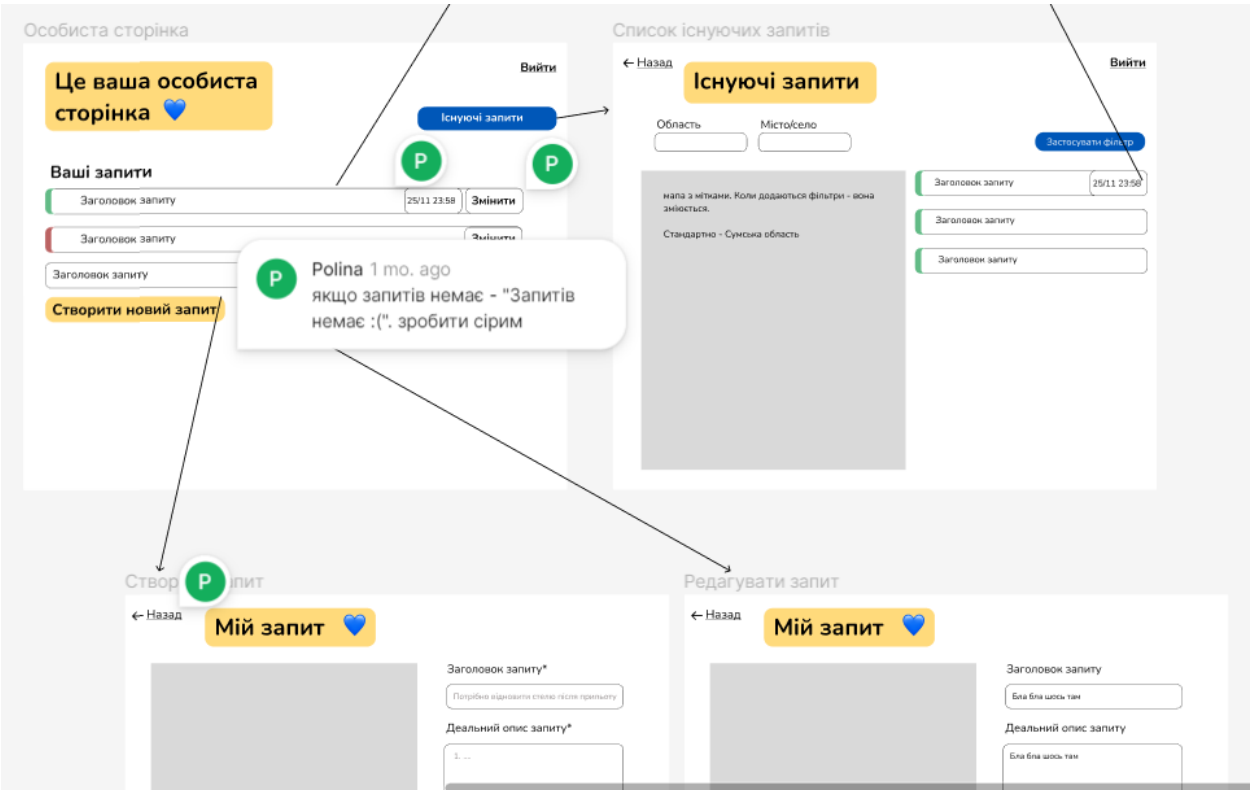

Рис. 3.4 – Макети «Особистої сторінки», «Списку існуючих запитів»

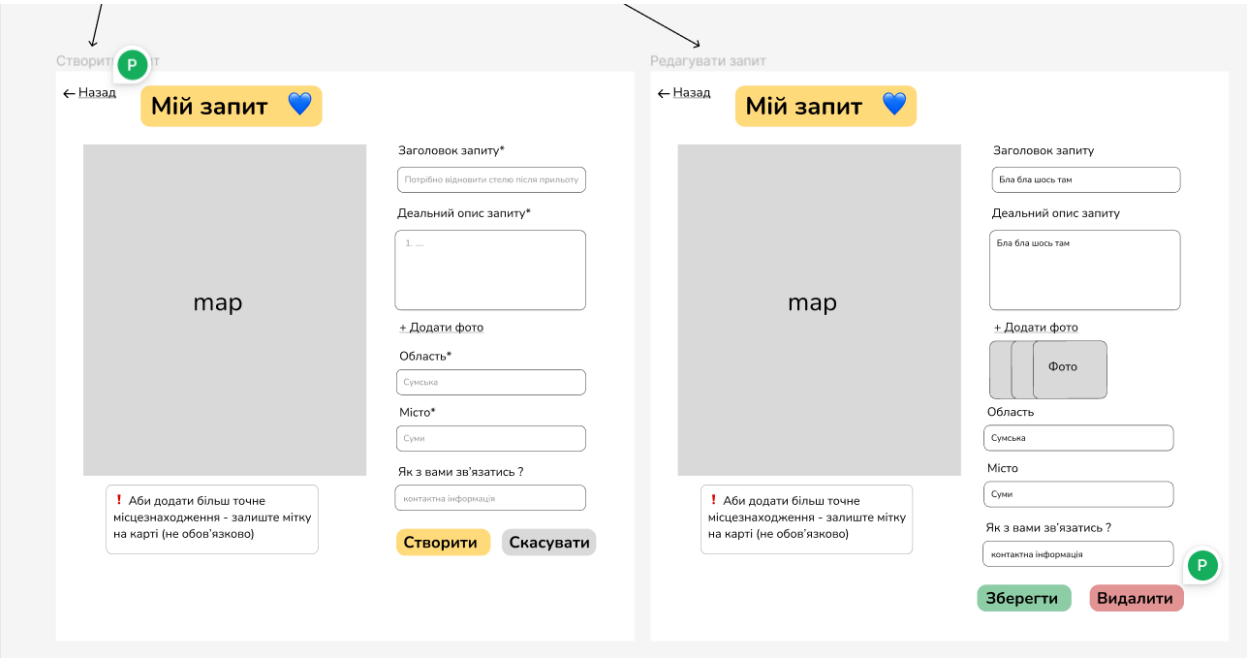

Рис. 3.5 – Макети «Створення запиту» та «Редагування запиту»

Наступним етапом є ініціалізація та налаштування проекту. Це означає, що треба завантажити середовище для програмування, обраний фреймворк, запустити локальний сервер, створити супер-адміна та налаштувати базу даних. Що і було зроблено.

3.2.2 База даних

Перед початком написання сторінок, а саме сторінки Авторизації та Реєстрації користувачів, необхідно описати сутність користувача в базі даних.

Було використано вбудовану структуру бази даних Django – SQLite3. Ця вбудована база даних вже містить у собі стандартну сутність User (користувач), але для даного проекту необхідно її розширити. Тому для цього було додано надналаштування.

models.py

main > models.py

```python
from django.db import models
from django.contrib.auth.models import User

class Account(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    is_active_account = models.BooleanField(default=False)
    activation_code = models.CharField(max_length=100)
    change_password_code = models.CharField(max_length=100, default='')

    def __str__(self):
        return self.user.email

class Request(models.Model):
    account = models.ForeignKey(Account, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    description = models.TextField()
    photo = models.ImageField(upload_to='requests', blank=True, null=True)
    district = models.CharField(max_length=100)
    city = models.CharField(max_length=100)
    contacts = models.CharField(max_length=100)
    created_at = models.DateTimeField(auto_now_add=True)
    is_closed = models.BooleanField(default=False)
    lat = models.FloatField()
    lng = models.FloatField()

    def __str__(self):
        return self.account.user.email + '|' + self.name
```

Рис. 3.6 – Файл models.py

У файлі models.py описано основні сутності та зв’язки між ними. А саме: Account – користувач та все, що необхідно для його створення. Request – сутність запиту. Зв’язок з таблицею Account створено за допомогою “ForeignKey”.

3.2.3 Авторизація та супутні дії

Даний веб-застосунок передбачає повну реєстрацію та подальшу авторизацію. Аби захистити веб-сервіс було впроваджено надсилання активаційного коду на пошту. Також якщо вже зареєстрований користувач забув свій пароль, від може легко відновити його. На рис. 3.7 зображено фрагменти коду файлу views.py з реалізацією надсилання активаційного коду та відновлення паролю за запитом.

views.py

main > views.py

```
190                 account.save()
191                 send_auth_email(email, activation_code, 'activate')
192                 return redirect('/main/login?error=check_email')
193         else:
194             return redirect('/main/register/')
195
196     def activate(request):
197         if request.method == 'GET':
198             code = request.GET.get('code')
199             if Account.objects.filter(activation_code=code).exists():
200                 account = Account.objects.get(activation_code=code)
201                 account.is_active_account = True
202                 account.save()
203                 return redirect('/main/login?error=account_activated')
204             else:
205                 return redirect('/main/login?error=wrong_code')
206         else:
207             return redirect('/main/login/')
208
209     def new_password(request):
210         if request.method == 'POST':
211             code = request.POST.get('code').strip()
212             password = request.POST.get('password').strip()
213             if Account.objects.filter(change_password_code=code).exists():
214                 account = Account.objects.get(change_password_code=code)
215                 user = account.user
216                 user.set_password(password)
217                 user.save()
218                 return redirect('/main/login?error=new_password')
219             else:
220                 return redirect('/main/change_password?error=wrong_code')
221         else:
222             return redirect('/main/change_password/')
223
224     def change_password_request(request):
225         if request.method == 'POST':
226             email = request.POST.get('email').strip()
227             if User.objects.filter(email=email).exists():
```

Рис. 3.7 – Частини коду для активаційного коду та відновлення паролю

Авторизаційний код створюється автоматично за допомогою 128-бітного алгоритму гешування (md5).

```
def generate_code():
    timestamp = int(time.time())
    return hashlib.md5(str(timestamp).encode('utf-
8')).hexdigest()
```

Для надсилання коду та посилання на відновлення паролю було використано модуль **yagmail.** Це модуль для Python, який використовується для надсилання листів. Він без проблем взаємодіє з Gmail і надсилає листи за SMTP протоколом.

Щоб імпортувати yagmail треба її спочатку імпортувати за допомогою - `import yagmail`. Нижче буде представлено фрагмент коду з views.py, де описано надсилання листа за допомогою модулю yagmail.

```
def send_auth_email(email, code, type):
    user = config['gmail_address']
    app_password = config['gmail_password']
    host = config['server_host']
    to = email

    if type == 'activate':
        subject = 'Activate account code from naPomich'
        content = ['Your activate account link: ' + host +
'main/activate?code=' + code]
    elif type == 'change_password':
        subject = 'Change password code from naPomich'
        content = ['Your change password link: ' + host +
'main/change_password?code=' + code]

    with yagmail.SMTP(user, app_password) as yag:
        yag.send(to, subject, content)
        return True
```

3.2.4 Інтеграція мапи

Як було зазначено у попередніх розділах за основу було обрано карту Leaflet. Для інтеграції карти перш за все необхідно інтегрувати її бібліотеку на кожній сторінці, де вона використовуватиметься. На рис. 3.8 виділено фрагменти коду у файлі *create_request.html*, які саме і здійснюють цю інтеграцію.

Рис. 3.8 – Інтеграція мапи

Відображення мапи на сторінці здійснюється за допомогою API Leaflet, яке є у відкритому доступі. Для цього було інтегровано токен доступу. Початкові координати вказані в Сумській області, тобто при створенні запиту спочатку буде показано саме Сумську область. При натисканні на мапу ставиться маркер, при спробі змінити координати мапи попередній маркер видаляється, а ставиться новий. Нижче буде приведено код із файлу *create_request.js*, у якому описано викладене вище.

```
let mymap = L.map('map').setView([50.907688, 34.796716], 13);
let marker;

document.getElementById('photo').addEventListener('change',
function () {
    let file = this.files[0];
```

```
    if (file) {
        document.getElementById('filename').innerHTML =
file.name;
        document.getElementById('photoBtnLabel').innerHTML = '+
Змінити фото';
    }
});

L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}
/{y}?access_token=pk.eyJ1IjoibWFwYm94IiwiYSI6ImNpejY4NXVycTA2emY
ycXBndHRqcmZ3N3gifQ.rJcFIG214AriISLbB6B5aw', {
    maxZoom: 18,
    id: 'mapbox/streets-v11',
    tileSize: 512,
    zoomOffset: -1
}).addTo(mymap);

mymap.on('click', function(e) {
    if (marker) {
        mymap.removeLayer(marker);
    }
    marker = L.marker(e.latlng).addTo(mymap);
    document.getElementById('lat').value = e.latlng.lat;
    document.getElementById('lng').value = e.latlng.lng;
});
```

3.2.5 Реалізація фільтру

Для зручного та швидкого пошуку в системі серед існуючих запитів було реалізовану фільтрування. Основний пошук ведеться по області та населеному пункту. Нижче приведено безпосередньо запит.

```
function loadData(district = '', city = '') {
    let xhr = new XMLHttpRequest();
    let url = '/get_requests/?district=' + district + '&city=' +
city;
    xhr.open('GET', url, true);
```

```
    xhr.send();
    xhr.onreadystatechange = function() {
        if (xhr.readyState != 4) return;
        if (xhr.status != 200) {
            alert(xhr.status + ': ' + xhr.statusText);
        } else {
            let data = JSON.parse(xhr.responseText);
            addMarkers(data);
            addRequests(data);
        }
    }
}
```

3.2.6 Огляд сторінок

У загальному вигляді розроблений веб-застосунок має мінімалістичний вигляд аби не перенавантажувати око людини і щоб не виникало труднощів з навігацією. Отже, перша сторінка коротко описує мету сервісу та те, що необхідно зробити спершу аби почати користуватися веб-сервісом.

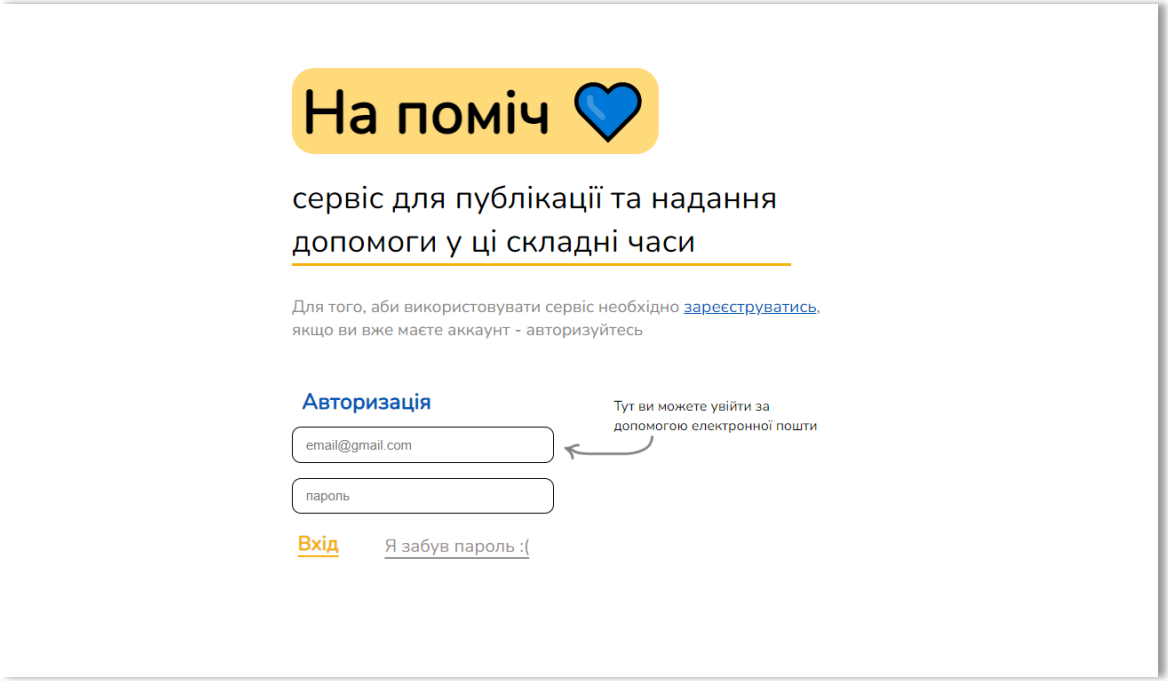

Рис. 3.9 – Перша сторінка

Для реєстрації необхідно ввести лише найголовніше, аби можна було ідентифікувати користувача. Також було додано невелику підказку того, який саме пароль прийме система.

Рис. 3.10 – Реєстрація

Запроваджена можливість відновлення паролю за допомогою сервісу yagmail.

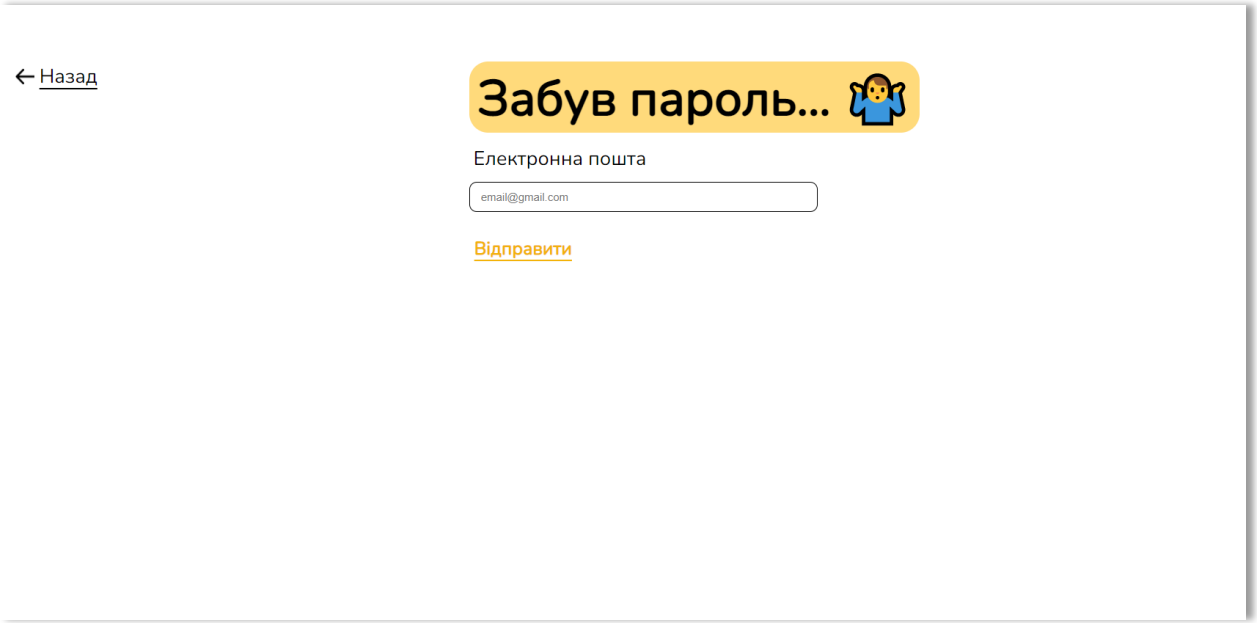

Рис. 3.11 – Початкова сторінка для відновлення паролю

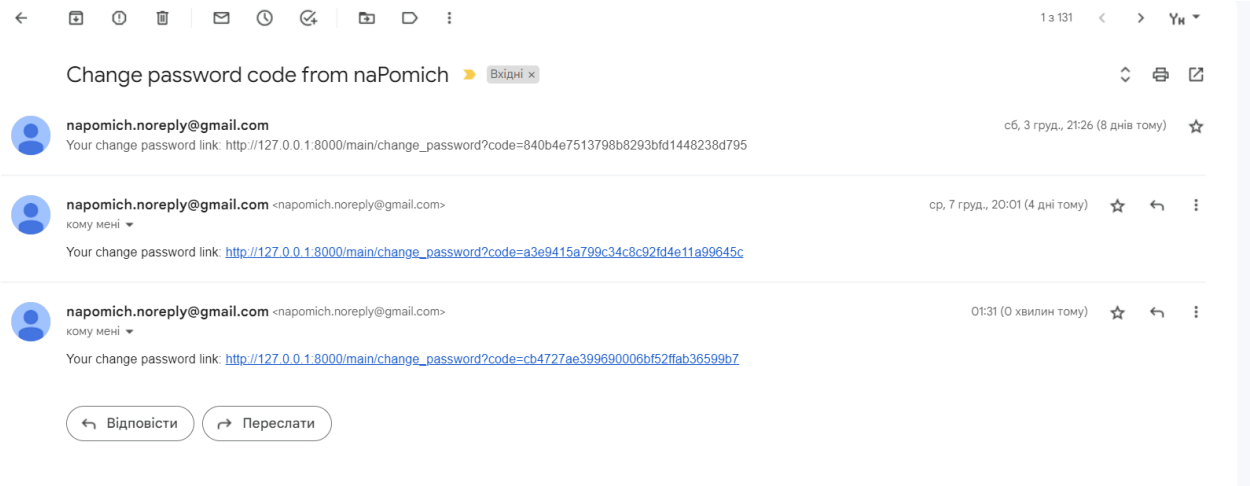

Рис. 3.12 – Посилання для зміни паролю

Після успішної авторизації користувач потрапляє на особисту сторінку. Головна мета – відобразити існуючі запити користувача. Усі запити мають статус: Активний/Неактивний, який відповідає зеленому або червоному кольорам. Якщо таких запитів ще не створено, буде відображено відповідне текстове повідомлення. На цій же сторінці є можливість перейти на сторінку всіх існуючих запитів від всіх зареєстрованих користувачів.

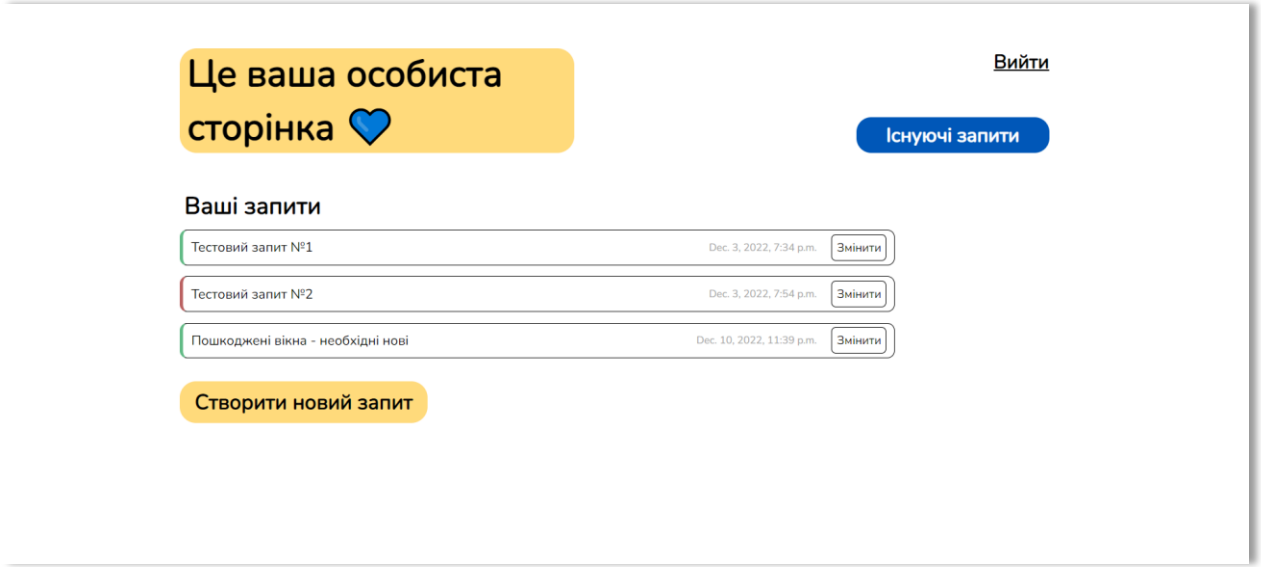

Рис. 3.13 – Особиста сторінка

Натиснувши на кнопку «Створити новий запит» користувач перенаправляється на сторінку створення запиту. Всі поля окрім фото обов’язкові. За бажанням користувач може вказати своє точне місцезнаходження на карті, але це також не є обов’язково.

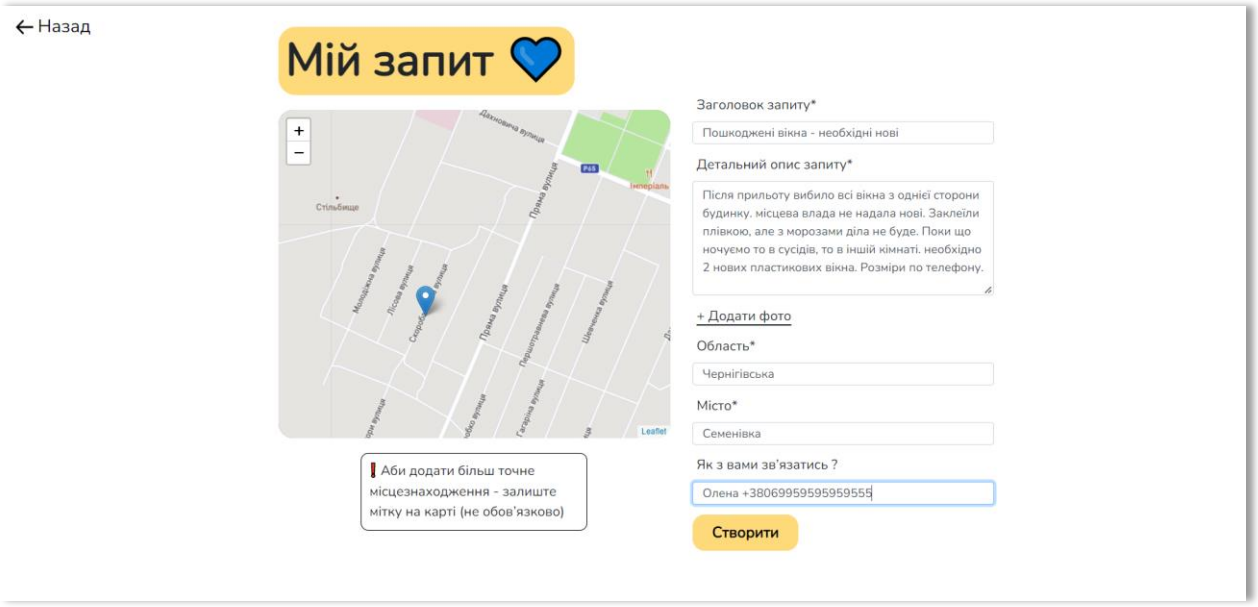

Рис. 3.14 – Створення запиту

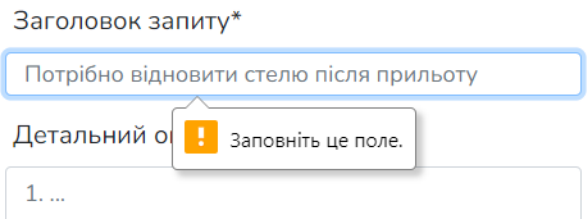

Рис. 3.15 – Попередження системи про обов'язковість поля

Також з особистого кабінету можна перейти на «Існуючі запити» - це сторінка з усіма активними запитами від всіх зареєстрованих користувачів.

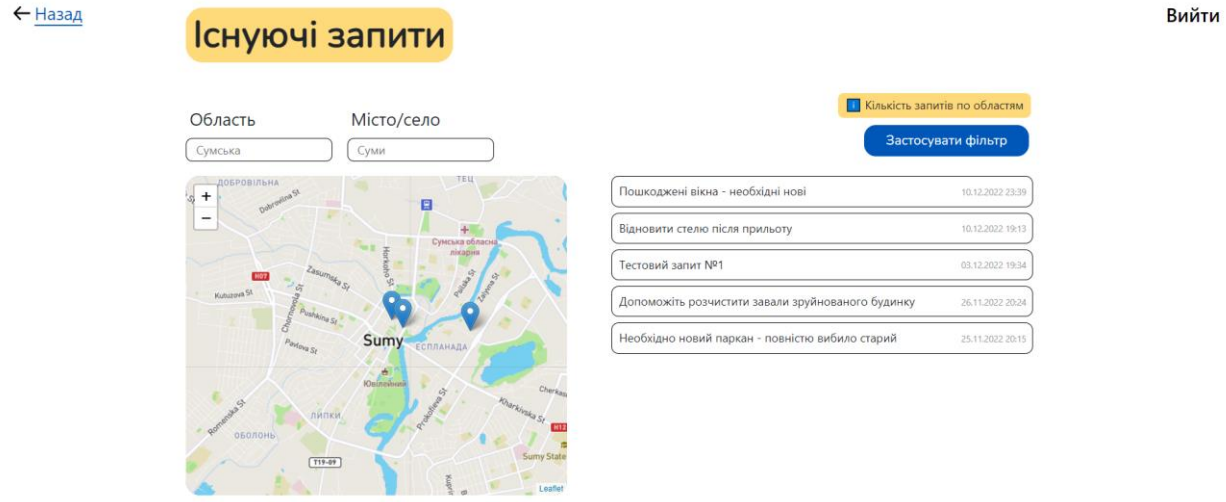

Рис. 3.16 – сторінка «Існуючі запити»

Натиснувши на будь-яку з міток на карті з’явиться активний заголовок і опис запиту. Можна натиснути на заголовок і автоматично відкриється даний запит.

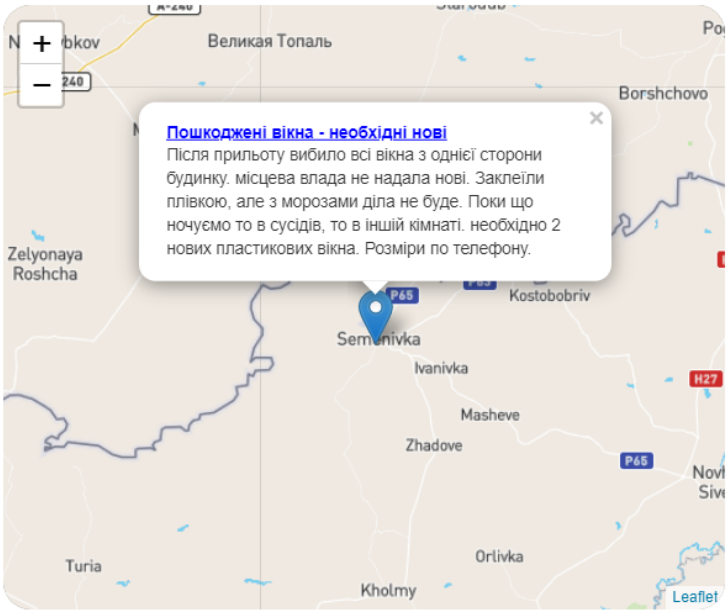

Рис. 3.17 – Інтерактивна мітка

Натиснувши на кнопку «Кількість запитів по областям» можна побачити коротку аналітику, яка допоможе узагальнити всі дані. У майбутньому планується додати пай-чарт для наочності та кластеризацію для мапи. За допомогою кластеризації можна побачити масованість запитів по ділянкам.

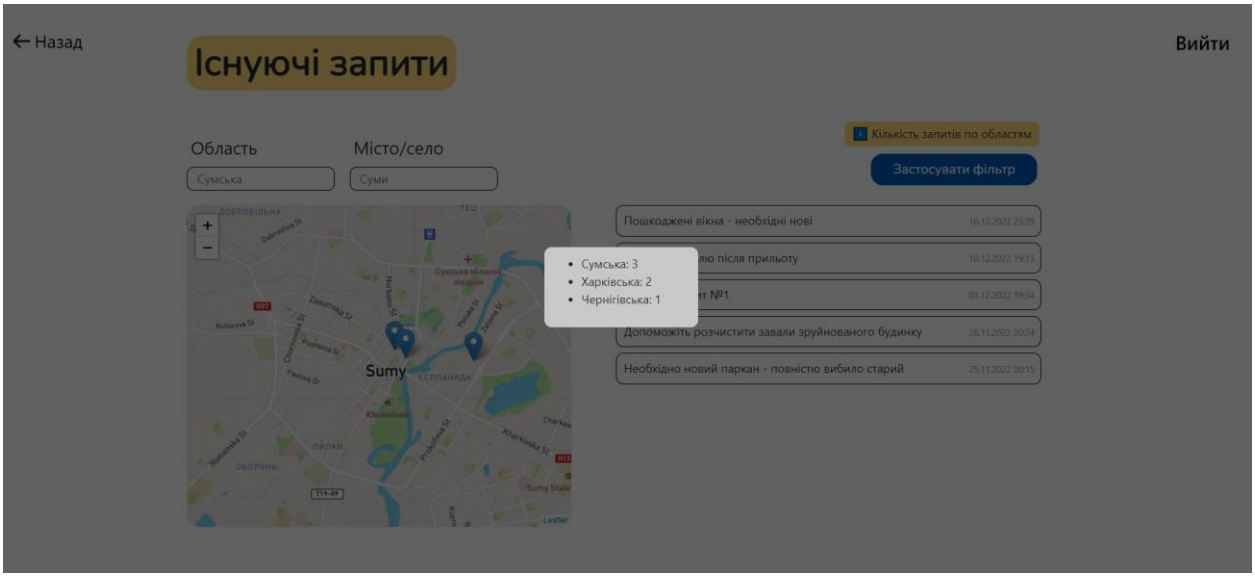

Рис. 3.18 – Коротка аналітика

Загальний вигляд активного запиту зображений на рисунку 3.19 нижче.

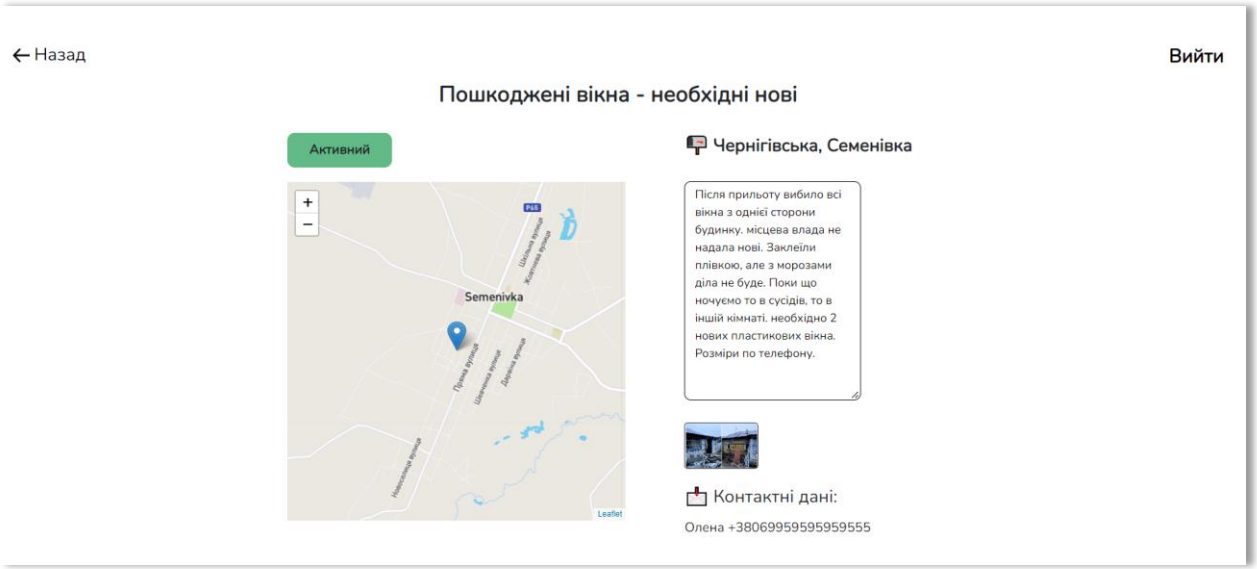

Рис. 3.19 – Загальний вигляд запиту

Запит можна відредагувати та зробити його неактивним натиснувши на кнопку «Змінити» на особистій сторінці, а потом – «Видалити». Таким же чином запит можна відновити натиснувши на відповідну кнопку при повторному редагуванні. Така можливість є звісно тільки у власника запиту і у адміністратора.

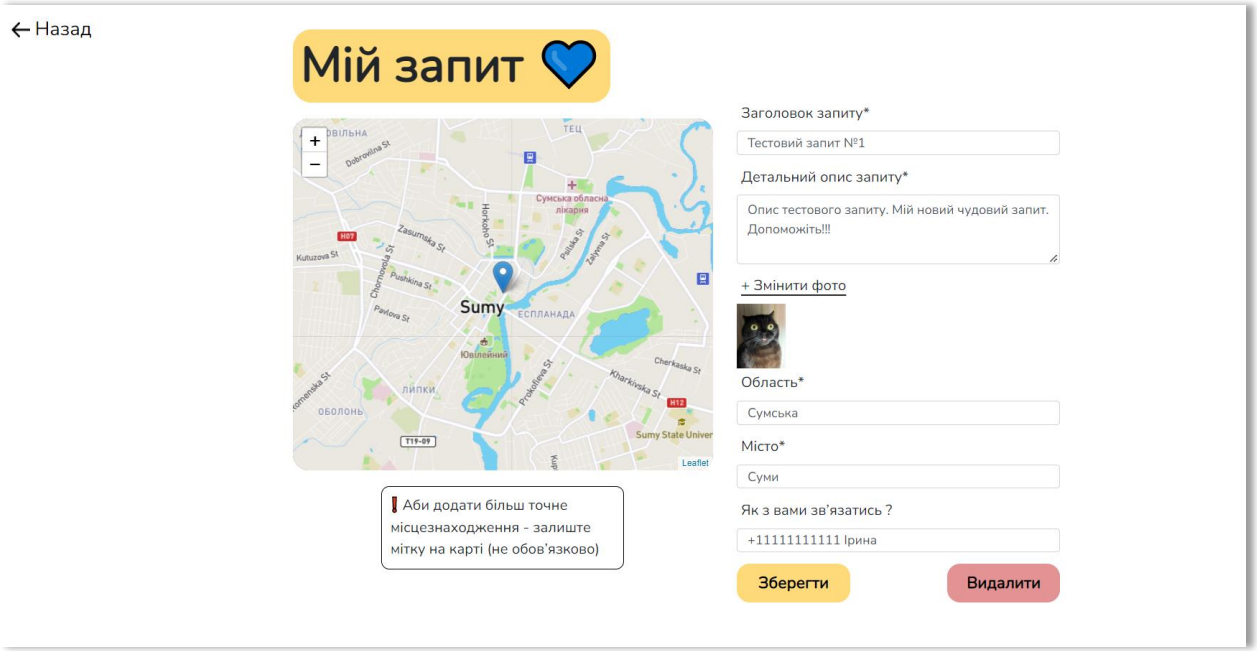

Рис. 3.20 – Редагування запиту

ВИСНОВКИ

Розроблений веб-сервіс для використання особами, які постраждали від воєнних дій на території України.

Реалізована можливість публікувати запити про допомогу різного характеру. За спостереженнями, людей найбільше турбує гуманітарна допомога у вигляді продуктів харчування та ліків, а також допомога у відбудові або розборі пошкодженого житла.

Аналітична функція застосунку дозволяє за допомогою карти і збору аналітики прослідкувати активність і ареали обстрілів. Публікування критичної інфраструктури може модеруватися адміністраторами застосунку.

Проведено аналіз предметної області, під час якого розглядалися існуючі сервіси та було проведено аналіз наявної літератури. Обрано засоби для вирішення поставленої задачі, розроблено макет, діаграму варіантів використання (Use Case) та ERD діаграму на етапі проектування.

Розроблено технологію з інтеграцією API мап, що надає наступні можливості:

- реєстрація нового користувача;
- авторизація та відновлення забутого паролю;
- створення запиту про допомогу;
- редагування створеного запиту;
- перегляд існуючих запитів;
- перегляд карти по обраним фільтрам;
- перегляд кількості запитів по областям.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Sumy Post. Сумщина – у трійці лідерів з відновлення руйнувань [Electronic resource]. 2022. URL: https://sumypost.com/sumynews/suspilstvo/sumshhyna-u-trijtsi-lideriv-z-vidnovlennya-rujnuvan/ (accessed: 04.12.2022).
2. Укрінформ. ОП розробив план оперативної відбудови соціальної інфраструктури України Fast Recovery [Electronic resource]. 2022. URL: ОП розробив план оперативної відбудови соціальної інфраструктури України Fast Recovery (accessed: 11.10.2022).
3. Alsobky A., Mousa R. Estimating free flow speed using google maps API: Accuracy, limitations, and applications // Advances in Transportation Studies. 2020. Vol. 50.
4. Fielbaum A., Jara-Diaz S. Assessment of the socio-spatial effects of urban transport investment using Google Maps API // J Transp Geogr. 2021. Vol. 91.
5. Bertolotto M., McArdle G., Schoen-Phelan B. Volunteered and crowdsourced geographic information: The openstreetmap project // Journal of Spatial Information Science. 2020. Vol. 20.
6. Laksono D., Aditya T. Utilizing a game engine for interactive 3D topographic data visualization // ISPRS Int J Geoinf. 2019. Vol. 8, № 8.
7. Saputra O.A., Ramdani F., Saputra M.C. Comparison Analysis of Google Maps, Wisepilot, and Here Wego with User-Centered Design (UCD) Approach & Cartography // 2018 4th International Symposium on Geoinformatics, ISyG 2018. 2019.
8. Horbiński T., Lorek D. The use of Leaflet and GeoJSON files for creating the interactive web map of the preindustrial state of the natural environment // Journal of Spatial Science. 2022. Vol. 67, № 1.
9. Farkas G. Applicability of open-source web mapping libraries for building massive Web GIS clients // J Geogr Syst. Springer Verlag, 2017. Vol. 19, № 3. P. 273–295.

10. TIOBE. TIOBE Index for December 2022 [Electronic resource]. 2022. URL: https://www.tiobe.com/tiobe-index/ (accessed: 04.12.2022).
11. PYPL. PYPL PopularitY of Programming Language [Electronic resource]. 2022. URL: https://pypl.github.io/PYPL.html (accessed: 04.12.2022).
12. Developer Survey Results 2018 [Electronic resource]. 2018. URL: https://insights.stackoverflow.com/survey/2018/#technology-most-loved-dreaded-and-wanted-frameworks-libraries-and-tools (accessed: 05.12.2022).
13. Ekaterina Zublenko. Why Django is the Best Web Framework for Your Project [Electronic resource]. 2019. URL: https://steelkiwi.com/blog/why-django-best-web-framework-your-project/ (accessed: 05.12.2022).
14. Olexandr Stepanov. Best practices working with Django models in Python [Electronic resource]. 2018. URL: https://steelkiwi.com/blog/best-practices-working-django-models-python/ (accessed: 06.12.2022).
15. Alistair A.R. Cockburn. Use cases, ten years later. 2002.

ДОДАТОК А

models.py

main > models.py

```python
from django.db import models
from django.contrib.auth.models import User

class Account(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    is_active_account = models.BooleanField(default=False)
    activation_code = models.CharField(max_length=100)
    change_password_code = models.CharField(max_length=100, default='')

    def __str__(self):
        return self.user.email

class Request(models.Model):
    account = models.ForeignKey(Account, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    description = models.TextField()
    photo = models.ImageField(upload_to='requests', blank=True, null=True)
    district = models.CharField(max_length=100)
    city = models.CharField(max_length=100)
    contacts = models.CharField(max_length=100)
    created_at = models.DateTimeField(auto_now_add=True)
    is_closed = models.BooleanField(default=False)
    lat = models.FloatField(null=True, blank=True)
    lng = models.FloatField(null=True, blank=True)

    def __str__(self):
        return self.account.user.email + '|' + self.name
```

Рис. 1 - Models.py

views.py

main > views.py

```python

from django.http import HttpResponse
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from django.db.models import Count
from .models import Account
from .models import Request
import time
import hashlib
import yagmail
import json
from django.http import JsonResponse

```

Рис. 2 – views.py (ч. 1)

views.py

main > views.py

```
15
16  f = open('main/config.json')
17  config = json.load(f)
18
19  def login_view(request):
20      return render(request, 'login.html', {'error_type': str(request.GET.get('error'))})
21
22  def register_view(request):
23      return render(request, 'register.html', {'error_type': str(request.GET.get('error'))})
24
25  def change_password_view(request):
26      return render(request, 'change_password.html', {'error_type': str(request.GET.get('error')), 'code': str(request.GET.get('code'
27
28  def logout_view(request):
29      logout(request)
30      return redirect('/main/login/')
31
32  @login_required(login_url='/main/login/')
33  def open_request_view(request):
34      back = 'main'
35      if request.GET.get('back'):
36          back = request.GET.get('back')
37
38      if request.GET.get('id'):
39          id = request.GET.get('id')
40          if Request.objects.filter(id=id).exists():
41              find_request = Request.objects.get(id=id)
42              return render(request, 'request.html', {'request': find_request, 'back': back})
43          else:
44              return render(request, 'request.html', {'error': 'not_found', 'back': back})
45      else:
46          return render(request, 'request.html', {'error': 'not_found', 'back': back})
47
48  @login_required(login_url='/main/login/')
49  def create_request_view(request):
50      return render(request, 'create_request.html')
51
52  @login_required(login_url='/main/login/')
```

Рис. 3 – views.py (ч. 2)

views.py

main > views.py

```python
@login_required(login_url='/main/login/')
def edit_request_view(request):
    if request.GET.get('id'):
        account = Account.objects.get(user=request.user)
        id = request.GET.get('id')
        if Request.objects.filter(id=id).exists():
            find_request = Request.objects.get(id=id)
            if find_request.account == account:
                return render(request, 'edit_request.html', {'request': find_request})
            else:
                return render(request, 'edit_request.html', {'error': 'not_allowed'})
        else:
            return render(request, 'edit_request.html', {'error': 'not_found'})

@login_required(login_url='/main/login/')
def requests_view(request):
    return render(request, 'requests.html')

@login_required(login_url='/main/login/')
def main_view(request):
    user_account = request.user.account
    user_requests = Request.objects.filter(account=user_account)
    return render(request, 'main.html', {'requests': user_requests})

@login_required(login_url='/main/login/')
def add_request(request):
    if request.method == 'POST':
        account = Account.objects.get(user=request.user)
        new_request = Request()
        new_request.account = account
        new_request.name = request.POST.get('name').strip()
        new_request.description = request.POST.get('description').strip()

        if request.FILES.get('photo') is not None:
            new_request.photo = request.FILES.get('photo')

        new_request.district = request.POST.get('district').strip()
        new_request.city = request.POST.get('city').strip()
```

Рис. 4 – views.py (ч. 3)

views.py

main > views.py

```
88            new_request.district = request.POST.get('district').strip()
89            new_request.city = request.POST.get('city').strip()
90            new_request.contacts = request.POST.get('contacts').strip()
91            if request.POST.get('lat') and request.POST.get('lng'):
92                new_request.lat = request.POST.get('lat')
93                new_request.lng = request.POST.get('lng')
94            new_request.save()
95            return redirect('/main/')
96
97    @login_required(login_url='/main/login/')
98    def get_requests(request):
99        if request.method == 'GET':
100           district = request.GET.get('district')
101           city = request.GET.get('city')
102           requests_list = []
103           if district and city:
104               requests = Request.objects.filter(district__iexact=district, city__iexact=city, is_closed=False)
105               requests_list = get_requests_list(requests)
106           elif district and not city:
107               requests = Request.objects.filter(district__iexact=district, is_closed=False)
108               requests_list = get_requests_list(requests)
109           elif not district and city:
110               requests = Request.objects.filter(city__iexact=city, is_closed=False)
111               requests_list = get_requests_list(requests)
112           else:
113               requests = Request.objects.filter(is_closed=False).order_by('-id')[:20]
114               requests_list = get_requests_list(requests)
115           return JsonResponse(requests_list, safe=False)
116       else:
117           return HttpResponse('error')
118
119   @login_required(login_url='/main/login/')
120   def update_request(request):
121       if request.method == 'POST':
122           account = Account.objects.get(user=request.user)
123           id = request.POST.get('id')
124           if Request.objects.filter(id=id).exists():
125               find_request = Request.objects.get(id=id)
```

Рис. 5 – views.py (ч. 4)

views.py

main > views.py

```
125             find_request = Request.objects.get(id=id)
126             if find_request.account == account:
127                 find_request.name = request.POST.get('name').strip()
128                 find_request.description = request.POST.get('description').strip()
129 
130                 if request.FILES.get('photo') is not None:
131                     find_request.photo = request.FILES.get('photo')
132 
133                 find_request.district = request.POST.get('district').strip()
134                 find_request.city = request.POST.get('city').strip()
135                 find_request.contacts = request.POST.get('contacts').strip()
136                 if request.POST.get('lat') and request.POST.get('lng'):
137                     find_request.lat = request.POST.get('lat')
138                     find_request.lng = request.POST.get('lng')
139                 find_request.save()
140                 return redirect('/main/')
141             else:
142                 return redirect('/main/')
143         else:
144             return redirect('/main/')
145 
146 @login_required(login_url='/main/login/')
147 def change_status(request):
148     if request.method == 'POST':
149         account = Account.objects.get(user=request.user)
150         id = request.POST.get('id')
151         if Request.objects.filter(id=id).exists():
152             find_request = Request.objects.get(id=id)
153             if find_request.account == account:
154                 find_request.is_closed = not find_request.is_closed
155                 find_request.save()
156                 return HttpResponse('ok')
157             else:
158                 return HttpResponse('not_allowed')
159         else:
160             return HttpResponse('not_found')
161 
162 @login_required(login_url='/main/login/')
```

Рис. 6 – views.py (ч. 5)

main > views.py

```
@login_required(login_url='/main/login/')
def get_requests_analitics(request):
    if request.method == 'GET':
        districts = Request.objects.values('district').annotate(count=Count('district')).order_by('-count')
        return JsonResponse(list(districts), safe=False)
    else:
        return HttpResponse('error')

def auth(request):
    if request.method == 'POST':
        email = request.POST.get('username').strip()
        password = request.POST.get('password').strip()
        user = authenticate(request, username=email, password=password)
        if user is not None:
            if user.account.is_active_account == False:
                return redirect('/main/login?error=check_email')
            else:
                login(request, user)
                return redirect('/main/')
        else:
            return redirect('/main/login?error=wrong_data')
    else:
        return redirect('/main/login/')

def register(request):
    if request.method == 'POST':
        name = request.POST.get('username').strip()
        email = request.POST.get('email').strip()
        password = request.POST.get('password').strip()
        if  User.objects.filter(email=email).exists():
            return redirect('/main/register?error=user_exist')
        else:
            activation_code = generate_code()
            user = User.objects.create_user(email, email, password)
            user.first_name = name
            user.save()
            account = Account(user=user, activation_code=activation_code)
```

Рис. 7 – views.py (ч. 6)

main > views.py

```
200                 send_auth_email(email, activation_code, 'activate')
201                 return redirect('/main/login?error=check_email')
202         else:
203             return redirect('/main/register/')
204 
205 def activate(request):
206     if request.method == 'GET':
207         code = request.GET.get('code')
208         if Account.objects.filter(activation_code=code).exists():
209             account = Account.objects.get(activation_code=code)
210             account.is_active_account = True
211             account.save()
212             return redirect('/main/login?error=account_activated')
213         else:
214             return redirect('/main/login?error=wrong_code')
215     else:
216         return redirect('/main/login/')
217 
218 def new_password(request):
219     if request.method == 'POST':
220         code = request.POST.get('code').strip()
221         password = request.POST.get('password').strip()
222         if Account.objects.filter(change_password_code=code).exists():
223             account = Account.objects.get(change_password_code=code)
224             user = account.user
225             user.set_password(password)
226             user.save()
227             return redirect('/main/login?error=new_password')
228         else:
229             return redirect('/main/change_password?error=wrong_code')
230     else:
231         return redirect('/main/change_password/')
232 
233 def change_password_request(request):
234     if request.method == 'POST':
235         email = request.POST.get('email').strip()
236         if User.objects.filter(email=email).exists():
```

Рис. 8 – views.py (ч. 7)

main > views.py

```
236         if User.objects.filter(email=email).exists():
237             user = User.objects.get(email=email)
238             change_password_code = generate_code()
239             user.account.change_password_code = change_password_code
240             user.account.save()
241             send_auth_email(email, change_password_code, 'change_password')
242             return redirect('/main/change_password?error=check_email')
243         else:
244             return redirect('/main/change_password?error=user_not_exist')
245     else:
246         return redirect('/main/change_password/')
247
248 def generate_code():
249     timestamp = int(time.time())
250     return hashlib.md5(str(timestamp).encode('utf-8')).hexdigest()
251
252 def send_auth_email(email, code, type):
253     user = config['gmail_address']
254     app_password = config['gmail_password']
255     host = config['server_host']
256     to = email
257
258     if type == 'activate':
259         subject = 'Activate account code from naPomich'
260         content = ['Your activate account link: ' + host + 'main/activate?code=' + code]
261     elif type == 'change_password':
262         subject = 'Change password code from naPomich'
263         content = ['Your change password link: ' + host + 'main/change_password?code=' + code]
264
265     with yagmail.SMTP(user, app_password) as yag:
266         yag.send(to, subject, content)
267         return True
268
269 def get_requests_list(requests):
270     requests_list = []
271     for request in requests:
272         requests_list.append({
```

Рис. 9 – views.py (ч. 8)

```
269 def get_requests_list(requests):
270     requests_list = []
271     for request in requests:
272         requests_list.append({
273             'id': request.id,
274             'name': request.name,
275             'description': request.description,
276             'contacts': request.contacts,
277             'is_closed': request.is_closed,
278             'district': request.district,
279             'city': request.city,
280             'lat': request.lat,
281             'lng': request.lng,
282             'created_at': request.created_at.strftime("%d.%m.%Y %H:%M"),
283         })
284     return requests_list
```

Рис. 10 – views.py (ч. 9)

requests.js

main > static > requests.js > addEventListener('click') callback

```
let mymap = L.map('map').setView([50.907688, 34.796716], 13);

L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token=pk.eyJ
    maxZoom: 18,
    id: 'mapbox/streets-v11',
    tileSize: 512,
    zoomOffset: -1
}).addTo(mymap);

window.onload = function() {
    loadData();
}

function loadData(district = '', city = '') {
    let xhr = new XMLHttpRequest();
    let url = '/get_requests/?district=' + district + '&city=' + city;
    xhr.open('GET', url, true);
    xhr.send();
    xhr.onreadystatechange = function() {
        if (xhr.readyState != 4) return;
        if (xhr.status != 200) {
            alert(xhr.status + ': ' + xhr.statusText);
        } else {
            let data = JSON.parse(xhr.responseText);
            addMarkers(data);
            addRequests(data);
        }
    }
}

function addMarkers(data) {
    mymap.eachLayer(function(layer) {
        if (layer instanceof L.Marker) {
            mymap.removeLayer(layer);
        }
    });
    for (let i = 0; i < data.length; i++) {
        if (data[i].lat == null || data[i].lng == null) {
```

Рис. 11 - Requests.js (ч. 1)

requests.js

main > static > requests.js > addEventListener('click') callback

```
    for (let i = 0; i < data.length; i++) {
        if (data[i].lat == null || data[i].lng == null) {
            continue;
        }
        let marker = L.marker([data[i].lat, data[i].lng]).addTo(mymap);
        let popupMarkup = '<div class="popup">' + '<div class="popup-name" onclick="openRequest(' + data[i].id +
        marker.bindPopup(popupMarkup);
    }
}

function addRequests(data) {
    let requests = document.getElementById('request-list');
    requests.innerHTML = '';
    if (data.length == 0) {
        requests.innerHTML = '<div class="empty-list">Немає запитів 🤷</div>';
        return;
    }
    for (let i = 0; i < data.length; i++) {
        let request = document.createElement('div');
        request.onclick = function() {
            openRequest(data[i].id);
        }
        request.className = 'request';
        request.id = 'request' + data[i].id;
        request.innerHTML = '<div class="request-name">' + data[i].name + '</div>' + '<div class="request-time">
        requests.appendChild(request);
    }
}

function openRequest(id) {
    let location = '../open_request?id=' + id + '&back=requests';
    window.location.href = location;
}

document.getElementById('submitSearch').addEventListener('click', function() {
    let district = document.getElementById('district').value;
    let city = document.getElementById('city').value;
```

Рис. 12 - Requests.js (ч. 2)

main > static > JS requests.js > addEventListener('click') callback

```
76
77  document.getElementById('analiticsBtn').addEventListener('click', function() {
78      let xhr = new XMLHttpRequest();
79      let url = '/get_requests_analitics/';
80      xhr.open('GET', url, true);
81      xhr.send();
82      xhr.onreadystatechange = function() {
83          if (xhr.readyState != 4) return;
84          if (xhr.status != 200) {
85              alert(xhr.status + ': ' + xhr.statusText);
86          } else {
87              let data = JSON.parse(xhr.responseText);
88              data = prepareData(data);
89              openAnalitics(data);
90          }
91      }
92  });
93
94  function openAnalitics(data) {
95      let analiticsModal = document.getElementById('analiticsModal');
96      analiticsModal.hidden = false;
97      let analiticsList = document.getElementById('analiticsList');
98      analiticsList.innerHTML = '';
99      for (let i = 0; i < data.length; i++) {
100         let analiticsItem = document.createElement('li');
101         analiticsItem.className = 'analitics-item';
102         analiticsItem.innerHTML = data[i].district + ': ' + data[i].count;
103         analiticsList.appendChild(analiticsItem);
104     }
105
106     analiticsModal.addEventListener('click', function(event) {
107         analiticsModal.hidden = true;
108     });
109 }
```

Рис. 13 - Requests.js (ч. 3)

```
112 function prepareData(data) {
113     let result = [];
114     for (let i = 0; i < data.length; i++) {
115         if (result.find(x => x.district == data[i].district.toLowerCase()) == undefined) {
116             let obj = {
117                 'district': data[i].district.toLowerCase(),
118                 'count': data[i].count
119             }
120             result.push(obj);
121         } else {
122             result.find(x => x.district == data[i].district.toLowerCase()).count += data[i].count;
123         }
124     }
125
126     for (let i = 0; i < result.length; i++) {
127         result[i].district = result[i].district[0].toUpperCase() + result[i].district.slice(1);
128     }
129
130     return result;
131 }
```

Рис. 14 - Requests.js (ч. 4)

main.js

main > static > main.js > editRequest

```
1   function openRequest(event, id) {
2       if (!event.target.classList.contains('edit-request')) {
3           let location = '../open_request?id=' + id + '&back=main';
4           window.location.href = location;
5       }
6   }
7
8   function editRequest(id) {
9       let location = '../edit_request?id=' + id;
10      window.location.href = location;
11  }
```

Рис. 15 - Main.js

create_request.js

main > static > create_request.js > ...

```
1   let mymap = L.map('map').setView([50.907688, 34.796716], 13);
2   let marker;
3
4   document.getElementById('photo').addEventListener('change', function () {
5       let file = this.files[0];
6       if (file) {
7           document.getElementById('filename').innerHTML = file.name;
8           document.getElementById('photoBtnLabel').innerHTML = '+ Змінити фото';
9       }
10  });
11
12  L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token=pk.eyJ1Ijoi
13      maxZoom: 18,
14      id: 'mapbox/streets-v11',
15      tileSize: 512,
16      zoomOffset: -1
17  }).addTo(mymap);
18
19  mymap.on('click', function(e) {
20      if (marker) {
21          mymap.removeLayer(marker);
22      }
23      marker = L.marker(e.latlng).addTo(mymap);
24      document.getElementById('lat').value = e.latlng.lat;
25      document.getElementById('lng').value = e.latlng.lng;
26  });
27
```

Рис. 16 - Create_request.js

edit_request.js

main > static > edit_request.js > id

```
let mymap = L.map('map').setView([50.907688, 34.796716], 13);
let marker;
let lat = document.getElementById('lat').value;
let lng = document.getElementById('lng').value;

if (lat != 'None' && lng != 'None') {
    mymap.setView([lat, lng], 13);
    marker = L.marker([lat, lng]).addTo(mymap);
}


L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token=pk.eyJ1IjoibWFwYm
    maxZoom: 18,
    id: 'mapbox/streets-v11',
    tileSize: 512,
    zoomOffset: -1
}).addTo(mymap);

mymap.on('click', function(e) {
    if (marker) {
        mymap.removeLayer(marker);
    }
    marker = L.marker(e.latlng).addTo(mymap);
    document.getElementById('lat').value = e.latlng.lat;
    document.getElementById('lng').value = e.latlng.lng;
});

document.getElementById('photo').addEventListener('change', function () {
    let file = this.files[0];
    if (file) {
        if (document.getElementById('image')) {
            document.getElementById('image').remove();
        }
        document.getElementById('filename').innerHTML = file.name;
        document.getElementById('photoBtnLabel').innerHTML = '+ Змінити фото';
    }
});
```

Рис. 17 - Edit_request.js (ч. 1)

```

function toggleStatus(event) {
    event.preventDefault();
    let id = document.getElementById('reqId').value;
    let url = '../change_status/';
    let xhr = new XMLHttpRequest();
    xhr.open('POST', url, true);
    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhr.setRequestHeader('X-CSRFToken', csrftoken);
    xhr.send('id=' + id);
    xhr.onload = function () {
        if (xhr.status === 200 && xhr.responseText === 'ok') {
            window.location.href = '../main/';
        }
    }
}
```

Рис. 18 - Edit_request.js (ч. 2)