

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Комп'ютерних наук та технологій

(повне найменування інституту, назва факультету)

Кафедра комп'ютерних систем та мереж

(повна назва кафедри)

Пояснювальна записка

до дипломної роботи

бакалавр

(рівень вищої освіти (освітньо-кваліфікаційний рівень))

на тему Розробка інтерактивної підсистеми екологічного моніторингу

Виконав: студент(ка) 4 курсу, групи КНТ-527

Спеціальності 123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Комп'ютерна інженерія»

Шевчук М.С.

(прізвище та ініціали)

Керівник Зеленьова І.Я

(прізвище та ініціали)

Рецензент Морщавка С.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повно підписування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти (освітній ступінь) бакалаврський
 Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
 Завідувач кафедри Кудерметов Р.К.

« » « » 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Шевчука Микити Сергійовича
(прізвище, ім'я, по батькові)

Тема проекту (роботи) Розробка інтерактивної підсистеми екологічного моніторингу

Виконавець проекту (роботи) Зеленьова Ірина Яківна, к. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Відомо, що наказом вищого навчального закладу від «17» березня 2021 року № 81
 прок подання студентом проекту (роботи) 12 травня 2021 року

Вхідні дані до проекту (роботи) наявні методи та алгоритми розробки тестових
ттів, архітектура проекту за шаблоном проектування Page Object

міст розрахунково-пояснювальної записки (перелік питань, які потрібно
 бити) 1) Аналіз технічного завдання, дослідження наявних методів та аналогі
тичного тестування вебсайтів;

робка архітектури проекту та написання тестових наборів;

длідження ефективності використання локаторів та оцінка часових витрат
ння різних версій сайту.

робка керівництва користувача.

ік графічного матеріалу (з точним зазначенням обов'язкових креслень)

руктурна схема-системи;

горитм роботи системи;

валізація гістограми;

будова проекції;

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посила консультанта	Підпис, дата	
		завдання	прийняв виконання завдання
1-3	Зеленьова І.Я., к.т.н., доцент		
Нормоконтроль	Щербак Н.В., ст. викл.		

7. Дата видачі завдання 01.03.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Прийняв
		01.03.2021 р.	
1	Огляд існуючого стану проблеми, виділення недоліків та переваг	15.03.2021 р.	
2	Розробка архітектури проєкту	20.03.2021 р.	
3	Розробка алгоритмів тестових наборів	01.04.2021 р.	
4	Реалізація тестових методів	10.04.2021 р.	
5	Дослідження ефективності використання докаторів	13.04.2021 р.	
6	Оцінка часових витрат на тестування сайту	15.04.2021 р.	
6	Оформлення отриманих результатів у ПЗ	20.04.2021 р.	
7	Оформлення графічного матеріалу	12.05.2021 р.	
8	Оформлення допоміжного матеріалу		

Студент

(підпис)

М.С. Шевчук

(ініціали та прізвище)

Керівник проєкту (роботи)

(підпис)

І.Я. Зеленьова

(ініціали та прізвище)

РЕФЕРАТ

ПЗ 76 с., 45 рис., 8 табл., 22 посилань

МОНІТОРИНГ, НЕСАНКЦІОНОВАНЕ ЗВАЛИЩЕ, ОПИС МІТКИ, КАРТА, МІТКИ, АВТОМАТИЗАЦІЯ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, EXPRESS, NODE-JS, REACT-ROUTER-DOM, MONGOOSE, MONGODB, JAVASCRIPT, REACT, MATERIALIZE CSS, ІНТЕРАКТИВНА ПІДСИСТЕМА ЕКОЛОГІЧНОГО МОНІТОРИНГУ

Об'єкт дослідження – інтерактивні системи онлайн моніторингу.

Предмет дослідження – методи автоматизації системи моніторингу несанкціонованих звалищ.

Мета роботи – розробка інтерактивної підсистеми екологічного моніторингу.

Дипломна робота складається з чотирьох розділів.

Перший розділ присвячено огляду існуючих рішень проблеми, аналізу програмного забезпечення призначеного для моніторингу несанкціонованих звалищ, аналізу архітектурних особливостей елементного базису.

У другому розділі обирається засіб проектування для роботи, показані основні типи даних та структурна схема програми, описано модулі розроблюваної інтерактивної підсистеми екологічного моніторингу, описано взаємодію компонентів системи.

У третьому розділі розроблено алгоритм роботи, описано повідомлення програмісту.

У четвертому розділі описано призначення програми, умови виконання та налаштування програми, розроблено керівництво користувача, описано тестування і результати.

ЗМІСТ

Перелік умовних позначень.....	7
Вступ.....	8
1 Аналіз сучасного стану проблеми.....	9
1.1 Аналіз предметної області.....	9
1.2 Огляд існуючих рішень проблеми.....	10
1.3 Порівняння існуючих рішень проблеми.....	13
1.4 Постановка завдання до роботи.....	15
2 Вибір та обґрунтування засобів проєктування системи.....	16
2.1 Структурна схема програми.....	16
2.2 Вибір інструментарію розробки програмного забезпечення.....	18
2.3 Опис модулів розроблюваної інтерактивної підсистеми екологічного моніторингу.....	24
3 Розробка клієнтської та серверної частин комплексу.....	35
3.1 Алгоритм функціонування програми.....	35
3.2 Основні рішення щодо реалізації компонентів системи.....	40
3.3 Підсумки опису програми.....	46
3.4 Призначення та умови застосування програми.....	46
3.5 Характеристики програми.....	47
3.6 Звернення до програми.....	49
3.7 Вхідні та вихідні дані.....	51
3.8 Повідомлення програмісту.....	51
4 Дослідження комплексу та рекомендації з використання.....	53
4.1 Призначення програми.....	53
4.2 Умови виконання програми.....	54
4.3 Виконання програми.....	55
4.4 Повідомлення оператора.....	58
4.5 Приклад роботи користувача з системою.....	60

4.6 Опис методики проведення тестування.....	63
4.7 Результати тестування.....	63
Висновки.....	66
Перелік джерел посилання.....	68
Додаток А.....	70

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ACID – Atomicity, Consistency, Isolation, Durability.

API – Application Programming Interface.

ER – Entity-Relationship.

GPL – General Public License.

GUI – graphical user interface.

IDE – Integrated Development Environment.

БД – База даних.

ОС – Операційна система.

ПП – Програмний продукт;

СКБД – Система керування базами даних.

ВСТУП

Проблема із відходами стоїть перед людством з того самого часу, як воно з'явилося на Землі, і з часом ця проблема стає все серйознішою. На сьогодні в Україні проблема сміттєзвалищ – одна з найважливіших і найактуальніших серед проблем забруднення навколишнього середовища. Це питання потребує негайного вирішення не тільки в Україні, а й у всьому світі. У кожному людському помешканні утворюється величезна кількість непотрібних матеріалів та виробів, починаючи від старих газет та журналів, порожніх консервних банок, пляшок, харчових відходів, обгортки та упаковок, закінчуючи битим посудом, зношеним одягом та поламаною побутовою чи офісною технікою. Кожного дня ми змушені стикатися з відходами: вдома, на вулиці. Всюди нас оточують папірці, обгортки з пластика, скло, целофан та ін. Із зростанням кількості міст та промислових підприємств постійно збільшується кількість відходів. Промислові і побутові відходи створюють безліч проблем, таких як транспортування, зберігання, утилізація та ліквідація.

Площі сміттєзвалищ в Україні займають більшу територію, ніж площа об'єктів природного заповідного фонду України. Площа території яке займає сміття в Україні складає 7% від всієї території нашої держави, а площа об'єктів природного заповідного фонду – близько 6,7% [1]. Санкціоновані та несанкціоновані сміттєзвалища продовжують розростатися з року в рік, а природні заповідники – майже ні.

За допомогою сучасних технологій люди можуть оперативно відстежити обстановку в деяких містах, наприклад за допомогою Google Maps, ми можемо відстежити чистоту вулиць, які знаходяться вздовж дороги. Також можливо відстежити маршрут або перебування той чи іншої сміттєвивозної машини.

З усього описаного вище витікає, що розробка програмного забезпечення для моніторингу несанкціонованого звалища сміття є дуже актуальною на даний момент і може задовольнити потреби широкого кола користувачів.

1 ОГЛЯД ІСНУЮЧОГО СТАНУ ПРОБЛЕМИ

1.1 Аналіз предметної області

Перед початком аналізу предметної області необхідно конкретизувати проблему, що розглядається. У нашому випадку такою проблемою є потреба у автоматизації процесу моніторингу несанкціонованого звалища сміття у місті, тобто потреба міських жителів та приватних або державних підприємств у даному програмному забезпеченні.

Переходячи до безпосереднього аналізу, необхідно детально розібрати процес додавання мітки збору сміття, а також оповіщення поштою при новому звалищі. Ознайомившись з робочими практиками та методологіями роботи подібних сайтів, можна коротко сказати про наявну потребу у програмному забезпеченні, що вирішувало би наступний спектр проблем:

- зберігання та редагування інформації щодо несанкціонованих звалищ сміття в Україні;
- перегляд усіх несанкціонованих звалищ сміття у країні;
- відправлення повідомлення на пошту при додаванні користувачем нового звалища.

Вирішення наведених вище задач, тобто часткова автоматизація рутинних справ, а також стимул для вирішення екологічних проблем України, це дозволить значно пришвидшити роботу компаній по вивозу сміття і певних організацій, які займаються екологічною проблемою середовища.

Проблема сміття в Україні стоїть досить гостро, при тому, що 95% всіх твердих відходів потрапляють на звалища. Експерти відзначають, що кожен українець щороку генерує близько 350 кг сміття. А в рік на полігони і неофіційні звалища вивозять приблизно 14 мільйонів тонн побутових відходів. Загальна площа всіх звалищ складає майже 5% території країни.

1.2 Огляд існуючих рішень проблеми

В цьому підрозділі наведено список готових програмних продуктів, що надають користувачам вирішення поставленої проблеми. Для кожного з розглянутих прикладів приводиться короткий опис, перелічені їх переваги та недоліки, а також наведені короткі підсумки з аналізу існуючих рішень [1-5].

Всі розглянуті програмні продукти були розроблені для використання у країнах СНД, в яких діджиталізація являється чим то новим, особливо у автоматизації для звичайних громадян. Програмне забезпечення, орієнтоване на використання у інших країнах не було розглянуте через значну різницю у мовах країн СНД та західної Європи а також Америки [5-8].

1.2.1 Система «Map Your Trash» («Cadiffian's»)

Система «Map Your Trash» розроблена компанією «Cadiffian's» [9], що займається пошуком, та вибиранням різних звалищ сміття по всій території Кадіффа. За переконаннями компанії, даний програмний продукт дозволяє пришвидшити роботу пошуку несанкціонованого звалища, а також повідомлення про це самій компанії, для подальшої утилізації сміття.

«Map Your Trash» надає користувачу наступний функціонал:

- перегляд несанкціонованих звалищ в Кадіффа;
- перегляд повних географічних координат кожної мітки звалища;
- моніторинг статусу звалища («Звалище ще не ліквідоване», «Звалище ліквідоване»);

Слід відзначити, що дане програмне забезпечення є абсолютно безкоштовним, але веде спостереження лише за Кадіффом, тобто не має можливості перегляду інших міст, або країн.

Графічний інтерфейс програми наведено на рисунку 1.1.

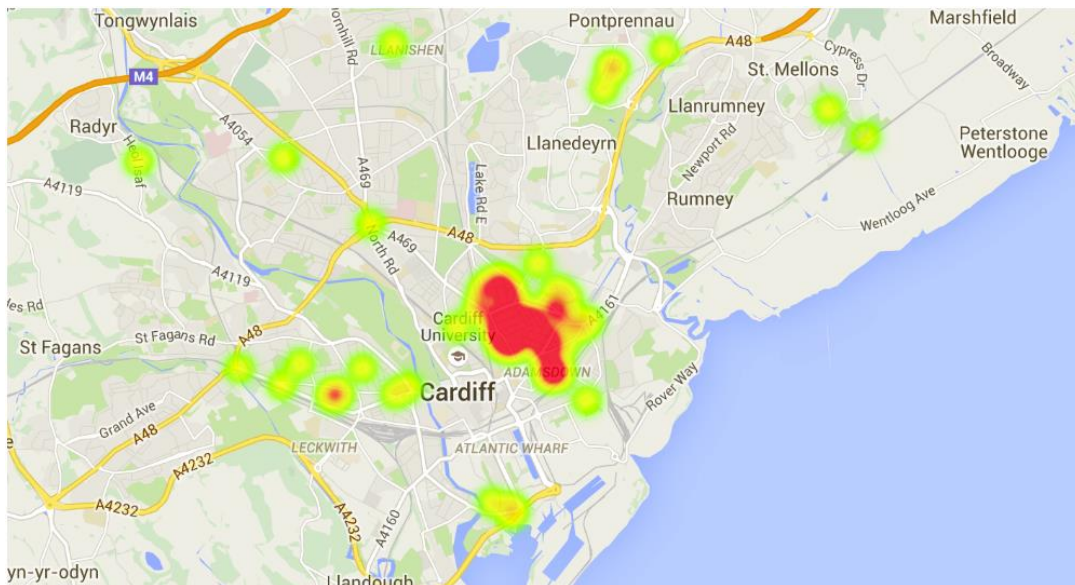


Рисунок 1.1 – Інтерфейс «Map Your Trash»

Серед недоліків даного програмного продукту можна виділити неможливість користувача залишати координати звалища, який він зміг найти, а також локальний моніторинг звалищ, якій можна відстежити лише на певних територіях; також у даному програмному забезпеченні відсутня фотографія звалища, або ж фотографія виконаної роботи, тому неможливо сказати існує звалище на даний момент у встановленій мітці чи ні.

1.2.2 Система «Інтерактивна мапа Мінекоенерго» («Міністерство енергетики та захисту довкілля України»)

Система «Інтерактивна мапа Мінекоенерго» розроблена компанією «Міністерство енергетики та захисту довкілля України» [10]. Як вже зрозуміло з назви компанії, це державна компанія України, яка займається безпекою довкілля. Мапа має дуже приємний дизайн, але не достатньо зрозумілий інтерфейс.

«Інтерактивна мапа Мінекоенерго» надає користувачу наступний функціонал:

- перегляд несанкціонованих звалищ в Україні;
- кластеризація кожної області за кількістю несанкціонованих звалищ;
- перегляд статусу кожного зі звалищ (виконано вивезення чи ні);
- можливість перегляду фотографій по регіонам;

- можливість перегляду дати додавання мітки;
- можливість додати своє звалище, при зверненні до адміністратора;
- можливість перегляду останнього контролю звалищ.

Графічний інтерфейс програми наведено на рисунку 1.2.

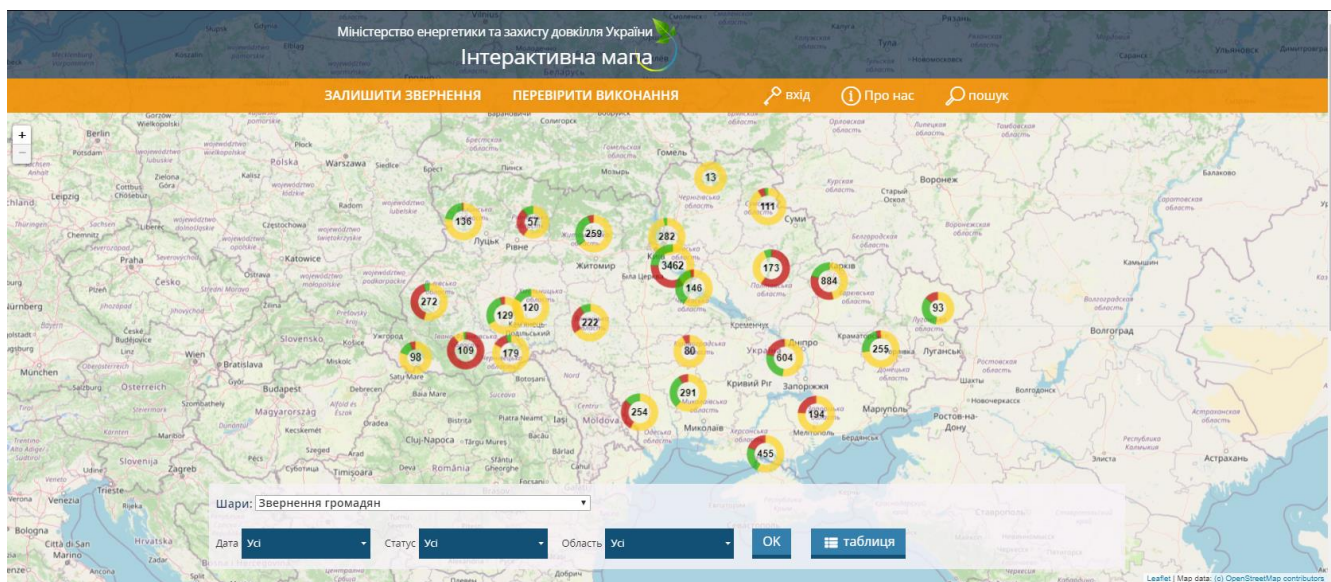


Рисунок 1.2 – Графічний інтерфейс системи «Інтерактивна мапа Мінекоенерго»

Серед недоліків даного програмного продукту можна виділити можливість додавання користувачем свого звалища лише при зверненні до адміністратора сайту. Після звернення треба чекати декілька діб. Також у даному програмному продукті відсутні фотографії при перегляді інформації про мітку.

1.2.3 Система «Trashout» («TrashoutNGO»)

Система «Trashout» розроблена компанією «TrashoutNGO» [11]. Даний програмний продукт дозволяє переглядати несанкціоновані звалища по всьому світі.

Як і в інших розглянутих прикладах, програма надає комплексне рішення для вирішення проблем автоматизації перегляду мітки несанкціонованого звалища на карту.

Серед переваг даного програмного забезпечення можна виділити кластеризацію звалищ по країнам.

Графічний інтерфейс програми наведено на рисунку 1.3.

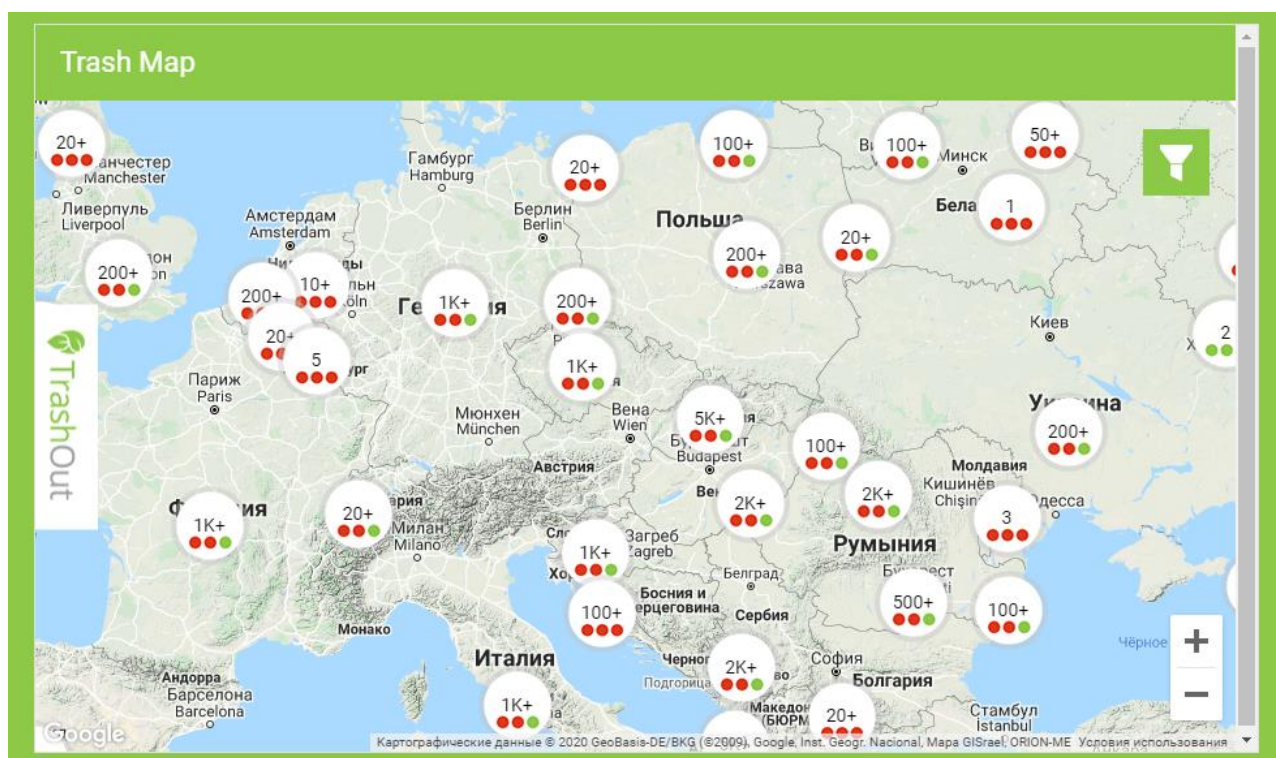


Рисунок 1.3 – Интерфейс системы «Trashout»

Серед недоліків даного програмного продукту можна назвати малий функціонал, порівняно з іншими програмами, відсутня можливість додавання мітки на карту, а також відсутність авторизації користувача у системі. Програма надає лише можливість перегляду міток несанкціонованих звалищ [11].

1.3 Порівняння існуючих рішень проблеми

По закінченню огляду існуючих рішень поставленої проблеми було проведено порівняння розглянутих варіантів реалізації програмного забезпечення. Через дублювання назв модулів/компонент розглянутих програм у якості назв розглянутого програмного забезпечення приводиться назва програмного продукту.

Результати порівняння наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння існуючих рішень проблеми

Метрики порівняння	Програми-аналоги		
	Map Your Trash	Інтерактивна мапа Мінекоенерго	Trashout
Додавання клієнтом нового звалища	-	+	-
Моніторинг виконання	+	+	+
Опис міток	+-	+-	+-
Інтерфейс	-	+-	-
Прегляд звалищ	+	+	+

Порівняння розглянутого програмного забезпечення було проведено з використанням наступних метрик:

- додавання клієнтом свого звалища – система додавання своєї мітки на карті, реалізовані рішеннями з підтримки цілісності даних у БД;
- моніторинг виконання – можливість моніторингу виконання робіт над несанкціонованим звалищем(робота утилізації вже почата, робота утилізації ще не почалась);
- опис міток – можливість опису кожної з мітки, назва міста, фотографія звалища, опис звалища;
- інтерфейс – наявність зручного користувацького інтерфейсу, що відповідає стандартам проектування UI/UX інтерфейсів (англ. «UI» – «User Interface») – «користувацький інтерфейс», «UX» – «User Experience» – «досвід

користування» [12], а також сучасним тенденціям проектування користувацьких інтерфейсів (таким, як наприклад, Google Material Design);

- перегляд звалищ – можливість перегляду користувачами усіх нових звалищ, які були додані.

Оцінювання відповідності тому чи іншому критерію відбувається за наступною шкалою:

- «+» – програма повністю відповідає заданому критерію;
- «+/-» – програма продукт частково відповідає заданому критерію;
- «-» – програма не відповідає заданому критерію.

Як видно з результатів порівняння, жоден з розглянутих варіантів вирішення проблеми не надає зручного користувацького інтерфейсу (або не відповідає сучасним стандартам проектування інтерфейсів) та можливості використання на різних пристроях, а отже попри всі свої переваги не може бути використано великим загалом користувачів.

1.4 Постановка завдання до роботи

Після проведеного аналізу готових рішень проблеми автоматизації моніторингу несанкціонованих звалищ, було формалізовано завдання до роботи, що полягає у створенні клієнт-серверного застосунку (оскільки розроблюване програмне забезпечення має використовувати єдине, централізоване сховище даних для всіх користувачів та надавати інтерфейс для роботи з нею), що відповідає наступним критеріям:

- надає можливість створення та редагування інформації про мітки, яку надає користувач;
- надає можливість перегляду усіх міток та інформації про них, а також авторизацію клієнта в системі;

- надає можливість повідомлення про реєстрацію нового несанкціонованого звалища, та оповіщення про це до спеціальних органів на електрону пошту;
- має зручний користувацький інтерфейс, що відповідає стандартам проектування UI/UX-інтерфейсів, а також Material Design;
- є кросплатформеним: може бути використано на UNIX-подібних операційних системах, таких як Linux, MacOS, Android, iOS, а також на ОС Windows.

На вході програма повинна отримувати авторизованого користувача, а також інформацію про несанкціоноване звалище (місце розташування звалища на карті, опис та фото).

Оцінка якості розробленого програмного забезпечення проводиться шляхом тестування, а також порівняння його на відповідність критеріям, зазначеним у пункті 1.3, табл.1.1 пояснювальної записки.

2 ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Структурна схема програми

Перед проєктуванням окремих модулів і компонентів інтерактивної підсистеми екологічного моніторингу, насамперед, треба визначитись з її базовою архітектурою. Вдалий вибір архітектури може вберегти розробника від прийняття помилкових рішень або необхідності витратити занадто багато часу на організацію роботи у системі, а не на реалізацію бізнес-логіки, тобто логіки роботи програми.

Після детального аналізу технічного завдання, було прийнято рішення про реалізацію клієнт-серверної архітектури, оскільки системою, яка розробляється, має користуватись велика кількість користувачів (клієнтів) у різних місцях (при

кожному виявлені нової звалища), а дані повинні зберігатись централізовано в одному місці (тобто, на загальному сервері).

Графічне представлення клієнт-серверної архітектури наведено на рисунку 2.1.

Сама система має бути представлена у вигляді веб-застосунку. Це позбавить користувачів від необхідності завантажувати програму на свої робочі пристрої (як у випадку з розробкою мобільних та Desktop (з англ. «Робочий стіл») застосунків), а розробників – від потреби реалізації різних застосунків під різні види пристроїв (персональні комп'ютери, ноутбуки, планшети, смартфони тощо) і різні типи операційних систем (Windows, Linux, MacOS, iOS, Android та інші) через можливість створення адаптивного користувацького інтерфейсу (тобто такого, що підлаштовується під робочу машину користувача).

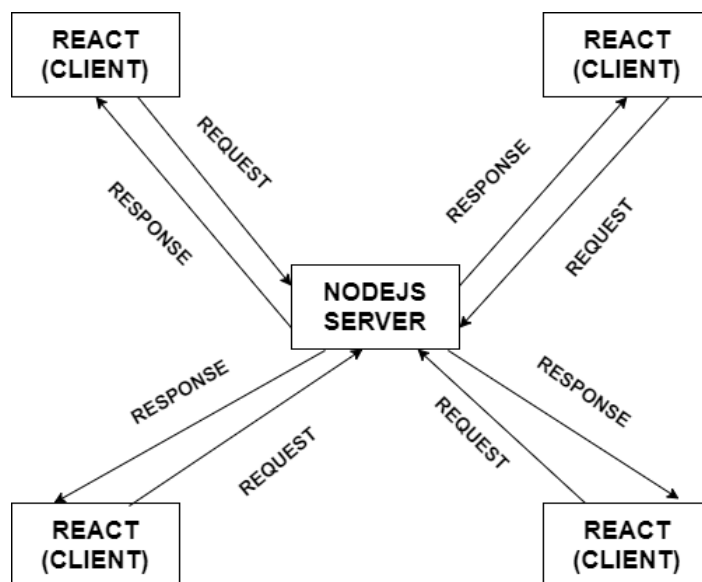


Рисунок 2.1 – Графічне представлення клієнт-серверної архітектури

Процес розробки окремих компонентів системи, а саме клієнтської та серверної її частин, має бути розділений між собою. Це забезпечить можливість тестування і подальшої модифікації цих компонентів незалежно один від одного, і, на відміну від монолітних систем, надасть змогу використовувати різні

інструменти для їх реалізації (мови програмування, програмні платформи, середовища розробки, програми і методології тестування тощо).

Опис структури кожного з компонентів програми, а також особливості їх проектування і інструментів для розробки наведено у наступних підрозділах пояснювальної записки.

2.2 Вибір інструментарію розробки програмного забезпечення

Як і вибір правильної архітектури, вибір відповідних інструментів може вберегти розробника від випадкових помилок, значно пришвидшити розробку системи і привести її у потрібний напрямок. До таких інструментів належать мова програмування, фреймворки, програмні платформи, сторонні бібліотеки, сервіси та програми, середовища розробки і таке інше.

Відповідно до визначеної у підрозділі 2.1 структури програми, а також рішення про роздільну розробку окремих компонентів системи, необхідно обрати інструменти для:

- розробки і тестування серверної частини;
- розробки і тестування клієнтської частини;
- розгортання системи;
- роботи з базою даних.

Окрім вищезазначених пунктів також необхідно визначитись з використанням допоміжних інструментів для розробки системи в цілому, формування документації тощо.

Підходячи до вибору треба враховувати наступні фактори:

- популярність того чи іншого інструменту;
- наявність готових рішень поставленої проблеми;

- можливість інтеграції конкретного інструменту з іншими, що використовуються під час розробки і роботи системи;
- досвід використання інструменту розробником.

Використовуючи вищезазначені критерії було обрано інструменти для розробки і підтримки програмного забезпечення для автоматизації моніторингу несанкціонованих звалищ. Результати порівняльного аналізу і обґрунтування кожного вибору наведено нижче.

2.2.1 Вибір інструментів для розробки і тестування серверної частини

На сьогоднішній день існує багато варіантів для написання веб-застосунків, представлених як спеціалізованими рішеннями, як, наприклад, мови програмування PHP, Perl і подібні, так і різноманітні бібліотеки, фреймворки та програмні платформи, написані на мовах програмування загального застосування, таких як C++, C#, Java, JavaScript, Python, Ruby та інші. Результати порівняння мов програмування наведено у таблиці 2.1

Таблиця 2.1 – Результати порівняння різних мов програмування

Метрики порівняння	Мови програмування				
	Javascript	HP	Ruby	C#	Python
Наявність готових фреймворків для написання серверної частини	+	+	+	+	+
Наявність готових фреймворків для написання клієнтської частини	+	-	-	-	-
Простота синтаксису	+	-	-	-	+-
Наявність динамічної типізації	+	-	+	+	+
Наявність async,await функцій	+	-	+	+	+
Автоматичне курування пам'яттю	+	+	+	+	+

З-поміж великого загалу мов програмування і їх діалектів, що надають готові рішення поставленої проблеми, було обрано мову програмування JavaScript [12 - 20]. Нижче наведені основні переваги цього вибору:

- простий синтаксис Javascript, що, при правильному проектуванні програм, читається подібно до англійської мови;
- основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.
- для забезпечення високого рівня абстракції і досягнення прийнятної ступеня крос-браузерності при розробці веб-додатків використовуються бібліотеки JavaScript. Вони являють собою набір багаторазово використовуваних об'єктів і функцій. Серед відомих JavaScript бібліотек можна відзначити React.js, Vue.js, Ember.js, Adobe Spry, AngularJS, Dojo, Extjs, jQuery, Mootools, Prototype, Qooxdoo (англ.), Underscore і Node.js.

2.2.2 Вибір інструментів для розробки клієнтської частини

На відміну від серверної частини, вибір мови програмування для написання клієнтської – більш проста задача. В світі існує лише одна мова програмування, яка має змогу виконуватись у браузері – це JavaScript (JS). Усі аналоги або є її діалектами (наприклад, TypeScript), або трансклюються у JS-код (наприклад, CoffeeScript, ClojureScript).

Відповідно до пункту 2.2.3 було прийнято рішення про використання компонентно-орієнтованого підходу під час написання клієнтської частини веб-застосунку. Дана парадигма є доволі поширеною, на сьогоднішній день існує безліч фреймворків, що дозволяють писати компонентно-орієнтований код. За даними опитування «State of JavaScript» 2019 року найпопулярнішими з-поміж них є React JS, Vue.js, Angular JS, Svelte, Ember [12 - 20].

Для написання даного програмного продукту було прийняте рішення про використання фреймворку React JS.

React (старі назви: React.js, ReactJS) – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників [21].

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux [12 - 22].

2.2.3 Вибір інструментів для розгортання системи

Розгортання системи – не менш важливий процес ніж написання і тестування програмного коду. Можливість переносити, масштабувати і доповнювати систему є дуже важливою у процесі підтримки програмного забезпечення.

З метою вирішення цієї проблеми було прийняте рішення про використання інструменту віртуалізації heroku.

Heroku — хмарна PaaS-платформа, що підтримує ряд мов програмування. Компанією Heroku володіє Salesforce.com [1]. Heroku, одна з перших хмарних платформ, з'явилась в червні 2007 року і спочатку підтримувала тільки мову програмування Ruby, але на даний момент список підтримуваних мов також включає в себе Java, Node.js, Scala, Clojure, Python і PHP. На серверах Heroku використовуються операційні системи Debian або Ubuntu (яка також заснована на Debian).

2.2.4 Вибір інструментів для роботи з базою даних

Під «інструментами для роботи з базою даних» слід розуміти систему керування базами даних, яка буде використовуватись для організації збереження інформації у системі.

Для використання у даному програмному продукті була обрана СКБД MongoDB, для роботи з котрою і була розроблена спеціально для модуля NodeJs, який є основним засобом для написання серверної частини веб-застосунку, що відповідає за роботою за базою даних.

MongoDB — документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів. MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який з реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується

автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин [22].

Основні можливості MongoDB:

- документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних);
- досить гнучка мова для формування запитів;
- динамічні запити;
- повна підтримка індексів;
- профілювання запитів;
- швидкі оновлення «на місці»;
- ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео;
- журналювання операцій, що модифікують дані в БД;
- підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардінг;
- може працювати відповідно до парадигми MapReduce [12 - 22].

2.2.5 Вибір допоміжних інструментів

Велику роль під час створення програмного забезпечення, крім інструментів для написання коду і розгортання системи, грають засоби для автоматизації процесу розробки. До таких відносяться середовища розробки, аналізатори коду, системи контролю версій, системи безперервної інтеграції та поставки (англ. «CI/CD» – «Continuous Integration / Continuous Delivery») та інші.

У якості середовища для розробки було обрано Visual Studio Code. Visual Studio Code – засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X. За основу

для Visual Studio Code використовуються напрацювання вільного проекту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовують браузерний рушій Chromium і Node.js. Примітно, що про використання напрацювань вільного проекту Atom на сайті Visual Studio Code і в пресрелізі і в офіційному блозі не згадується. Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки [21]. Для вирішення задачі контролю версій під час розробки проекту було обрано Git – централізовану систему, що дозволяє зберігати стан програмного забезпечення на різних етапах його розробки. Також, робота з Git надає можливість використання одного з віддалених сховищ Git-репозиторіїв, таких як, наприклад, Github, Bitbucket, Gitlab та інші. Система безперервної інтеграції та доставки дозволяє автоматизувати процес тестування коду після його завантаження до Git-репозиторію, пакування проекту до Docker-контейнерів та їх завантаження до віддаленого сховища Docker Hub. У якості віддаленого сховища для Git-репозиторіїв було обрано сервіс Gitlab, що має власну систему безперервної інтеграції та доставки Gitlab CI, яка є однією з найпопулярніших і найпростіших у використанні. Для конфігурування системи використовуються yaml-файли, подібно до того, як це відбувається з програмним забезпеченням Ansible, обраним для розгортання на сервері.

2.3 Опис модулів розроблюваної інтерактивної підсистеми екологічного моніторингу

2.3.1 Проектування серверної частини застосунку

Серверна частина програмного продукту (англ. «Backend») має обробляти запити, що надсилаються клієнтською частиною, опрацьовувати дані, що

приходять на сервер, організувати їх для збереження у системі, реалізовувати роботу зі спільною базою (сховищем) даних (БД).

Відповідно до прийнятих на поточний момент практик розробки веб-застосунків, даний компонент системи має використовувати MVC-архітектуру (від англ. «Model-View-Controller»). Графічне представлення цієї архітектури наведено на рисунку 2.2.

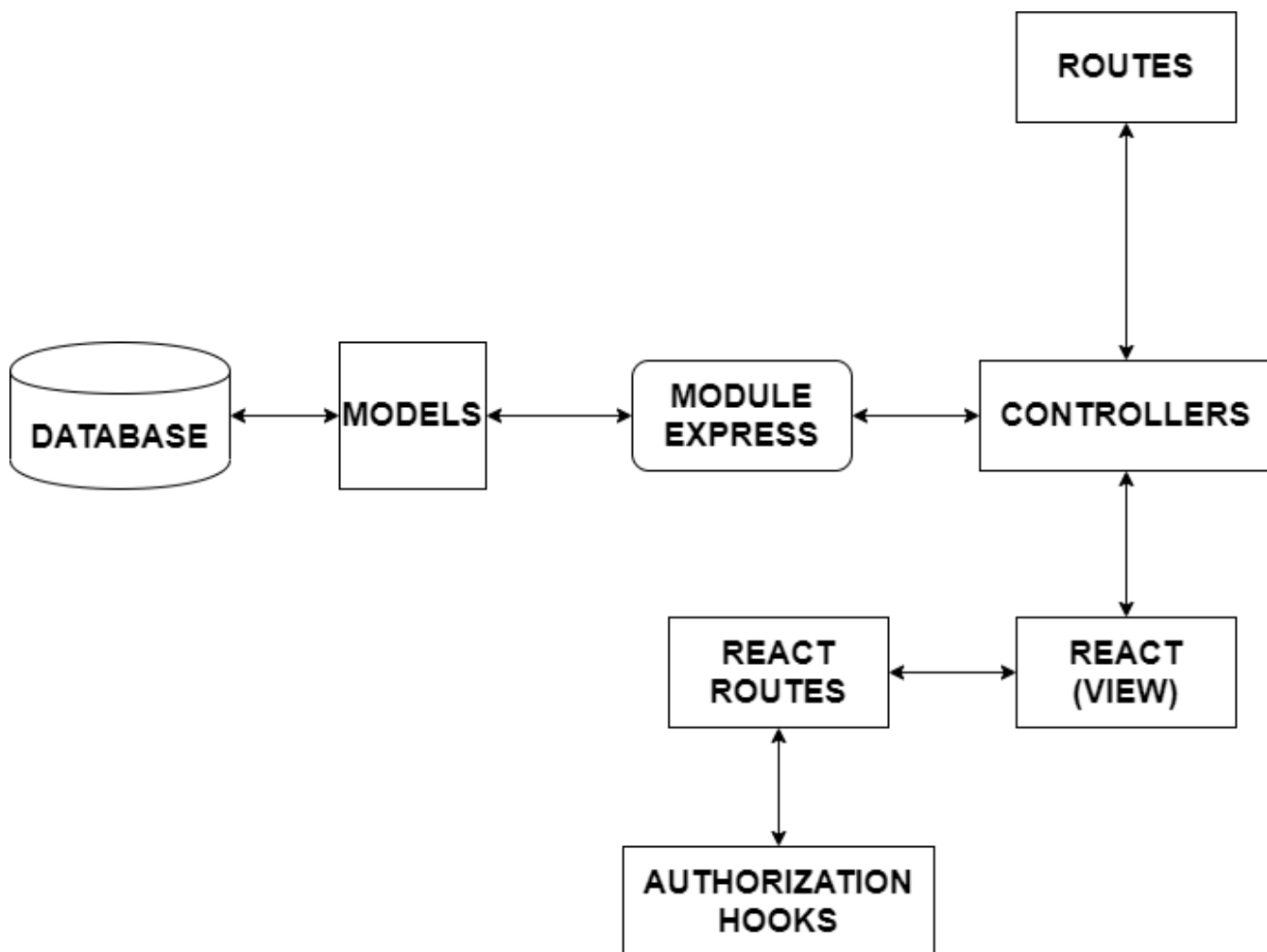


Рисунок 2.2 – Графічне зображення MVC-архітектури

Використання подібного шаблону проектування дозволить подрібнити серверну частину веб-застосунку на ще дрібніші модулі, розробка і тестування яких може йти незалежно один від одного, а, отже, ймовірна модифікація системи не призведе до значних проблем.

Нижче наводиться коротка характеристика виділених модулів:

- модель (англ. «Model») – частина програми, що відповідає за реалізацію усієї бізнес-логіки, тобто логіки роботи з даними, їх обробки, форматування, зберігання у базі даних тощо;
- представлення (англ. «View») – частина програми, яку видно клієнту (користувачу), тобто будь-які текстові дані (неформовані, або приведені до якогось формату, як наприклад, HTML, JSON, XML та інші) і файли, що програма видає у результаті своєї роботи;
- контролер (англ. «Controller») – частина програми, яка зв’язує між собою модель і представлення: використовуючи інтерфейс моделі оновлює представлення для користувача.

2.2.2 Проектування бази даних

Важливою частиною проектування будь-якої системи є рішення про організацію даних в ній. Для системи, що розробляється, було створено ER-діаграму (англ. «Entity Relation»), яка зображена на рисунку 2.3.

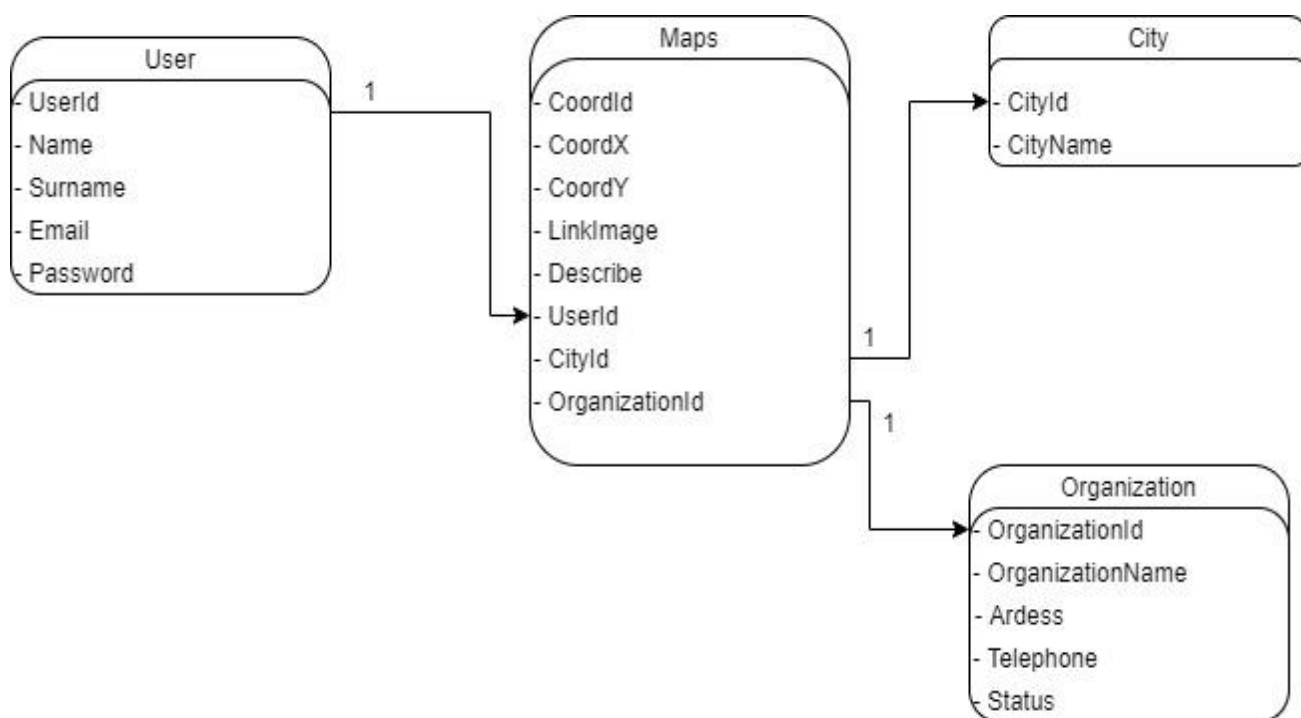


Рисунок 2.3 – ER-діаграма сутностей предметної області

ER-діаграма описує концептуальну схему зв'язків між виділеними сутностями предметної області. Вона дозволяє не прив'язуючись до конкретної реалізації бази даних чи її системи керування (СКБД).

Нижче наведено перелік виділених сутностей предметної області:

- користувач (англ. User) – особа, що має намір користуватися усім функціоналом сайту моніторингу несанкціонованих звалищ сміття в Україні.
- мапа (англ. Map) – інформація, про усі мітки на карті, їх координати X та координати Y, також посилання на фотографію, опис даної мітки, назву міста а також електронна адреса організації, яка займається вивозом сміття.
- місто (англ. City) – інформація, про місто в якому знаходиться несанкціоноване звалище.
- організація (англ. Organization) – інформація про організацію, яка займається ліквідацією звалищ у даному місті, її назва, адреса, телефон, та статус активності.

Нижче наведено таблиці опису ER – діаграми розробленої бази даних (таб. 2.2 – 2.5)

Таблиця 2.2 – Атрибути сутності КОРИСТУВАЧ

№ Атрибут	Атрибут	Тип атрибуту
1	UserId	Type(ObjectId)
2	Name	Type(String)
3	Surname	Type(String)
4	Email	Type(String)
5	Password	Type(String)

Таблиця 2.3 – Атрибути сутності МАПА

№ Атрибут	Атрибут	Тип атрибуту
1	CoordX	Type(Number)
2	CoordY	Type(Number)
3	Linkimage	Type(String)
4	Describe	Type(String)
5	UserId	Type(ObjectId)
6	CityId	Type(ObjectId)
7	OrganizationId	Type(ObjectId)
8	CoordsId	Type(ObjectId)

Таблиця 2.4 – Атрибути сутності МІСТО

№ Атрибут	Атрибут	Тип атрибуту
1	CityId	Type(ObjectId)
2	CityName	Type(String)

Таблиця 2.5 – Атрибути сутності ОРГАНІЗАЦІЯ

№ Атрибут	Атрибут	Тип атрибуту
1	OrganizationId	Type(ObjectId)
2	OrganizationName	Type(String)
3	Adress	Type(String)
4	Telephone	Type(String)
5	Status	Type(Boolean)

Нижче наведено таблиця опису зв'язків між сутностями (таб. 2.6):

Таблиця 2.6 – Зв'язки між сутностями

Сутність №1	Сутність №2	Зв'язок між сутностями
КОРИСТУВАЧ	МАПА	Один-до-багатьох
МАПА	МІСТО	Один-до-багатьох
МАПА	ОРГАНІЗАЦІЯ	Один-до-багатьох

Нижче наведено перелік усіх Атрибуті, та їх функцій у розробці:

- UserId – унікальний ідентифікатор Id користувача;
- Name – Ім'я користувача;
- Surname – Прізвище користувача;
- Email – електронна адреса користувача;
- Password – пароль користувача від аканту;
- CoordX – координата по X мітки;
- CoordY – координата по Y мітки;
- Linkimage – посилання на фотографію мітки;
- Describe – опис мітки;
- CoordsId – унікальний ідентифікатор мітки;
- CityId – унікальний ідентифікатор міста;
- CityName – назва міста;
- OrganizationId – унікальний ідентифікатор організації;
- OrganizationName – назва організації;
- Address – адреса організації;
- Telephone – телефон організації;

Status – статус організації (робоча чи ні).

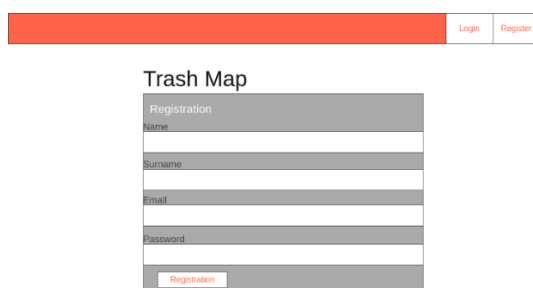
2.3.3 Проектування клієнтської частини застосунку

Клієнтська частина програмного продукту (англ. «Frontend») відповідає за візуальне представлення системи, механізми та інструменти взаємодії користувача з нею.

На відміну від серверної частини веб-застосунку, під час написання клієнтської сторони найкращою практикою на даний момент є використання компонентно-орієнтованого підходу, тобто розбиття усього інтерфейсу за на окремі компоненти: сторінки, блоки, форми для заповнення, таблиці, списки, елементи керування і т.і. Розбиття складної структури веб-сторінки на більш дрібні і прості частини дозволяє пришвидшити розробку, полегшити подальшу модифікацію програми, покращує можливості повторного використання коду, а також підсилює модульність системи.

Нижче наведено перелік основних компонентів клієнтської частини, що включає у себе список сторінок:

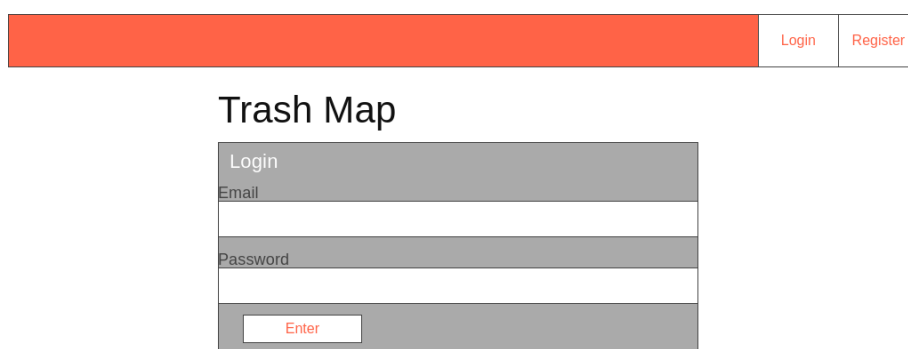
- сторінка реєстрації, що включає у себе можливість реєстрації користувача (рис. 2.4, 2.8);



The screenshot shows a web interface with a red header bar containing 'Login' and 'Register' links. Below the header is a form titled 'Trash Map' with a sub-header 'Registration'. The form contains input fields for 'Name', 'Surname', 'Email', and 'Password', followed by a 'Registration' button.

Рисунок 2.4 – Сторінка реєстрації

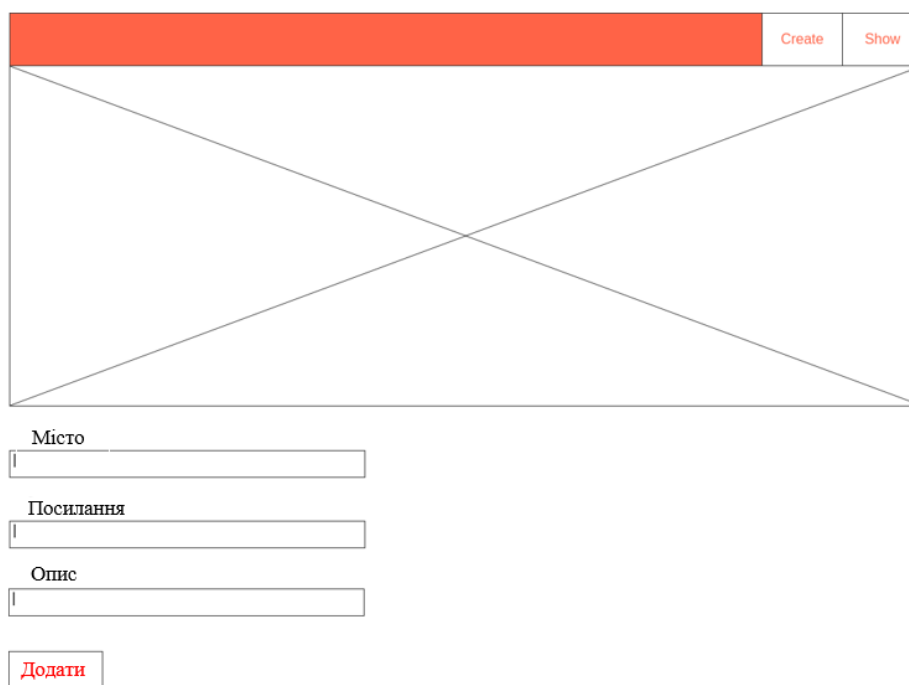
- сторінка авторизації, що включає у себе можливість авторизації користувача (рис. 2.5, 2.9);



The screenshot shows a web interface with a red header bar containing 'Login' and 'Register' links. Below the header is a form titled 'Trash Map' with a sub-header 'Login'. The form contains input fields for 'Email' and 'Password', followed by an 'Enter' button.

Рисунок 2.5 – Сторінка авторизації

- сторінка додавання інформації та координат звалища, що включає у себе можливість додавання зареєстрованим користувачем додавати мітку на карті та опису звалища (рис. 2.6, 2.10);
- сторінка перегляду координат та даних сміття, що включає в себе можливість перегляду інформації про марки(рис. 2.7, 2.11);



Місто

Посилання

Опис

Додати

Рисунок 2.6 – Сторінка додавання координати та даних сміття

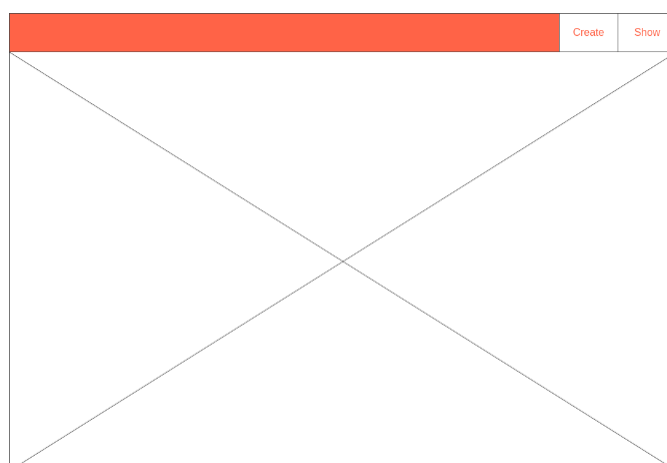


Рисунок 2.7 – Сторінка перегляду координат та даних сміття

	Login	Register
Name		
Surname		
Email		
Password		
Registration		

Рисунок 2.8 – Сторінка реєстрації (мобільний варіант)

	Login	Register
Email		
Password		
Увійти		

Рисунок 2.9 – Сторінка авторизації (мобільний варіант)

Create Show

Місто

Посилання

Опис

Додати

Рисунок 2.10 – Сторінка додавання координати та даних сміття (мобільний варіант)

Create Show

Рисунок 2.11 – сторінка перегляду координат та даних сміття (мобільна версія)

Після визначення основних компонентів клієнтської частини було створено вайрфрейми майбутніх сторінок. Вайрфрейм (англ. «Wireframe») – це образ дизайну низької точності, не сильно деталізований рисунок дизайну в чорно-білому

вигляді з упором не на візуальну складову, а на структуру і зміст. Основна мета створення вайрфреймів полягає в тому, щоб зобразити загальну функціональність самого продукту. Вони відображають такі речі, як навігація, основні групи змісту, кнопки, керуючі елементи і загальний макет. Це спосіб візуалізації того, як елементи інтерфейсу будуть відображатися для користувачів на найважливіших екранах.

2.3.4 Проектування взаємодії компонентів системи

Окрім внутрішньої організації програми велику роль відіграє її зовнішнє представлення, інтерфейс її взаємодії з клієнтською частиною застосунку. Найпопулярнішим, і в багатьох моментах найкращим, варіантом вирішення проблеми організації взаємодії між окремими компонентами системи є розробка RESTful застосунків (від англ. аббревіатури «REST» – «Representational state transfer»).

Нижче наведені основні правила (або архітектурні обмеження) створення RESTful застосунків:

- 1) використання клієнт-серверної архітектури (рішення про використання якої вже було прийнято на етапі проектування).
- 2) відсутність стану, тобто кожен запит містить всю необхідну інформацію для його обробки, і не покладається на те, що сервер знає щось з попереднього запиту.
- 3) хешування, тобто дані, які передаються сервером, повинні містити інформацію про те, чи можна їх хешувати, і якщо можна, то як довго.
- 4) однорідний інтерфейс, який зменшує зв'язність між компонентами і сервісами які вони надають і дозволяє нескладно змінювати компоненти при потребі. Це досягається кількома точнішими обмеженнями:
 - ідентифікація ресурсів: URI запит повинен точно визначати ресурс, що очікується та якого формату повинна бути відповідь;

- маніпуляція ресурсами через представлення: якщо клієнт має представлення ресурсу, включаючи метадані, він має достатньо інформації, щоб модифікувати або видаляти цей ресурс;
- самоописові повідомлення: кожне повідомлення має включати достатньо інформації, щоб обробити повідомлення;
- гіпермедіа як рушій стану застосунку (HATEOAS): Клієнт, що має доступ до REST сервісу, повинен отримати всі наявні сервіси та методи за допомогою гіперпосилань.

5) Шари абстракції: кожен компонент потрапляє в якийсь шар, і спілкується лише з компонентами в шарі під ним або в шарі над ним. Обмеження знання системи одним шаром зменшує складність компонентів.

6) Запитування коду (необов'язкове): клієнти повинні дозволяти розширювати свою функціональність дозволяючи завантаження додаткового коду в формі аплетів чи скриптів.

Як видно з переліку, використання подібної архітектури дозволить створити простий механізм взаємозв'язку окремих компонентів програми. Для організації цього зв'язку буде використовуватись текстовий формат JSON, що легко сприймається людиною, є містким через незначне використання спеціальних символів (на відміну від таких форматів як XML чи HTML), а тому доволі просто обробляється програмно (більшість сучасних мов програмування мають власні інструменти для парсингу, тобто синтаксичного аналізу, даного формату).

3 РОЗРОБКА КЛІЄНТСЬКОЇ ТА СЕРВЕРНОЇ ЧАСТИН КОМПЛЕКСУ

3.1 Алгоритм функціонування програми

Як було визначено ще на етапі проектування, структурно програма складається з двох окремих модулів: клієнтської і серверної частини. Обмін даними

і взаємодія цих компонентів реалізується за рахунок використання протоколу передачі даних HTTP, призначений для надсилання запитів клієнтом і відповідей на ці запити сервером.

Структура взаємодії між компонентами системи зображена на рисунку 3.1.

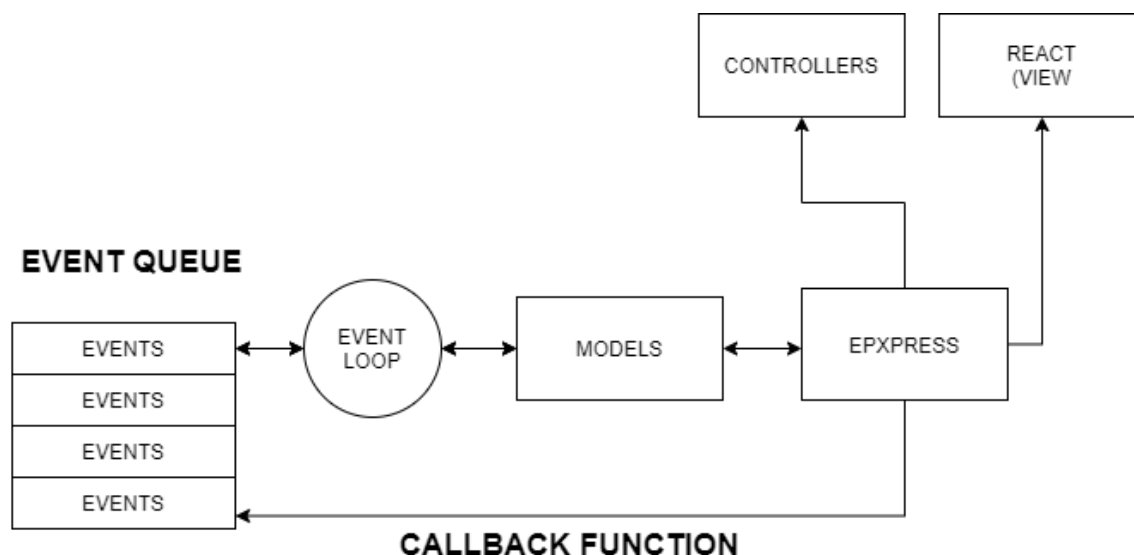


Рисунок 3.1 – Структура взаємодії компонентів системи

Відправляючи запит, клієнт початково буде отримувати згенеровані клієнтським застосунком HTML-сторінки (а також файли стилів .css та файли коду .js, що відносяться до цієї сторінки). Потім, з використанням Fetch API (браузерного програмного інтерфейсу для відправки HTTP-запитів) мови програмування JavaScript, клієнт буде надсилати запити до серверу на отримання чи обробку необхідних даних (інформація, що зберігається у базі даних і необхідна для заповнення поточної сторінки, або дані, відправлені на сервер для обробки, подальшого зберігання тощо).

З метою реалізації аутентифікації і налаштувань прав доступу до тих чи інших ресурсів веб-сервісу були використані токени авторизації. Клієнт, відправляючи дані для авторизації (адресу ел. пошти і пароль), отримує ключ (токен) для подальшого звернення до серверу від свого імені. Така практика є широко розповсюдженою при побудові клієнт-серверних застосунків.

3.1.1 Алгоритм функціонування серверної частини

Порядок прийому та обробки запитів до серверу регламентується стандартом WSGI (з англ. «Web Server Gateway Interface»), розробленого з метою опису взаємодії веб-застосунків, написаних на мові програмування Javascript.

Схема роботи WSGI-застосунків зображена на рисунку 3.2.

Потрапивши до Javascript-застосунку запит спочатку обробляється проміжними шаром (англ. «Middleware»), що видозмінює запити перед їх потраплянням до відповідного контролеру і відповіді, після його роботи. Необхідний контролер обирається в залежності від того, до якої адреси був направлений запит. Контролер, використовуючи інтерфейс моделі для роботи з даними і реалізації бізнес-логіки, тобто правил обробки інформації, відправляє відповідь клієнту, в нашому випадку, в текстовому форматі JSON з вказанням коду статусу HTTP (наприклад, 200 у випадку вдалого запиту, 400 – у випадку некоректно наданих даних, 403 через заборону звернення і так далі). Алгоритм взаємодії окремих модулів серверної частини (представлення, контролерів, моделі) зображено на рисунку 2.3.

Для приведення отриманих даних до Javascript-об'єктів, а також структурування відповідей до формату JSON у NodeJs використовуються серіалізатори (англ. «serializers»). Вони також відповідають за початкову валідацію даних, їх відповідність висунутим вимогам [17].

Django інтуїтивно надає дані, зберігаючись в базі даних, але API повинен передавати інформацію в менш складній структурі. Хоча дані представлені як екземпляри класів Model, їх необхідно перевести в формат JSON для передачі через API.

Сериалізатор DRF виробляє це перетворення. Коли користувач передає інформацію (наприклад, створення нового екземпляра) через API, серіалізатор бере дані, перевіряє їх і перетворює в щось, що Django може скласти в екземпляр моделі. Аналогічним чином, коли користувач звертається до інформації через API, відповідні екземпляри передаються в

сериализатор, который преобразовывает их в формат, который может быть легко передан пользователю как JSON.

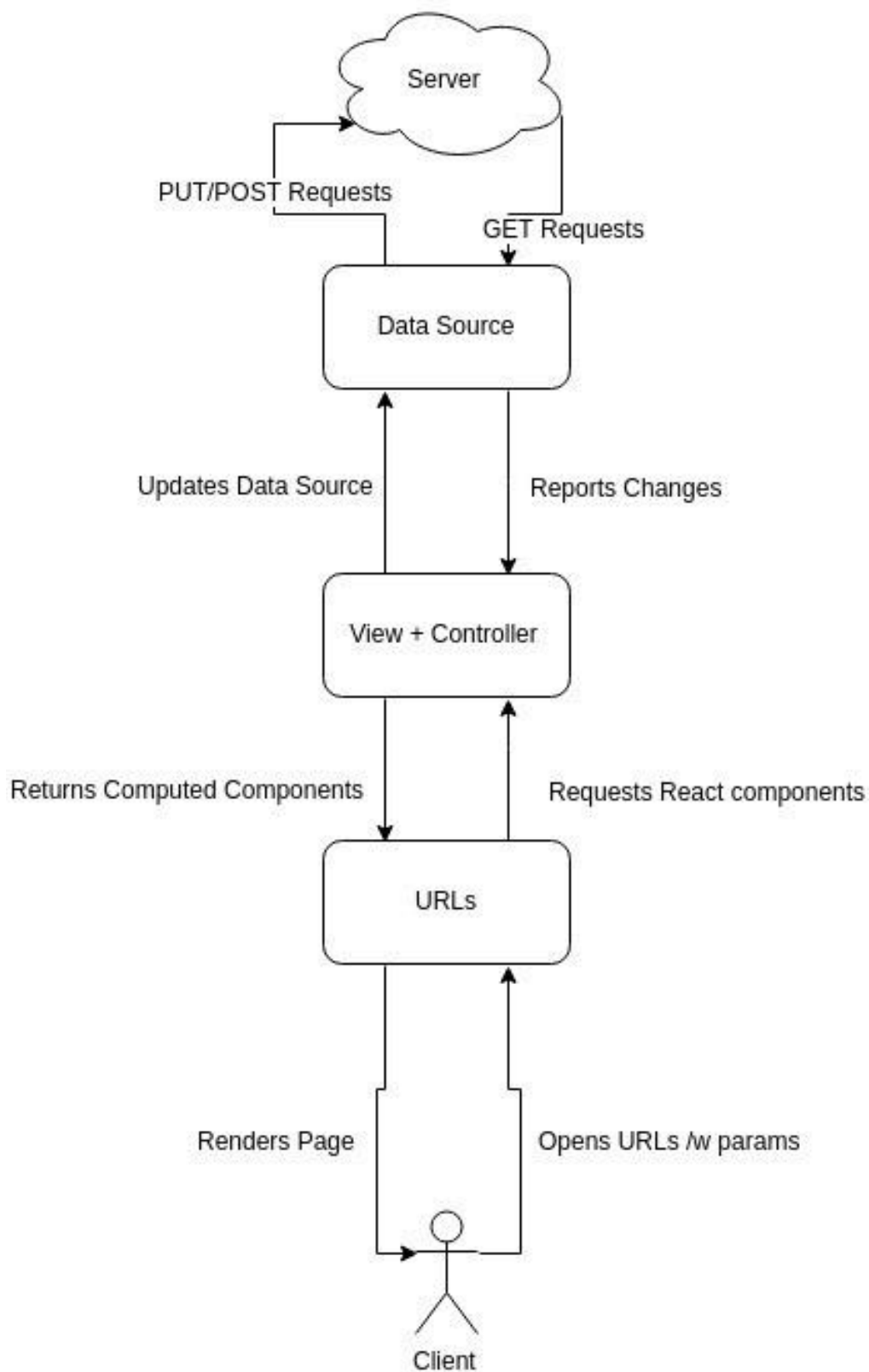


Рисунок 3.2 – Схема работы WSGI-застосунків

Важливою частиною проектування будь-якої системи є рішення про організацію даних в ній. Для системи, що розробляється, було створено ER-діаграму (англ. “Entity Relationship”), яка зображена на рисунку 3.3

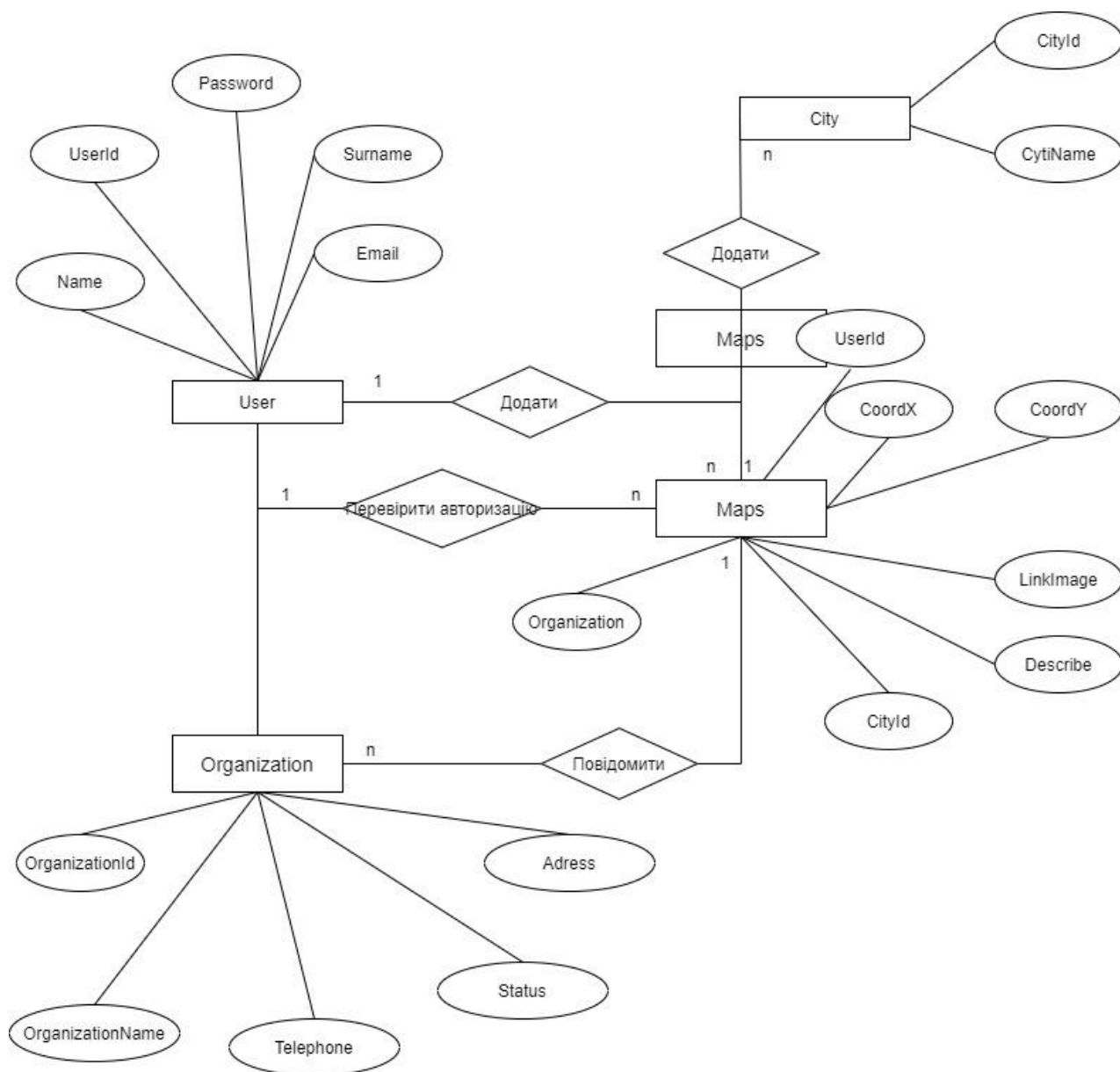


Рисунок 3.3 - ER-діаграма сутностей предметної області

3.1.2 Алгоритм функціонування клієнтської частини

Алгоритм отримання даних користувачем під час звернення до клієнтської частини застосунку подібний до того, як це відбувається у серверній частині. Перейшовши за необхідним посиланням (URL, з англ. – «Uniform Resource Locator»), клієнт отримує дані відповідного компоненту (який може включати в

себе інші компоненти). Компонент у React являє собою HTML-розмітку та інкапсульовані з нею стилі CSS і програмний код мовою JavaScript з використанням React API. Дані у компоненті React оновлюються реактивно, тобто при зміні значення певної змінної оновлюється HTML-код, прив'язаний до неї. Це дозволяє уникнути постійній перезавантажень сторінки під час оновлення даних і досягти відчуття плавності в користувача системи.

3.2 Основні рішення щодо реалізації компонентів системи

3.2.1 Рішення щодо реалізації серверної частини

NodeJs надає можливість розбиття застосунків на окремі структурні частини, та реалізують у собі архітектуру MVC [15]. Нижче наведено перелік виділених частин:

- Routs – призначений для роботи з маршрутизаціями зі сторони серверу;
- Middlewears – призначений для роботи з запитами, і фреймворком для генерації відповіді;
- Models – призначений для роботи з моделями, тобто з базою даних;
- Config – призначений для збереження констант, які не повинні попасти у головний код у чистому виді
- Apps – призначений для призначення інструкцій серверу NodeJs додатку;

У директорії з програмним кодом також наявний модуль «package.json», в якому описані основні налаштування модулів програми, ініціалізація кожного з модулів а також деякі їх константи.

Структурно кожен з компонентів включає наступні файли:

- App.js – файл з набором інструкцій для серверу NodeJs;

- Auth.routes.js – файл з набором маршрутизації для усіх маршрутів, зв'язаних з роботою з авторизацією та реєстрацією;
- Maps.routes.js – файл з набором маршрутизації для усіх маршрутів, зв'язаних з роботою з мапами;
- Maps.js – файл моделі бази даних в якому описуються карти та їх параметри;
- User.js – файл моделі бази даних в якому описуються користувачі;
- auth.middlewear.js – перевіряє чи є користувач авторизований у системі, чи ні;
- default.js – описує константи, які не повинні потрапити до основного коду у чистому виді.

Оскільки Nodejs не надає інструментарію авторизації та реєстрації, під час реалізації серверної частини було використано модуль JWTwebToken – для хешування паролів а також модуль express-sessions для збереження токена в сесії та кеші програми [5].

Для реалізації добування захищеної інформації з фалу config під час розробки проекту був використаний модуль config [6].

Для обробки запитів на виконання CRUD операцій до серверу, реалізована модель заявок за відносним посиланням /routes/, наступна:

- при відправлені POST запиту за посиланням /routes/auth.routes/login буде перевірена валідація полей авторизації, пошук користувача у базі даних, та занесення відповіді у JSON форматі до серверу;
- при відправлені POST запиту за посиланням /routes/auth.routes/register буде перевірена валідація полей реєстрації, та додавання користувача до бази даних, при невдалій перевірці валідації помилка буде занесена у JSON форматі до серверу;
- при відправлені GET запиту за посиланням /routes/maps/map буде виконано запит до бази даних, та повернена у JSON форматі даних про мітки на карті;

– при відправленні POST запиту за посиланням `/routes/auth.routes/register` буде перевірена валідація полів інформації про мітку на карті, та додавання інформації про мітку до бази даних, при невдалій перевірці валідації помилка буде занесена у JSON форматі до серверу.

Взаємодія визначених модулів у серверній частині застосунку показана на рисунку 3.3

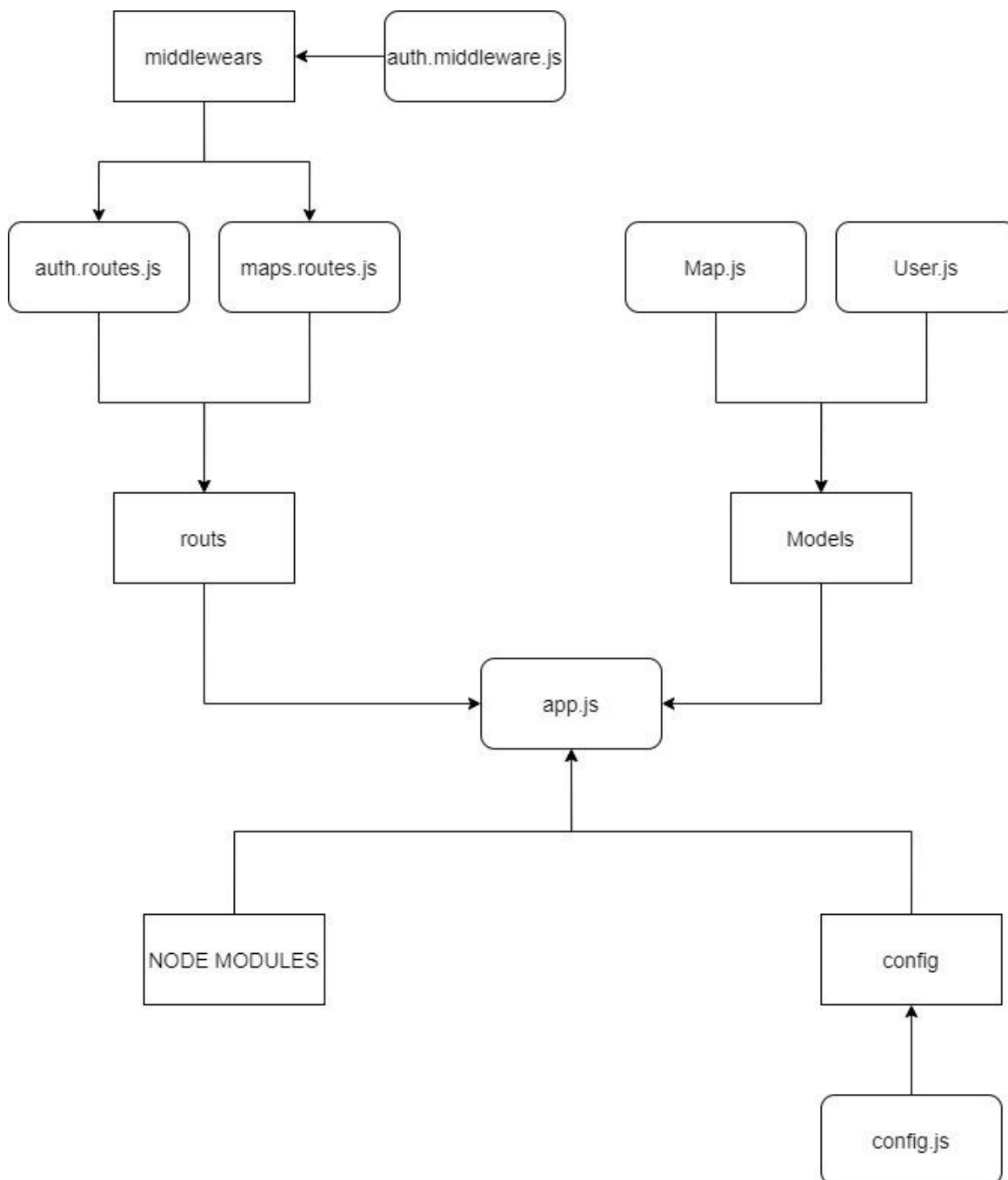


Рисунок 3.4 – Схема взаємодії розроблених модулів серверної частини

3.2.2-Рішення щодо реалізації клієнтської частини

Звичайний React-застосунок має наступну структуру файлів і директорій:

- «package.json» – перелік встановлених залежностей, а також режимів виконання застосунку
- «node_modules/» – директорія для зберігання файлів залежностей (сторонніх бібліотек, фреймворків тощо);
- «public/» – зібраний (скомпільований) проект у готовому вигляді, файли, що змінюються під час зборки знаходяться у директорії public/build/ ;
- «src/» – директорія з кодом програми, саме тут знаходяться файли main.js та App, що є точкою входу у програму.
- «components/» – загальні компоненти системи, такі як Шапка Navbar, MapsPage та Loader;
- «pages» – директорій з усіма сторінками проекту;
- «routes/» – сторінки сайту та компоненти, що відносяться до нього.
- «hooks» – директоріями з перевітками на авторизацію користувача а також показ графічного компонента з помилками серверної частини.

Взаємодія модулів клієнтської частина наведена на рисунку 3.4

Взаємодія визначених модулів наведена на рисунку 3.5

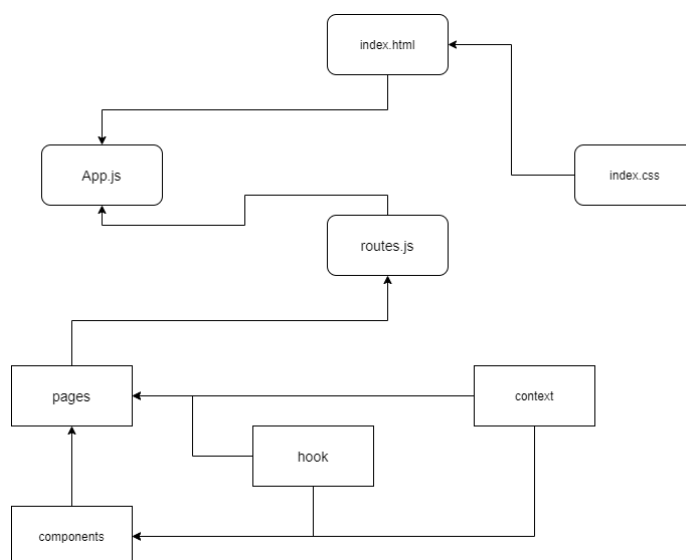


Рисунок 3.5 – Схема взаємодії розроблених модулів клієнтської частини

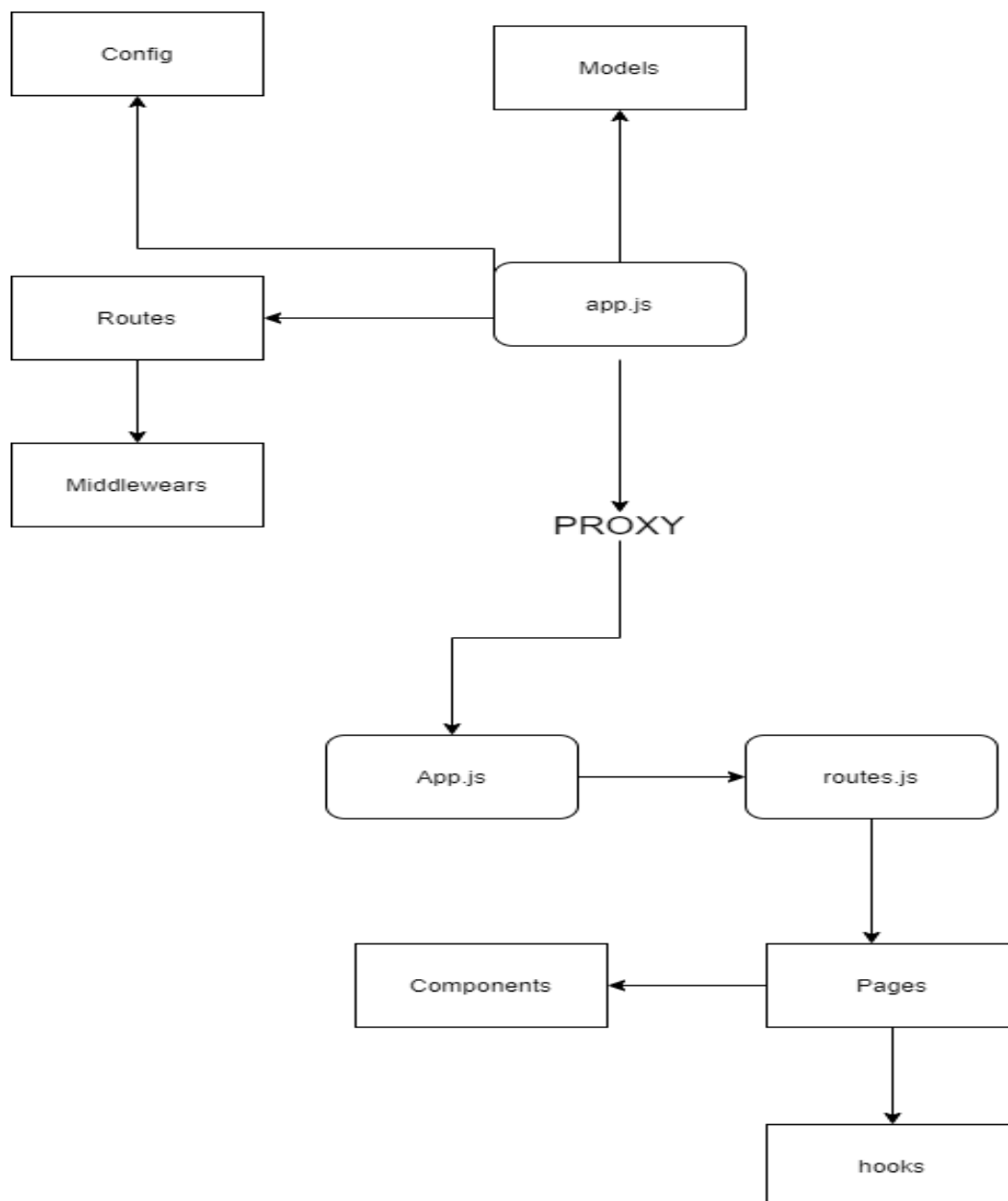


Рисунок 3.6 – Схема взаємодії окремих розроблених модулів

Для верстки (створення дизайну) сайту було використано CSS-фреймворк Materialize CSS, що включає в себе багато готових компонентів (таких, наприклад, як спливаючі повідомлення, діалогові вікна, готові компоненти форм тощо), надає можливості 12-колонної розмітки сайту та створення адаптивного інтерфейсу, як того вимагає завдання до роботи [18].

Для нормальної роботи авторизації у систему, були написані так звані React-Hooks, які дозволяють зберігати хешований токен, у продовж заданого часу.

Для роботи з певними запитами з клієнтської частини на серверну, було написано `http.hook.js`. Цей модуль дозволяє віддавати з клієнтської частини запити (такі як POST чи GET), а також приймати чи віддавати данні.

Для налаштування роутингу (встановлення відповідності URL-адресів та відображаємих компонентів) було використано бібліотеку «`react-router-dom`».

Для зберігання інформації про авторизованого користувача, а також його токена для проведення запитів до серверу використовується локальне сховище браузера, частина браузерного API.

Для коректної роботи програми, було розроблено `middleware`, який дозволяє передавати з клієнтської частини до серверної дані про користувача який пройшов авторизацію на сайті.

Для використання мап було використано «`Google maps API`», для цього треба було зареєструватися у системі «`Google Cloud Platform`», та підтвердити свої платіжні реквізити. Одразу після цього, вам буде надана можливість підключення послуги «`Google Maps API`» для роботи з мапами, а також «`Google Geolocation API`», для роботи з геоданими користувача.

`Google Maps API` не являє собою повний безкоштовний додаток, у разі підключення платіжних реквізитів, зразу на аккаунт користувача «`Google Cloud Platform`», буде зачислено 300 доларів. При використанні мапи 1000 разів, з аккаунту буде списано 7 доларів, а при використанні пошуку геолокації, з аккаунту користувача платформи, буде списано 5 доларів. Якщо місячна сума використаних грошей перевище 300 доларів, то у кінці місяця з платіжних реквізитів буде списана сума, яка перевищує ліміт.

`Google Maps API` надає можливість додавання міток на мапі, а також надає майже повний функціонал для роботи з мапою. Для можливості повного функціоналу роботи з «`Google Maps`», треба підключити «`Google Geolocation API`».

3.3 Підсумки опису програми

В ході виконання етапу реалізації програми було реалізовано визначені раніше компоненти програми. Для спрощення процесу розробки, були використані наявні готові рішення у вигляді модулів для роботи з NodeJs та React, що описані у пункті 3.2

По завершенню етапу реалізації отримано програмне забезпечення, що відповідає всім функціональним і не функціональним вимогам, що висуваються до нього згідно зі складеним раніше технічним завданням. Розроблений програмний продукт відповідає наявним сьогодні тенденціям розробки веб-застосунків і може бути легко модифікований в майбутньому.

3.4 Призначення та умови застосування програми

3.4.1 Призначення та функції

Дане програмне забезпечення призначене для моніторингу несанкціонованого звалища сміття та додавання несанкціонованого звалища до бази даних, та оповіщення про це спеціальних органів.

Функції програмного забезпечення, які розроблені у даному ПЗ:

- додавання міток на карту та опис несанкціонованого звалища;
- перегляд усіх добавлених міток користувачем;
- оповіщення про додавання нового звалища до карти;
- авторизація користувачів на сайті;
- реєстрація користувачів на сайті;
- можливість перегляду деяких сторінок лише авторизованим користувачам;

- валідація даних;
- при введення некоректних даних повідомлення про помилку користувачу;
- після спливу деякого часу відправлення повідомлення на пошту до прокуратури;
- після спливу деякого часу можливість відправлення повторного повідомлення до спеціальних органів прибирання сміття.

3.4.2 Умови виконання програми

Для того, щоб виконання програмного засобу було коректне, необхідні такі параметри системи а також такі периферійні пристрої як:

- одноядерний процесор з тактовою частотою 2.4 ГГц. Та більше;
- обсяг оперативної пам'яті не менше ніж 512Мб;
- жорсткий диск не менш ніж з 50 Гб. пам'яті;
- монітор;
- блок живлення не менш ніж 240 ватт;
- клавіатура;
- Windows 7 та вище.

3.5 Характеристики програми

Система розрахована на цілодобовий режим роботи протягом 365 днів на рік. Засобами контролю моніторингу є автоматизовані вбудовані в платформу програмні компоненти, які самостійно виконують такі дії в разі помилки:

- Якщо відсутнє підключення до бази даних, то система видає помилку адміністратору у консолі серверу, про те що підключення відсутнє;
- Якщо виникла якась системна помилка, то система видає помилку адміністратору у консолі серверу, про те яка сама помилка виникла.

Для відновлення програми потрібно перезапустити сервер та запустити програму через консоль серверу за допомогою команди «`npm run server`».

Обсяг пам'яті, якій займає програма – 270 Мб.

Програма міститься на жорсткому диску сервера.

Програма написана на мові програмування як Javascript, за допомогою серверу NodeJs.

Перелік файлів з яких складається програма:

- `App.js` – файл з набором інструкцій для серверу NodeJs;
- `Auth.routes.js` – файл з набором маршрутизації для усіх маршрутів, зв'язаних з роботою з авторизацією та реєстрацією;
- `Maps.routes.js` – файл з набором маршрутизації для усіх маршрутів, зв'язаних з роботою з мапами;
- `Maps.js` – файл моделі бази даних в якому описуються карти та їх параметри;
- `User.js` – файл моделі бази даних в якому описуються користувачі;
- `auth.middlewear.js` – перевіряє чи є користувач авторизований у системі, чи ні;
- `default.js` – описує константи, які не повинні потрапити до основного коду у чистому виді.
- «`package.json`» – перелік встановлених залежностей, а також режимів виконання застосунку
- «`node_modules/`» – директорія для зберігання файлів залежностей (сторонніх бібліотек, фреймворків тощо);
- «`public/`» – зібраний (скомпільований) проект у готовому вигляді, файли, що змінюються під час зборки знаходяться у директорії `public/build/` ;
- «`src/`» – директорія з кодом програми, саме тут знаходяться файли `main.js` та `App`, що є точкою входу у програму.
- «`components/`» – загальні компоненти системи, такі як Шапка `Navbar`, `MapsPage` та `Loader`;

- «pages» – директорій з усіма сторінками проекту;
- «routes/» – сторінки сайту та компоненти, що відносяться до нього.
- «hooks» – директоріями з перевітками на авторизацію користувача а також показ графічного компонента з помилками серверної частини.
- Routs – призначений для роботи з маршрутизаціями зі сторони серверу;
- Middlewears – призначений для роботи з запитами, і фреймворком для генерації відповіді;
- Models – призначений для роботи з моделями, тобто з базою даних;
- Config – призначений для збереження констант, які не повинні попсти у головний код у чистому виді
- Apps – призначений для призначення інструкцій серверу NodeJs додатку.

3.6 Звернення до програми

Управління вмістом системи відбувається лише через скачану програму. Для того, щоб передати повне управління над програмою потрібно закатити папку з програмою, та сервер NodeJs, а з ним буде встановлений npm, після цього треба встановити деякі модулі, такі як:

- Express – модель для роботи серверної частини проекту [19];
- Jsonwebtoken – модуль для генерації токену [20];
- Express-validator – модель для спрощеної валідації;
- Bcryptjs – модуль для хешування паролей користувача [21];
- Confing – модель для роботи з файлом ;
- Mongoose – модуль для роботи з базою даних;
- Nodemailer – модуль для відсилання повідомлення поштою;

- React – модуль для роботи клієнтської частини серверу;
- Google-maps-react – модуль для роботи з гугл мапами [22];
- Jquery – модуль для спрощеної реалізації адаптивної верстки;
- React-helmet – модуль для роботи з <head> у React;
- Nodemon – модуль для перегрузки серверу NodeJs після кожного змінення;
- Concurrently – модуль для прксування клієнтської частини на серверну частину;
- React-router-dom – модуль для реалізації маршрутизації React [22].

Після цього у коді файлу config.js потрібно встановити адрес електронної пошти з якої треба вислати повідомлення а також пароль від цієї електронної пошти, лістинг 3.1.

Лістинг 3.1 – Адрес електронної пошти

```
"Email": "graduant2020@gmail.com",
"EmailPassword": "240500240500"
```

Також у коді програми у файлі config.js програміст повинен вказати електрону адресу прокуратури, лістинг 3.2.

Лістинг 3.2 – Адресс прокуратури

```
"EmailProk": "Prokuratura@gmail.com",
```

Також у коді файлу «maps.routs.js» треба вказати Email на який буде відправлене повідомлення, лістинг 3.3.

Лістинг 3.2 – Email на який буде відправлене повідомлення

```
to: config.get('Email')
```

3.7 Вхідні та вихідні дані

Вхідними даними програми є Ім'я користувача, фамілія користувача, телефон користувача, пароль користувача, пошта користувача, назва міста, посилання на несанкціонованого звалища, опис звалища.

Вихідними даними програми є результат хешування паролю а також повний опис мітки яка знаходиться на Google мапі, а також сформоване повідомлення на пошту певної спеціальної організації.

3.8 Повідомлення програмісту

Усі повідомлення, що може отримати програміст представлений нижче:

- відсутнє підключення до бази даних – Означає, що програма не може підключитися до бази даних mongoDb;
- під час виникнення якоїсь системної помилки, помилка буде виведена до консолі серверу, а також строку програми на якій виникла помилка програми, та її опис (рис. 3.7).

Приклад системних помилок показано на рисунку 3.7 – 3.8.

```

[0] Server has been started
[0] [nodemon] restarting due to changes...
[0] [nodemon] starting `node app.js`
[0] Error: Configuration property "mongoUri" is not defined
[0]   at Config.get (C:\Users\tapac\Desktop\Projecct\node_modules\config\lib\config.js:182:11)
[0]   at start (C:\Users\tapac\Desktop\Projecct\app.js:18:39)
[0]   at Object.<anonymous> (C:\Users\tapac\Desktop\Projecct\app.js:36:1)
[0]   at Module._compile (internal/modules/cjs/loader.js:1133:30)
[0]   at Object.Module._extensions..js (internal/modules/cjs/loader.js:1153:10)
[0]   at Module.load (internal/modules/cjs/loader.js:977:32)
[0]   at Function.Module._load (internal/modules/cjs/loader.js:877:14)
[0]   at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:74:12)
[0]   at internal/main/run_main_module.js:18:47
[0] Отсутствует подключение к базе данных!!
[0] [nodemon] clean exit - waiting for changes before restart

```

Рисунок 3.7 – Приклад системних помилок під час виконання програми

```
[0] MongooseError [MongooseServerSelectionError]: Could not connect to any servers in your MongoDB Atlas cluster. Make sure your current IP address is on your Atlas cluster's IP whitelist: https://docs.atlas.mongodb.com/security-whitelist/.
[0]   at new MongooseServerSelectionError (C:\Users\rapac\Desktop\Projecct\node_modules\mongoose\lib\error\serverSelection.js:24:11)
[0]   at NativeConnection.Connection.openUri (C:\Users\rapac\Desktop\Projecct\node_modules\mongoose\lib\connection.js:823:32)
[0]   at Mongoose.connect (C:\Users\rapac\Desktop\Projecct\node_modules\mongoose\lib\index.js:333:15)
[0]   at start (C:\Users\rapac\Desktop\Projecct\app.js:18:24)
[0]   at Object.<anonymous> (C:\Users\rapac\Desktop\Projecct\app.js:36:1)
[0]   at Module._compile (internal/modules/cjs/loader.js:1133:30)
[0]   at Object.Module._extensions..js (internal/modules/cjs/loader.js:1153:10)
[0]   at Module.load (internal/modules/cjs/loader.js:977:32)
[0]   at Function.Module._load (internal/modules/cjs/loader.js:877:14)
[0]   at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:74:12)
[0]   at internal/main/run_main_module.js:18:47 {
[0]   reason: TopologyDescription {
[0]     type: 'ReplicaSetNoPrimary',
[0]     setName: null,
[0]     maxSetVersion: null,
[0]     maxElectionId: null,
[0]     servers: Map {
[0]       'cluster0-shard-00-02-h5nch.mongodb.net:27017' => [ServerDescription],
[0]       'cluster0-shard-00-00-h5nch.mongodb.net:27017' => [ServerDescription],
[0]       'cluster0-shard-00-01-h5nch.mongodb.net:27017' => [ServerDescription]
[0]     }
[0]   }
}
```

Рисунок 3.8 – Приклад системних помилок під час підключення до бази даних

Клієнтська система, чи Frontend, була розроблена з дотриманням компонентно-орієнтованої парадигми програмування, використовуючи фреймворк REACT-JS, що набирає популярності в останні роки.

React (іногда React.js или ReactJS) — JavaScript-библиотека[4] с открытым исходным кодом для разработки пользовательских интерфейсов. React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций[5 - 7]. React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL [8].

Якщо використовувати тільки React, стан програми може стати неконсистентним. Бізнес-логіка розмазується по ієрархії компонентів. Потрібно писати більше колбеков для передачі на верхній рівень інформації про стан компонентів. Компоненти програми стає важче перевикористати.

Тому для управління станом часто використовують бібліотеки на додаток до React. Тоді, в термінах опису архітектури, React в нашому MVC- або MVVM-додатку відповідає за View (відображення) плюс загальний каркас додатка. А

робота з Model будується за допомогою бібліотеки управління станом. Наприклад, такий як Mobx.

Бібліотека Redux - це спосіб управляти статусом додатки. Він заснований на декількох концепціях, вивчивши які, ви легко зможете вирішити проблеми з умовою.

GraphQL - це мова запитів для API-інтерфейсів і середовище, в якому вони виконуються. За допомогою GraphQL можна отримати дані з API і передати їх в додаток (від сервера до клієнта). GraphQL розробили в 2012 році в Facebook як альтернативу REST. Нагадаємо, що REST - це архітектурний стиль, який описує правила взаємодії між клієнтом і сервером. При цьому використовуються HTTP-запити, стандарт побудови URL, формати даних JSON і XML. REST API видає відповідь на запит у форматі JSON. Потім відповідь потрібно проаналізувати, вибрати конкретні дані і витягти їх. GraphQL ж дозволяє передати в додаток відразу потрібні дані за один запит, навіть якщо вони знаходяться в декількох джерелах. Завдяки цьому технологія добування даних GraphQL зручніше і практичніше, ніж REST [8].

4 ДОСЛІДЖЕННЯ КОМПЛЕКСУ ТА РЕКОМЕНДАЦІЇ ПО ВИКОРИСТАННЮ

4.1 Призначення програми

Дане програмне забезпечення призначене для екологічного моніторингу та додавання несанкціонованого звалища до бази даних, та оповіщення про це спеціальних органів.

Функції програмного забезпечення, які розроблені у даному ПЗ:

- додавання міток на карту та опис несанкціонованого звалища;
- перегляд усіх добавлених міток користувачем;

- оповіщення про додавання нового звалища до карти;
- авторизація користувачів на сайти;
- реєстрація користувачів на сайті;
- можливість перегляду деяких сторінок лише авторизованим користувачам;
- валідація даних;
- при введення некоректних даних повідомлення про помилку користувачу;
- після спливу деякого часу відправлення повідомлення на пошту до прокуратури;
- після спливу деякого часу можливість відправлення повторного повідомлення до спеціальних органів прибирання сміття.

4.2 Умови виконання програми

Для того, щоб виконання програмного засобу було коректне, необхідні такі параметри системи а також такі периферійні пристрої як:

- одноядерний процесор з тактовою частотою 2.4 ГГц. Та більше;
- обсяг оперативної пам'яті не менше ніж 512Мб;
- жорсткий диск не менш ніж з 50 Гб. пам'яті;
- монітор;
- блок живлення не менш ніж 240 ватт;
- клавіатура;
- Windows 7 та вище.

4.3 Виконання програми

Для запуску програмного забезпечення необхідно відкрити головний файл програми «Project» та вписати в консоль «npm run dev».

Для нормальної роботи програми у оператора повинно бути встановлений браузер з найновішою версією Javascript.

Звертання до програми відбувається у той момент, коли користувач перейшов на любую сторінку сайту.

Список назв полів та інформація, яка повинна бути введена оператором представлені у таблиці 4.1

Таблиця 4.1 – Перелік назв полів та правильна їх інформація

Поле	Коректна інформація
coordsX	Координата X мітки, яка буде автоматично виставлено при нанесення мітки на карті
coordsY	Координата Y мітки, яка буде автоматично виставлено при нанесення мітки на карті
city	Назва міста яка повинна бути вказана при додавання мітки на карту
link	Посилання на фотографію яка повинна бути вказана при додавання мітки на карту
describe	Описання мітки, яка повинна бути вказана при додавання мітки на карту
email	Email-адрес оператора який повинний бути вказане при реєстрації чи авторизації
password	Password оператора який повинний бути вказане при реєстрації чи авторизації

Для використання функцій програми оператор повинен перейти на головну сторінку сайту «localhost:3000», у нього з'являється сторінка авторизації, а також у правому верхньому куті, можливість вибрати сторінку реєстрації, якщо оператор вже зареєстрований у системі, то він може увійти у свій аккаунт у системі, якщо ж ні то зареєструватися.

Після успішного входу оператору надається можливість відвідати дві сторінки, це сторінка додавання нового сміттєзвалища, а також сторінка перегляду інформації про нові сміттєзвалища.

Для додавання нового звалища оператор повинен відвідати сторінку додавання звалища, ввести потрібну інформацію, та додати нове несанкціоноване звалище.

Для перегляду звалища, оператор повинен перейти на сторінку перегляду звалищ, натиснути на любу з міток та подивитися усю потрібну інформацію, яка знаходиться у інформаційному вікні.

Для можливості перегляду своєї геопозиції, оператор повинен натиснути на червону кнопку, яка знаходиться у лівому нижньому куті сорінки перегляду звалищ (рис.4.1), а також правому нижньому куті сторінки додавання звалищ (рис. 4.2), після цього оператор отримає спеціальну мітку на мапі, зі вказаної нової графічної міткою на мапі (рис. 4.3). Для видалення, повторного оповіщення спеціальних органів, чи зміни статусу мітки на виконану, треба відкрити інформаційне вікно даної мітки, та натиснути на відповідну кнопку (рис 4.4).

Інтерфейси зображено на рисунках 4.1 – 4.4.

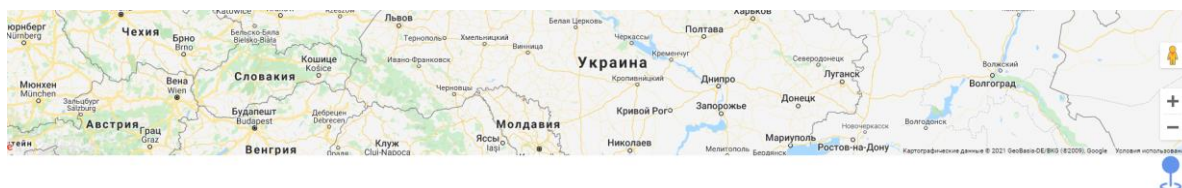


Рисунок 4.1 – Місцезнаходження кнопки пошуку геолокації на сторінці перегляду звалищ.

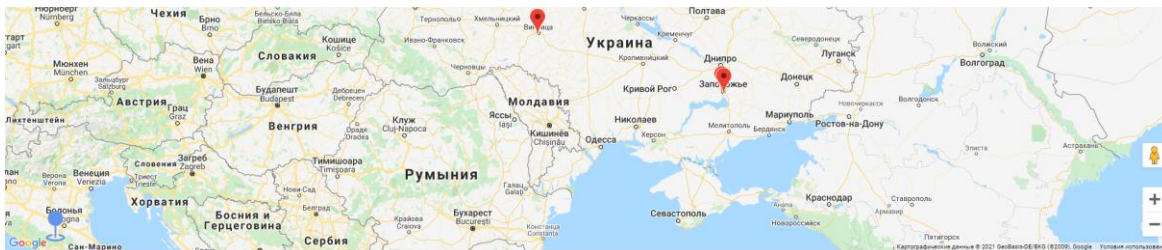


Рисунок 4.2 – Місцезнаходження кнопки пошуку геолокації на сторінці додавання звалищ.

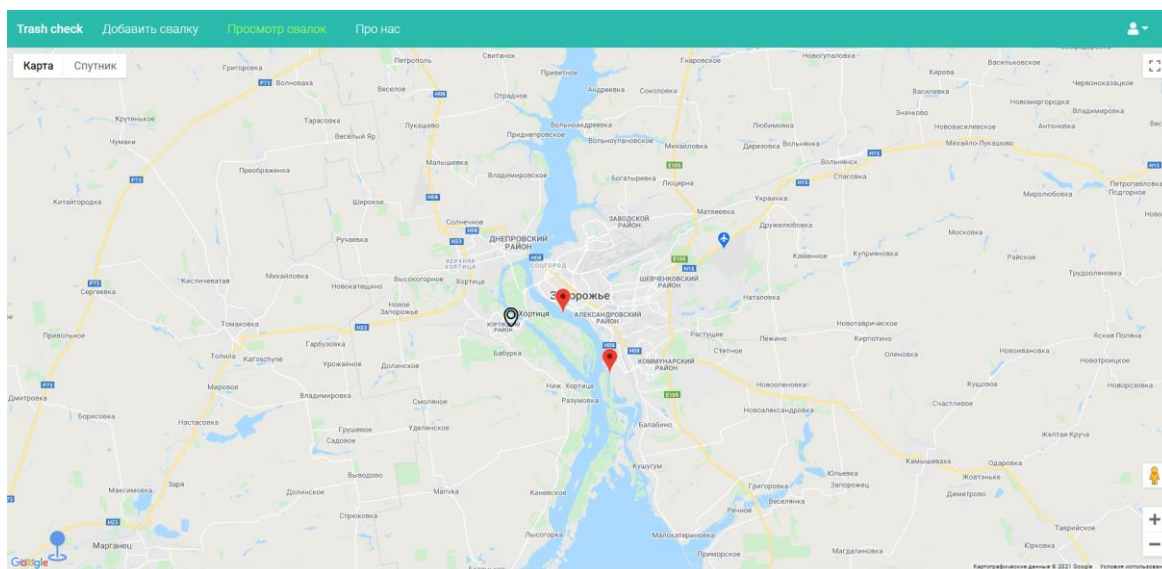


Рисунок 4.3 – Спеціальна мітка на карті, яка показує дійсну геопозицію оператора

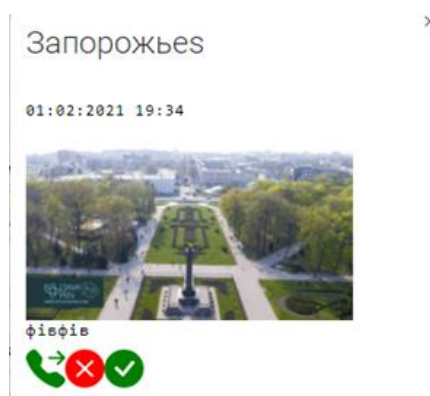


Рисунок 4.4 – Інформаційне вікно, з відповідними кнопками

4.4 Повідомлення оператора

Усі повідомлення, що може отримати оператор представлені нижче:

- «Будь ласка виберете точку на карті !!» – Означає, що оператор не вказав мітку на карті, при додаванні нового звалища;
- «Будь ласка, введіть назву міста вашої точки !!» – Означає, що оператор не вказав назву міста, при додаванні нового звалища;
- «Будь ласка, введіть посилання вашої точки !!» – Означає, що оператор не вказав посилання на фотографію, при додаванні нового звалища;
- «Будь ласка, введіть опис до вашої точки !!» – Означає, що оператор не вказав опис несанкціонованого звалища, при додаванні нового звалища;
- «Щось пішло не так спробуйте знову!» – Означає, що при роботі програми виникла серверна помилка;
- «Некоректні дані при реєстрації» – Означає, що при вводі даних при реєстрації оператор ввів некоректні данні;
- «Такий користувач вже існує» – Означає, що при реєстрації оператор ввів логін, який вже є в системі;
- «Некоректні дані при вході у систему» – Означає, що при авторизації оператор ввів некоректні дані (рис. 4.5);
- «Користувач не знайдений» – Означає, що при авторизації логін користувача не був знайдений (рис. 4.6);
- «Невірний пароль спробуйте знову!» – Означає, що при авторизації оператор ввів неправильний пароль (рис. 4.7);
- «Немає авторизації» – Означає, що оператор не є авторизований у системі (рис. 4.8).

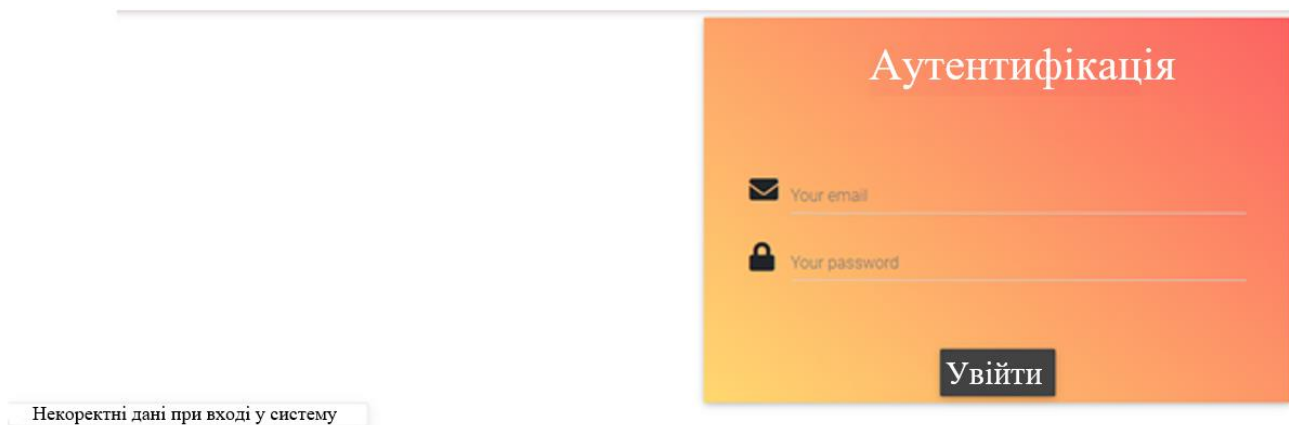


Рисунок 4.5 – Інтерфейс помилки при вказанні некоректних даних при вході користувача у систему

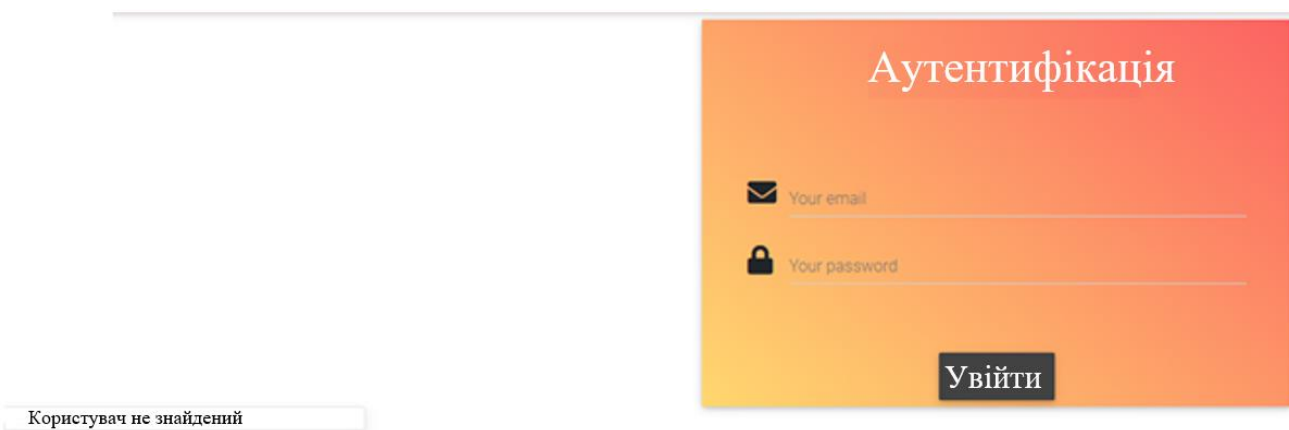


Рисунок 4.6 – Інтерфейс помилки при вказанні некоректних даних при вході користувача у систему



Рисунок 4.7 – Інтерфейс помилки при вказанні некоректних даних при вході користувача у систему



Рисунок 4.8 – Інтерфейс помилки при авторизації користувача

4.5 Приклад роботи користувача з системою

Під час переходу оператору на головну сторінку сайту «localhost:3000», у нього з'являється сторінка авторизації (рис. 4.9), якщо оператор вже зареєстрований у системі, то він може ввести свою пошту у поле email, а потім пароль у поле password які знаходяться у формі Входу яка знаходиться посередині сторінки та натиснути кнопку «Вход», якщо ж у оператора немає зареєстрованого аккаунту, то тоді треба перейти на сторінку реєстрації(рис. 4.10) ввести свої данні у форму реєстрації яка знаходиться посередині сторінки та натиснути кнопку «Регистрация».

Після успішного входу в систему, оператор переходить на сторінку «Додати звалище»(рис. 4.11), оператор може поставити мітку на мапі, якщо ж мітка на карті поставлена з помилкою, оператор може зразу ж її переставити натиснувши на мапу, далі оператор повинен ввести назву міста, посилання на фотографію звалища а також опис несанкціонованого звалища, після цього потрібно натиснути на кнопку «Добавить», та після цього оператора буде автоматично переправлено до сторінки «Просмотр свалок», а також відправлене повідомлення на електронну пошту спеціальних органів по прибиранню сміття.

На сторінці для перегляду звалищ (рис.4.12), оператор може переглянути кожен мітку, яка є на карті, для цього йому потрібно натиснути на будь-який з червоних маркерів на мапі, після цього над маркером з'явиться вікно(див.рис. 5.20), в якому буде описано усі параметри точки, а саме: місто, фотографія та опис звалища.

На сторінці для додавання власного фото онлайн, користувач може натиснути на значок фотоапарата на сторінці «Додати звалище».

Якщо ж оператор захоче вийти з системи, потрібно лише натиснути на кнопку «Logout», яка знаходиться у правому верхньому куті сайту.

Інтерфейси зображено на рисунках 4.9 – 4.14

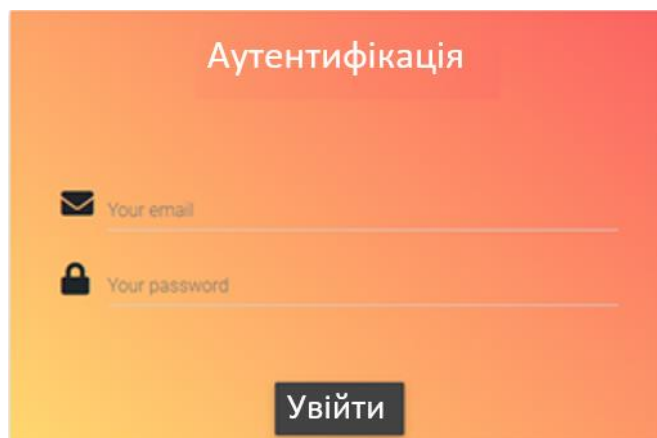


Рисунок 4.9 – Інтерфейс сторінки Авторизації

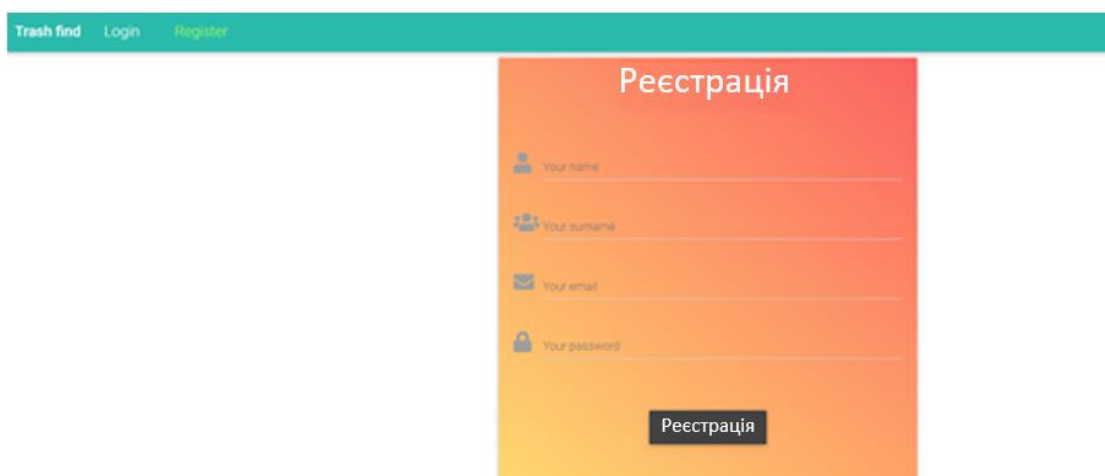


Рисунок 4.10 – Інтерфейс сторінки Реєстрації

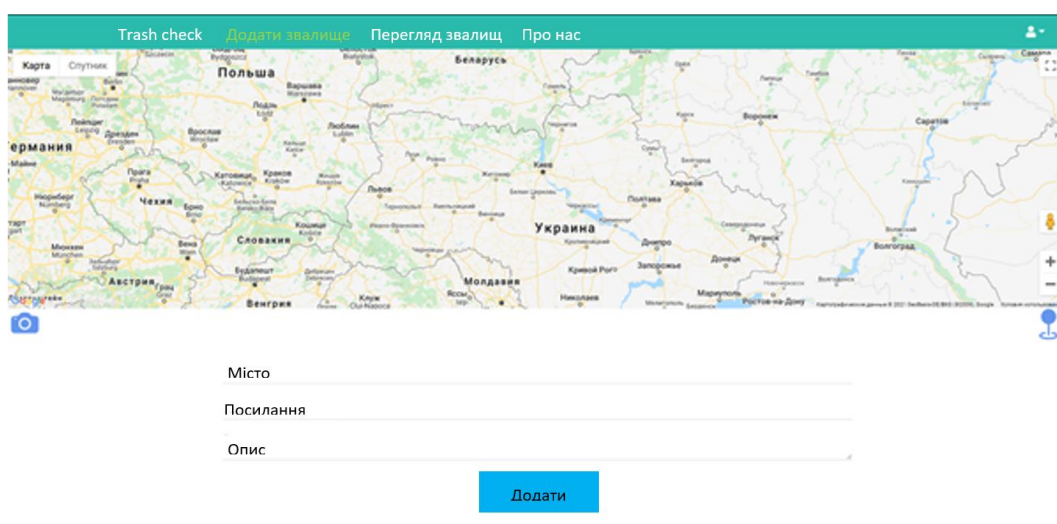


Рисунок 4.11 – Інтерфейс сторінки додавання мітки

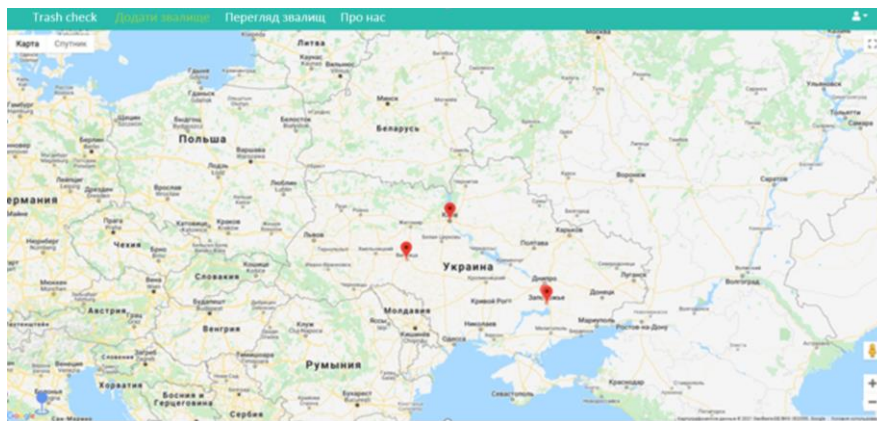


Рисунок 4.12 – Інтерфейс сторінки Перегляду міток

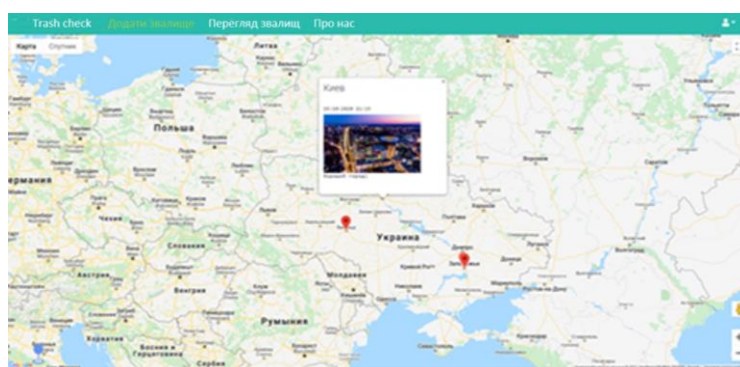


Рисунок 4.13 – Інтерфейс перегляду конкретної мітки

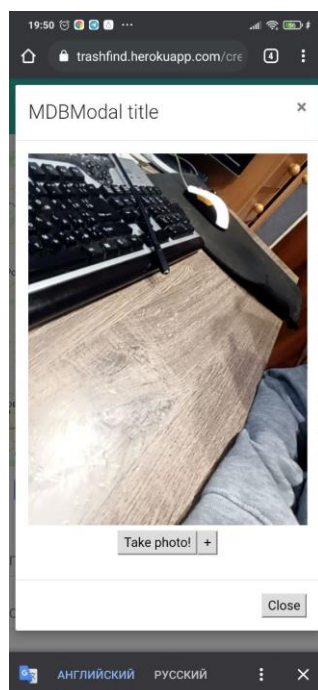


Рисунок 4.14 – Інтерфейс перегляду компонента «додти фото»

4.6 Опис методики проведення тестування

Тестування клієнтської та серверної частини було проведено мануально, тобто власноруч. Таким чином вдалося уникнути написання складної логіки написання автоматичних тестів для графічного інтерфейсу користувача, а також було проведено оцінку якості верстки веб-сайту.

4.7 Результати тестування

Для клієнтської частини тестування було проведено мануально. За допомогою інструментів розробника, вбудованих у браузер Google Chrome було перевірено адаптивність верстки веб-застосунку, те, як графічний інтерфейс змінюється при зміні ширини дисплею. Перевірено зовнішній вигляд з мобільних пристроїв і планшетів, а також персональних комп'ютерів.

Було перевірено реакцію системи на введення як коректних, так і некоректних даних, її поведінку під час помилок валідації. Процес тестування наведено нижче.

Якщо під час введення даних у поля реєстрації користувач не заповне всі поля, та натисне кнопку реєстрації, то отримає повідомлення «Некоректні дані при реєстрації» (рис 4.15);

Якщо під час додавання мітки на мапу, користувач не поставить мітку на мапу, та натисне кнопку додавання мітки, то отримає повідомлення «Будь ласка виберіть точку на карті» (рис. 4.16)

Якщо під час додавання мітки на мапу, користувач не введе опис мітки, та натисне кнопку додавання мітки, то отримає повідомлення «Будь ласка, введіть опис до вашої точки» (рис. 4.17)

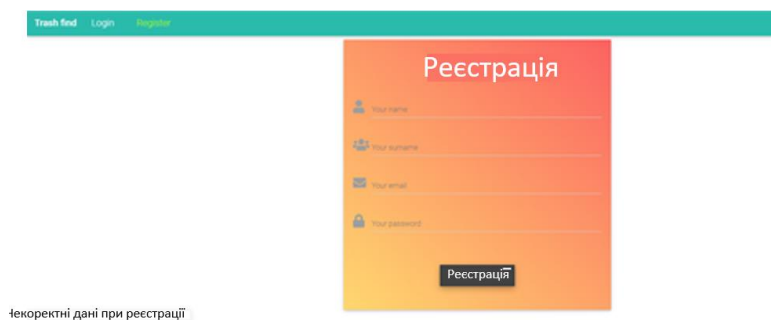


Рисунок 4.15 – Реакція на помилку під час заповнення форми реєстрації

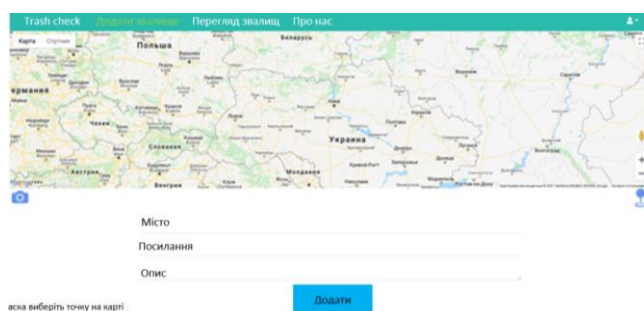


Рисунок 4.16 – Реакція на помилку під час заповнення форми додавання мітки на мапу

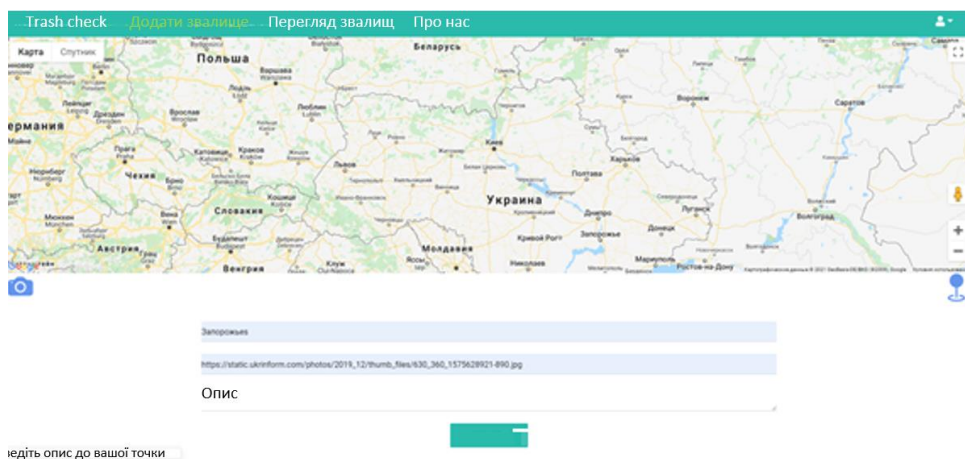


Рисунок 4.17 – Реакція на помилку під час заповнення форми додавання мітки на мапу

Якщо під час введення даних у поля входу в систему користувач не заповне всі поля, та натисне на кнопку входу, то отримає повідомлення «Некоректні дані при вході в систему» (рис 4.18);

Якщо під час введення даних у поля входу в систему користувач введе коректну електронну адресу, та натисне кнопку реєстрації, але некоректний пароль, то отримає повідомлення «Невірний пароль, спробуйте знову» (рис 4.19).

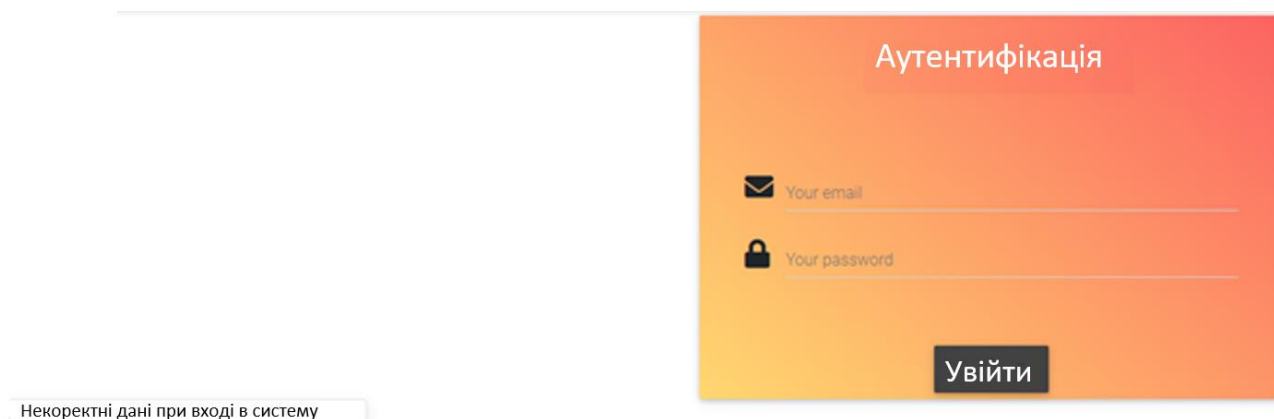


Рисунок 4.18 – Реакція на помилку під час заповнення форми входу

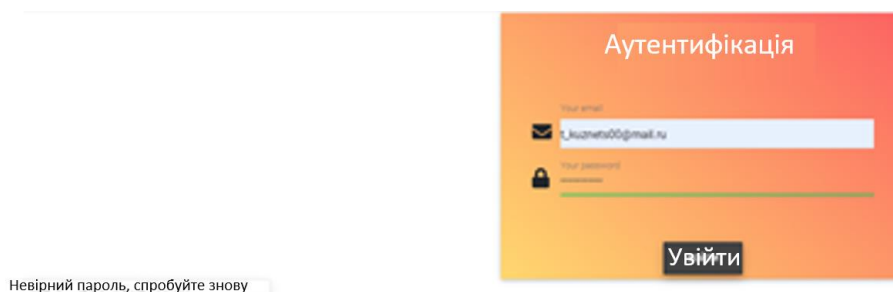


Рисунок 4.19 – Реакція на помилку під час введення паролю користувачем.

Отримане програмне забезпечення є ліпшим за розглянути на першому етапі аналоги. Воно може бути використано усіма людьми у яких є інтернет, а при необхідності легко модифіковане чи використане під час розробки більш значної системи.

В результаті виконання роботи розроблено інтерактивну підсистему екологічного моніторингу, що відповідає всім вимогам, поставленим у технічному завданні до проекту, а також допоможе країні та її населенню боротися проти несанкціонованих звалищ сміття, та тримати екологічну ситуацію у країні в нормі.

ВИСНОВКИ

В ході проектування і розробки програмного забезпечення було проведено аналіз предметної області, огляд готових рішень проблеми. За результатами проведеного аналізу було виявлено, що усі наявні аналоги є комплексними рішеннями проблем онлайн моніторингу несанкціонованого звалища сміття. Крім того, розглянуті застосунки були доволі застарілими, а деякі навіть могли бути використані лише локально, не звертаючись до серверу. Ознайомившись с перевагами готового програмного забезпечення, а також визначивши його недоліки було сформоване задачу розробки і оформлено технічне завдання.

Згідно з розробленим наприкінці етапу аналізу вимог технічним завданням було проведено проектування майбутньої програмної системи. Було прийняте рішення про дотримання клієнт-серверної архітектури, а також окрему розробку клієнтської і серверної частини веб-застосунку.

Клієнтська система, чи Frontend, була розроблена з дотриманням компонентно-орієнтованої парадигми програмування, використовуючи фреймворк REACT-JS, що набирає популярності в останні роки. Для пришвидшення та полегшення процесу верстки було прийнято рішення про використання Materialize CSS, за допомогою якого можна створювати компоненти графічного інтерфейсу, що відповідають стандартам Material Design. Також клієнтська частина є повністю адаптивною, та підтримує усі сучасні телефони, у яких у браузері встановлена найновіша версія Javascript.

Для реалізації серверної частини, або Backend, було використано Nodejs + express, написаний мовою програмування Javascript. З його допомогою було реалізовано REST API для обробки даних у системі. Під час розробки серверної частини було використано технології, що відповідають сучасним тенденціям розробки.

Для забезпечення безпеки користувачів, була використана бібліотека Vscript, за допомогою якої до бази даних зберігається лише хешований пароль.

Отримане програмне забезпечення є ліпшим за розглянути на першому етапі аналоги. Воно може бути використано усіма людьми у яких є інтернет, а при необхідності легко модифіковане чи використане під час розробки більш значної системи.

В результаті виконання роботи розроблено інтерактивну підсистему екологічного моніторингу, що відповідає всім вимогам, поставленим у технічному завданні до проекту, а також допоможе країні та її населенню боротися проти несанкціонованих звалищ сміття, та тримати екологічну ситуацію у країні в нормі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Сфера поводження з відходами залишається для України вкрай проблемною [Електрон. ресурс]. – Режим доступу: <https://ecopolitic.com.ua/news/v-gosekoinspektsii-nazvali-naibolee-zasorennnye-oblasti-v-ukraine/>
2. Національний кадастр антропогенних викидів [Електрон. ресурс]. – Режим доступу: <http://www.seia.gov.ua/seia/doccatalog/document?id=138881>
3. Мониторинг несанкционированных мест скопления отходов с использованием космических снимков [Электрон. ресурс]. – Режим доступа: <https://media.neliti.com/media/publications/311588-monitoring-of-unauthorized-places-of-was-bdd5e39e.pdf>
4. Система космічного моніторингу за сміттєзвалищами твердих побутових відходів [Електрон. ресурс]. – Режим доступу: <http://journals.nupp.edu.ua/sunz/article/view/983>
5. Стихійні звалища: шкода здоров'ю і довкіллю [Електрон. ресурс]. – Режим доступу: <https://mezhova.otg.dp.gov.ua/ua/novini-ta-podiyi/novini/stihijni-zvalishcha-shkoda-zdorovyu-i-dovkilliyu>
6. Стихійні сміттєзвалища – проблема сьогодення [Електрон. ресурс]. – Режим доступу: <https://ns-plus.com.ua/2020/04/13/23286/>
7. Стихійні сміттєзвалища та шляхи боротьби з порушниками [Електрон. ресурс]. – Режим доступу: <https://globynska-gromada.gov.ua/news/1589981882/>
8. Створювати стихійні звалища – заборонено [Електрон. ресурс]. – Режим доступу: <http://myrghorod.pl.ua/news/stvorjuvaty-styhijni-zvalyscha-zaboroneno>
9. Map Your Trash [Електрон. ресурс]. – Режим доступу: <https://globalewaste.org/map/>
10. Інтерактивна мапа Мінекоенерго [Електрон. ресурс]. – Режим доступу: <https://ecomapa.gov.ua/>
11. Trashout [Електрон. ресурс]. – Режим доступу: <https://rabota-psy.livejournal.com/999500.html>

12. NPM: Javascript [Электрон. ресурс]. – Режим доступа:
<https://learn.javascript.ru/>
13. NPM: React [Электрон. ресурс]. – Режим доступа:
<https://www.npmjs.com/package/react>
14. NPM: MongoDB [Электрон. ресурс]. – Режим доступа:
<https://www.npmjs.com/package/mongodb>
15. Visual-studio-code [Электрон. ресурс]. – Режим доступа:
<https://code.visualstudio.com/>
16. NodeJs [Электрон. ресурс]. – Режим доступа:
<https://nodejs.org/uk/download/>
17. NPM: React-materialize [Электрон. ресурс]. – Режим доступа:
<https://www.npmjs.com/package/react-materialize>
18. NPM: Express [Электрон. ресурс]. – Режим доступа:
<https://www.npmjs.com/package/express>
19. NPM: JsonWebToken [Электрон. ресурс]. – Режим доступа:
<https://www.npmjs.com/package/jsonwebtoken>
20. NPM: Bcrypt [Электрон. ресурс]. – Режим доступа:
<https://www.npmjs.com/package/bcrypt>
21. Google API: Google maps API [Электрон. ресурс]. – Режим доступа:
<https://developers.google.com/maps/documentation>.
22. NPM: React-router-dom [Электрон. ресурс]. – Режим доступа:
<https://www.npmjs.com/package/react-router-dom>

ДОДАТОК А

А1. Листинг функции app.js

```
const express = require('express')
const config = require('config')
const mongoose = require('mongoose')

const app = express()

app.use(express.json({extended: true}))

app.use('/api/auth', require('./routes/auth.routes'))
app.use('/api/maps', require('./routes/maps.routes'))

const PORT = config.get('port') || 5000

async function start()
{
  try
  {
    await mongoose.connect(config.get("mongoUri"),
    {
      useNewUrlParser: true,
      useFindAndModify: false,
      useUnifiedTopology: true
    })
    app.listen(PORT, ()=>
    {
      console.log("Server has been started")
    })
  }
  catch(e)
  {
    console.log(e)
  }
}
start()
```

А2. Листинг функции auth.routes.js

```
const {Router} = require('express')
const User = require('../models/user')
```

```

const router = Router()
const jwt = require('jsonwebtoken')
const {check, validationResult} = require('express-validator')
const bcrypt = require('bcryptjs')
const config = require('config')
const auth = require('../middlewares/auth.middleware')
const Map = require('../models/Maps')

//api/auth/register
router.post('/register',
[
    check('email', 'Некорректный email').isEmail(),
    check('password', 'Минимальная        длина        пароля        6
СИМВОЛОВ!').isLength({min:6})
],
async (req, res)=>
{
    try
    {
        const errors = validationResult(req)

        if(!errors.isEmpty())
        {
            return res.status(400).json({
                errors: errors.array(),
                message: 'Некорректные данные при регистрации'
            })
        }

        const {email,password} = req.body

        const canddate = await User.findOne({email})

        if(canddate)
        {
            return res.status(400).json({message: 'Такой
ПОЛЬЗОВАТЕЛЬ уже существует'})
        }

        const hashedPassword = await bcrypt.hash(password,12)
        const user = new User({email,password: hashedPassword})

        await user.save()

        res.status(201).json({message: 'Пользователь создан!'})

    }
    catch(e)
    {
        res.status(500).json({message: 'Что то пошло не так,
попробуйте снова!'})
    }
})

//api/auth/login

```

```

router.post(
  '/login',
  [
    check('email', 'Введите корректный
email').normalizeEmail().isEmail(),
    check('password', 'Введите пароль').exists()
  ],
  async(req, res)=>
  {
    try
    {
      const errors = validationResult(req);
      if(!errors.isEmpty())
      {
        return res.status(400).json({
          errors: errors.array(),
          message: 'Некорректные данные при входе в
систему'
        })
      }
      const {email,password} = req.body

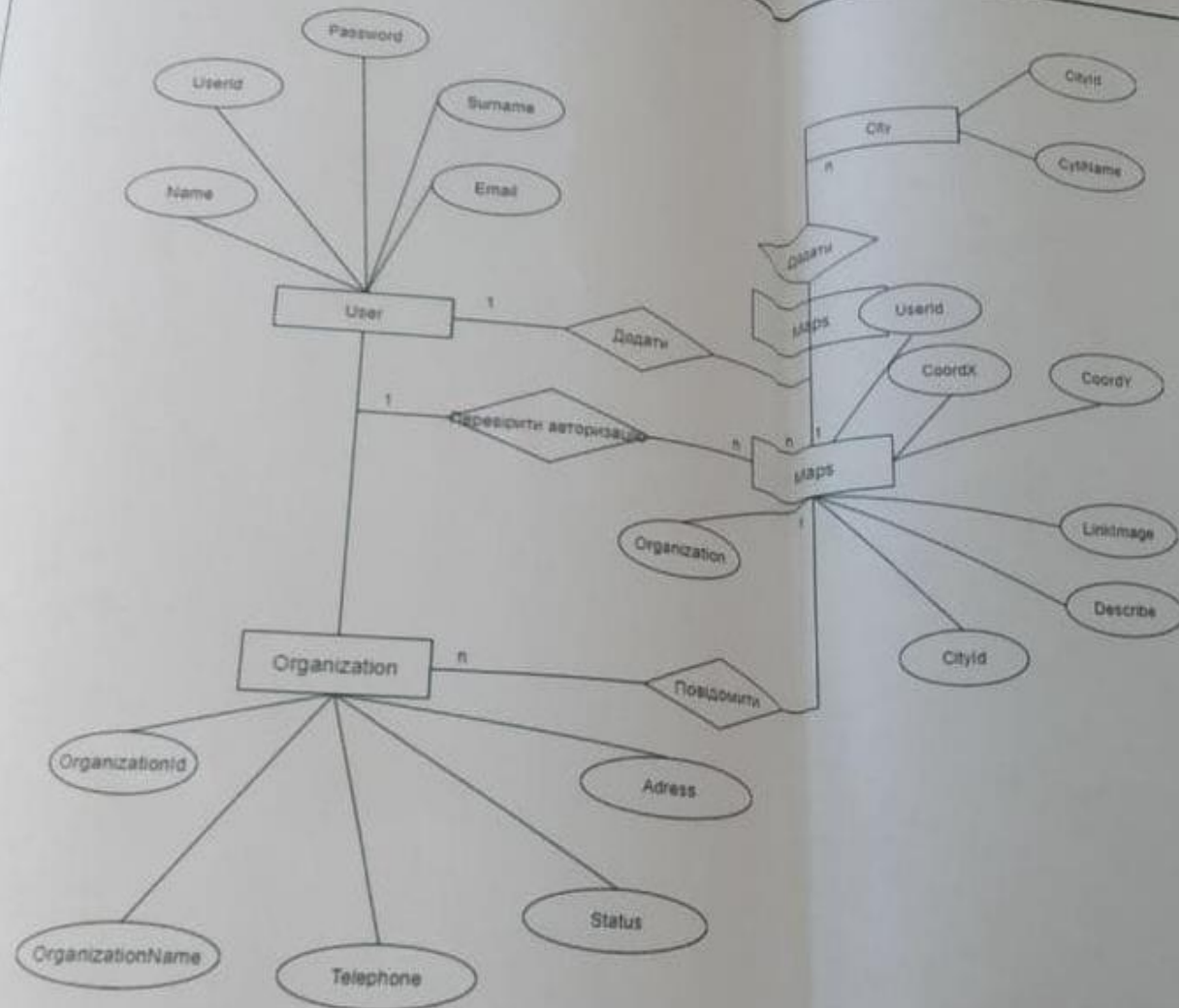
      const user = await User.findOne({email})

      if(!user)
      {
        return res.status(400).json({message: 'Пользователь
не найден'})
      }
      const isMatch = await
bcrypt.compare(password,user.password)
      if(!isMatch)
      {
        return res.status(400).json({message: 'Неверный
пароль попробуйте снова!'})
      }

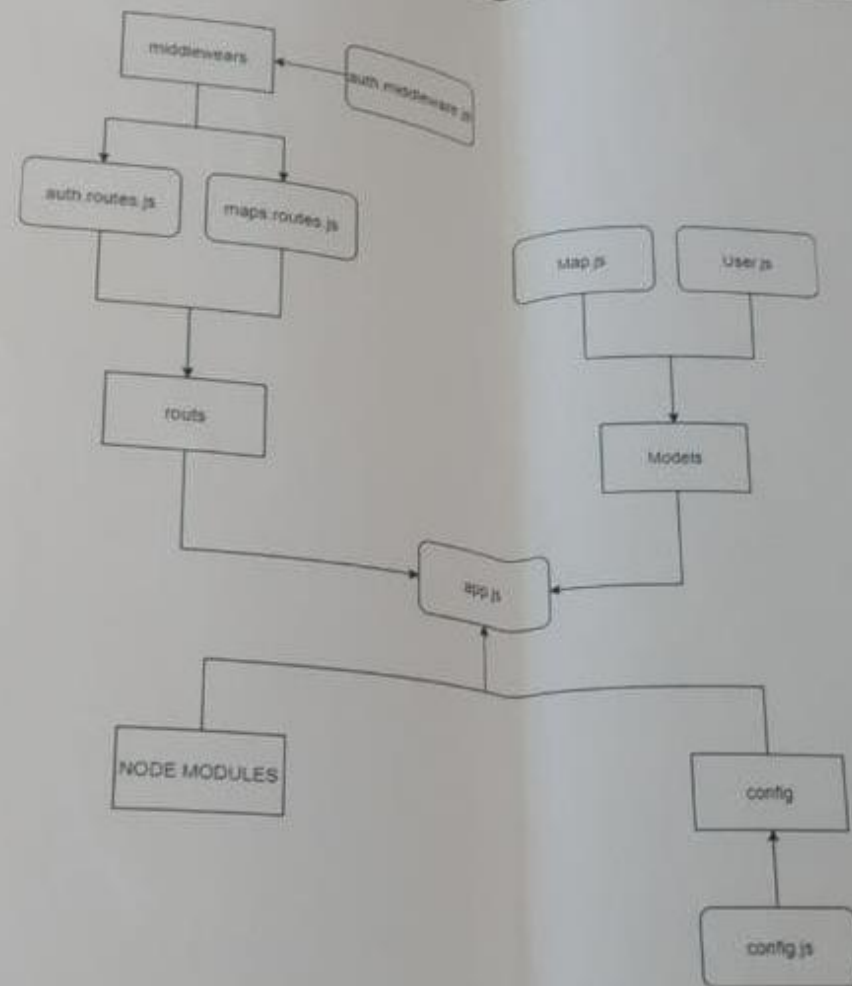
      const token = jwt.sign(
        { userId: user.id },
        config.get('jwtSecret'),
        {expiresIn:'1h'}
      )
      res.json({token, userId: user.id})

    }
    catch(e)
    {
      res.status(500).json({message:'Что то пошло не так,
попробуйте снова!'})
    }
  })
module.exports = router

```



					13.02070849.00044 ПЛ/1		
					Розробка інтерактивної підсистеми екологічного моніторингу		
					ER-діаграма сутностей предметної області		
Зм.	Арх.	№ докум.	Підпис	Дата	Листопад	Місяць	Місяць
Розробка	Щербань М.С.						
Перевірка	Делецько І.Я.						
Т.напр.							
Н.контр.	Щербань Н.В.						
Утверждає	Кудерник Р.Х.						
					ЗНТУ зр. ННТ-527		



					13.02070849.00044 ПЛ2		
					Розробка інтерактивної підсистеми екологічного моніторингу		
					Схема власної розробки модуля серверної частини		
Зм.	Апр.	На докум.	Підпис.	Дата	Листопад	Місто	Місяць/рік
Розробка	Шевчук М.С.						
Перевірка	Зеленюк І.Я.						
Т.контр.							
Н.контр.	Шевченко Н.В.						
Утвердження	Григоренко Р.К.						
					ЗНТУ зр. КНТ-527		

1. Атрибути сутності «КОРИСТУВАЧ»

№	Атрибут	Тип атрибуту
1	UserId	Type(ObjectId)
2	Name	Type(String)
3	Surname	Type(String)
4	Email	Type(String)
5	Password	Type(String)

3. Атрибути сутності «МІСТО»

№ Атрибуту	Атрибут	Тип атрибуту
1	CityId	Type(ObjectId)
2	CityName	Type(String)

5. Зв'язки між сутностями

Сутність №1	Сутність №2	Зв'язок між сутностями
КОРИСТУВАЧ	МАПА	Один-до-багатьох
МАПА	МІСТО	Один-до-багатьох
МАПА	ОРГАНІЗАЦІЯ	Один-до-багатьох

2. Атрибути сутності «МАПА»

№ Атрибуту	Атрибут	Тип атрибуту
1	CoordX	Type(Number)
2	CoordY	Type(Number)
3	LinkImage	Type(String)
4	Describe	Type(String)
5	UserId	Type(ObjectId)
6	CityId	Type(ObjectId)
7	OrganizationId	Type(ObjectId)
8	CoordId	Type(ObjectId)

4. Атрибути сутності «ОРГАНІЗАЦІЯ»

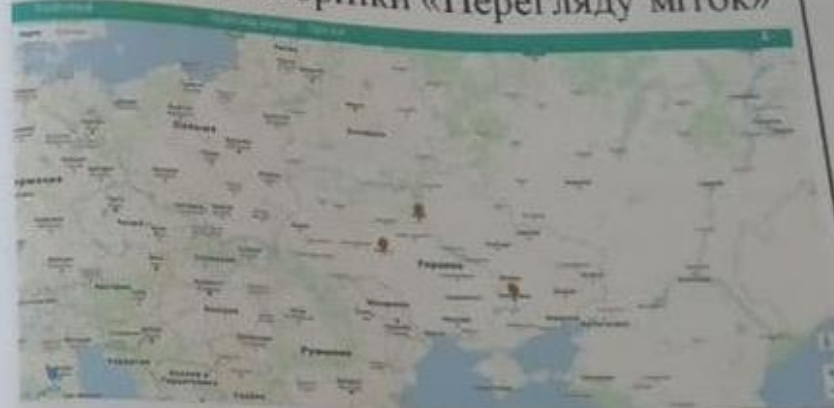
№ Атрибуту	Атрибут	Тип атрибуту
1	OrganizationId	Type(ObjectId)
2	OrganizationName	Type(String)
3	Address	Type(String)
4	Telephone	Type(String)
5	Status	Type(Boolean)

					13.02070849.00044 ПЛЗ		
Зам.	Куп.	№ докум.	Підпис	Дата	Розробка інтерактивної підсистеми екологічного моніторингу		
Розробка	Щербак М.С.				Атрибути сутності		
Перевірка	Володимир І.В.						
Т.контр.							
Н.контр.	Щербак Н.В.						
Утверджені	Буденко І.В.						
					ЗНТУ зр. КНТ-527		

1. Інтерфейс сторінки додавання мітки



2. Інтерфейс сторінки «Перегляду міток»



3. Інтерфейс перегляду конкретної мітки



4. Інтерфейс перегляду компонента «Додати фото»



					13.02070849.00044 ПЛ4		
					Розробка інтерактивної підсистеми екологічного моніторингу Інтерфейс перегляду		
Зм.	Апр.	Мі Діагн.	Лінійн.	Діагн.			
Розробка	Щербак М.С.						
Перевірка	Іванюк І.І.						
Технік							
М. конст.	Щербак М.С.						
М. конст.	Іванюк І.І.						
					ЗНТУ зр. КНТ-527		