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

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Наталія АУШЕВА

«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 122 «Комп'ютерні науки»

за освітньо-професійною програмою «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: « Інтерактивна система пошуку і реєстрації пунктів прийому та переробки вторинної сировини »

Виконала:
студентка IV курсу, групи ТМ-82
Мавроді Юлія Андріївна _____

Керівник:
Старший викладач
Дацюк Оксана Антонівна _____

Рецензент:

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ — 2022

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 «Комп’ютерні науки»

Освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
”___”_____2022р.

ЗАВДАННЯ

на дипломну роботу студенту
Мавроді Юлії Андріївні
(прізвище, ім’я, по батькові)

1. Тема роботи «Інтерактивна система пошуку і реєстрації пунктів прийому та переробки вторинної сировини»
керівник роботи Дацюк Оксана Антонівна, старший викладач
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”10” червня 2022р. №
2. Строк подання студентом роботи 10 червня 2021 р.

3. Вихідні дані до роботи мова програмування JavaScript, фреймворк Angular, оточення Node.js, бібліотеки Express, середовище розробки WebStorm. Розроблено з використанням онлайн-сервісу для розробки інтерфейсів та прототипування Figma, бібліотекою компонентів Angular Material та онлайн-редактору MapBox.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати проблематику сортування, збирання або переробки сміття, розглянути існуючі програмні рішення для моніторингу пунктів прийому вторинної сировини, розробити універсальний програмний продукт для пошуку і реєстрації пунктів прийому та переробки вторинної сировини..

5. Перелік ілюстративного матеріалу

Постановка задачі, Архітектура системи, Інтерфейс програм-аналогів, Діаграма прецедентів, концептуальна модель бази даних, скріншоти інтерфейсу користувача, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	завдання прийняв

7. Дата видачі завдання ”10” вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	25.05.2022	виконано
2.	Вивчення та аналіз задачі	21.02.2022-21.03.2022	виконано
3.	Розробка архітектури та загальної структури системи	21.03.2022-04.04.2022	виконано
4.	Розробка структур окремих підсистем	05.04.2022-12.04.2022	виконано
5.	Програмна реалізація системи	13.04.2022-25.05.2022	виконано
6.	Оформлення пояснювальної записки	26.05.2022-29.05.2022	виконано
7.	Захист програмного продукту	04.06.2022	виконано
8.	Передзахист	06.06.2022 – 09.06.2022	виконано
9.	Захист	20.06.2022 – 30.06.2022	виконано

Студент _____ Мавроді Ю.А.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Дацюк О.А.
(підпис)

АНОТАЦІЯ

Розроблена інтерактивна система здійснює реєстрацію, перегляд та пошук пунктів прийому та переробки вторинної сировини на інтерактивній мапі. В залежності від ролі користувача, система надає можливість створення нової, редагування та видалення існуючої мітки або фільтрації пунктів прийому за потрібними критеріями. Систему можна використати при реєстрації інших категорій, для пошуку небезпечних зон або пунктів дії Червоного Хреста.

Пояснювальна записка складається зі вступу, п’яти розділів, висновку, списку використаних джерел; містить 72 сторінки, 32 рисунки та 1 додаток. Список використаних джерел включає 8 бібліографічних найменувань та 19 електронних ресурсів.

Ключові слова: інтерактивна мапа, збір вторинної сировини, переробка вторинної сировини, веб-сервіс, фільтрація пунктів прийому.

ABSTRACT

The developed interactive system carries out registration, viewing and search of points of reception and processing of secondary raw materials on the interactive map. Depending on the role of the user, the system provides the ability to create a new label, edit and delete existing or filter reception points according to the desired criteria. The system can be used when registering other categories to search for dangerous areas or points of action of the Red Cross.

The explanatory note consists of an introduction, five chapters, a conclusion, and a list of sources used; contains 72 pages, 32 figures and 1 appendix. The list of used sources includes 8 bibliographic titles and 19 electronic resources.

Keywords: interactive map, collection of secondary raw materials, processing of secondary raw materials, web service, filtering of reception points.

ЗМІСТ

ВСТУП

Збір вторинної сировини є основою для покращення екологічної ситуації у будь-якому населеному пункті. Насамперед одним із важливих аспектів є ефективність та якість здійснення збору вторинної сировини та вивезення сміття у великих місцях. Щодня сотні та тисячі тон різних відходів вирушають на міські звалища з великих та малих українських міст. Життєдіяльність сучасного міста, наприклад такого як Київ, пов'язана зі споживанням величезної кількості товарної продукції, будівництвом, знесенням та реконструкцією будівель та споруд різного призначення, ремонтом та обслуговуванням кілометрів торгових та офісних площ, що є наслідком утворення тисяч тон відходів вторинної сировини, яка потребує переробки.

Для вирішення цієї проблеми було запроваджено інтерактивну систему пошуку і реєстрації пунктів прийому та переробки вторинної сировини. Програмний продукт дозволяє користувачу здійснити перегляд та пошук пунктів прийому на інтерактивній мапі. В залежності від ролі користувача, система має функціонал додавання нової, редагування та видалення існуючої мітки або фільтрації пунктів прийому за потрібними критеріями. Окрім багатого функціоналу є можливість перегляду рекомендацій щодо сортування вторинної сировини.

Задача розробки даного модулю є досить актуальною, оскільки в Україні ще дуже мало пунктів прийому відсортованого сміття або люди про них не знають, і розширений функціонал проекту допоможе у комфортному та швидкому пошуку потрібного пункту прийому або в створенні нового.

Також, особливо у період воєнного стану в Україні, дана інтерактивна система може бути використана для позначення небезпечних зон, де могли залишитися залишки або уламки вибухонебезпечних предметів чи будь-яке інше мілітарі-обладнання, яке потребує утилізації.

З іншого боку, насьогодні дуже важливим є питання проінформованості

населення щодо отримання будь-якого виду допомоги. Тож задача швидкого та доступного отримання місцезнаходження пунктів дії Червоного Хреста є наразі досить актуальною інформацією. Дану інтерактивну мапу можна використовувати саме для надання такого роду інформації.

1 ЗАДАЧА РОЗРОБКИ ІНТЕРАКТИВНОЇ СИСТЕМИ ПОШУКУ І РЕЄСТРАЦІЇ ПУНКТІВ ПРИЙОМУ ТА ПЕРЕРОБКИ ВТОРИННОЇ СИРОВИНИ

Метою даного проекту є розробка інтерактивної системи, що надасть користувачеві можливість здійснити реєстрацію, перегляд та пошук пунктів прийому та переробки вторинної сировини на інтерактивній мапі. Програмний продукт буде комфортним рішенням для створення нової, редагування та видалення існуючої мітки або фільтрації пунктів прийому за потрібними критеріями, в залежності від ролі користувача. Також розробка продукту буде корисна при реєстрації інших категорій відходів, які потребують обов’язкової утилізації або при швидкому пошуку пунктів дії Червоного Хреста.

Для досягнення поставленої мети необхідно виконати наступні задачі:

1. Провести аналіз існуючих рішень, присутніх на ринку.
2. Розробити інтерактивну систему на базі інтерактивної мапи, що передбачає:
    - можливість реєстрації пунктів прийому та переробки вторинної сировини, небезпечних зон або пунктів дії Червоного Хреста;
    - відображення усіх пунктів у вигляді міток на мапі;
    - можливість зручного перегляду детальної інформації потрібного пункту;
    - швидкий пошук потрібних пунктів;
3. Розробити можливість реєстрації користувача у системі:
    - вхід до особистого кабінету;
    - вихід з особистого кабінету;
    - перегляд детальної інформації користувача;
    - передбачити різний функціонал програми, в залежності від ролі

користувача у системі;

4. Розробити окремий функціонал для ролі адміністратора інтерактивної системи, а саме:
   - перегляд повного списку пунктів прийому та переробки вторинної сировини, небезпечних зон або пунктів дії Червоного Хреста;
   - пошук по списку за ключовими словами;
   - додавання нового пункту;
   - редагування існуючого пункту;
   - видалення пункту;
   - перегляд детальної інформації потрібного пункту;
5. Розробити функціонал для ролі користувача інтерактивної системи, що передбачає:
   - зручну та швидку розширену фільтрацію за різними критеріями;
   - перегляд допоміжної інформації з сортування сміття;

Зручність використання та зрозумілий користувацький інтерфейс також є одними з найважливіших вимог до розробки програмного продукту.

2 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

Як вже було зазначено вище, проблематика ефективності та якості здійснення збору вторинної сировини та вивезення сміття є одним із важливих аспектів для покращення екологічної ситуації у будь-якому населеному пункті. І відповідно в 21 сторіччі люди потребують допоміжного програмного продукту, який дозволить здійснити пошук, перегляд або реєстрацію пунктів прийому і переробки вторинної сировини, зробивши їх простими та зручними.

Для ефективної розробки такої інтерактивної системи, перед реалізацією програмного продукту було виконано дослідження існуючих аналогів в області інтегрованих середовищ розробки. Було проаналізовано які саме переваги слід впровадити та яких недоліків слід уникати при створенні системи. Для порівняння та аналізу схожих програмних продуктів було вирішено розглянути наступні інтерактивні мапи:

- Locator.ua;
- Сортуй;
- Київкомунсервіс.

Розглянемо та проаналізуємо кожну систему окремо.

2.1 Веб-застосунок Locator.ua

Веб-застосунок призначений для пошуку організацій та послуг різних сфер діяльності людини. Перш за все, при використанні даного ресурсу, користувач потрапляєте до головної сторінки, де може побачити мапу, меню з різними категоріям вибору, наприклад авто, бізнес, готелі здоров’я краса,

культура, магазини та інш, поле для вводу запиту, за яким може здійснюватися пошук, головну функціональну інформацію, список для вибору населеного пункту, кнопку «додати організацію» та вікно реєстрації/входу до особистого кабінету (рис 2.1).

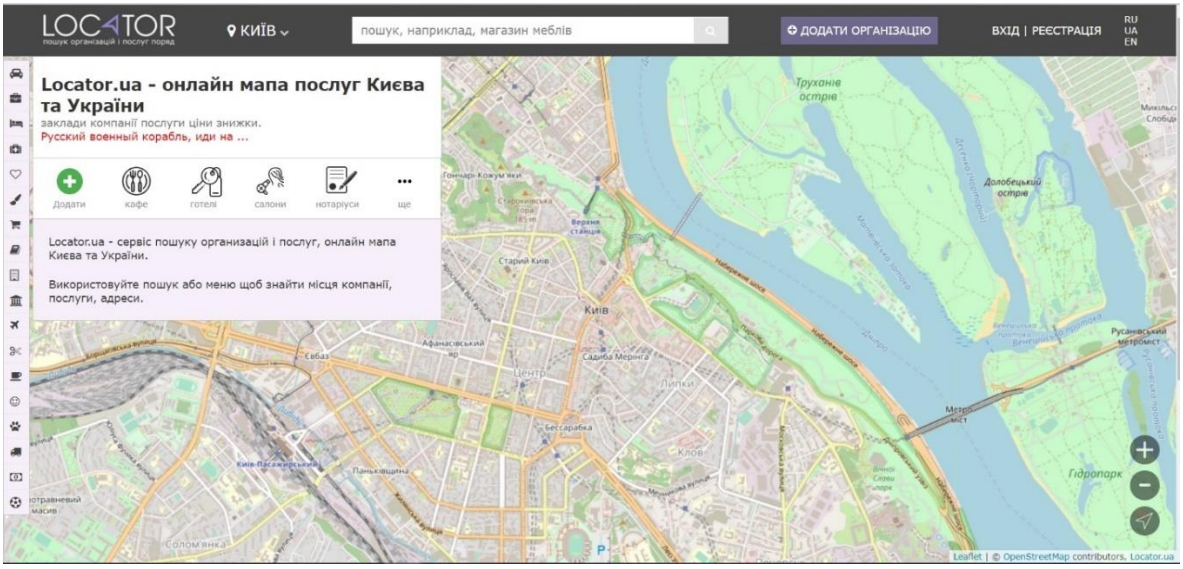

Рисунок 2.1 – Головна сторінка веб-застосунку Locator.ua

При пошуку необхідної категорії організацій та послуг, користувач має змогу переглянути у вигляді списку та вибрати потрібний йому пункт (рис 2.2).

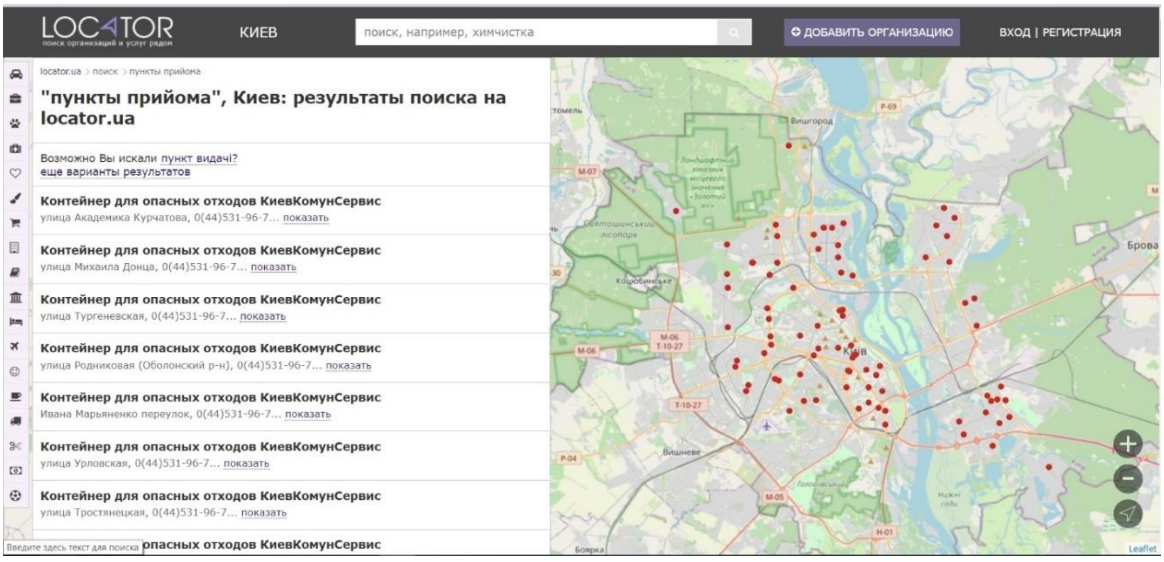

Рисунок 2.2 –Сторінка пошуку пунктів прийому вторинної сировини веб-застосунку Locator.ua

Тобто, враховуючи вищеперерахований функціонал сервісу, можна виокремити певні переваги та недоліки системи.

Серед переваг сервісу можна виділити:

- багатофункціональність застосунку, а саме можливість вибору в меню або пошук будь-якої категорії організацій чи послуг. Тобто користувач може використовувати один застосунок при різних потребах;
- великий вибір міст, де функціонує дана інформаційна система;
- можливість додати певну організацію чи послугу саме авторизованому користувачу. Це знижує ризик можливості додавання на сайт помилкової чи неперевіреної інформації;

З недоліків системи, враховуючи предметну область роботи, можна перерахувати:

- відсутність фільтрації за будь-якими типами вторинної сировини. Тобто користувач не має змоги вибрати необхідний йому тип сміття, що не дасть змоги швидко знайти пункт прийому чи переробки, де збирають саме потрібний тип вторинної сировини.
- відсутність інформації, що може бути необхідна користувачу, при пошуку потрібного йому пункту прийому та переробки вторинної сировини, при наведенні на маркер цього пункту на мапі. Тож користувач системи не в змозі, переглядаючи мапу з пунктами, вибрати потрібний йому за типом збираної сировини;
- при перегляді застосунку на пристроях, з різною шириною екрана, спостерігається погана адаптивність сайту;

2.2 Мобільний застосунок «Сортуй»

Мобільний застосунок «Сортуй» призначений для визначення типу вторинної сировини та надавання інформації з приводу того, як підготувати її до переробки. Розроблений для доступної та швидкої допомоги користувачу визначити куди відправити сміття — на сортувальну станцію чи на сміттєзвалище (рис 2.3).

Застосунок містить як універсальні правила сортування, так і адаптовані до умов популярних пунктів прийому. Має зручний функціонал пошуку потрібного типу вторинної сировини. Наразі застосунок підтримує специфічні умови сортування операторів із 10+ міст, серед яких Київ, Львів, Одеса, Харків, Миколаїв, Херсон, Чернігів та інші. Також «Сортуй» має актуальні та оновлені правила сортування і інформацію про події, пов’язані зі збиранням специфічних видів сміття. Визначити куди і коли найзручніше здавати вторинну сировину, підкаже інтерактивна вбудована мапа пунктів прийому.

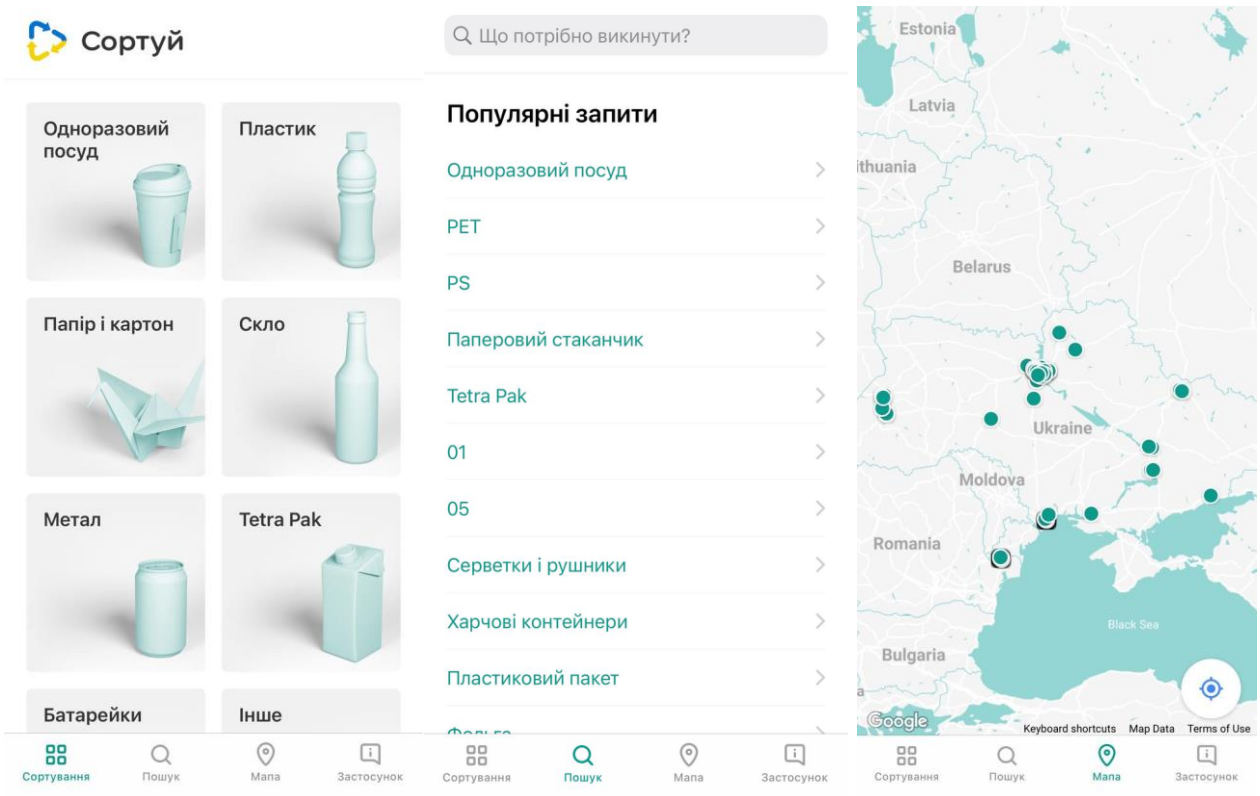

Рисунок 2.3 – Вкладки «Сортування», «пошук» та «Мапа» мобільного

застосунку «Сортуй»

Проаналізувавши функціонал мобільного застосунку, можна виділити переваги та недоліки системи.

Серед переваг «Сортуй» було визначено:

- детальний опис правил сортування, які дають загальне уявлення про типи вторинної сировини, які найчастіше приймаються на переробку в Україні. Ці універсальні правила дають розуміння принципів сортування відходів;
- продуманий і комфортний у використанні інтерфейс користувача, який дозволяє здійснити перегляд та пошук необхідної інформації за допомогою розробленого інтерактиву;

До недоліків мобільного застосунку можна віднести:

- відсутність фільтрації за типами вторинної сировини, як і у випадку з веб-застосунком, який було розглянуто вище;
- неможливість використання сервісу на комп’ютері, так як «Сортуй» є мобільним застосунком;

2.3 Інтерактивна мапа від сервісу «Київкомунсервіс»

Інтерактивна мапа є допоміжним сервісом для комунального підприємства виконавчого органу Київради (Київської міської державної адміністрації).

Метою діяльності підприємства є: забезпечення реалізації заходів щодо розвитку системи поводження з відходами у місті Києві, рішень та розпоряджень міської ради та її виконавчого органу з питань запобігання або зменшення обсягів утворення відходів, їх збирання, перевезення, зберігання, оброблення, видалення та утилізації та, розміщення та знешкодження [1].

До основних обов’язків КП «Київкомунсервіс» входить:

- розвиток системи поводження з відходами у місті Києві;
- надання послуг щодо своєчасного та повного вивезення побутових відходів від місць їх утворення до визначених місць утилізації;
- координація та контроль за діяльністю підприємств-перевізників ТПВ;
- впровадження роздільного збору побутових відходів в столиці .

Інтерактивна мапа наділена можливістю пошуку та перегляду місцерозташування контейнерів для різного типу відходів. Реалізований функціонал дозволяє легко відфільтрувати представлені на мапі контейнери за різними критеріями, а саме за адресою будинка, районом міста, місцезнаходженням та за типом контейнера (рис 2.4). Також є можливість перегляду допоміжної інформації з типами контейнерів, розташованих на мапі (рис 2.5). Для користувачів даного сервісу реалізована реєстрація та вхід до особистого кабінету. Тож зважаючи на розглянутий функціонал інтерактивної мапи, можна виділити певні переваги та недоліки системи.

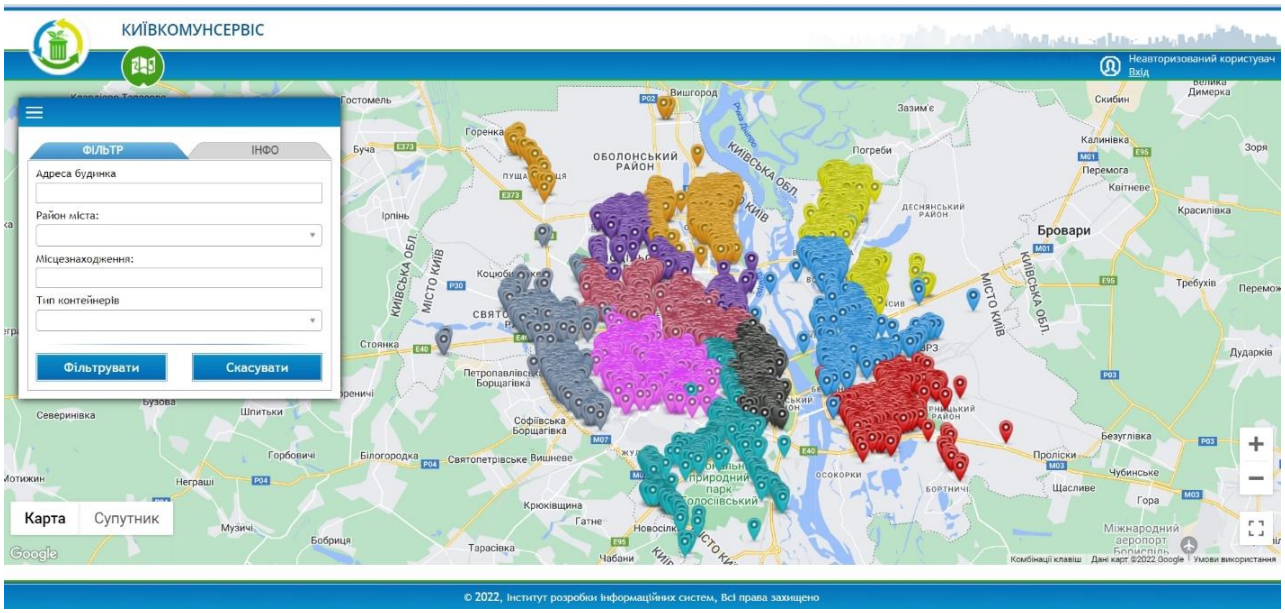

Рисунок 2.4 – Функціонал інтерактивної мапи «Київкомунсервіс»

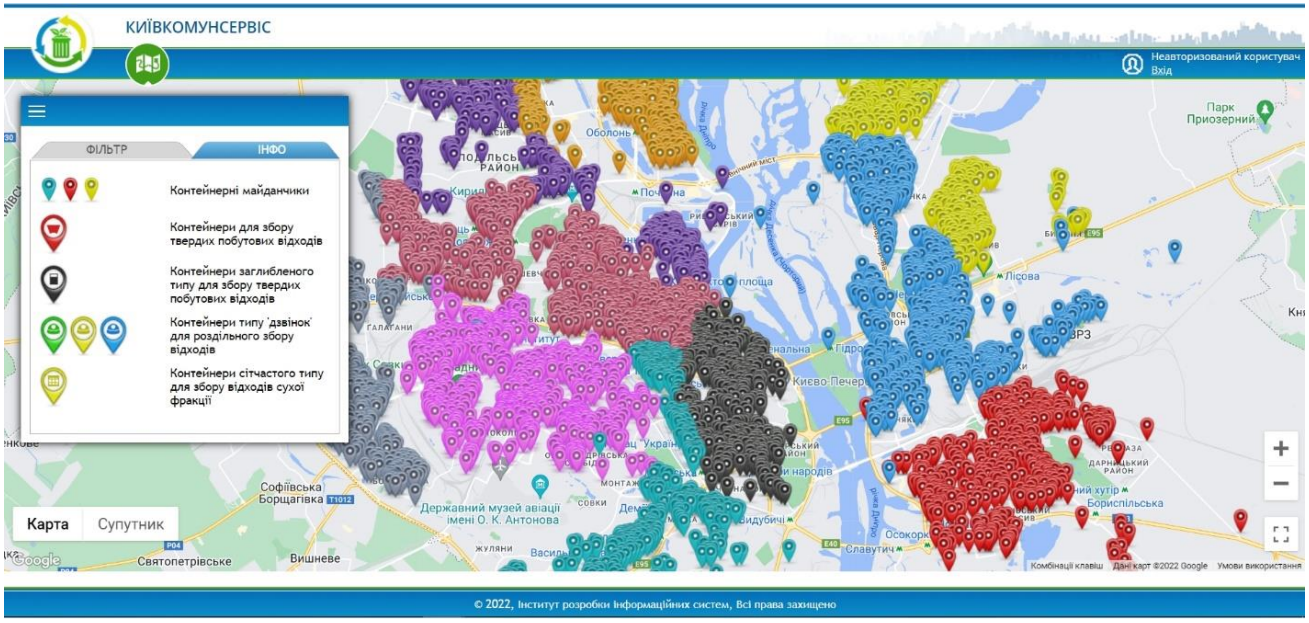

Рисунок 2.5 – Інформаційна довідка інтерактивної мапи «Київкомунсервіс»

До переваг даного сервісу можна віднести зручну фільтрацію за різними характеристиками контейнерів, що допомагає легко знайти потрібний на мапі;

Мапа більш придатна для використання саме юридичними особами, ніж приватними. На мапі розташовані місцезнаходження різного типу контейнерів, проте пункти прийому та переробки вторинної сировині на ній не відображаються. Згідно предметної області роботи, даний аспект можна вважати недоліком.

2.4 Висновки до розділу

В даному розділі було проведено аналіз предметної області роботи, розглянуто існуючі аналоги інтерактивної системи пошуку і реєстрації пунктів прийому та переробки вторинної сировини та сформовані основні недоліки даних сервісів. Серед яких є відсутність фільтрації за різними типами вторинної сировини; відсутність інформації, що може бути необхідна користувачу, при пошуку потрібного йому пункту прийому та переробки вторинної сировини,

при наведенні на маркер цього пункту на мапі; неможливість використання сервісу на комп’ютері або неадаптований дизайн системи для різної ширини екрану монітора.

Тож було описано функціонал системи, що розробляється, згідно аналізу переваг і недоліків існуючих аналогів, націлений на швидку, зручну та комфортну роботу з програмним продуктом. Основними критеріями системи є:

- можливість фільтрації за різними типами вторинної сировини;
- присутність усієї необхідної інформації пункту прийому та переробки вторинної сировини, при наведенні на маркер цього пункту на мапі;
- можливість додавання, редагування та видалення пунктів прийому зареєстрованим користувачам системи;
- опис правил сортування, які дають загальне уявлення про типи вторинної сировини, які найчастіше приймаються на переробку в Україні;
- можливість використовувати систему за допомогою комп’ютера та адаптований дизайн під різну ширину екрана девайса;
- продуманий і комфортний у використанні інтерфейс користувача, який дозволяє здійснити перегляд та пошук необхідної інформації за допомогою розробленого інтерактиву;

3 ЗАСОБИ РОЗРОБКИ ІНТЕРАКТИВНОЇ СИСТЕМИ ПОШУКУ І РЕЄСТРАЦІЇ ПУНКТІВ ПРИЙОМУ ТА ПЕРЕРОБКИ ВТОРИННОЇ СИРОВИНИ

Для реалізації інформаційної системи було обрано найбільш зручні та комфортні у використанні засоби розробки. Проаналізувавши існуючі програмні рішення, було обрано мову програмування Javascript в операційній системі Windows, що найбільш підходить для вирішення поставленої задачі.

Макет інтерактивної системи виконувався в онлайн-сервісі для розробки інтерфейсів та прототипування Figma.

При розробці клієнтського застосунку було обрано фреймворк Angular, з використанням мови TypeScript, бібліотеку компонентів Angular Material та інтерактивну мапу MapBox.

Для роботи з серверною частиною використовувалась мова програмування JavaScript, за допомогою оточення Node.js, та бібліотеки Express.

В якості середовища розробки було обрано редактор коду WebStorm.

3.1 Середовище розробки WebStorm

WebStorm — це інтегроване середовище для розробки на JavaScript та пов'язаних з ним технологіях від компанії JetBrains, що розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. Підтримує мови JavaScript, CoffeeScript, TypeScript та Dart [2].

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління

версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується коректний рефакторинг.

Основними можливостями є:

- інтеграція з системи керування версіями Subversion, Git, GitHub, Perforce, Mercurial, CVS підтримуються з коробки з можливістю побудови списку змін і відкладених змін;
- інтеграція з системами відстеження помилок; починаємо п.п. з малої літери, тому що попередній пункт закінчився «;»
- модифікація файлів .css, .html, .js з одночасним переглядом результатів (Live Edit, в деяких джерелах ця функціональність називається «редагування файлів на льоту» або «в реальному часі» або «без перезавантаження сторінки»);
- віддалене розгортання за протоколами FTP, SFTP, на монтованих мережевих дисках тощо з можливістю автоматичної синхронізації; в тексті є нерозривні пробіли
- можливості Zen Coding і Emmet; в тексті є нерозривні пробіли
- підтримка Angular, React, Vue.js, Node.js, Meteor, Ionic, Cordova, React Native, Electron та інш;

3.2 Мова програмування JavaScript

JavaScript (скорочено JS) — динамічна, об'єктно-орієнтована мова програмування . Реалізація стандарту ECMAScript.

JavaScript найчастіше використовується у браузері , що надає можливість виконання коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером , змінювати структуру та зовнішній вигляд веб-сторінки [3].

З появою Node.js мова програмування JavaScript також використовується для програмування на стороні сервера (подібно до таких мов програмування, як Java, PHP), розробки ігор , стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF -документів тощо.

JavaScript має C-подібний синтаксис, але в порівнянні з мовою Сі має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів функції як об'єкти першого класу обробка винятків
- автоматичне приведення типів
- автоматичне прибирання сміття
- анонімні функції

JavaScript містить декілька вбудованих об'єктів: Global , Object , Error, Function, Array, String, Boolean , Number , Math, Date , RegExp та інші. Містить набір вбудованих операцій, які, строго кажучи, не обов'язково є функціями або методами, а також набір вбудованих операторів, що управляють логікою виконання програм [4].

Наразі JavaScript, є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професіональних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-любителів. Поява AJAX змінила ситуацію та повернула увагу професійної спільноти до мови. В результаті, були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та налагодження), створені бібліотеки та

фреймворки , а з появою платформи Node.js поширилося використання JavaScript поза браузером [5].

3.3 Фреймворк Angular

Angular – це JavaScript фреймворк від компанії Google, який призначений для розробки односторінкових програм.

Легкий, швидкий та доступний front-end фреймворк Angular став популярний у величезної кількості розробників. Він дає можливість будувати інтерактивні та динамічні веб-програми, для яких потрібно набагато менше зусиль та коду.

Angular є структурною основою для динамічних веб-застосунків, яка дозволяє використовувати HTML як мову шаблону, а потім розширити синтаксис HTML для вираження компонентів програми. За допомогою прив'язки даних та запровадження залежності можна виключити більшу частину коду, який довелося б писати [6].

Особливості фреймворку Angular:

- мова програмування TypeScript, яка використовується в якості мови шаблону;
- хоча Typescript є основною мовою для Angular, програми можна також писати за допомогою таких мов як Dart або JavaScript;
- підтримка Google;
- націлений на розробку односторінкових додатків, тобто SPA-рішень (Single Page Application). Як приклад можна навести популярні програми для соціальних мереж (Twitter, Instagram та Facebook);
- надає клієнтську MVC-інфраструктуру, яка допомагає у запуску та

створенні динамічних додатків із сучасним рівнем якості;

- програми, написані на Angular, сумісні з різними браузерами. Angular автоматично обробляє JavaScript, що підходить для кожного браузера;
- чистий і точний дизайн інтерфейсу користувача;
- проста маршрутизація;
- структура Angular полегшує розширення синтаксису HTML і легко створює повторні компоненти за директивами;
- в цілому, Angular є фреймворкос для створення великомасштабних, високопродуктивних та простих у обслуговуванні веб-додатків [7].

Також варто відзначити, що Angular пропонує рендеринг на стороні сервера, який прискорює завантаження початкової сторінки і, отже, покращує SEO, спрощуючи сканування динамічних сторінок. Швидке відображення сторінок значно покращує сприйняття веб-застосунків для наступного покоління, написаних в рамках Angular. Також, за допомогою даного фреймворку можна:

1. Легко тестувати код
2. Легко створювати персоналізовані об'єктні моделі документа (Document Object Model, DOM).
3. Моделювати дані обмежено для використання невеликих моделей даних, що робить код простим та легким для тестування.

3.4 Онлайн-редактор Figma

Figma (Фігма) – це графічний онлайн-редактор для спільної роботи. У ньому можна створювати інтерактивні прототипи сайтів та мобільних додатків, інтерфейс або елементи інтерфейсу програми, векторні ілюстрації. У Figma всі

документи зберігаються у хмарі. Завдяки цьому в редакторі можна колективно працювати над макетами та відкривати їх за посиланням, без завантаження [8].

3.5 Онлайн-сервіс MapBox

MapBox – це онлайн-сервіс, призначений для створення та публікації своїх власних карт. Лише десять років тому картографічні програми були доступні лише професіоналам і коштували чималих грошей. Проте ситуацію змінив Google, запустивши свій сервіс Google Maps, який абсолютно вільно надає всім охочим докладні карти і супутникові знімки. Сьогодні Google Maps має десятки послідовників і конкурентів, які в деяких аспектах його перевершують. Наприклад, сервіс MapBox, який може допомогти створити свою власну електронну карту або схему та опублікувати зручним для вас способом [9].

Сервіс MapBox є функціональним і водночас простим інструмент для розміщення в Інтернеті різної картографічної інформації, як індивідуальним користувачам, так і громадським організаціям.

3.6 Платформа NodeJs

Node.js (Node) – це платформа з відкритим вихідним кодом для роботи з мовою JavaScript, побудована на движку Chrome V8. Вона дозволяє писати серверний код для веб-застосунків і динамічних веб-сторінок, а також програм командного рядка. В основі платформи - подієво-керована модель з неблокуючими операціями введення-виводу, що робить її ефективною та легкою [10].

Платформу використовують fronted-розробники, backend-розробники та інші. Вона дозволяє написати програму для різних ОС: Linux, OS X та Windows, може використовуватись для створення API. Також Node.js застосовується для розробки крос-платформних програм: наприклад, списку завдань, який повинен працювати на різних платформах, синхронізувати дані в реальному часі та відправляти на мобільний пристрій. Node.js використовується при створенні сервісів з постійним обміном інформацією з користувачем: соціальних мереж, онлайн-ігор, чатів, систем спільної роботи над проектом, онлайн-редакторів тексту та ін [11].

Node.js є основою Internet of Things, або просто IoT. Платформа допомагає керувати приладами та створювати сервери, здатні одночасно обробляти велику кількість запитів.

3.7 Фреймворк express.js

Express – це фреймворк для Node.js, який реалізує шар функцій, необхідних створення ефективних додатків і API. Його використання значно скорочує написання коду, отже, зменшується час, що витрачається на розробку [12].

Node.js Express встановлюється через пакетний менеджер NPM.

Основні можливості Express:

- надає зручний та короткий метод визначення маршруту;
- спрощена обробка для отримання параметрів HTTP-запиту;
- високий рівень підтримки механізму шаблонів, проста візуалізація динамічних HTML-сторінок;
- надає механізм проміжного програмного забезпечення для ефективного керування HTTP-запитами;

- наявність великої кількості стороннього проміжного програмного забезпечення для розширення функції [13];

3.8 Мова гіпертекстової розмітки HTML5 та каскадні таблиці стилів CSS3

Як відомо, найпростіший код для веб-сайтів генерується мовою гіпертекстової розмітки HTML (HyperText Markup Language). Цей інструмент дозволяє структурувати різні блоки на сторінці та розміщувати тексти, малюнки, таблиці, списки і т. д. HTML не використовує логічні функції, зате дозволяє формувати та викладати на сторінках інформацію. Саме з HTML працюють браузери, які підтримують стандартні команди (теги та атрибути) цієї мови, і виводять їх у певному вигляді.

HTML містить сотні так званих тегів чи дескрипторів. Кожен тег використовує певні правила та має властивості, якими визначається його відображення на сторінці.

На практиці, при створенні веб-ресурсів за допомогою HTML розробник знайомий з елементами мови гіпертекстової розмітки і прописує всі необхідні блоки [14] .

HTML – це база, де побудований весь Інтернет. Інші інструменти є надбудовами і дозволяють додавати більш складний функціонал.

HTML5 – сучасна версія мови гіпертекстової розмітки, яка отримала низку доповнень та нових можливостей. До складу робочої групи її розробників входять представники таких гігантів IT-сфери, як Google, Apple, IBM, Microsoft та інші.

Багато експертів вважають HTML5 не продовженням розвитку HTML мови, а новою відкритою програмною платформою, яка застосовується для

створення веб-інтерфейсів, що використовують текстові та мультимедійні інструменти (відеоролики, графіка, аудіозаписи та інші елементи). При цьому в п'ятій версії зберігається зворотна сумісність і зручність читання коду для користувачів.

Але хорошим доповненням для HTML стали CSS (Cascading Style Sheets) – каскадні таблиці стилів, що містять опис особливостей зовнішнього вигляду веб-документа, створеного на основі інструментів гіпертекстової розмітки. В основному CSS використовується для візуального оформлення веб-сторінок. Тобто HTML задає структуру ресурсу, а таблиці стилів визначають те, як він виглядатиме [15].

Саме CSS відповідають за привабливість та дизайн сторінок. Ця технологія дозволяє "навести красу". Вже судячи з назви вона працює зі стилями (квітами, шрифтами, формами тощо)

CSS може застосовуватися не тільки в мовах розмітки HTML та XHTML, але й працювати у форматі XML (враховуючи документи SVG та HUL).

CSS3 – третя версія таблиці каскадних стилів, значно розширила можливості попередніх поколінь. Особливість CSS3 у тому, що з її допомогою можна створювати анімовані елементи самостійно JS. Вона включає підтримку різних градієнтів та тіней, використовує нові форми згладжування тощо [16].

Використання набору технологій HTML5+CSS3 відкриває розробникам величезний арсенал програмних засобів, з якими значно легше стало створювати зручні, стильні, сучасні та функціональні веб-ресурси.

3.9 Система керування реляційними базами даних MySQL

MYSQL – це система керування реляційними базами з відкритим вихідним кодом (СКРБД). Використовується модель клієнт-сервер. Є дуже популярною у використанні, бо має такі ознаки:

1. Простота. Існує можливість зміни вихідного коду.
2. Продуктивність. Система підтримує багато моделей кластерних серверів. Пропонується оптимальна швидкість, як обробки великої аналітики, так ведення електронної комерції.
3. Стандартизація. Має підтримку різних галузей, тобто великі ресурси для розробників. Підтримується промисловий стандарт. Користувачі швидко отримують замовлене програмне забезпечення.
4. Безпека. Система доступу до облікових записів дозволяє гарантувати збереження баз даних та високий клас їх безпеки. Доступне шифрування пароля та перевірка на основі хоста.

Базова структура системи клієнт-сервер дуже проста. Клієнт підключається до сервера через мережу. Він запитує з інтерфейсу користувача (GUI). Якщо сервер зрозумілий інструкції, він видає інформацію.

У середовищі MySQL відбуваються такі процеси:

- створюється база даних MySQL, визначається відношення таблиці;
- клієнти роблять запити, використовуючи команди SQL;
- сервер відповідає запитаною інформацією, яка надсилається клієнту.

Популярними інтерфейсами є SequelPro, DBVisualizer, MySQL WorkBench та Navicat DB Admin Tool. Серед них є безкоштовні та ті, якими можна користуватися на комерційних умовах. Багато інтерфейсів сумісні з популярними операційними системами. Деякі працюють лише з macOS [17].

3.10 Інструмент проектування MySQL Workbench

MySQL Workbench - це кросплатформовий інструмент проектування

реляційних баз даних з відкритим вихідним кодом, який додає функціональність та спрощує розробку MySQL та SQL. Він об'єднує проектування, розробку, створення, адміністрування та обслуговування SQL, а також пропонує графічний інтерфейс для структурованої роботи з базами даних [18].

MySQL Workbench надає можливості для управління моделями баз даних, такими як:

- створення графічної моделі;
- зворотній інжиніринг живих баз даних у моделі (моделювання даних);
- пряма інженерна модель у скрипт/живу базу даних і більше;

Існує декілька систем керування реляційними базами даних, таких як Microsoft SQL Server, Microsoft Access, Oracle та DB2. MySQL Workbench пропонує деякі переваги, які слід враховувати під час вибору інструменту.

MySQL підтримує кілька механізмів зберігання, кожен зі своїми специфікаціями, на відміну інших інструментів. Він також пропонує високу продуктивність завдяки своєму дизайну та простоті.

Він також відомий своєю рентабельністю. Версія спільноти безкоштовна для користувачів, а корпоративна версія має низьку ліцензійну плату.

3.11 Висновки до розділу

У розділі було розглянуто засоби розробки модуля інтерактивної системи пошуку і реєстрації пунктів прийому та переробки вторинної сировини. Для створення якісного програмного продукту проаналізовано доступні засоби розробки та обрано ті, які дозволять виконати роботу зручно та ефективно. Для вирішення поставленої задачі було обрано вищеперераховані інструменти, а саме

онлайн-редактор Figma для створення макету системи, фреймворк Angular для створення та роботи з клієнтською частиною, онлайн-сервіс MapBox для імплементування інтерактивної мапи. Для роботи з серверною частиною використовувалась мова програмування JavaScript, за допомогою оточення Node.js, та бібліотеки Express. MySQL для керування базою даних. Всі засоби добре взаємодіють один з одним і підтримуються в середовищі WebStorm.

4 ОПИС РЕАЛІЗАЦІЇ ІНТЕРАКТИВНОЇ СИСТЕМИ ПОШУКУ І РЕЄСТРАЦІЇ ПУНКТІВ ПРИЙОМУ ТА ПЕРЕРОБКИ ВТОРИННОЇ СИРОВИНИ

Розроблена інтерактивна система пошуку і реєстрації пунктів прийому та переробки вторинної сировини має архітектуру клієнт–сервер, яка створюється сервером, веб–клієнтом та базою даних (рис 4.1).

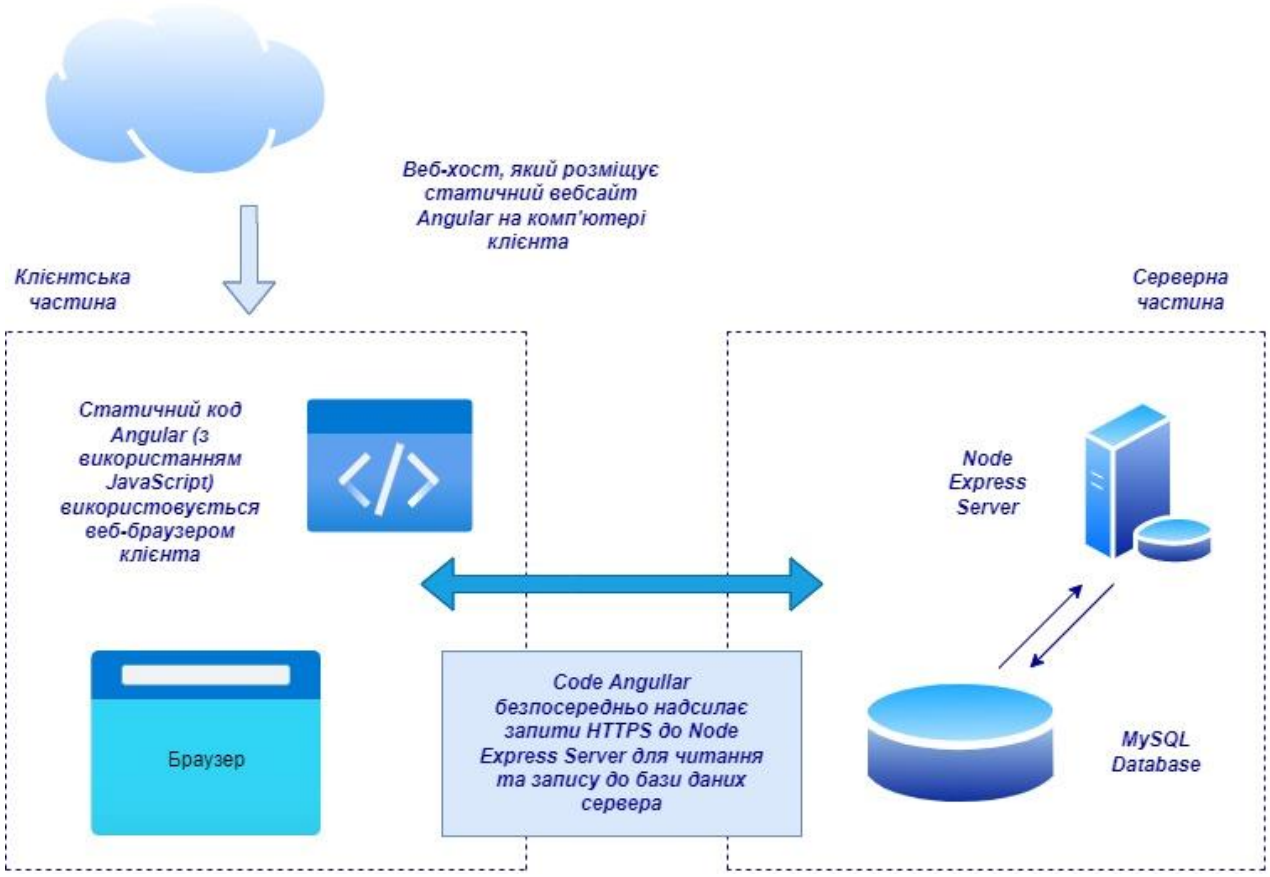

Рисунок 4.1 – Системна архітектура клієнт-сервер

Тож система матиме три компоненти:

1. Інтерфейс користувача, орієнтований на клієнта, що відображається через веб-браузер: для цього буде використовуватися Angular для створення статичних веб-сторінок.
2. База даних на стороні сервера для зберігання всієї інформації про користувача: для цього буде використовуватися MySQL.

3. Сервер Node Express для взаємодії з нашою базою даних та обслуговування та отримання інформації на/з нашого клієнтського веб-сайту.

4.1 Опис архітектури системи

Для покращеної розробки системи, після аналізу існуючих програмних рішень поставленої задачі та перед розробкою програмного забезпечення, було виконано макет інформаційної системі у онлайн-редакторі Figma. Розроблені елементи інтерфейсу та інтерактивний прототип сайту було взято за основу при створені системи (рис. 4.2).

Архітектура створеної інформаційної системи націлена на зручне використання будь-якому користувачеві. Є простою та зрозумілою людям різного віку та досвіду користування сайтами. Тобто система спроектована для зрозумілого та комфортного використання як для молодого покоління, так і для людей похилого віку.

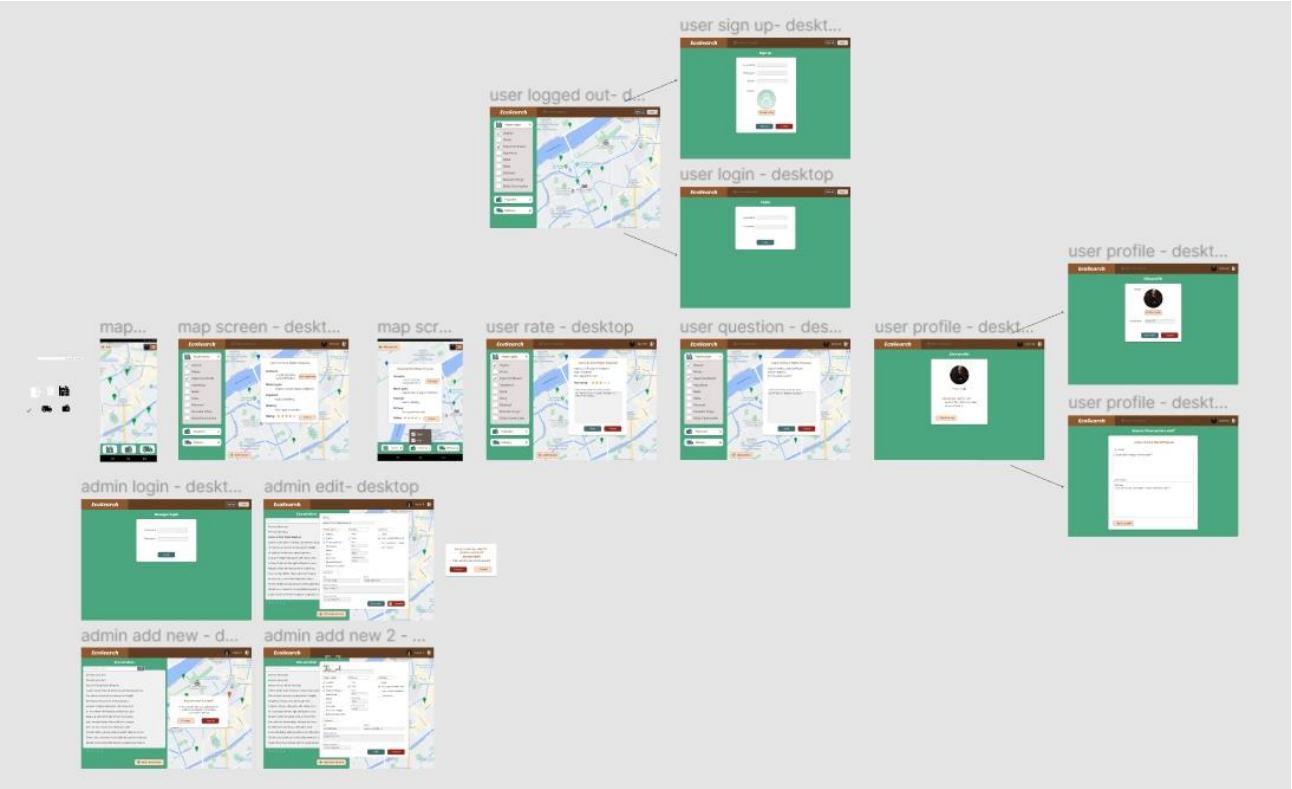

Рисунок 4.2 – Макет інформаційної системи

За даним макетом можна визначити основні сторінки сайту або принцип навігації. Також наглядним є розподілення ролей у системі та різний функціонал, який буде реалізовано відповідно до кожного користувача.

4.2 Діаграма прецедентів

Для наглядного відображення відношень між дійовими особами в системі використовується діаграма прецедентів (рис 4.3).

Цей граф показує відношення між акторами, та прецеденти з якими вони взаємодіють. Видно, що серед основних осіб, які можуть взаємодіяти з інтерактивною системою є звичайний користувач та адміністратор [19].

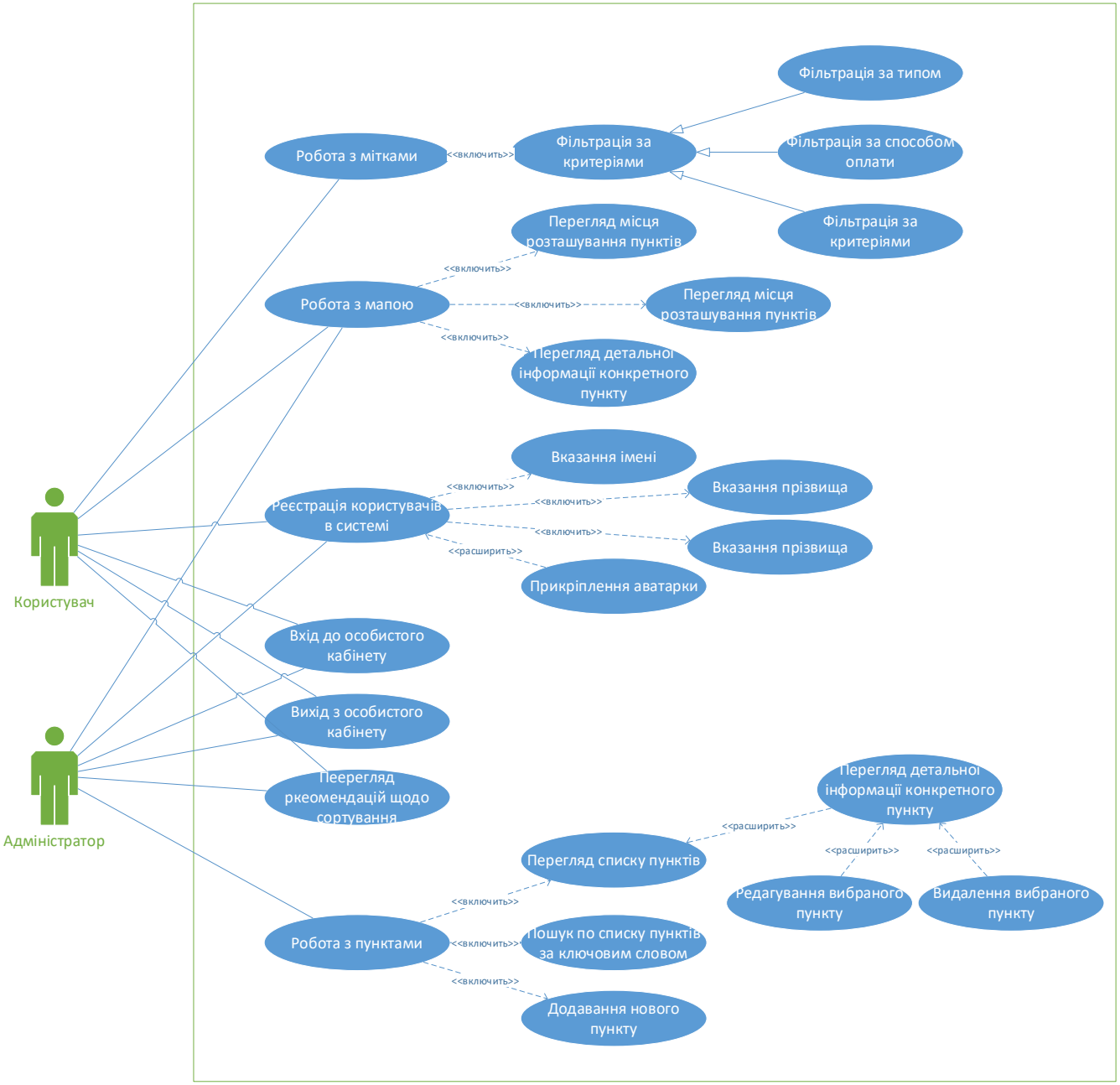

Рисунок 4.3 – Діаграма прецедентів

Відповідно до діаграми будь-який користувач інформаційної системи має змогу пошуку та перегляду пунктів прийому та переробки вторинної сировини, небезпечних зон або пунктів дії Червоного Хреста, має можливість зручного перегляду детальної інформації потрібного пункту та допоміжної інформації з сортування сміття. Також обидва типи користувачів можуть зареєструватися у системі, з подальшою можливістю входу та виходу з особистого кабінету.

Звичайний користувач має змогу зручно та швидко профільтрувати

пункти прийому та переробки вторинної сировини, небезпечних зон або пунктів дії Червоного Хреста за різними розширеними критеріями.

Користувач-адміністратор має особливість у перегляді повного списку пунктів, пошуку за цим списком, перегляду детальної інформації потрібного пункту, додаванням нового пункту, редагуванням або видаленням існуючого.

4.3 Реалізація бази даних

Розроблена інформаційна система використовує реляційну базу даних. Ці дані організовані у вигляді набору таблиць, які складуються зі стовпців та рядків. У таблицях зберігається інформація про об'єкти, які представлені у базі даних. У кожному стовпці таблиці зберігається певний тип даних, у кожній комірці – значення атрибута. Кожний рядок таблиці є набором пов'язаних значень, які відносяться до одного об'єкту чи сутності. Кожен рядок у таблиці може бути позначений унікальним ідентифікатором, який називається первинним ключем, а рядки з декількох таблиць можуть бути пов'язані за допомогою зовнішніх ключів.

Одним з головних етапів проектування бази даних є розробка її концептуальної моделі, за допомогою якої можна відобразити предметну область системи, охарактеризувати її основні сутності та наглядно продемонструвати взаємозв’язок між ними. Нижче наведена концептуальна модель бази даних, що використовувалася для збереження даних розроблюваної інформаційної системи (рисунок 4.4) [20].

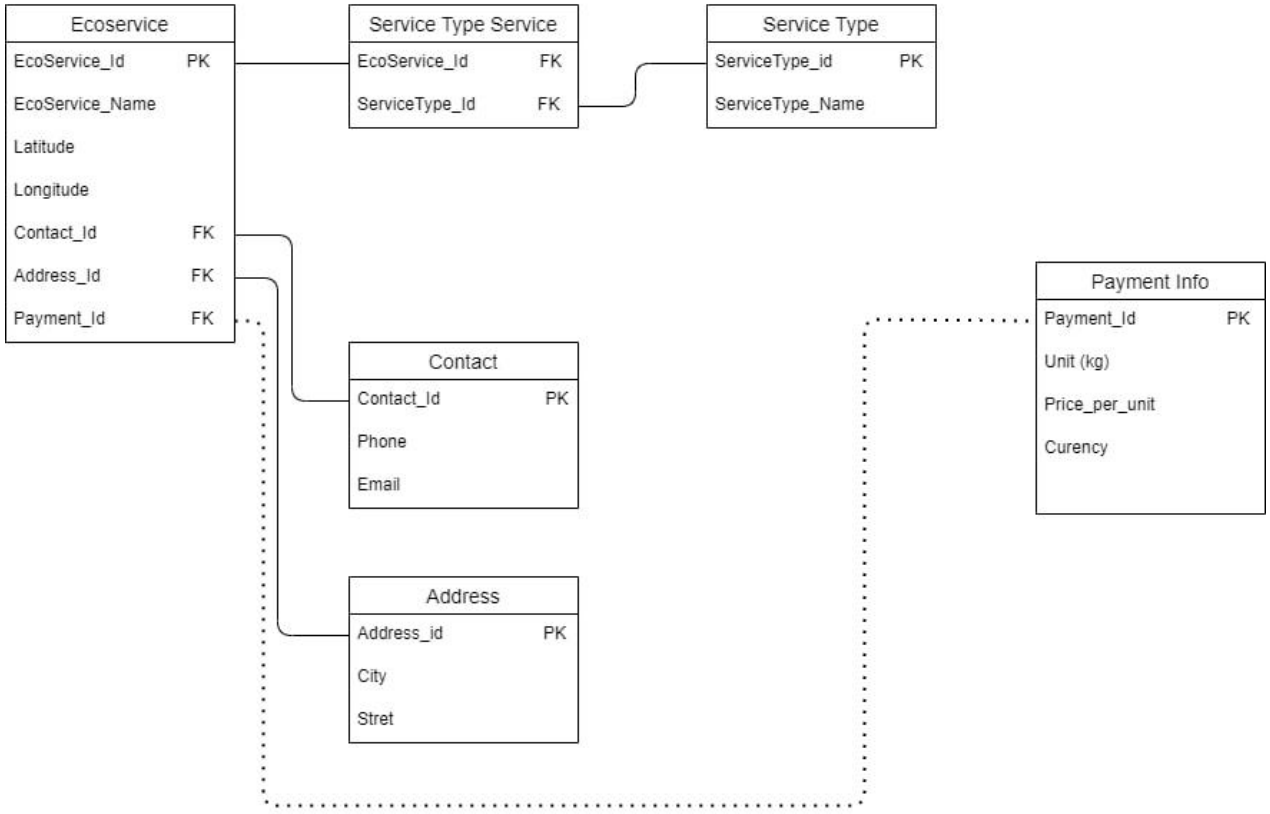

Рисунок 4.4 – Концептуальна модель БД

База даних складається з 6 таблиць:

- **Ecoservice**: зберігає у собі інформацію про пункти прийому та переробки вторинної сировини: ідентифікатор, назва пункту прийому та переробки, координата широти, координата довготи, id контакту, id адреси та id способу оплати;
- **Service Type**: зберігаються типи пунктів прийому та переробки вторинної сировини, а саме ідентифікатор та назва;
- **Service Type Service**: таблиця, яка має зв’язок багато до багатьох, використовується для зв’язування таблиць Ecoservice та Service Type.
- **Contact**: зберігаються контактна інформація пункту, має такі поля: ідентифікатор, номер телефону та email;
- **Address**: таблиця, що зберігає повну адресу пункту: ідентифікатор,

місто та вулиця.

– **Payment info**: таблиця, яка містить інформацію про спосіб оплати та має такі поля: ідентифікатор, одиниця (кг), ціна за одиницю та валюта.

4.4 Серверна частина

Реалізація серверної частини була виконана за допомогою середовища виконання JavaScript - Node.js та для більш зручної побудови системи - Express.

Сервер має наступний функціонал:

- обробляє запити від клієнтської частини веб-сайту;
- комунікація з БД;
- динамічно завантажує зображення для сайту.

Файлова структура серверної частини організована так, щоб було логічне відокремлення однієї сутності від іншої та має наступний вигляд (рисунок 4.5).

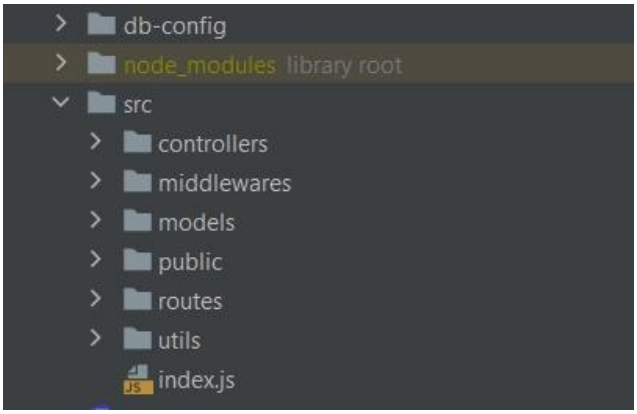

Рисунок 4.5 – Структура проекту серверної частини

- **db-config** зберігає файли налаштування доступу до бази даних системи;
- **node_modules** зберігає модулі проекту, враховуючи усі бібліотеки та файли встановлені на серверній частині, для забезпечення коректної роботи системи;
- **controllers** зберігає класи, методи якого є обробниками маршрутів. При

цьому контролер створюється для кожної взаємопов'язаної сукупності маршрутів, наприклад, для всіх маршрутів, які відповідають виконання дій над пунктами;

- **middlewares** містить в собі функції, що мають доступ до об'єкта запиту, об'єкта відповіді і наступної функції проміжної обробки в циклі “запит-відповідь” програми;
- **models** зберігає моделі, які описують структуру даних, що зберігаються в БД. Важливо зауважити, через моделі переважно йде взаємодія з БД;
- **public** – це папка, яка містить всі файли, до яких користувач має доступ як image, video,.;
- **routes** містить описи всіх маршрутів, причому будь-яка взаємопов'язана сукупність маршрутів виноситься в окремий файл;
- **utils** містить в собі файли, які використовуються в різних частинах створеної системи;
- **index.js** зберігає лише основні функції проміжної обробки та функцію запуску Node.js програми.

4.5 Клієнтська частина

Розробка клієнтського застосунку була реалізована за допомогою фреймворку Angular, з використанням мови TypeScript, бібліотеки компонентів Angular Material та інтерактивної мапи MapBox. Клієнтська частина призначена для комунікації з сервером, тобто для відправки та отримання даних. Загальна файлова структура клієнтської частини має наступний вигляді (рисунок 4.6).

Рисунок 4.6 – Розміщення файлів на клієнті

Умовно, файлову структуру можна поділити на декілька категорій.

4.5.1 Файли конфігурації робочого простору

Усі проекти в робочій області мають загальний контекст конфігурації CLI (набір параметрів, які настроюють інтерфейс командного рядка Angular). Верхній рівень робочої області містить файли конфігурації всього робочого простору, файли конфігурації для кореневого рівня та підпапки для вихідного коду програми та файлів тестування.

До таких файлів відносяться:

- **node_modules** надає пакети npm для всього робочого простору. Залежності всієї папки node_modules видно всім проектам;
- **src** є вихідними файлами для проекту кореневої програми;

- **.editorconfig** є налаштуванням для редакторів коду;
- **angular.json** містить параметри конфігурації CLI за промовчанням для всіх проектів у робочій області, враховуючи параметри конфігурації для інструментів зборки, обслуговування та тестування, які використовує CLI;
- **package-lock.json** надає інформацію про версію для всіх пакетів, встановлених у node_modules клієнтом npm;
- **tsconfig.json є** базовою конфігурацією TypeScript для проектів у робочій області. Всі інші конфігураційні файли успадковуються від цього базового файлу;

4.5.2 Вихідні файли програми

Файли на верхньому рівні src/ підтримують тестування та запуск програми. Підпапки містять джерело програми та конфігурацію конкретної програми.

- **app** містить файли компонентів, в яких визначено логіку та дані програми;
- **assets** містить файли образів та інших активів, які потрібно скопіювати під час збирання програми;
- **environments** містить параметри конфігурації збору для конкретних цільових середовищ;
- **favicon.ico** іконка, яку можна використовувати для програми на панелі закладок;
- **index.html** основна HTML-сторінка, яка відображається, коли хтось відвідує web-сторінку;
- **main.ts** є основною точкою входу для програми. Компілює програму

за допомогою JIT-компілятора та завантажує кореневий модуль програми (AppModule) для запуску у браузері.;

- **polyfills.ts** надає полізаповнені скрипти для підтримки браузера;
- **styles.sass** списки CSS-файлів, що поставляють стилі для проекту. Розширення відображає стиль препроцесора, який був налаштуваний для проекту;
- **test.ts** головна відправна точка для модульних випробувань, з деякою специфічною кутовою конфігурацією;

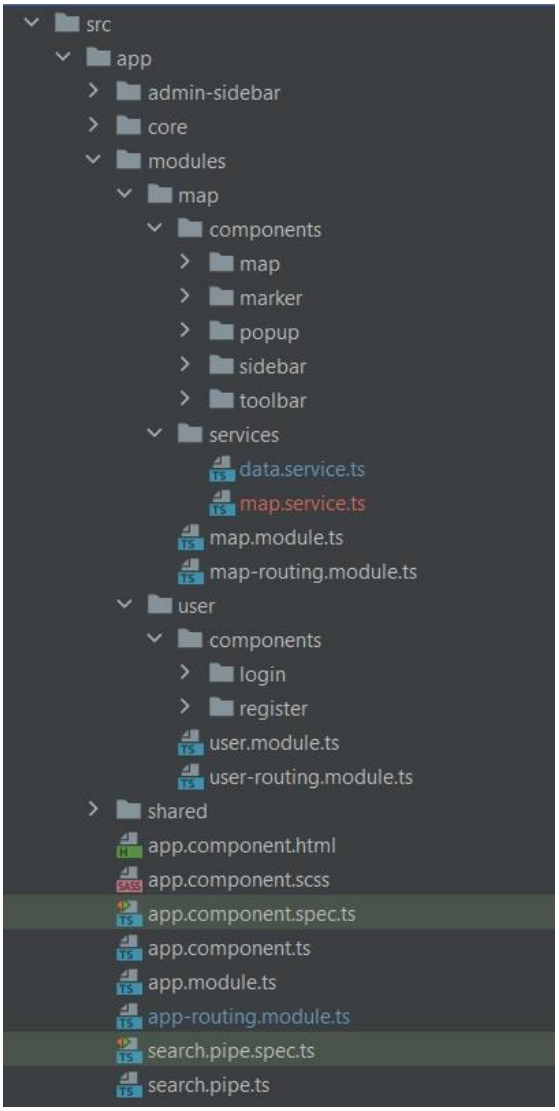

Рисунок 4.7 – Розміщення файлів на клієнті

У папці src/ папка app/ містить логіку та дані проекту. Тут знаходяться компоненти, шаблони та стилі (рис. 4.7), а саме:

- **app.component.ts** визначає логіку кореневого компонента програми AppComponent. Подання, пов'язане з цим кореневим компонентом, стає коренем ієрархії уявлень у міру додавання компонентів та служб у програму;
- **app.component.html** визначає шаблон HTML, пов'язаний із кореневим AppComponent;
- **app.component.css** визначає базову таблицю стилів CSS для кореневого AppComponent;
- **app.component.spec.ts** **в**изначає юніт-тест для кореневого AppComponent;
- **app.module.ts** визначає кореневий модуль AppModule, який повідомляє Angular, як збирати програму;
- **core** містить:
  1) компоненти хедера, аватара користувача, іконок та лінку на хедері для перегляду допоміжної інформації з сортування сміття;
  2) інтерфейси користувача системи (рис. 4.8) та пункту прийому та переробки вторинної сировини (рис. 4.9);
  3) сервіси для авторизації користувача та отримання даних з серверу.

```
export interface UserInterface {
  username: string;
  password: string;
  firstName?: string;
  lastName?: string;
}

export interface AuthResponseInterface {
  access_token: string;
}
```

Рисунок 4.8 – Інтерфейс користувача системи

```ts
export interface City {
  id: string;
  name: string;
}

export interface GeoLocation {
  lng: number;
  lat: number;
}

export interface Photo {
  id: string | number;
  url: string; // link to photo on server
}

type wasteType =
  | 'organic'
  | 'paper'
  | 'cardboard'
  | 'hazardous'
  | 'metal'
  | 'glass'
  | 'electrical'
  | 'reusable things'
  | 'bulky'
  | 'construction'
  | 'plastic'
  | 'dangerous areas'
  | 'Red Cross action point';

export interface ecoServiceEntity {
  id: string | number; // unique identifier
  name: string; // name of organization
  cityId?: string | number; // id of city where it located
  geo: GeoLocation; // GeoLocation coordinates object
  wasteTypes: wasteType[];
  phone?: string; // (380) 989-88-99,
  address?: string; // 'Расстанный переулок, д. 1',
  photos?: Photo[]; // array of photos or empty [] if no
  payed: boolean; // 'free' | 'payed';
  paidComment?: string; // some price info
  delivery: 'none' | 'apartment' | 'street' | 'location';
}
```

Рисунок 4.9 – Інтерфейс пункту прийому та переробки вторинної сировини

- admin-sidebar містить компоненти для додавання, видалення, оновлення сервісу, підтвердження дій адміністратора та сам компонент сторінки адміністратора (рис 4.10)

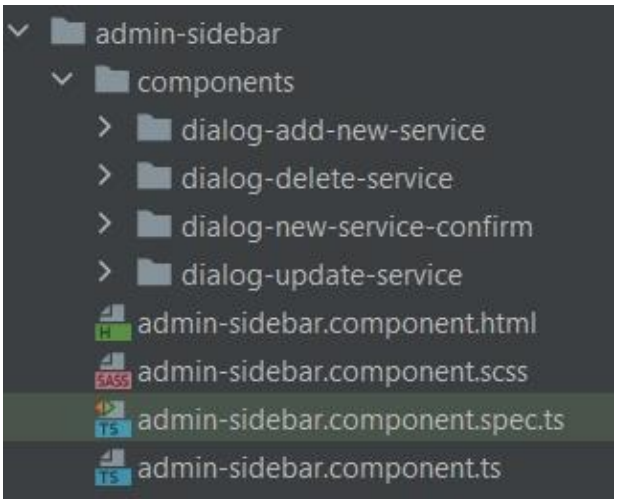

Рисунок 4.10 – Структура компоненту admin-sidebar

- **modules** містить**:**

1) компонент map, в якому зберігається компоненти карти, маркеру, спливаючого вікна, бокової панелі для звичайного користувача та панелі інструментів з фільтрами; сервіси для обробки даних з БД; модуль, де підключаються, для більш комфортної роботи з мапою, модулі, компоненти, директиви, що дозволяють форматувати результат у рядковій інтерполяції; модуль для маршрутізації компонента map;

2) компонент user, який містить компонент реєстрацій користувача у системі та входу до особистого кабінету; модуль, де підключаються, для більш комфортної роботи з користувачами, модулі, компоненти, пайпи та тп; та модуль для маршрутізації компонента user;

- **shared** визначає:

1) material.module.ts для винесення підключення модулів бібліотеки Angular Material у окремий допоміжний модуль;
2) shared.module.ts для винесення підключення потрібних для коректної роботи системи у окремий допоміжний модуль;

- **search.pipe.ts** містить реалізацію пайпу пошуку потрібного пункту прийому та переробки вторинної сировини за певним ключовим

4.5.3 Файли конфігурації програм

Спеціальні файли конфігурації для кореневої програми знаходяться на кореневому рівні робочої області.

Конфігураційні файли TypeScript для конкретного проекту успадковуються від файлу tsconfig.json для всього робочого простору.

- **.browserslistrc** налаштовує спільне використання цільових браузерів та версій Node.js між різними інтерфейсними інструментами;
- **karma.conf.js** містить конфігурацію карми для конкретної програми;
- **tsconfig.app.json** містить конфігурацію TypeScript для конкретної програми, враховуючи параметри компілятора шаблонів TypeScript та Angular;
- **tsconfig.spec.json** конфігурація TypeScript для тестів програми.

словом;

5 ІНСТРУКЦІЯ РОБОТИ КОРИСТУВАЧА З ІНТЕРАКТИВНОЮ СИСТЕМОЮ ПОШУКУ І РЕЄСТРАЦІЇ ПУНКТІВ ПРИЙОМУ ТА ПЕРЕРОБКИ ВТОРИННОЇ СИРОВИНИ

Інтерактивна система створена для здійснення реєстрації, перегляду та пошуку пунктів прийому та переробки вторинної сировини на інтерактивній мапі. Має зрозумілу та просту систему користування. Користувачі, з різними ролями в системі, мають однаковий інтерфейс, але різний функціонал системи.

Розроблений програмний продукт використовує для роботи веб-браузер, тобто для успішного користування всім функціоналом веб-додатку користувач повинен мати лише доступ до веб-браузера та інтернет підключення, перейшовши за відповідним посиланням. Застосунок підтримує останні версії усіх сучасних браузерів, а саме Google Chrome, Mozilla Firefox, Apple Safari ta Microsoft Edge.

5.1 Робота користувача з системою

Після переходу за посиланням, користувач потрапляє на головну сторінку (рис. 5.1). На ній відображається інтерактивна мапа з пунктами прийому та переробки вторинної сировини, панель зі списками різних категорій пунктів за якими може відбуватися фільтрація міток на мапі, кнопка реєстрації користувача в системі, кнопка входу до особистого кабінету та посилання на допоміжну інформацію з сортування сміття.

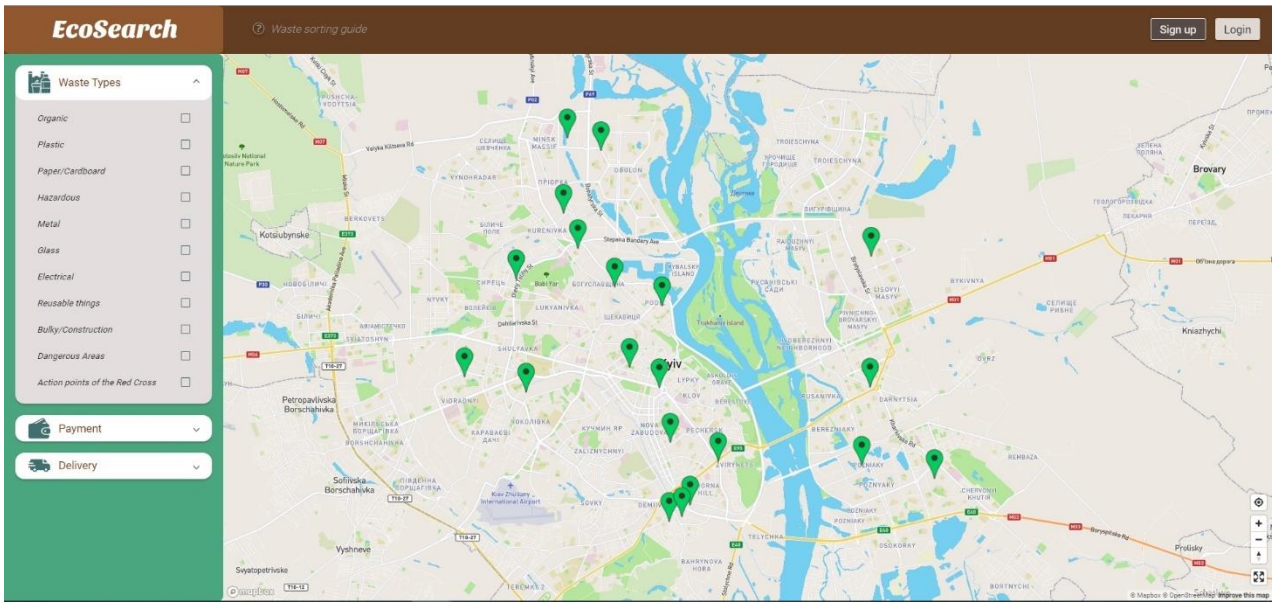

Рисунок 5.1 – Головна сторінка інформаційної системи

5.1.1 Фільтрація пунктів прийому та переробки

Користувач системи має змогу відфільтрувати пункти прийому та переробки вторинної сировини за потрібними йому критеріями. В залежності від обраного типу, на мапі відображаються лише ті пункти, які відповідають запиту користувача (рис. 5.2). Фільтрація може відбуватися не лише за декількома потрібними типами одночасно (рис 5.3), але й об’єднувати у запиті різні категорії пошуку потрібного пункту (рис. 5.4).

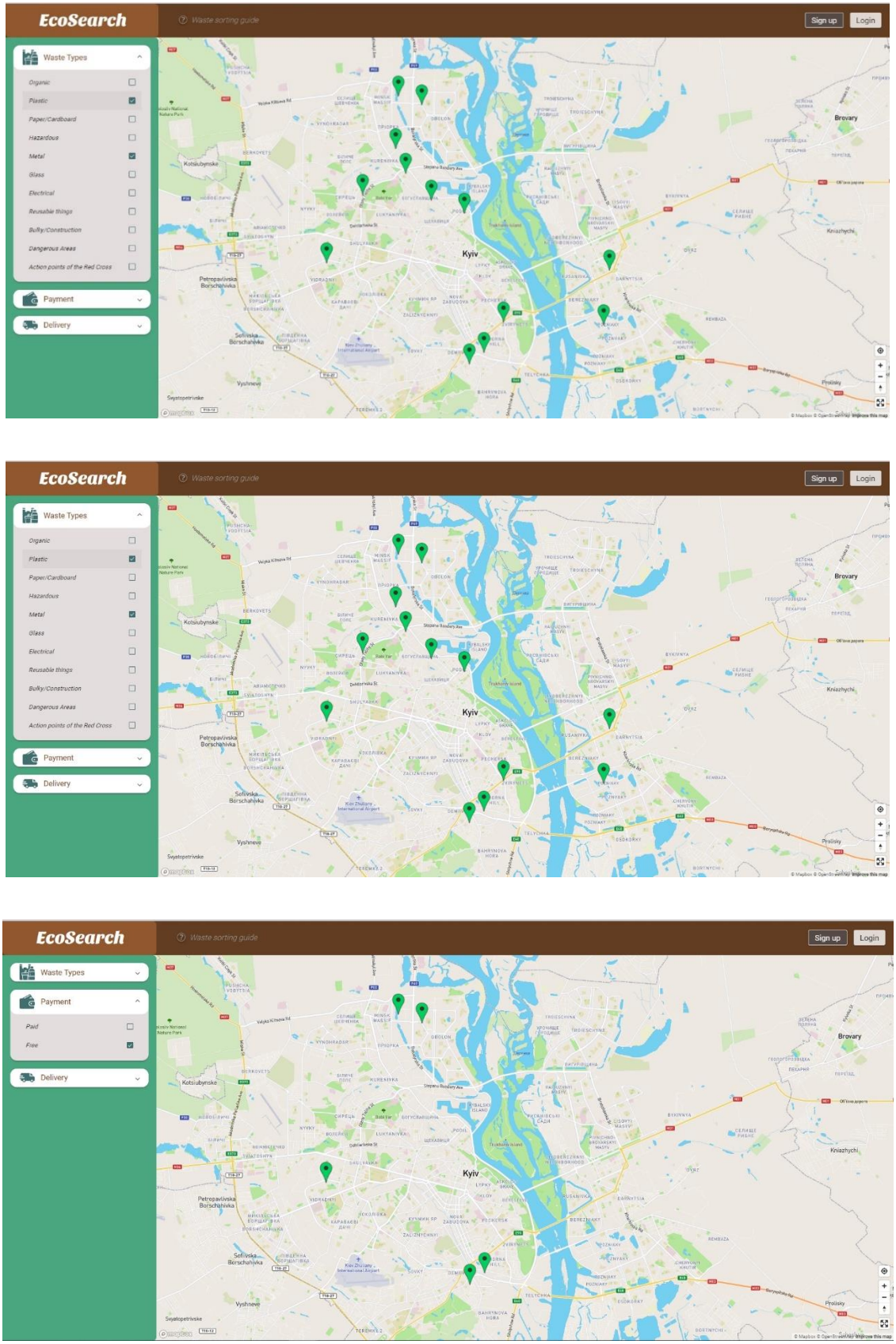

Рисунок 5.2 – 5.4 – Фільтрація за різними категорії та типами пунктів прийому та переробки вторинної сировини

5.1.2 Робота з інтерактивною мапою

Інтерактивна мапа наділена дуже зручним та комфортним у використанні функціоналом. Користувач може легко керувати мапою: збільшувати, віддаляти, пересувати або повертати її, для зручності користування. Також користувач має змогу подивитися головну інформацію пункту прийому та переробки вторинної сировини, натиснувши на потрібним йому пункт (рис 5.5).

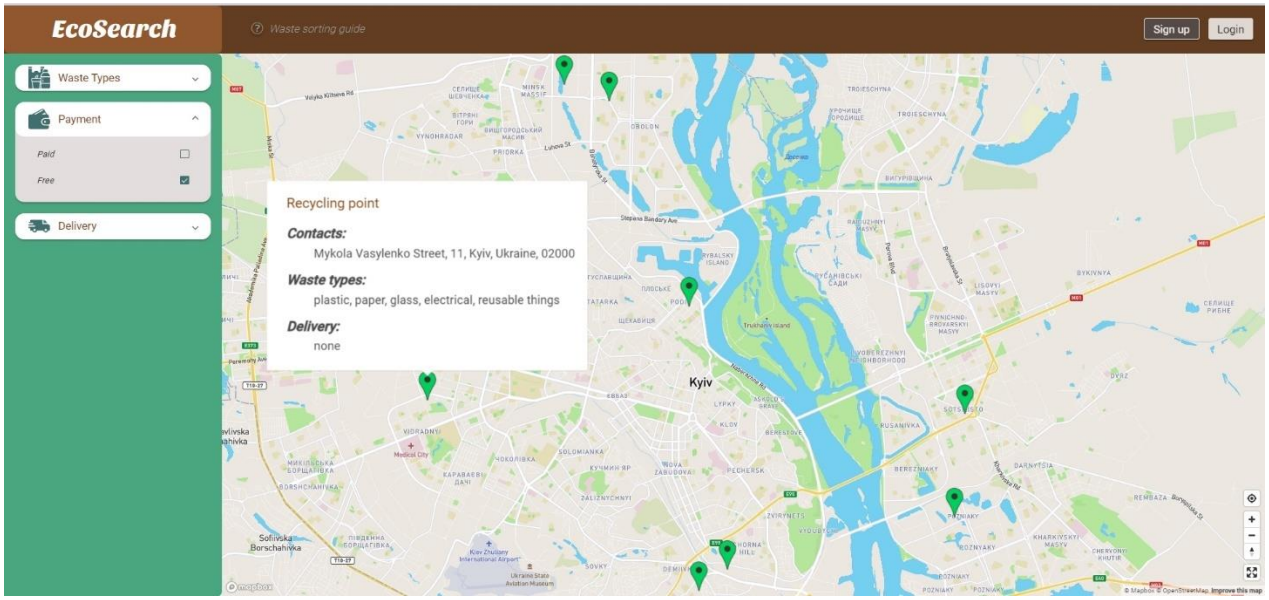

Рисунок 5.5 – Детальна інформація пункту прийому та переробки

5.1.3 Реєстрація користувача в системі

Для забезпечення розширеного доступу до певного функціоналу, користувачу надається можливість реєстрації у системі (рис. 5.6).

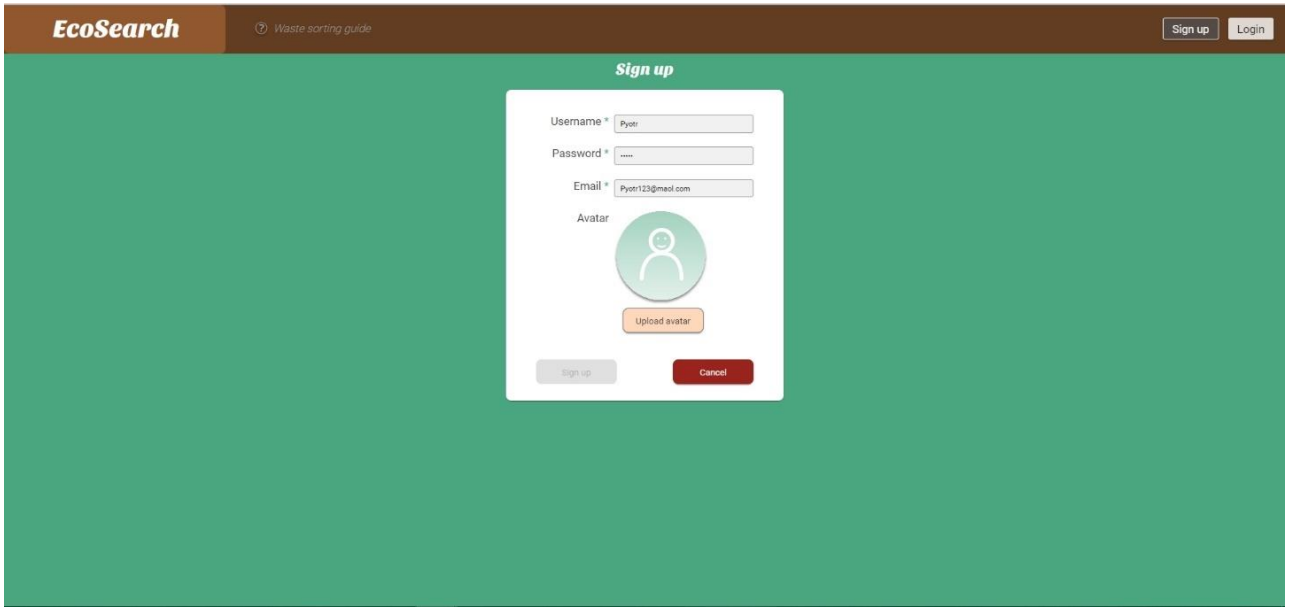

Рисунок 5.6 – Сторінка реєстрації користувача в системі

Після натискання кнопки «sign up», користувач потрапляє на сторінку реєстрації в системі. Форма реєстрації складається з чотирьох полів – ім’я користувача, пароль, адреса електронної пошти та необов’язкового поля прикріплення аватара користувача. Всі дані, що вводяться до форми валідуються перед відправкою, щоб уникнути створення користувача з незаповненими даними або з даними неправильного формату.

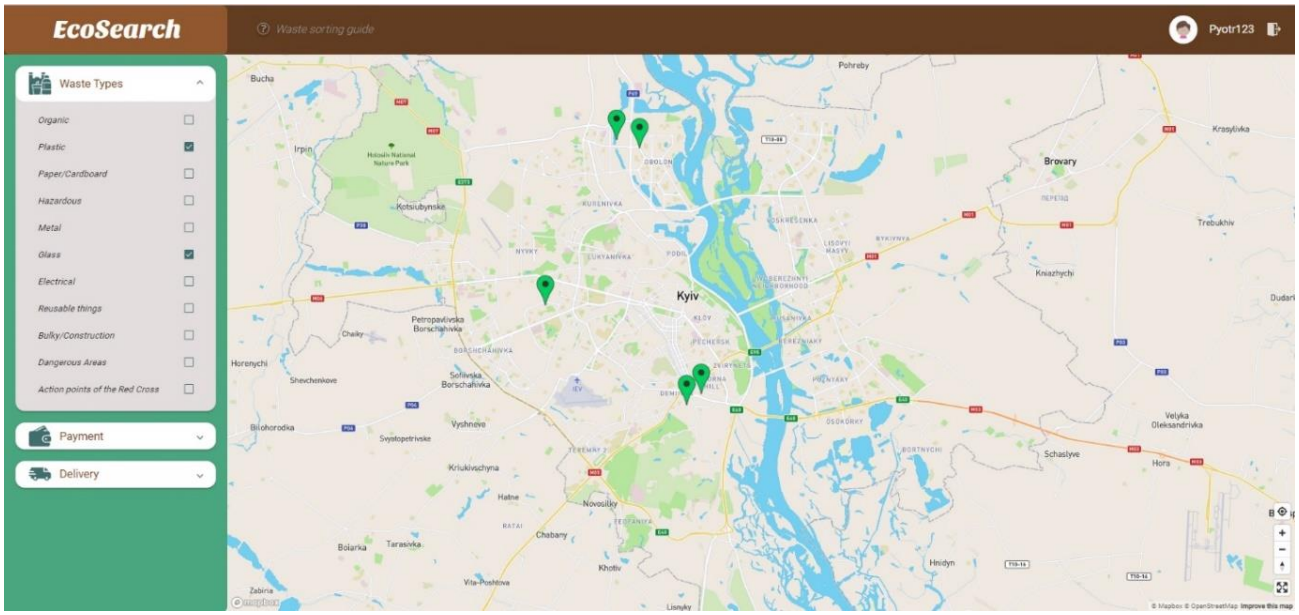

Рисунок 5.7 – Сторінка зареєстрованого користувача в системі

Після коректного вводу та перевірці даних, натиснувши на активну кнопку «sign up», користувач перенаправляється на головну сторінку, за допомогою свого облікового запису (рис 5.7).

Користувач має змогу вийти з особистого кабінету, натиснувши з правої сторони від його імені на кнопку виходу, та буде перенаправлений до сторінки входу в систему.

5.1.4 Вхід користувача в систему

Потрапивши на сторінку авторизації в системі, зареєстрований користувач повинен ввести свій логін та пароль (рис. 5.8). Після входу до свого особистого кабінету, в залежності від ролі користувача в системі, відкриється потрібна сторінка для подальшої роботи з сервісом.

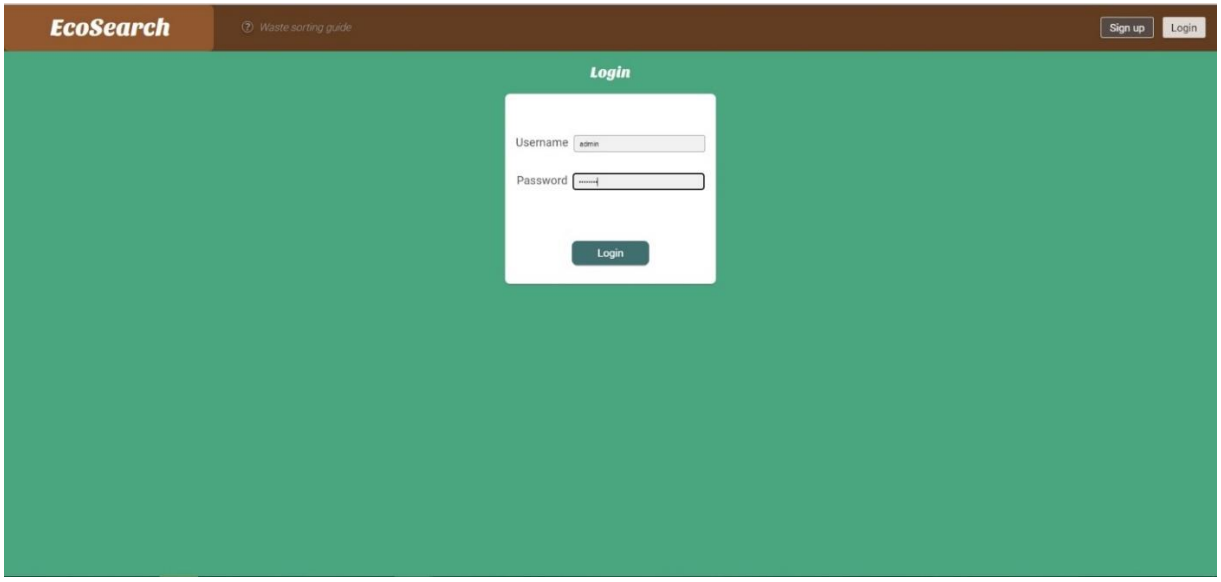

Рисунок 5.8 – Сторінка входу користувача в особистий кабінет

5.2 Робота адміністратора з системою

Увійшовши до облікового запису, як адміністратор системи, користувач потрапляє на відповідну сторінку з розширеним функціоналом перегляду, додавання, оновлення та видалення пунктів прийому та переробки вторинної сировини. Користувач може бачити панель зі списком доступних пунктів, поле для швидкого пошуку за ключовим словом та кнопку «Add new service» для додавання нового сервісу до списку (рис. 5.9).

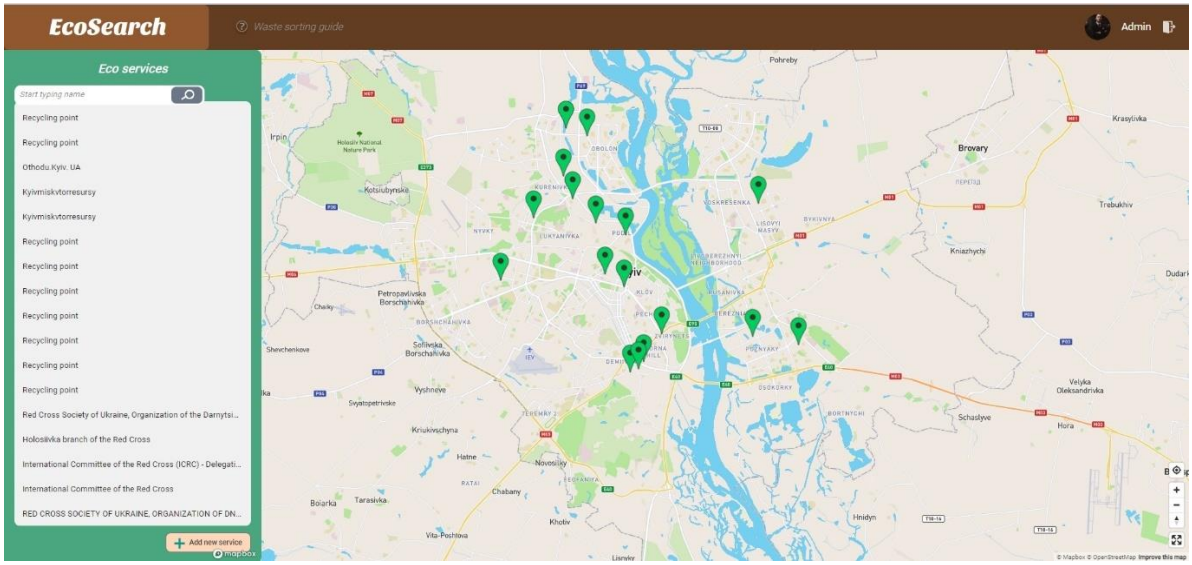

Рисунок 5.9 – Головна сторінка адміністратора інформаційної системи

5.2.1 Поле для пошуку сервісів

Функціонал пошуку потрібного пункту прийому та переробки вторинної сировини або пункту дії Червоного Хреста реалізовано наступним чином.

Користувач пише у полі вводу потрібну йому назву або будь-яку іншу характеристику та здійснює пошук за списком присутніх сервісів у ньому (рис 5.10). Це є зручним інструментом для роботи з великою кількістю інформації, бо робить користування системою більш швидким та зручним процесом.

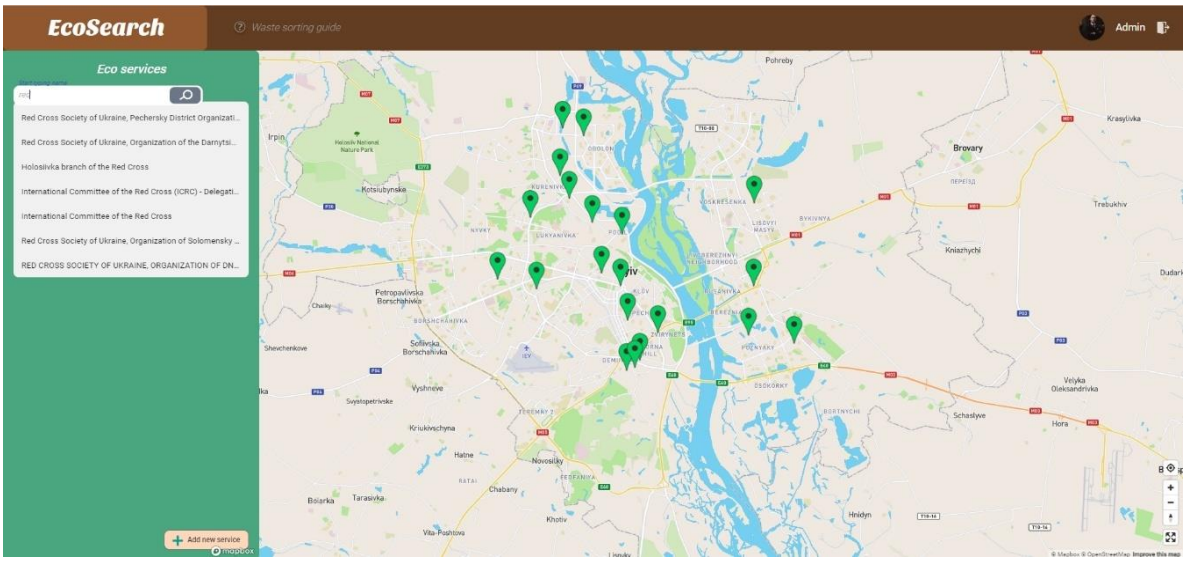

Рисунок 5.10 – Поле пошуку сервісів

5.2.2 Додавання нового пункту

Якщо перед користувачем стоїть задача створення нового пункту та відображення його на мапі, він легко може це зробити натиснувши на відповідну кнопку «Add new service». Після натискання цієї кнопки, відкриється діалогове вікно для перевірки дійсності створення нового сервісу (рис 5.11).

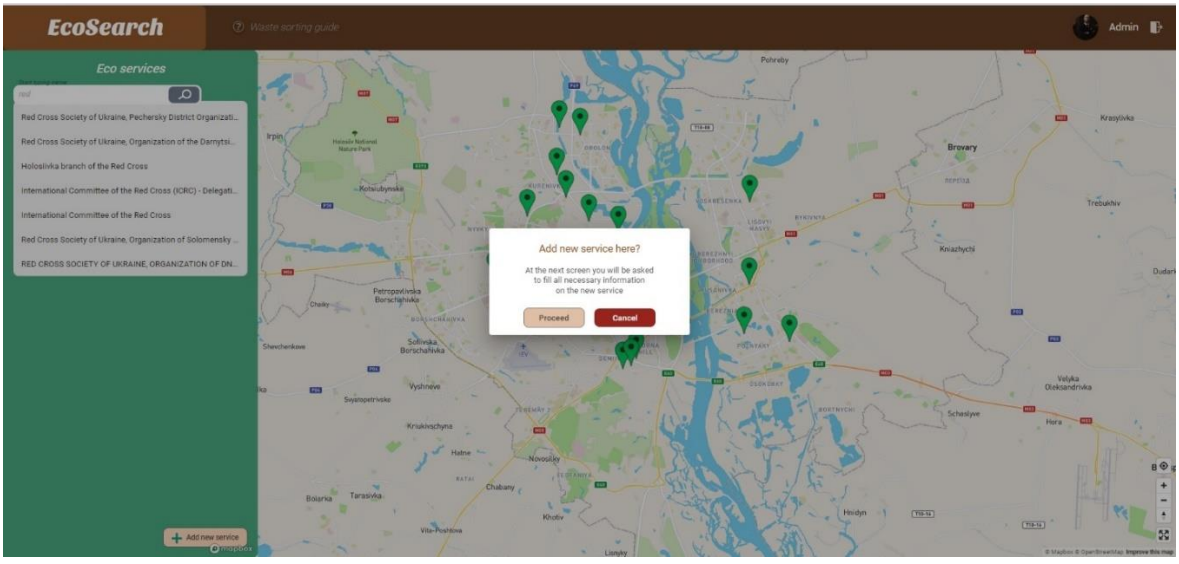

Рисунок 5.11 – Діалогове вікно перевірки дійсності створення нового пункту

Після підтвердження даної операції відкриється форма заповнення нового пункту прийому або пункту дії Червоного Хреста або у випадку відміни користувач повертається до головного меню..

Після коректного заповнення всіх обов’язкових полів форми (рис.5.12) та перевірки внесених даних на предмет створення пункту з незаповненими даними або з даними неправильного формату, користувач може переглянути створений пункт на інтерактивній мапі (рис. 5.13)

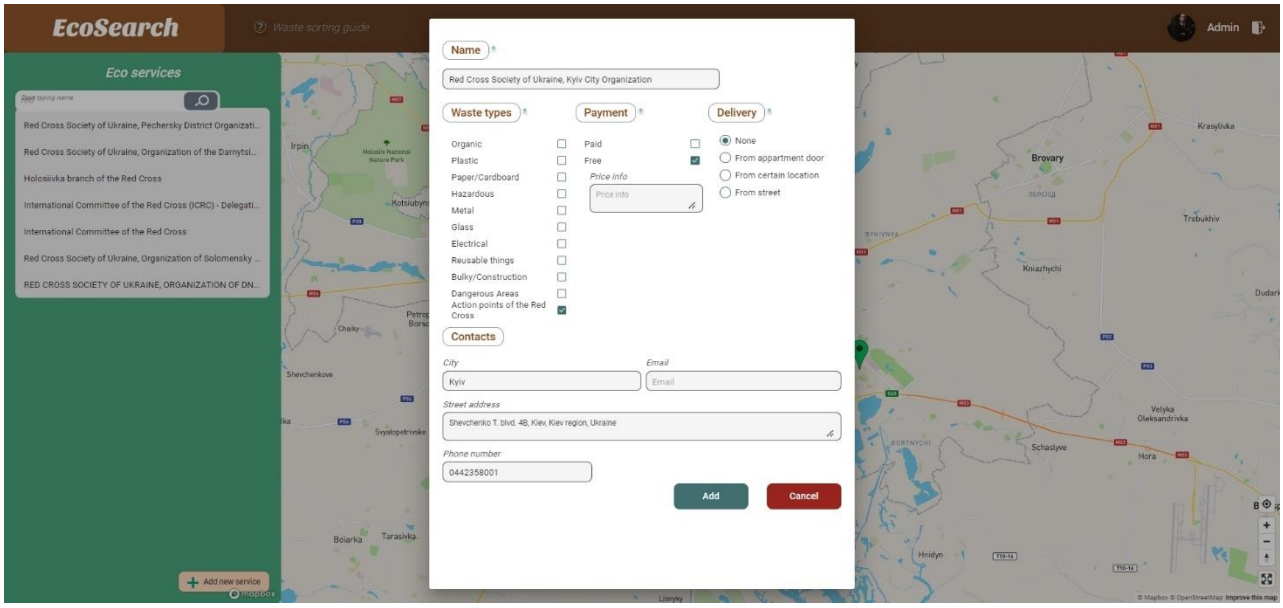

Рисунок 5.12 – Створення нового пункту

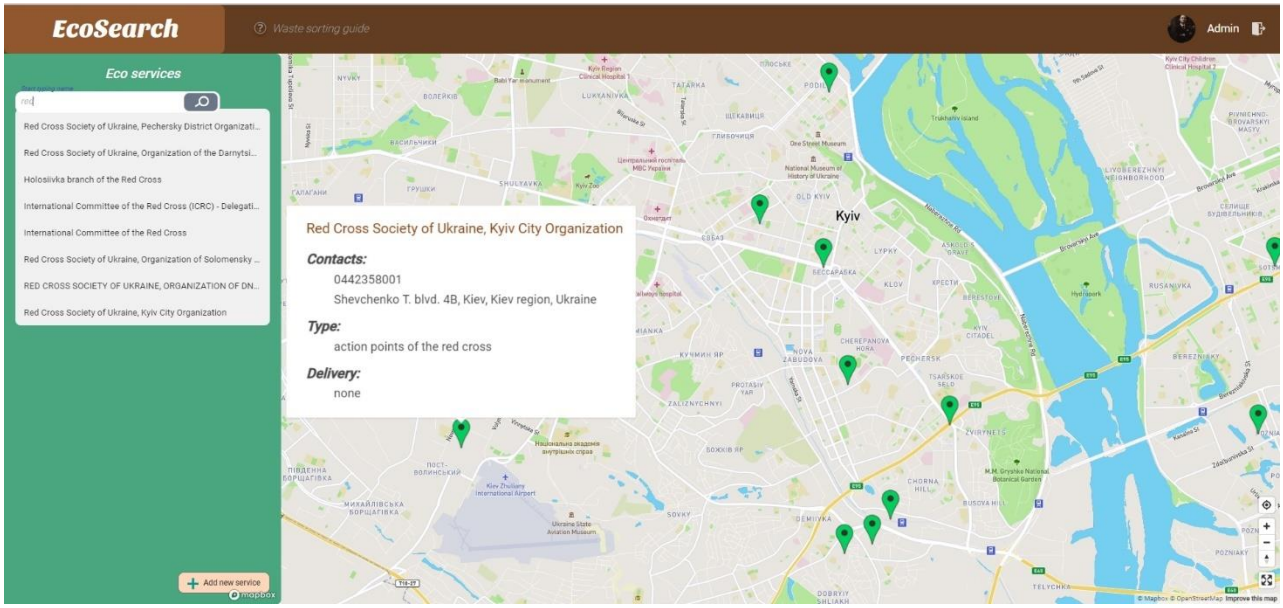

Рисунок 5.13 – Відображення нового пункту на мапі

5.2.3 Редагування існуючого пункту

Функціонал системи дозволяє редагувати вже існуючі пункти. Для цього користувачу необхідно вибрати зі списку потрібний пункт та натиснути на нього. Після чого відкриється форма, заповненими відповідно до вибраного пункту полями, з можливістю внесення змін (рис. 5.14).

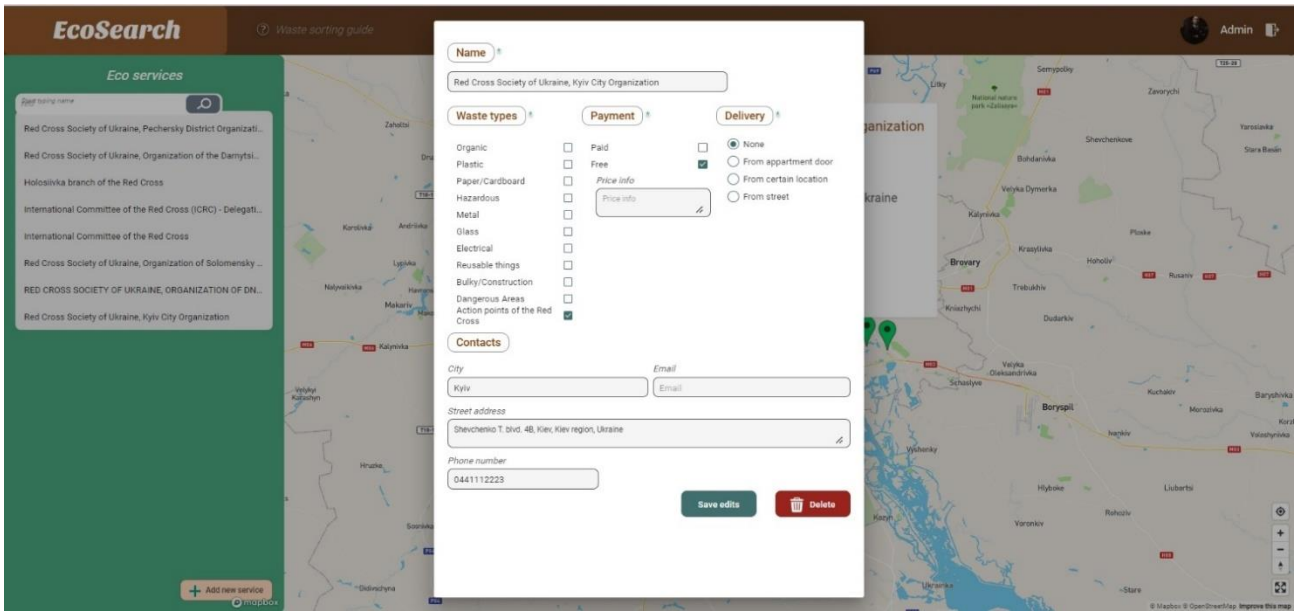

Рисунок 5.14 – Редагування існуючого пункту

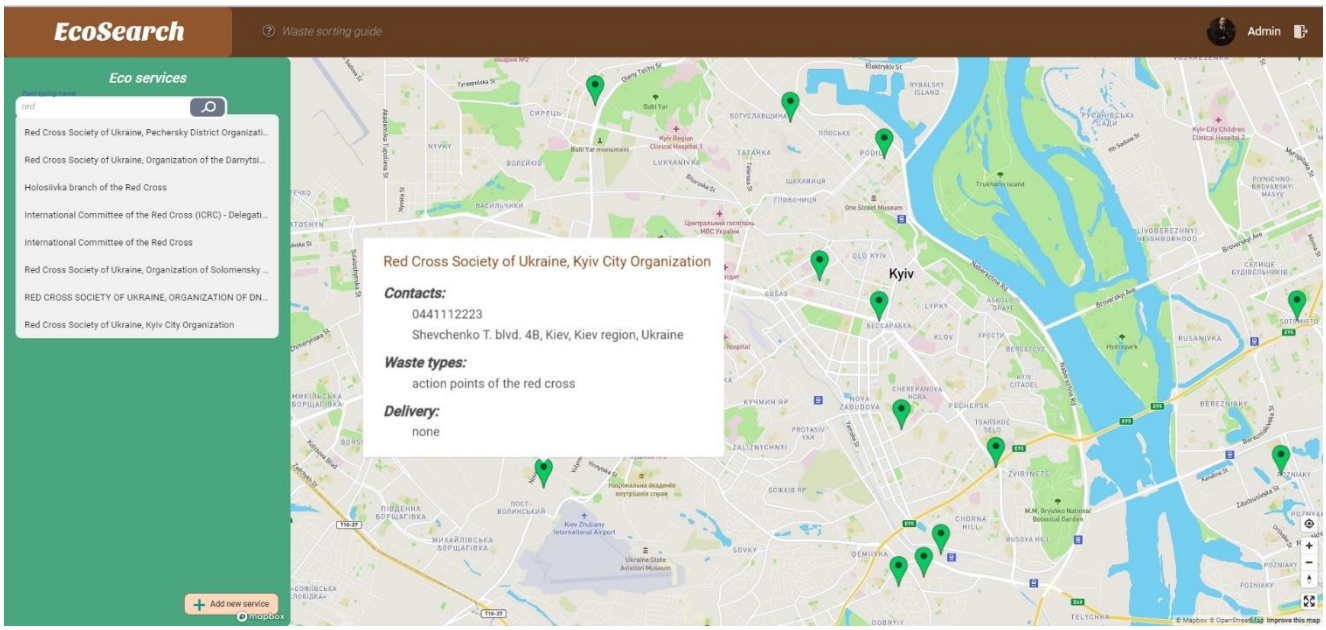

Рисунок 5.15 – Відображення зміненого пункту на мапі

Як і у випадку з додаванням нового пункту, при редагуванні існуючого потребується коректне заповнення всіх обов’язкових полів форми та перевірка внесених даних. Користувач знову може переглянути змінений пункт на інтерактивній мапі (рис. 5.15)

5.2.4 Видалення існуючого пункту

Для видалення застарілого пункту користувачу необхідно так само вибрати зі списку пунктів необхідний та натиснути на нього. У відкритому діалоговому вікні натиснути «Delete» (рис. 5.16) та перевірити видалення застарілого пункту з інтерактивної мапи (рис. 5.17).

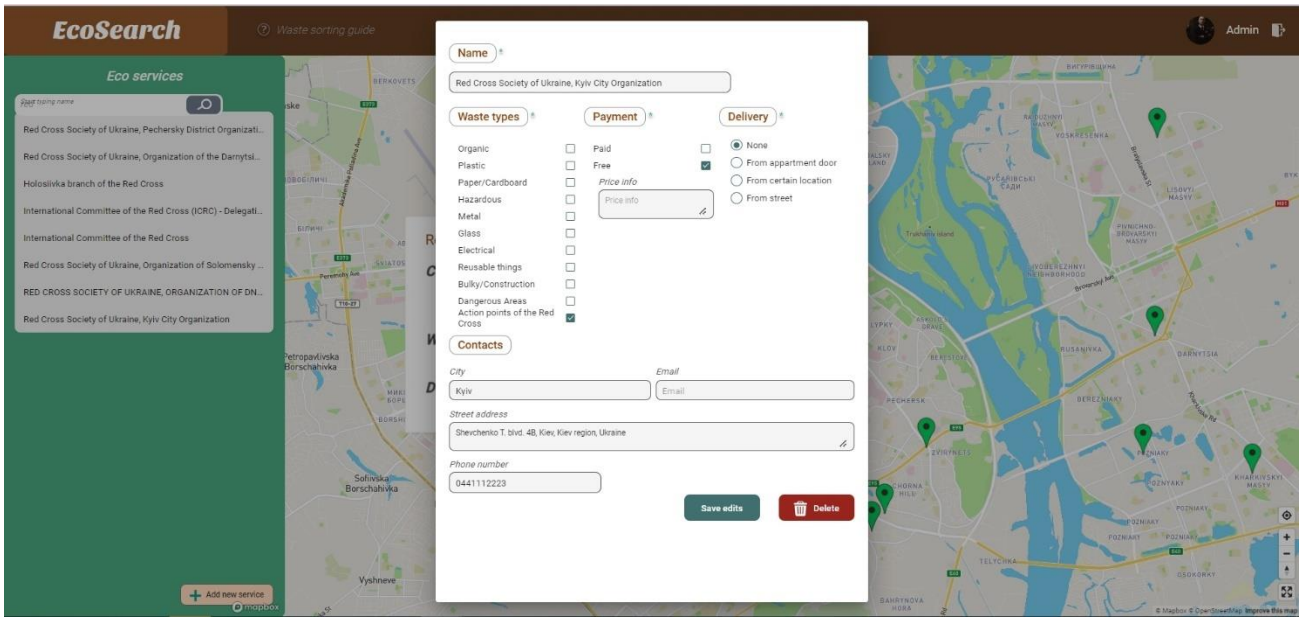

Рисунок 5.16 – Видалення існуючого пункту

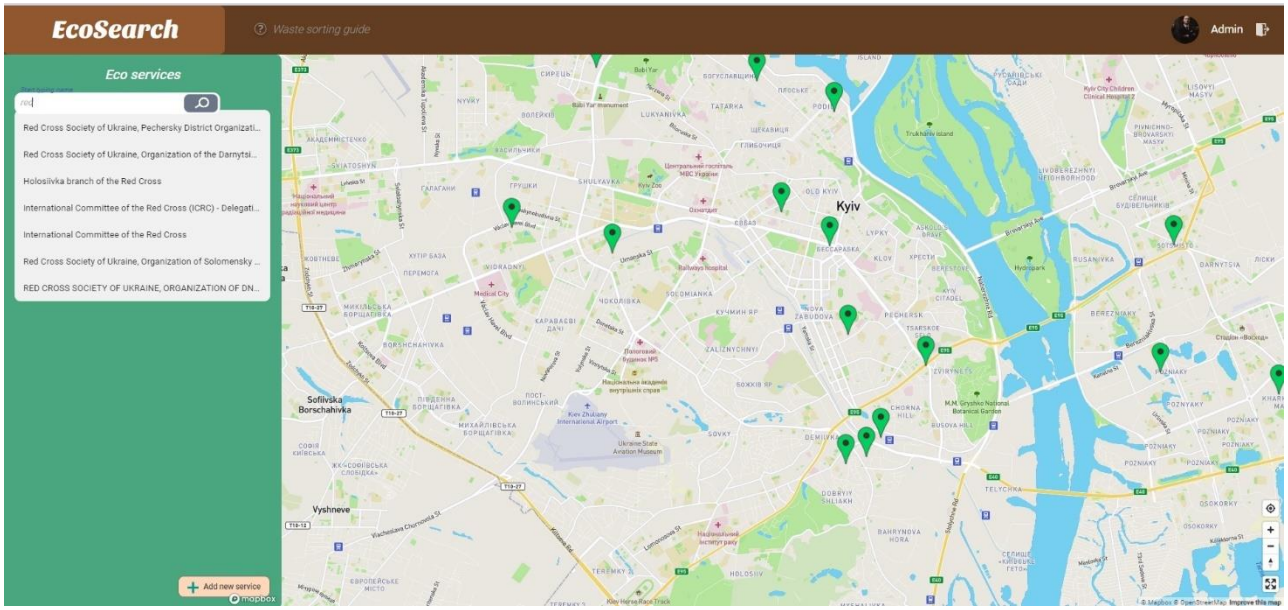

Рисунок 5.17 – Видалення пункту з інтерактивної мапи

5.3 Висновки до розділу

У п’ятому розділі було описано системні вимоги, що необхідні для коректного встановлення та роботи програми. Також було детально розглянуто всі варіанти взаємодії з розробленим інформаційною системою користувачів з різними ролями. Було продемонстровано зручний у використанні функціонал програми та адаптований під різні девайси дизайн продукту.

ВИСНОВКИ

У ході виконання дипломної роботи було проаналізовано існуючі аналоги проблематики предметної області роботи, визначено їх основні переваги і недоліки та на основі аналізу сформовано функціонал програмного продукту.

Розроблена інформаційна система пошуку і реєстрації пунктів прийому та переробки вторинної сировини надає користувачу можливості здійснити перегляд та пошук пунктів прийому на інтерактивній мапі. В залежності від ролі користувача, система має різний функціонал, а саме: додавання нової, редагування та видалення існуючої мітки або фільтрації пунктів прийому за потрібними критеріями. Також реалізована можливість перегляду рекомендацій щодо сортування вторинної сировини.

Доповненням до існуючого функціоналу розроблюваної системи стала можливість позначення небезпечних зон, де могли залишитися залишки або уламки вибухонебезпечних предметів чи будь-яке інше мілітарі-обладнання, яке потребує утилізації. Також застосунок пропонує використовувати інтерактивну мапу, як спосіб інформування громадян про місцезнаходження пунктів дії Червоного Хреста, що є наразі досить актуальною проблемою.

При розробці клієнтського застосунку було обрано фреймворк Angular, з використанням мови TypeScript, бібліотеку компонентів Angular Material та інтерактивну мапу MapBox. Для роботи з серверною частиною використовувалась мова програмування JavaScript, за допомогою оточення Node.js, та бібліотеки Express. В якості середовища розробки було обрано редактор коду WebStorm.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційна сторінка. *Київкомунсервіс* : веб-сайт. URL: https://kks.kyiv.ua/pro-na/ (дата звернення: 22.02.2022).

2. Документація WebStorm. *JetBrains* : веб-сайт. URL: https://www.jetbrains.com/ru-ru/webstorm/ (дата звернення: 26.05.2022).

3. Flanagan D. JavaScript: The Definitive Guide : tutorial. O'Reilly Media, Inc., 2006. 1018 p.

4. Виразний Javascript. *Karmazzin* : веб-сайт. URL: https://karmazzin.gitbooks.io/eloquentjavascript_ru/content/ (дата звернення: 26.05.2022).

5. Резиг Джон, Бибо Беэр Секреты javascript ниндзя : навч. посіб. «Вильямс», 2013. 416 с.

6. Що таке Angular і навіщо все його вчать? *Agilie* : веб-сайт. URL: https://agilie.com/ru/blog/chto-takoie-angular-i-zachiem-iegho-vsie-uchat (дата звернення: 26.05.2022).

7. Yakov Fain Angular Development with TypeScript : tutorial. 2nd Edition. Manning; 2nd edition 2017. 560p.

8. Figma: огляд програми для веб-дизайну. *SendPulse* : веб-сайт. URL: https://sendpulse.com/ru/blog/figma (дата звернення: 26.05.2022).

9. Документація MapBox. *MapBox* : веб-сайт. URL: https://www.mapbox.com/about/company (дата звернення: 27.05.2022).

10. Документація NodeJS. *Blog.Skillfactory* : веб-сайт. URL: https://blog.skillfactory.ru/glossary/node-js/ (дата звернення: 27.05.2022).

11. Херрон Девід. Node.js. Розробка серверних веб-додатків на JavaScript : підручник. 2012. 144с

12. Чепегін І. Д. Серверний JavaScript – переваги та недоліки node.Js. И. 61 Д. Чепегін. 2020. №1. С. 18–20

13. Документація Express.js. *Node.js* : веб-сайт. URL: https://nodejsdev.ru/doc/express/ (дата звернення: 27.05.2022).

14. Робсон Е., Фрімен Е. Вивчаємо HTML, XHTML і CSS : навч. посібник. O'Reilly Media, 2019. 720с

15. Макфарланд Д. Нова велика книга CSS : підручник. O'Reilly Media,, 2016. 720с.

16. Технології HTML5 та CSS3. *Astwellsoft* : веб-сайт. URL: https://astwellsoft.com/ru/blog/tehnology/html5-css3.html (дата звернення: 27.05.2022).

17. Що таке MySQL? *Freehost.ua*: веб-сайт. URL: https://freehost.com.ua/faq/wiki/chto-takoe-mysql/ (дата звернення: 27.05.2022).

18. MySQL Workbench, навчальний посібник: повний посібник з інструменту СУБД. *Best Programmer* : веб-сайт. URL: https://bestprogrammer.ru/baza-dannyh/uchebnoe-posobie-po-mysql-workbench-polnoe-rukovodstvo-po-instrumentu-subd (дата звернення: 28.05.2022).

19. Навчальний посібник з діаграми прецедентів (Посібник з прикладами). *Creately* : веб-сайт. URL: https://creately.com/blog/ru (дата звернення: 28.05.2022).

ДОДАТОК А

Інтерактивна система пошуку і реєстрації пунктів прийому та переробки вторинної сировини

Текст програмного модулю
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ_82303

Аркушів 10

Київ — 2022

map.component.html

//модуль для імплементації інтерактивної мапи в систему, зокрема для відображення мапи на сторінці.

```
<mgl-map
  (mapClick)="mapClickEventHandler($event)"
  (mapLoad)="mapLoadEvent($event)"
  [center]="center"
  [style]="style"
  [zoom]="zoom"
>
  <mgl-control mglFullscreen position="bottom-right"></mgl-control>
  <mgl-control mglNavigation position="bottom-right"></mgl-control>
  <mgl-control mglGeolocate position="bottom-right"></mgl-control>
  <app-marker
    *ngFor="let es of ecoStations"
    [es]="es"
    [events]="eventsSubject.asObservable()"
    [map]="map"
  ></app-marker>
</mgl-map>
<app-sidebar></app-sidebar>
<app-toolbar></app-toolbar>
```

map.component.ts

// модуль для реалізації імплементації функціоналу мапи.

```
import {
  AfterViewInit,
  Component,
  OnChanges,
```

```
  OnInit,
  ViewEncapsulation,
} from '@angular/core';
import * as mapboxgl from 'mapbox-gl';
import { Subject } from 'rxjs';
import { DataService } from '../../services/data.service';

@Component({
  selector: 'app-map',
  templateUrl: './map.component.html',
  styleUrls: ['./map.component.scss'],
  encapsulation: ViewEncapsulation.None,
})
export class MapComponent implements OnInit {
  public map!: mapboxgl.Map;
  public ecoStations!: any;
  public center!: any;
  public zoom!: any;
  public style!: string | undefined;
  eventsSubject: Subject<any> = new Subject<any>();

  constructor(private dataService: DataService) {
    this.ecoStations = this.dataService.store$.value.ecoStations;
    this.center = this.dataService.store$.value.mapDefaults.center;
    this.zoom = this.dataService.store$.value.mapDefaults.zoom;
    this.style = this.dataService.store$.value.mapDefaults.style;
  }

  ngOnInit(): void {
```

```
    this.dataService.store$.subscribe((store) => {
      this.ecoStations = store.ecoStations;
    });
  }

  mapLoadEvent(map: mapboxgl.Map) {
    this.map = map;
  }

  mapClickEventHandler(event: any) {
    this.eventsSubject.next(event);
  }

  markerClickEventHandler(event: any) {
    console.log(event);
  }
}
```

marker.conmponent.html

```
// модуль реалізації маркерів (пунктів прийому вторинної сировини), відображення їх на інтерактивній мапі
<mgl-marker [lngLat]="[es.geo.lng, es.geo.lat]" anchor="bottom" #myMarker>
  <div class="marker" (click)="markerClickHandler()"></div>
</mgl-marker>
<mgl-popup
  [marker]="myMarker"
  maxWidth="auto"
  [closeButton]="false"
```

```
  anchor="bottom"
  [offset]="45"
  class="mgl-popup"
>
  <app-popup [es]="es"></app-popup>
</mgl-popup>
```

marker.component.ts

// модуль реалізації функцій маркеру. Відповідає за взаємодію з ним, визначення розташування на мапі, передачу даних до html сторінки

```
import {
  Component,
  EventEmitter,
  Input,
  OnDestroy,
  OnInit,
  Output,
  ViewEncapsulation,
} from '@angular/core';
import { Observable, Subscription } from 'rxjs';
import { ecoServiceEntity } from '../../../../core/interfaces/ecoService.entity';

@Component({
  selector: 'app-marker',
  templateUrl: './marker.component.html',
  styleUrls: ['./marker.component.scss'],
  encapsulation: ViewEncapsulation.None,
})
```

```
export class MarkerComponent implements OnInit, OnDestroy {
  private eventsSubscription!: Subscription;
  @Input() events!: Observable<any>;
  @Input() es!: ecoServiceEntity;
  @Input() map!: mapboxgl.Map;
  clicked = new EventEmitter();

  constructor() {}

  ngOnInit(): void {
    this.eventsSubscription = this.events.subscribe((data) => {});
  }

  ngOnDestroy() {
    this.eventsSubscription.unsubscribe();
  }

  markerClickHandler() {
    this.map.flyTo({
      offset: [0, 70],
      center: this.es.geo,
      essential: true,
      animate: true,
      zoom: 9.5,
    });
  }
}
```

sidebar.component.html

// модуль для відображення меню користувача на головній сторінці системи, а саме відображення категорій фільтрації та їх типів, за якими користувач може фільтрувати пункти на мапі.

```
<mat-sidenav-container class="sidenav-container">
  <mat-sidenav mode="side" opened class="sidenav" disableClose="true">
    <form [formGroup]="filtersFormGroup">
      <mat-accordion class="accordion">
        <mat-expansion-panel
          [expanded]="panelOpenState === 0"
          (opened)="setPanelOpenState(0)"
          class="panel-container"
        >
          <mat-expansion-panel-header>
            <mat-panel-title>
              <img alt="waste type icon" src="/assets/types.svg" />
              <p>Waste Types</p>
            </mat-panel-title>
          </mat-expansion-panel-header>
          <mat-selection-list formControlName="types">
            <mat-list-option
              *ngFor="let wasteType of toolbarData.types"
              [value]="wasteType"
              (click)="filterChangeHandler()"
            >
              {{ wasteType }}
            </mat-list-option>
          </mat-selection-list>
```

```
  </mat-expansion-panel>
</mat-accordion>

<mat-accordion class="accordion">
  <mat-expansion-panel
    [expanded]="panelOpenState === 1"
    (opened)="setPanelOpenState(1)"
    class="panel-container"
  >
    <mat-expansion-panel-header>
      <mat-panel-title>
        <img alt="payment icon" src="assets/payment.svg" />
        <p>Payment</p>
      </mat-panel-title>
    </mat-expansion-panel-header>
    <mat-selection-list formControlName="payment">
      <mat-list-option
        *ngFor="let payment of toolbarData.payment"
        [value]="payment"
        (click)="filterChangeHandler()"
      >
        {{ payment }}
      </mat-list-option>
    </mat-selection-list>
  </mat-expansion-panel>
</mat-accordion>

<mat-accordion class="accordion">
  <mat-expansion-panel
```

```
        [expanded]="panelOpenState === 2"
        (opened)="setPanelOpenState(2)"
        class="panel-container"
      >
        <mat-expansion-panel-header>
          <mat-panel-title>
            <img alt="delivery icon" src="assets/delivery.svg" />
            <p>Delivery</p>
          </mat-panel-title>
        </mat-expansion-panel-header>
        <mat-radio-group formControlName="delivery">
          <mat-radio-button
            *ngFor="let delivery of toolbarData.delivery"
            (click)="filterChangeHandler(delivery)"
            [value]="delivery"
          >
            {{ delivery }}
          </mat-radio-button>
        </mat-radio-group>
      </mat-expansion-panel>
    </mat-accordion>
  </form>
 </mat-sidenav>
</mat-sidenav-container>
```

sidebar.component.ts

// модуль для отримання категорій фільтрації та їх типів з серверу та подальше використання в системі. Також реалізація функціоналу панелі, відкриття та закриття списків з типами кожної категорії.

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import {
  FormBuilder,
  FormControl,
  FormGroup,
  Validators,
} from '@angular/forms';
import { DataService } from '../../services/data.service';

@Component({
  selector: 'app-sidebar',
  templateUrl: './sidebar.component.html',
  styleUrls: ['./sidebar.component.scss'],
})
export class SidebarComponent implements OnInit {
  public panelOpenState = 0;
  public toolbarData!: any;
  public filtersFormGroup!: FormGroup;

  constructor(private dataService: DataService, private fb: FormBuilder) {
    this.toolbarData = this.dataService.store$.value.toolbarData;
  }

  ngOnInit(): void {
    this.filtersFormGroup = this.fb.group({
      types: new FormControl(this.dataService.store$.value.filters.types),
      payment: new FormControl(this.dataService.store$.value.filters.payment),
      delivery: new FormControl(this.dataService.store$.value.filters.delivery),
    });
```

```
  }

  filterChangeHandler(delivery = '') {
    if (delivery) {
      this.filtersFormGroup.setValue({
        ...this.filtersFormGroup.value,
        delivery,
      });
    }
    this.dataService.updateFilters(this.filtersFormGroup.value);
    this.dataService.filterPoints();
  }

  setPanelOpenState(index: number) {
    this.panelOpenState = index;
  }
}
```