КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-застосунок для сортування сміття»

Виконала _____
(Підпис)
Дуля Дарина Василівна
(прізвище, ім'я, по батькові)

Керівник к.т.н., доц. Міронова В.Л.
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2021

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

Назва теми: «Веб-застосунок для сортування сміття»

Освітня програма: Прикладне програмування
Спеціальність: Комп'ютерні науки

ПІБ	Підпис
Дуля Дарина Василівна	

Назва роботи українською та англійською мовами

Веб-застосунок для сортування сміття Web application for garbage sorting

Мета бакалаврської роботи, завдання

Мета бакалаврської роботи: підвищення рівню еко-свідомості громадян.

План роботи:
1. Огляд підходів до розробки і впровадження веб-застосунків
2. Аналіз архітектурних рішень і вибір програмних засобів для реалізації веб-застосунків
3. Реалізація веб-застосунку для сортування сміття

Міронова В.Л., доц., к. т. н.: _____

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	22.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

АНОТАЦІЯ

До бакалаврської дипломної роботи Дулі Дарини Василівни на тему:
«Веб-застосунок для сортування сміття»

Дана дипломна робота присвячена проектуванню та розробленню динамічного односторінкового веб-застосунку для сортування сміття.

Для розроблення веб-застосунку було проведено ґрунтовне дослідження актуальних технологій та аналіз наявних аналогічних застосунків для виявлення сильних і слабких сторін. Стек технологій для розроблення веб-застосунку було сформовано з огляду на їх відповідність сучасним тенденціям, впливу на швидкодію та кросбраузерність застосунку.

Загальний обсяг роботи: 68 сторінки, 21 рисунок, 1 таблиця, 35 джерел, 15 додатків.

Ключові слова: сортування сміття, веб-застосунок, single page application, React.js, Material-UI, React-Redux.js, React-Leaflet.js.

ANNOTATION

To the bachelor's thesis of Dula Daryna Vasylivna on the topic:

"Web application for garbage sorting"

This thesis is devoted to the design and development of a dynamic one-page web application for sorting garbage.

To develop the web application, a thorough study of current technologies and analysis of existing similar applications to identify strengths and weaknesses was conducted. The stack of technologies for web application development was formed taking into account their compliance with modern trends, the impact on the speed and cross-browser application.

Total volume of work: 68 pages, 21 figures, 1 table, 35 sources, 15 appendices.

Keywords: garbage sorting, web application, single page application, React.js, Material-UI, React-Redux.js, React-Leaflet.js.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

ООН – Організація Об’єднаний Націй

ВООЗ – Всесвітня організація охорони здоров’я.

ЗМІ – засоби масової інформації.

DOM – document object model.

MVC – Model-View-Controller.

HTML – HyperText Markup Language.

CSS3 – Cascading Style Sheets.

GUI – graphical user interface.

SPA – single page application.

MPA – multi page application.

UX – user experience.

HTML – HyperText Markup Language.

СКБД – система керування базами даних.

VPC – Virtual Private Cloud.

ВСТУП

Актуальність дипломної роботи витікає із поглиблення екологічної кризи, експоненційному зростанні відходів діяльності людини, а особливо відходів пластикового походження, що загрожує ареалам тваринного світу.

В наш час веб-застосунки є ефективним джерелом інформації, особливо серед молодого покоління, що більше сприймає як достовірне джерело Інтернет, а не телебачення. Веб-застосунки дозволяють ділитись інформацію з широким колом зацікавлених людей, надавати майданчик для обміну інформацією.

Практична частина дипломної роботи представлятиме собою проектування та розроблення веб-застосунку для сортування сміття, що міститиме розділ з інформаційним матеріалом, блог для обміну дослідом сортування сміття, мапу пунктів прийому вторинної сировини, фото та відеоматеріали, що відповідають тематиці.

Мета бакалаврської роботи: розробка веб-застосунку, що сприятиме підвищенню рівня еко-свідомості громадян.

Завдання дипломної роботи:

- огляд підходів до розробки і впровадження веб-застосунків;
- аналіз архітектурних рішень і вибір програмних засобів для реалізації веб–застосунків;
- реалізація веб-застосунку для сортування сміття.

Об’єктом дипломної роботи є веб-застосунок для сортування сміття.

Предметом дипломної роботи є програмні засади розроблення веб-застосунку.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Огляд сучасного стану проблеми

Однією з найактуальніших проблем сьогодення є екологічна криза. Дана проблема пов’язана з різким збільшенням кількості населення, що призводить до надмірного використання вичерпних ресурсів і нераціонального використання відновлюваних ресурсів планети. У результаті цього, вже близько пів століття тисячі великих і малих організацій займаються вирішенням найгостріших екологічних проблем, намагаються зменшити вплив людини на навколишнє середовище і на різних рівнях запобігають розвитку кризових ситуацій.

За даними Організації Об’єднаних націй (надалі ООН) [1] і Всесвітньої організації охорони здоров’я (надалі ВООЗ) [2], у результаті неконтрольованого скидання відходів до водойм щороку до Світового океану потрапляє щонайменше 8 мільйонів тон пластикових відходів. Найкращим доказом масштабів проблеми є результати експедиції Віктора Вескова, яка виявила поліетиленове сміття у Безодні Челленджера, що є найглибшою точкою планети.

Статистика використання пластику, надана ВООЗ, є дуже песимістичною:

- щороку у світі використовується близько 500 мільярдів поліетиленових пакетів;
- за останнє десятиліття було виготовлено більше пластику ніж за минуле століття;
- 50% використаного пластику є одноразовим;
- щохвилини у світі купується 1 мільйон пластикових пляшок.

Згідно з оцінкою Ellen MacArthur Foundation [3] до 2040 року обсяг пластику, що потрапляє до океану, збільшиться втричі, а його запаси в океані – в чотири рази. Так, до 2050 року кількість пластику в океані може перевищити кількість риби.

Пластик знаходиться на всіх рівнях харчового ланцюга. Морські мешканці сприймають його за елемент звичної середи існування, заплутуються в ньому або

споживають його, що призводить до смерті. Як результат, його знаходять в сотнях тисяч морських мешканців, він потрапляє і до тарілок любителів морепродуктів.

Пластик є не лише фізичним забруднювачем. У 2004 році морським біологом університету Плімута у Великій Британії був введений термін «мікропластик» для позначення частинок синтетичних полімерів розміром до 5мм. Існує первинний мікропластик, що являє собою мікрофрагменти пластику, виготовлені спеціально для використання у засобах гігієни. Він потрапляє у Світовий океан і навколишнє середовище стічними водами. Іншим видом мікропластику є вторинний – результат розпаду пластикових відходів під впливом води і ультрафіолетових променів. Такі мікрочастинки потрапляють у середину представників флори та фауни, а через них і до організму людини. На сьогоднішній день проводиться багато досліджень, присвячених мікропластику у навколишньому середовищі і його впливу на живі організми. За дослідженнями ВООЗ [4] у очищеній водопровідній та бутильованій воді виявлений мікропластик, що викликає занепокоєння щодо його впливу на здоров'я людини.

У зв'язку з пандемією COVID-19 різко збільшилася кількість використаних одноразових засобів індивідуального захисту, саме масок та рукавичок. За прогнозами ООН [5] близько 75% відходів, пов'язаних з пандемією, потраплять на звалища або плаватимуть у морях. Окрім екологічної шкоди, існують ризики повторного інфікування при сортуванні, викиду великої кількості токсинів у навколишнє середовище у разі спалювання.

За даними Міністерства розвитку громад і територій [6] за 2019 рік в Україні (без урахування даних окупованих територій) утворилося понад 10 мільйонів тонн сміття, з яких лише 6,1% були утилізовані. Наразі лише 1462 населених пункти мають пункти роздільного прийому побутових відходів. Однак, більшість людей навіть за наявності такої можливості не сортують сміття. Основним підґрунтям даної проблеми в Україні є низький рівень фінансування і зацікавленості держави за рахунок чого Міністерство захисту довкілля та природніх ресурсів України не може забезпечити достатній рівень

інформованості населення. Таким чином, проблема отримання вичерпної і достовірної інформації лягає на плечі кожного українця.

Багато людей надають перевагу різноманітним засобам масової інформації (надалі ЗМІ), однак, щороку все більше людей обирають Інтернет як джерело пошуку інформації. За даними дослідження Дослідницького холдингу Factum Group Ukraine [7], проведеними у 2019 році, 71% населення України є регулярними користувачами мережі Інтернет.

На сьогоднішній день питання екології вимагає розгляду широкого спектру проблем і наслідків. З даної точки зору жоден засіб масової інформації не може витрачати стільки ресурсів задля повного інформування громадян лише з одного питання, адже в будь-якому випадку дана інформація не зацікавить всю аудиторію ЗМІ, що призведе до її зменшення і, відповідно, збитків. Створення тематичного веб-застосунку, в свою чергу, має ряд переваг, серед яких:

- можливість широкого висвітлення інформації в різних аспектах – сайт може містити ряд тематичних статей з повним розкриттям екологічної кризи України і планети в цілому, теоретичними і практичними порадами щодо можливості дій кожного українця вже сьогодні, додатковою інформацією щодо рекомендацій організації сортування сміття вдома, місць прийому вторинної сировини тощо;
- цілодобовий доступ до інформаційного ресурсу який забезпечується завдяки використанню всесвітньої мережі Інтернет;
- доступ до ресурсу з будь-якого пристрою, що має можливість підключення до Інтернету – веб-застосунок не вимагає конкретної операційної системи і наявності додаткового програмного забезпечення, окрім браузеру.

1.2 Огляд і аналіз існуючих програмних рішень

На сьогоднішній день багато веб-сайтів тією чи іншою мірою піднімають питання екологічної кризи загалом і сортування сміття зокрема. Варто зазначити, що більшість інформаційних ресурсів такого типу є сайтам певних організацій,

що мають за мету рекламу власної продукції, станції сортування вторинної сировини тощо. Відповідно, до таких джерел інформації варто ставитись критично.

Для огляду та аналізу існуючих веб-застосунків мною було обрано наступні ресурси.

«Поводження з відходами» [8] – інформаційних ресурс, присвячений проблемі сміття на Львівщині. Цей сайт містить достатню кількість тематичних статей про конкретні проблеми з різними видами відходів на території області. Також на даному ресурсі присутній свіжий огляд екологічних новий, в тому числі про будування сміттєпереробного комплексу, ряд тематичних статей-порад щодо сортування сміття. На сайті міститься декілька видів інтерактивних карт з пунктами прийому вторинної сировини.

«Україна без сміття» [9] – офіційний веб-сайт станції сортування сміття «NoWaste Recycling Station». Даний ресурс містить у великій кількості тематичні статті, присвячені станції сортування (інформація про те, які типи вторинної сировини приймають і переробляють, графік роботи станції, актуальні вакансії тощо), послугам (можна скористатися послугами кур’єра, надіслати пластик поштою, «зелений офіс» – сервіс для великих компаній, спрямований на те, щоб зробити їх офіси екологічними тощо), проектам компанії, матеріалам, розділ блогу (містить статті, присвячені роботі станції переробки, а також декілька під розділів, присвячених екології та переробці сміття) і магазин з власною продукцією.

RecycleMap [10] – веб-застосунок, що являє собою некомерційний проект, що має на меті популяризувати ідею сортування сміття і зробити її доступнішою для пересічної людини. Даний ресурс являє собою інтерактивну мапу з відміченими на ній пунктами прийому різних видів вторинної сировини.

Офіційний портал Міністерства захисту довкілля та природних ресурсів України [11] представляє собою інформаційний веб-ресурс, присвячений екології України і роботі Міністерства. Даний сайт містить достатню кількість різноманітних новинних статей, присвячених ряду екологічних проблем

України. Варто зазначити, що більшість статей, присвячених питанню сміття, сформовані не у вигляді висвітлення проблеми і просування ідей її вирішення, а у форматі звинувачень, що не викликають у читача прихильності і зацікавленості.

Believe.Earth [12] – інформаційний портал Бразильської соціально-екологічної організації Alana. Метою даного ресурсу є поширення соціально-екологічних ідей у маси і побудова партнерських стосунків по всьому світові. Даний сайт містить широкий вибір тематичних статей про екологічно чисту енергію, проблеми продуктів харчування і шляхи їх вирішення, планування та інновації у великих містах, вплив використання ресурсів планети, проблеми транспорту і надмірне використання пластику та шляхів подолання даної проблеми тощо.

На основі проведеного аналізу існуючих веб-застосунків, присвячених тематиці сортування сміття, я можу зробити висновки, що на сьогоднішній день в Україні дана тематика є актуальною, однак, багато таких ресурсів використовують екологічну проблему, як рекламний аспект власної продукції. Жодний з переглянутих Українських сайтів не розкриває проблематику екологічної кризи повною мірою, обмежуючись певними локальними даними. Часто на таких ресурсах використовується і агресивна реклама, спрямована на залякування аудиторії. З урахуванням, що даним сайтом скористається людина, яка хоче більше дізнатися про екологічні проблеми і шляхи їх рішення, звинувачення можуть призвести до негативного враження.

Чудовим прикладом поширенням ідей є бразильський сайт Believe.Earth, який розкриває певні проблеми і шляхи вирішення за таким принципом:

- широке висвітлення глобальної проблеми (дані статті спрямовані на демонстрацію об’ємів існуючих проблем і важливості змін для їх зменшення і подолання);

- висвітлення різноманітних рішень (загальне визначення рівнів і шляхів подолання конкретних глобальних проблем для визначення чіткої картини існуючих рішень);
- індивідуальні історії (статті, присвячені реальним людям і їх щоденному вкладові у вирішення тієї чи іншої глобальної проблеми людства, - допомагають користувачу усвідомити необхідність дій кожного і дізнатися більше про даний вид рішення).

1.3 Огляд підходів до розробки і впровадження веб-застосунків

Веб-застосунок – це прикладне програмне забезпечення, у якому клієнтом виступає браузер, а сервером – веб-сервер. У порівнянні з веб-сайтом, веб-застосунок є динамічним і виконує певні прикладні завдання.

На сьогоднішній день веб-застосунки дуже поширені і використовуються мільйонами людей щодня. Вони поєднують у собі досвід побудови user-friendly графічних інтерфейсів десктопних та мобільних застосунків з простотою доступу до них з будь-якого веб-браузеру. Самі ці переваги забезпечують значний попит на веб-застосунки серед широкого спектру підприємств.

Популярність напрямку веб розробки спричиняє розвиток і різноманіття існуючих технологій. Скористаємося статистикою використання веб-фреймворків за 2020 рік (рисунок. 1), наданою Stack Overflow[13].

Найбільш використовуваним веб-технологією стала jQuery [14] – бібліотека з відкритим кодом, яка є легким та гнучким інструментом для роботи з HTML, анімацією та обробкою подій. Дана бібліотека досить популярна серед початківців за рахунок своєї зрозумілості і відсутності високих вимог до знання JavaScript. jQuery дозволяє спрощувати процес роботи з DOM-елементами. На сьогоднішній день jQuery все рідше використовується на реальних проектах і залишається лише на підтримці старих версій браузерів.

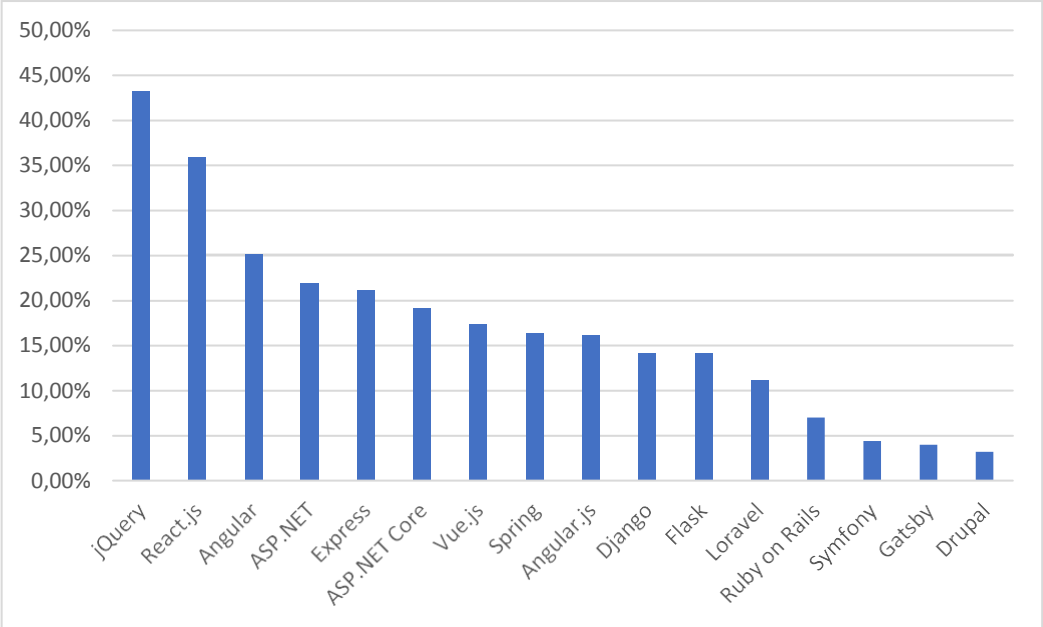


Рисунок 1 – Популярність веб-технологій

Свого часу Angular почав масове витіснення jQuery. Цей фреймворк був розроблений компанією Google і покликаний вирішити проблеми поєднання сучасних технологій і перевірених часом концепцій. Він пропонує багато функціональних рішень, які допомагають значно спрощувати структуру коду. Angular є унікальним своєю двосторонньою прив'язкою даних, що забезпечує синхронізацію між моделлю і представленням в реальному часі. Тобто будь-які зміни моделі відображаються на представленні і навпаки. Іншою важливою особливістю фреймворку є використання шаблону MVC (Model-View-Controller). Однак, як ми можемо бачити зі статистики, на сьогоднішній день, Angular активно замінюється бібліотекою React.js.

React.js – бібліотека, розроблена Facebook для виправлення проблем з підтримкою старого коду через постійне додавання нового функціоналу. Оскільки React є бібліотекою, то може бути підключеним до проекту частково, що дозволяє використовувати дану технологію на вже існуючих і функціонуючих проектах та не вимагає переписування усього коду, завдяки чому такі великі веб-застосунки як Facebook, можуть поступово впроваджувати її у своє ПЗ. React пропонує користувачам нову структуру, яка є швидше MV (model-view), ніж MVC. Це означає, що ми маємо модель і вигляд, вони можуть

взаємодіяти між собою, але React не визначає яким чином. Основною філософією даної технології є розбиття сторінки на компоненти, які можна повторно використовувати. Компоненти використовують JavaScript для взаємодії всередині себе та з іншими компонентами.

Для будь-якого веб-застосунку необхідна наявність серверної частини, яка буде забезпечувати його динамічність. Найпопулярнішими веб-фреймворками для написання бекенду на сьогоднішній день є ASP.NET і Express.js. Обидва фреймворки забезпечують можливість створення веб-серверів. Основною їх відмінністю є використання різних мов програмування: C# і JavaScript, що, за часту і зумовлює вибір розробників.

Відзначимо також два основні підходи [15] до реалізації веб-застосунків, а саме, односторінковий застосунок (single page application – SPA) та багатосторінковий застосунок (multi page application - MPA).

Односторінковий застосунок вважається більш сучасним підходом до розроблення програмного забезпечення. Такий підхід використовують інформаційно-технологічні гіганти галузі – Google, Facebook, Samsung тощо. SPA являє собою застосунок, що працює всередині браузера та не вимагає повторного (ререндеру) завантаження сторінки під час свого виконання.

На противагу цьому підходу використовується багатосторінковий підхід до розроблення програмного забезпечення. Такий підхід вважається більш класичним, однак він вимагає перезавантаження сторінки за кожного звернення або переходу. Цей підхід і досі вважається найкращим для компаній з великим об’ємом даних, хоча і потребує значно більше ресурсів.

Розглянемо переваги та недоліки обох підходів розроблення веб-застосунків.

Основним критерієм, з точки зору користувача, у веб-застосунку є швидкодія. SPA має перевагу в цьому напрямку. За рахунок відсутності необхідності повного ререндеру відображуваної сторінки односторонній застосунок відпрацьовує швидше за рахунок перезавантаження конкретних

компонентів. MPA працює повільніше через необхідність повного перезавантаження сторінки під час отримання нової інформації від застосунку.

Другим критерієм є залежність фронтенд та бекенд частин проекту. У SPA використовується принцип декомпозиції, що означає розподіл фронтенду та бекенду. Такі програми використовують розроблені програмістом API для звернення до сервера, отримання та відображення даних. MPA використовує більшу залежність між фронтендом та бекендом. В даному підході ці частини проекту є більш монолітними.

Третім критерієм є пошукова оптимізація. Даний критерій в першу чергу цінний для компаній та проектів, спрямованих на велике охоплення аудиторії, швидке просування та комерціалізацію. Пошукова оптимізація є слабкою стороною SPA. Не зважаючи на прогрес у галузі інформаційних технологій більшість пошукових систем і досі не розпізнають JavaScript, який є основою SPA. Оскільки пошукові системи завантажують та ранжують HTML-файли веб-сторінок даний критерій є перевагою MPA, що дозволяє краще позиціонувати проекти розроблені саме цим підходом. В тому числі ця перевага використовує можливість прописувати метатеги на кожній сторінці, що позитивно впливає на рейтинг пошукових систем.

Четвертим критерієм є досвід користувача (user experience – UX). Досвід користувача поєднує в собі поняття зручності та інтуїтивної зрозумілості інтерфейсу та адаптивності. SPA вважаються більш ефективними для мобільного досвіду. Тобто інтерфейс SPA є більш адаптивним для телефонів та планшетів. В тому числі варто відзначити меншу кількість даних, яку необхідно передати. MPA залишається фаворитом для настільних персональних комп’ютерів, однак з розширенням можливостей односторінкових застосунків це вже не можна вважати аксіомою.

П’ятим критерієм є безпека. Само собою зрозуміло, що чим більше компонентів у проекту тим більше необхідно докладати зусиль для захисту. MPA вимагає проектування та розроблення контуру захисту для убезпечення кожної сторінки та бібліотеки проекту. SPA простіше захищати, для цього необхідно

убезпечити точки звернення до даних у застосунку. Проте це ж є і слабким місцем SPA оскільки JavaScript більш схильний до хакерських атак і потребує ретельного написання.

Шостим критерієм є процес розробки. Перевагою SPA є внутрішній код, який можна багаторазово використовувати. Завдяки цьому можна знизити час розробки та кількість дублюючого коду. Також перевагою є декомпозиція фронтенду та бекенду у SPA, в той час як у MPA ці частини проекту маю чітко розроблюватись паралельно одне одному.

Сьомий критерії – залежність від JavaScript. Завдяки своїй гегемонії JavaScript став «кров'ю» та «м'язами», які забезпечують рух SPA. Це призводить до необхідності підтримки великої кількості додаткових бібліотек та певних обмежень на старих версіях браузерів, що не підтримують JavaScript. Цей критерій є суб'єктивним, адже в останні роки Python та Blazor від Microsoft активно пропонують шляхи максимального заміщення JavaScript.

1.4 Постановка задачі

Метою написання кваліфікаційної роботи бакалавра є розробка веб-застосунку для підвищення рівню еко-свідомості громадян. Даний веб-застосунок має являти собою SPA (single page application) з функціоналом для надання користувачам вичерпної та достовірної інформації щодо екологічної кризи, необхідності сортування і переробки сміття, місцезнаходженнях і типах пунктів прийому вторинної сировини у місті Києві.

Даний веб-застосунок має містити:

- інформацію про мету сайту;
- мотиваційну частину, присвячену розкриттю проблематики сортування сміття;
- інформацію щодо світового досвіду сортування сміття країн Європи заради демонстрації ефективних рішень, що застосовуються роками і приносять свої результати;

- інтерактивну карту пунктів прийому вторинної сировини у місті Києві з можливістю перегляду додаткової інформації про кожен пункт прийому (назва, адреса, опис, типи вторинної сировини, що приймається, робочі години тощо);
- передбачити можливість фільтрувати пункти прийому за типами вторинної сировини;
- інформацію про кожен тип вторинної сировини (що переробляється, як тощо);
- слайдер з мотиваційними баннерами;
- блог сайту з рекомендаціями людей, що сортують сміття щодня;
- форму зворотнього зв’язку з можливістю додати власну історію до блогу сайту.

Для забезпечення кросбраузерності веб-застосунок має коректно відображатися в останніх версіях сайту, а саме:

- IE (Internet Explorer): 11+;
- Google Chrome: 58+;
- Edge: 13+;
- Safari: 9+;
- Opera: 55+;
- Firefox: 54+.

Висновки до розділу 1

При написанні розділу мною було проведено ґрунтовний аналіз екологічної ситуації в Україні та світі, в контексті проблем накопичення відходів загалом, та твердих пластикових відходів зокрема. Було розглянуто статистичні дані Організації Об’єднаних Націй та Всесвітньої Організації Охорони Здоров’я, а також дослідницькі матеріали поважних агенцій, таких як – Ellen MacArthur Foundation. З огляду на проведену роботу, я вважаю свою тему дипломної цілком обґрунтованою в контексті отриманої інформації.

Наступним кроком було розглянуто ряд сайтів з інформування суспільства про проблему екологічної кризи та методів персональної боротьби у вигляді сортування власних побутових відходів.

Для глибшого розуміння наявних підходів до сучасної побудови веб-застосунків мною було проведено аналіз реалізації Single page application та Multi page application, порівняно їх переваги та недоліки. Що до технологій було проведено огляд найбільш затребуваних та популярних фреймворків та бібліотек, що представлені сьогодні на ринку веб-розробки та використовувались у продуктах огляд яких було проведено. Це технології фронтенд та бекенд частини проекту, а також баз даних, що зберігаюсь масиви інформації.

На підставі отриманих знань, інформації про проблематику та огляду технологій було прийнято рішення про формулювання постановки задачі для подальшої роботи над застосунком.

Підсумовуючи зазначу, що з огляду на технології та сучасні тренди у розробці фінальний варіант постановки задачі має вигляд веб-застосунку, що базується на принципі Single page application, визначено його функціонал та принцип кросбраузерності для можливості безперешкодного доступу з будь-якого пристрою.

РОЗДІЛ 2. АНАЛІЗ ТА ОБҐРУНТУВАННЯ СТЕКУ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБЛЕННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ СОРТУВАННЯ СМІТТЯ

Відповідно до постановки завдання необхідно провести аналіз та обґрунтувати вибір технологій та інструментів для розробки програмного забезпечення з урахуванням специфіки веб-проектування та інтеграції з хмарним середовищем. Оскільки в сучасній веб-розробці сайти давно перестали нести статично відображувану інформацію в проекті буде передбачено клієнтську частину (фронтенд), серверну частину (бекенд) та реалізовано базу даних із подальшим розміщенням на хмарному ресурсі.

2.1 Аналіз інструментів та засобів розробки фронтенд частини веб-застосунку

Реалізація фронтенду на проекті потребує обов’язкового використання класичних інструментів веб-розробки, таких як мова гіпертекстової розмітки HTML5 (HyperText Markup Language) [16] та каскадної таблиці стилів CSS3 (Cascading Style Sheets) [17].

Відповідно до сучасних тенденцій переважна більшість веб-застосунків являють собою SPA [18], тобто односторінковий застосунок, контент якого динамічно змінюється завдяки використанню скриптової мови програмування JavaScript. Порівнюючи зі звичними «багатосторінковими» сайтами, SPA надає переваги у швидкості, адже не потребує перезавантаження всієї сторінки, а змінює лише необхідні блоки сайту без зайвих затримок. На сьогоднішній день існує широкий вибір технологій, що спрощують розробку односторінкових веб-застосунків.

У процесі розробки веб-застосунку для сортування сміття будемо використовувати React.js [19]. Головна філософія даної бібліотеки полягає у розбитті GUI (graphical user interface) на окремі функціональні компоненти. Так, на рисунку 2 ми можемо спостерігати логіку розбиття сторінки на окремі компоненти, що можуть в подальшому бути перевикористаними.

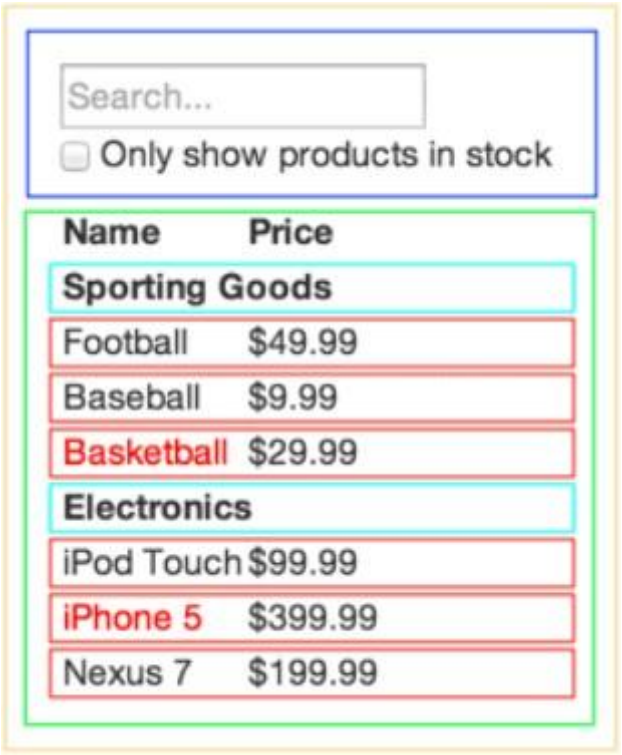


Рисунок 2 – Розбиття на компоненти

Завдяки використанню композиції компоненти застосунку можуть «вкладатися» один у одного, створюючи кінцевий вигляд користувацького компоненту. При використанні React.js, фактично, ми маємо лише одну HTML-сторінку – index.html, яка містить єдиний div-блок, у який буде динамічно підвантажено фінальний вигляд необхідного компоненту.

SPA передбачає єдину фізичку сторінку, однак, для висвітлення великого об’єму інформації, ми маємо необхідність розбиття веб-застосунку на логічні сторінки і створення функціоналу для зручної навігації між ними. Для цього React пропонує використання бібліотеки react-router-dom.js [20], яка відповідає за структури шляхів сайту і їх відповідність конкретним компонентам. Завдяки використанню цієї бібліотеки ми маємо можливість не лише динамічно перемикатися між різними сторінками-компонентами веб-застосунку, але й переходити між логічними сторінками, використовуючи URL-адреси, вбудовувати внутрішні навігаційні посилання у застосунок.

При роботі з елементами в React.js, на відміну від багатьох бібліотек і фреймворків, не розділяється логіка виводу даних і логіка користувацького

інтерфейсу. Це забезпечується за рахунок використання спеціального синтаксичного цукру JSX [21]. Так, фактично, при розробці будь-якого компоненту в одному файлі буде міститися JavaScipt-логіка виводу даних і JSX-логіка побудови графічного інтерфейсу.

```
export const MainTitle = ({ icon, title }) => {
  const classes = useStyles();
  return (
    <Box
      className={classes.title}
      display="flex"
      flexDirection="row"
      justifyContent="center"
      alignItems="center"
    >
      {icon && <img src={icon} alt="" className={classes.img} />}
      <Typography variant="h4" align="center">
        {title}
      </Typography>
    </Box>
  );
};
```

Рисунок 3 – приклад використання JSX

Хоча JSX, фактично, є JavaScript, але базується на стандартному HTML5 і відповідає за візуальне відображення функціоналу застосунку, стилізованого шляхом використання CSS. Під час розробки будь-якого клієнто-орієнтованого застосунку постає питання стилізації і дизайну. На сьогоднішній день одним з найпопулярніших підходів до дизайну є матеріальний дизайн [22], запропонований компанією Google для спрощення стилізації власних застосунків. Даний підхід базується на простоті і фізичних характеристиках речей: фактурі, текстурі, світлі, тіні тощо. Для написання нашого веб-застосунку використаємо матеріальну графічну бібліотеку для React – Material-UI [23]. Дана бібліотека надає можливість створення власних тем застосунків і широкий вибір готових стилізованих компонентів, що в рази спрощує процес проектування графічного інтерфейсу нашого застосунку і безпосередньої розробки власних компонентів.

Стандартні HTML-елементи форм є самостійно керованими, тобто оновлюють власний стан відразу при будь-яких змінах у елементі. React-елементи форми [24] не можуть самостійно контролювати власний стан і потребують “батьківського” контролю на рівні компоненту. Для спрощення логіки роботи з формами при написанні веб-застосунку для сортування сміття скористаємося бібліотеками Formik.js [25] та Yup.js [26]. Бібліотека Formik.js спростить логіку роботи елементів форми і форми в цілому. За допомогою Yup.js ми створимо власну схему об’єкту форми, за допомогою якої зможемо перевіряти введені користувачами дні на валідність.

Оскільки технічним завданням на розробку передбачено впровадження інтерактивної карти пунктів прийому вторинної сировини, під час розробки нами буде використано JavaScript-бібліотеку Leaflet.js [27] і дочірню бібліотеку React-Leaflet.js [28]. Дані бібліотеки пропонують швидкі і зручні рішення для створення власних інтерактивних карт з використанням відкритої бази місцевості OpenStreetMap [29]. Також скористаємося бібліотекою react-leaflet-control.js [30] для додавання власних елементів до інтерактивної карти.

Для роботи з глобальним станом веб-застосунку скористаємося принципом Redux і бібліотекою React-Redux.js [31]. Основна логіка роботи з Redux (рисунок 4) полягає у створенні власного сховища (Store) для збереження необхідних станів застосунку, що дає змогу отримувати доступ до глобальних станів з будь-якого UI-компонента. Зміна глобального стану відбувається при обробці певної події (натиск на кнопку, завантаження сторінки, зміна введених даних тощо), завдяки виклику спеціальної функції-дії, що є певною формою звернення до необхідного ред’юсера (зазвичай глобальну логіку роботи зі станами розділяють на декілька окремих ред’юсерів для спрощення структури застосунку). У випадку необхідності надсилання додаткових запитів на бекенд частину застосунку додаються додаткові прошарки (middlewares), після відпрацювання яких відбувається необхідна зміна станів.

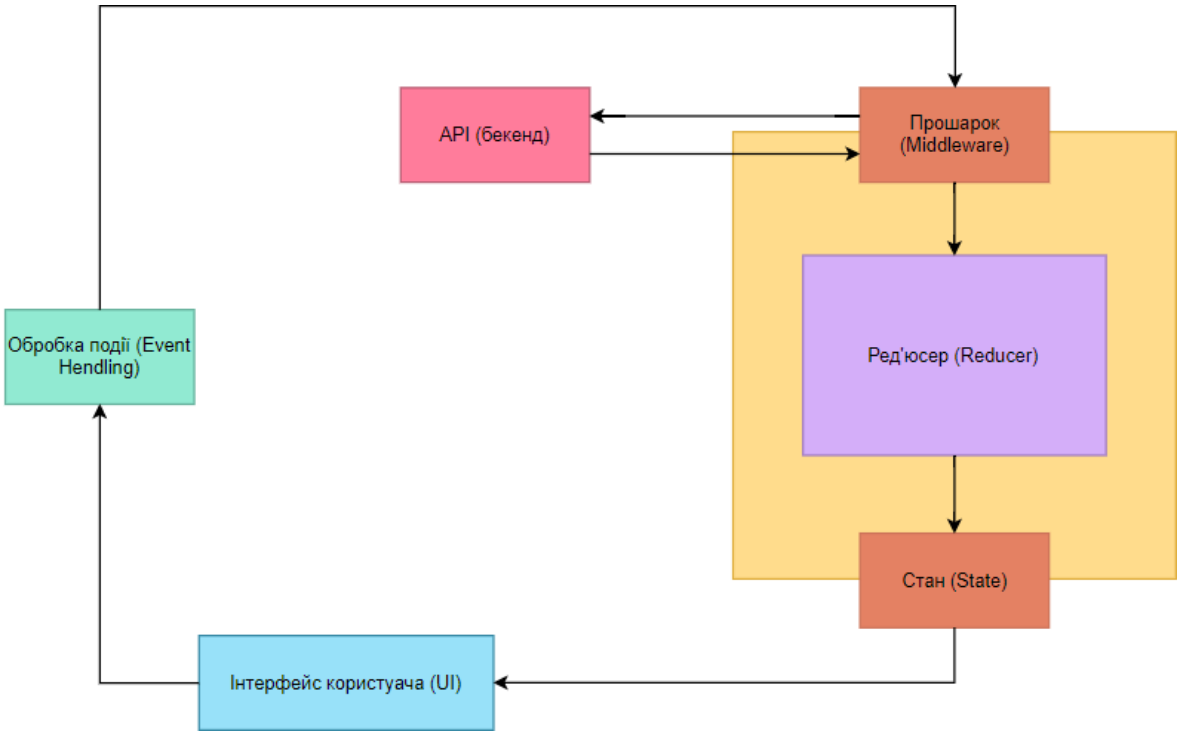


Рисунок 4 – Схема роботи з Redux

2.2 Аналіз інструментів та засобів розробки бекенд частини веб-застосунку

Розробка сучасного веб-застосунку вимагає наявності динамічних даних, які знаходяться на сервері. В першу чергу для реалізації такого роду завдання необхідно обрати базу даних.

На сьогоднішній день існує велика кількість різноманітних баз даних, кожна з яких має власні переваги та недоліки. Оскільки мовою розробки нашого веб-застосунку є JavaScript, а структура майбутньої БД не вимагає створення ряду пов’язаних між собою структур, за базу даних нами була обрана MogoDB [32] – сучасна документно-орієнтована система керування базами даних (далі СКБД). Дана СКБД надає розробникам переваги зберігання даних у звичних форматах об’єктів – а саме у форматі, схожому на JSON (JavaScript Object Notation). Завдяки цьому MongoDB підтримує використання вкладених масивів та об’єктів і дозволяє працювати з гнучкими та динамічними схемами даних.

MongoDB має власну мову запитів, що дозволяє виконувати різноманітні пошукові запити.

MongoDB має переваги для використання у проектах, що можуть динамічно змінювати свою бізнес-модель та/або структуру.

Першою перевагою MongoDB є відсутність чіткої схеми. Це проявляється в тому, що в якості сутності бази даних використовується не таблиця з реляційними зв’язками, а набір різноструктурних документів, що об’єднані виключно за типом самого файлу. Така структура вимагає більшої уваги від свого проектувальника, однак напрочуд легко піддається оперативним змінам без глобального переосмислення структури бази даних.

Другою перевагою є легке горизонтальне масштабування бази даних. Таким чином за необхідністю MongoDB може легко перерозподілити колекції даних в залежності від наданого простору розгортання. Це стало можливим завдяки легкому налаштуванню балансувальника, що є нативною частиною MongoDB на противагу реляційним базам, де такі балансувальники зазвичай є пропрієтарною складовою програмного продукту конкретної компанії, тобто такий підхід не буде універсальним.

Третьою перевагою є багатий арсенал агрегацій, що стає можливим завдяки власній мові написання запитів. MongoDB дозволяє проводити групування зі складними умовами, map-reduce та перетворення документа в колекції на льоту. Власна мова MongoDB в порівнянні з SQL більш громіздка однак привносить логічну структурованість у запити.

Таблиця 1 – Порівняння написання запитів на SQL та MongoDB [33]

SQL	MongoDB

SELECT cust_id,
SUM(li.qty) as qty
FROM orders o,
order_lineitem li
	db.orders.aggregate([
	{ $unwind: "$items" },
	{
	$group: {

WHERE li.order_id = o.id GROUP BY cust_id	_id: "$cust_id", qty: { $sum: "$items.qty" } } }])

Четвертою перевагою є деномінація. Фактично деномінація є доповненням до відсутності схеми. Тобто архітектору не важливо дотримуватись чіткої структури збереження даних в документах, інформацію можна зберігати в тому вигляді, в якому зручніше з нею працювати. Ця перевага вимагає від архітектора послідовності у проектуванні щоб убезпечитись від надмірного дублювання інформації.

Останньою перевагою використання MongoDB є проста логіка індексації сутностей. Завдяки чому відпадає необхідність повного сканування бази даних в процесі роботи, що зменшує час обробки запиту та звернення до необхідних даних.

З огляду на концепцію веб-розробки та необхідності доступу до даних з будь-якого місця та пристрою необхідно організувати ефективне збереження бази даних. Оскільки при роботі з базами даних масив інформації може бути надзвичайно об’ємним нам необхідно скористатись перевагами хмарного середовища.

MongoDB пропонує власну службу хмарних баз даних для застосунків Atlas. Завдяки цій системі, в тому числі, можна забезпечити перевагу ефективної горизонтальної масштабованості бази даних. Сам Atlas можна розгорнути на Google Cloud, Amazon Web Services або Microsoft Azure.

MongoDB Atlas виконує нативний контроль безпеки та конфіденційності даних шляхом трирівневої системи захисту:

1. Ізоляція – кластери MongoDB Atlas розгортаються у Virtual Private Cloud (віртуальній приватній хмарі) із спеціальними брандмауерами. Доступ до VPC надається за спеціальним списком IP-адрес.

2. Авторизація – передбачає попереднє створення користувачів або їх груп з певними ролями, відповідно до яких в подальшому можуть бути проведені маніпуляції з базою, доступ до даних та операції з ними.
3. Наскрізне шифрування – весь мережевий трафік шифрується на транспортному рівня.

Для написання серверної частини веб-застосунку сортування сміття нами був обраний веб-фреймворк Express.js. Даний фреймворк є базовим для платформи Node.js, яка дозволяє компілювати JavaScript код поза середовищем браузера. Express.js використовується для написання невеликої кількості базового функціоналу серверу, а саме:

- Написання обробників для різноманітних HTTP та HTTPS запитів;
- Інтеграція з механізмами рендерингу вигляду ("view") для генерації відповідей сервера, шляхом вставлення необхідних даних у відповідні шаблони;
- Встановлення загальних налаштувань сервера, таких як порт для підключення, маршрутизація шляхів, розміщення шаблонів тощо;
- Створення проміжних шарів (middleware) для додаткової обробки запитів.

Сам по собі Express.js є достатньо мінімалістичним і легким у використанні, він користується великою популярністю і має вичерпну документацію, що в рази спрощує процес написання серверу. У зв’язку з простотою даного фреймворку, виникає необхідність інтеграції додаткових бібліотек і фреймворків для розширення функціоналу.

Для роботи з базою даних скористаємося npm пакетом Mongoose.js. За допомогою даного пакету ми маємо можливість взаємодіяти з локальною або віддаленою базою даних MongoDB. Першочергово, для безпосередньої роботи з сутностями необхідно створити чітку схему даних колекції. Надалі з утвореної схеми створюємо спеціальну Mongoose-модель – клас, за допомогою якого ми будемо працювати з сутностями БД. В результаті ми отримуємо функціонал для отримання, запису або видалення даних.

Висновки до розділу 2

У цьому розділі, відповідно до поставленого технічного завдання, було проведено обґрунтування стеку технологій для використання на фронтенд та бекенд частині проекту, бази даних та хмарного середовища для її зберігання.

З отриманої в попередньому розділі інформації, мною було прийнято рішення використовувати мову гіпертекстової розмітки HTML для основного каркасу, як універсальний інструмент, каскадні таблиці стилів CSS для стилізації та мову програмування JavaScript для фронтенд та бекенд частин проекту.

Також, для ефективнішої роботи із стилізації будемо використовувати Material-UI, бібліотеку для розробки інтерфейсу користувача, що представляє собою «матеріальний» (об’ємний) дизайн та є чудовим доповненням React.js. Для спрощення роботи з формою скористаємося бібліотеками Formik.js та Yup.js.

Одним з найбільш містких за функціоналом елементів проекту є мапа. Для її створення використаємо дочірню бібліотеку Leaflet.js – React-Leaflet.js для роботи з картою.

Для бекенд частини, що відповідатиме за механізм обробки запитів, використаємо фреймворк Express.js.

Великий об’єм інформації, водночас, є достатньо однотиптим, тому було прийнято рішення використати нереляційну базу даних MongoDB та пакетом Mongoose.js для роботи з нею.

Великий об’єм інформації підіймає питання місця збереження інформації для роботи та можливого доповнення чи змін у них. Для гнучкості проекту було прийнято рішення використати одну з провідних хмарних платформ Google Cloud, на якій розгорнуто систему керування базами даних MongoDB Atlas.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-ЗАСТОСУНКУ СОРТУВАННЯ СМІТТЯ

3.1 Проектування веб-застосунку для сортування сміття

Одним з основних етапів проектування будь-якого веб-застосунку є розробка дизайну. На сьогоднішній день веб-дизайн є окремою галуззю веб розробки, що відповідає за зручність інтерфейсу з точки зору досвіду користувача (UX), розробляє макети для адаптивності застосунків та впроваджує мету інтуїтивності використання програмного продукту.

При розробці концепції дизайну веб-застосунку для сортування сміття за допомогою веб-ресурсу Coolors [34] була підібрана нейтральна кольорова гама (рисунок 5).

Рисунок 5 – Кольорова гама веб-застосунку для сортування сміття

Відповідно до визначеної мети веб-застосунку було обрано слоган “Carpe Diem”, що перекладається з латинської як “лови день” і є крилатим висловом, що означає “використовуй мить, яку маєш сьогодні”. У контексті глобальної екологічної кризи даний слоган підкреслює необхідність дій кожної людини вже сьогодні.

Логотипи веб-застосунку для сортування сміття було створено за допомогою онлайн платформи графічного дизайну Canva [35] на основі обраної кольорової гами та слогану (рисунок 6).

Рисунок 6 – Логотипи веб-застосунку для сортування сміття

Перед початком розробки необхідно сформулювати базове уявлення UI/UX каркасу веб-застосунку. Для цієї мети в проектуванні програмного забезпечення існує поняття wireframe (дослівно “каркас”) або більш розповсюджене blueprint (чернетка). Відповідно до постановки задачі розділимо функціонал веб-застосунку на логічні одиниці - сторінки:

1. Головна сторінка сайту (додаток А) - має містити три компоненти, що є спільними для всіх сторінок сайту, а саме: хедер (header) з навігаційними посиланнями на інші сторінки, слайдер зображень у вигляді банерів під навігаційним меню, футер (footer) сайту, що містить логотип веб-застосунку додаткові навігаційні посилання сайтом і інформацію про сайт. Головна сторінка містить такі унікальні блоки:
   - блок фото-посилань - кожне фото відповідає відповідній сторінці сайту, при натиску відбувається перехід за посиланням;
   - п’ять блоків-статей з коротким описом кожної сторінки сайту.

- “Поділитися своєю історією” є останнім блоком сайту і являє собою форму зворотнього зв’язку, за допомогою якої користувач може залишити власну історію на сторінці рекомендацій.

2. Екологічна криза (додаток Б) є мотиваційно-інформаційним розділом веб-застосунку, присвячений розкриттю екологічної проблеми в Україні та світі. Окрім обов’язкових компонентів дана сторінка має містити такі блоки:
   - сім інформаційних блоків у форматі міні-статей;
   - два блоки з відео-матеріалами.
3. Культура сортування (додаток В) – інформаційна сторінка сайту, присвячена розкриттю досвіду сортування сміття країн Європи заради демонстрації ефективних рішень, що застосовуються роками і приносять свої результати. Окрім обов’язкових компонентів дана сторінка має містити:
   - шість інформаційних блоків у форматі міні-статей;
   - блок з відеоматеріалом;
   - блок з фото матеріалом.
4. Техніки сортування (додаток Г) – інформаційна сторінка сайту, що має на меті надання користувачу вичерпної та достовірної інформації щодо технік сортування. Окрім базових блоків сторінка має містити:
   - інформаційно-мотиваційний блок з міні-статтею та зображенням-посиланням для завантаження додаткового файлу у форматі PDF з порадами та рекомендаціями як стати екологічно свідомим;
   - блок-список (checklist) з порадами як почати сортувати сміття вже сьогодні;
   - блок з вкладками (tabs), кожна з яких відповідає за певний тип матеріалу вторинної сировини. Передбачено наявність таких вкладок (tab): пластик, папір, скло, метал, органіка та інші матеріали. При обранні однієї з вкладок з’являється наступний блок сайту, що

містить інформацію про перелік типів вторинної сировини, що піддається переробці, не піддається тощо.

5. Карта (додаток Д) має на меті надання користувачу достовірної інформації щодо наявних пунктів прийому вторинної сировини на території міста Києва. Окрім базових компонентів дана сторінка має містити: інтерактивну карту міста Києва та області з відміченими на ній пунктами прийому вторинної сировини. У верхньому правому кутку карти має знаходитися кнопка “Фільтер”, при натиску на яку має відкриватися бічне меню з переліком усіх видів вторинної сировини для фільтрації міток на карті. При натиску на мітку повинно відкриватися popup з детальною інформацією про пункт прийому вторинної сировини (назва, адреса, опис, робочі години, перелік матеріалів до прийому), при натиску на іконку вторинної сировини повинно відкриватися нова сторінка (додаток Е) з детальною інформацією щодо вторинної сировини (що саме приймається, чому варто переробляти, як переробляється). Дана сторінка також має кнопку для повернення до перегляду карти і додатковий список посилань на інші матеріали.
6. Рекомендації (додаток Ж) – сторінка блогу сайту з історіями користувачів. Окрім базових блоків дана сторінка має містити:
   - Блоки з історіями – картки історій користувачів, що містять іконку з першою літерою імені користувача, ім’я та електронну адресу, зображення та початок історії користувача. При натиску на блок відкривається сторінка історії, що являє собою повний огляд історії (додаток И).
   - “Поділитися власною історією” – блок з формою зворотнього зв’язку, завдяки якій користувач має можливість залишити власну історію на сторінці блогу сайту. Форма містить такі обов’язкові до заповнення поля: ім’я, адреса електронної пошти та історія, а також не обов’язкове поле – фото. Адреса електронної пошти є

обов’язковим полем до заповнення для забезпечення зворотного зв’язку. Завдяки цьому в подальшому адміністрація сайту матиме можливість за необхідності зв’язатися з користувачем для уточнень, внесення змін до історії тощо.

Відповідно до завдання на розробку, ми маємо необхідність у збереженні певних даних на стороні серверу. Збереження потребують такі структури даних як інформація про пункти прийому вторинної сировини та історії користувачів.

```
const placeSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  categories: [
    {
      type: String,
    },
  ],
  description: String,
  address: String,
  phone: String,
  workingHours: String,
  coordinates: [
    {
      type: Number,
      required: true,
    },
  ],
});
```

Рисунок 7 – Схема даних про місце прийому вторинної сировини

Для представлення даних про пункт прийому вторинної сировини спроектуємо наступну схему (рисунок 7) з такими полями як:

– name – ім’я (назва) пункту прийому вторинної сировини, представлене у вигляді обов’язково поля типу строки (String);
– categories – перелік категорії матеріалів та сировин, які приймаються у даному пункті, представлені у вигляді масиву строк;

– description – опис пункту прийому вторинної сировини (будь-яка додаткова інформація про пункт прийому), поле питу строка;
– address - адреса пункту прийому вторинної сировини, строка;
– phone – контактний номер телефону пункту прийому вторинної сировини;
– workingHours – додаткове поле, що містить дані про робочі години пункту прийому вторинної сировини, представлене у вигляді строки;
– coordinates – географічні координати місце розташування пункту прийому вторинної сировини на карті у форматі масиву чисел.

Відповідно до технічного завдання на розробку на серверній (бекенд) частині застосунку має бути GET запит для отримання всіх точок прийому вторинної сировини, GET запит для отримання пунктів прийомів конкретної сировини. Також варто додати POST запити для додавання однієї або групи (масиву) точок прийому вторинної сировини для спрощення процесу занесення інформації до бази даних.

```
const storySchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
  },
  story: {
    type: String,
    required: true,
  },
  image: { data: Buffer, contentType: String },
});
```

Рисунок 8 – Схема даних історії користувача

Для представлення даних блогу (історій користувачів) спроєктуємо схему (рисунок 8) з такими полями:

- name – ім’я користувача, обов’язкове поле у форматі строки;
- email – адреса електронної пошти користувача, обов’язкове поле у форматі строки;
- story – історія користувача, обов’язкове поле формату строки;
- image – додаткове необов’язкове поле для збереження зображення до історії.

Відповідно до поставленого завдання для розробки застосунку для сортування сміття на серверній частині веб-застосунку необхідно передбачити такі запити:

- POST запит для додавання користувачем нової історії;
- GET запит на отримання всіх існуючих історій для їх подальшого відображення на сторінці блогу (“Рекомендації”) сайту;
- GET запит на отримання конкретної статті за її ідентифікатором (для відображення на сторінці перегляду статті.

3.2 Розробка веб-застосунку для сортування сміття

На першому етапі розробки, відповідно до визначених сторінок сайту, розробимо роутинг веб-застосунку (рисунок 9), де кожному шляху сайту буде відповідати певний компонент графічного інтерфейсу користувача. Завдяки створеному компоненту навігації react-router-dom самостійно буде визначати який з запропонованих компонентів сайту відображати в залежності від шляху, а перехід між сторінками сайту буде відбуватися за допомогою верхнього меню сайту.

```
const routes = Object.freeze({
  home: '/',
  crisis: '/crisis',
  culture: '/culture',
  techniques: '/techniques',
  map: '/map',
  mapInfo: '/map/:category',
  recommendations: '/recommendations',
  story: '/recommendations/:id',
});
```

```
<Switch>
  <Route exact component={Home} path={routes.home} />
  <Route exact component={Crisis} path={routes.crisis} />
  <Route exact component={Culture} path={routes.culture} />
  <Route exact component={Techniques} path={routes.techniques} />
  <Route exact component={Map} path={routes.map} />
  <Route exact component={MapInfo} path={routes.mapInfo} />
  <Route exact component={Recommendations} path={routes.recommendations} />
  <Route exact component={Story} path={routes.story} />
  <Redirect to={routes.home} />
</Switch>
```

Рисунок 9 – Роутинг веб-застосунку

Відповідно до визначеного роутингу маємо наступну UML-діаграму компонентів нашого веб-застосунку для сортування сміття (рисунок 10). Ми маємо шість головних компонентів, кожен з яких має спільні компоненти верхньої (header) та нижньої (footer) навігації сайту і є пов’язаними між собою за допомогою внутрішніх посилань (Link) сайту. Кожна сторінка має слайдер зображень і власні унікальні блоки зі статичними та динамічними даними.

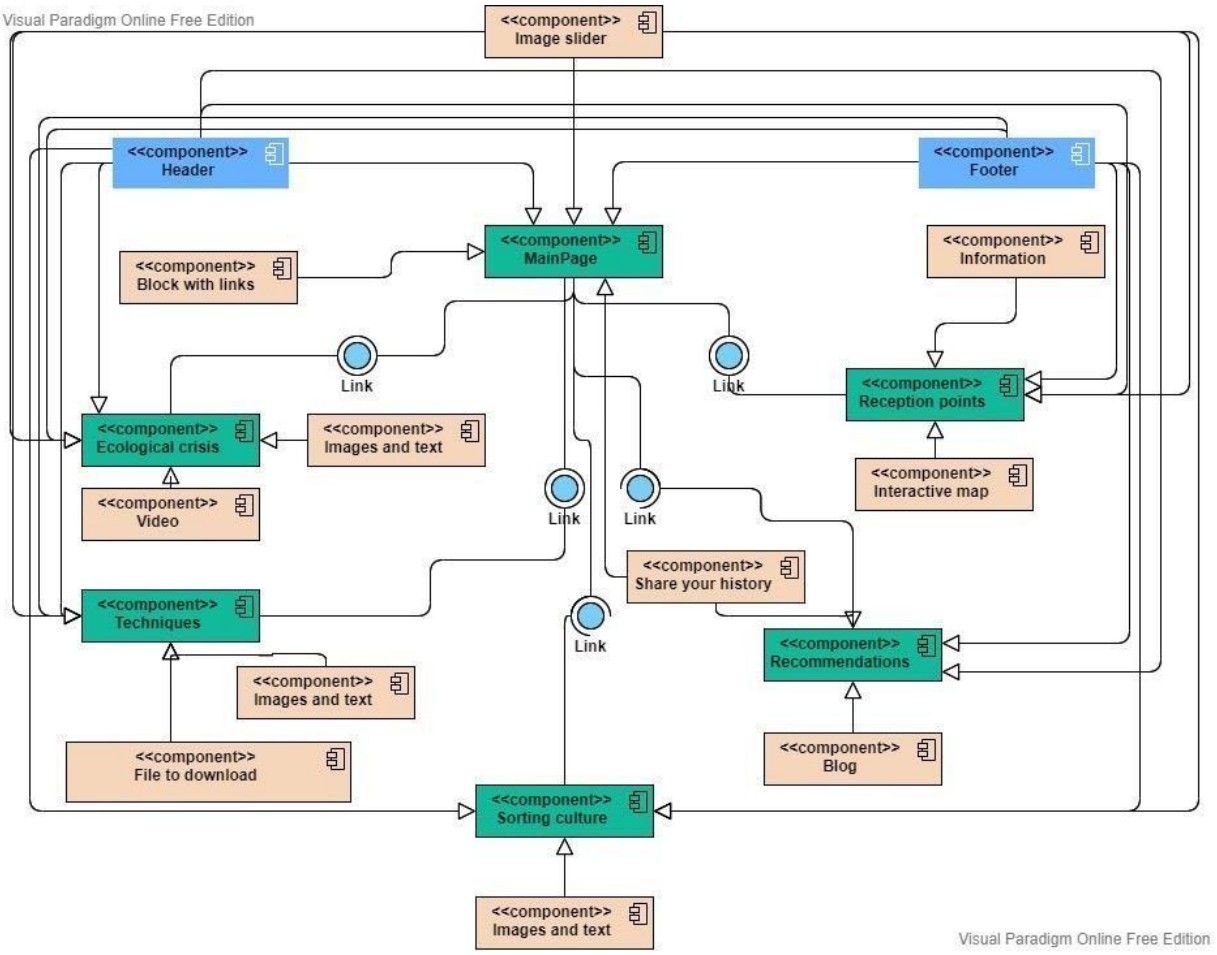


Рисунок 10 – UML-діаграма компонентів застосунку

Найважливішим глобальним компонентом розроблюваного веб-застосунку для сортування сміття є верхнє навігаційне меню сайту (додаток К). Воно являє собою горизонтальний елемент з посиланнями на всі логічні сторінки сайту. Основною складністю розробки даного компоненту є забезпечення адаптивності для різних розмірів екранів шляхом хования (hide) більшості посилань і винесення їх на додаткове бічне меню за допомогою компоненту Material-UI – Drawer.

З огляду на розроблені каркаси було створено ряд додаткових компонентів для відображення блоків статей та їх складових:

– статті з заголовком і іконкою – Article;
– статті, що має додатковий фото або відео контент, який знаходиться праворуч або ліворуч від безпосереднього заголовку і тексту – TwoPartArticle;
– набору двох або трьох міні-статей – Articles.

Під час розробки фронтенд частини веб-застосунку за допомогою бібліотеки react-responsive-carousel.js було створено адаптивний слайдер зображень, що є одним з головних мотиваційних блоків сайту. Даний компонент являє собою карусель банерів, в якому зображення змінюють одне одне шляхом натиску на бічні кнопки або вибору зображення з черги внизу блоку.

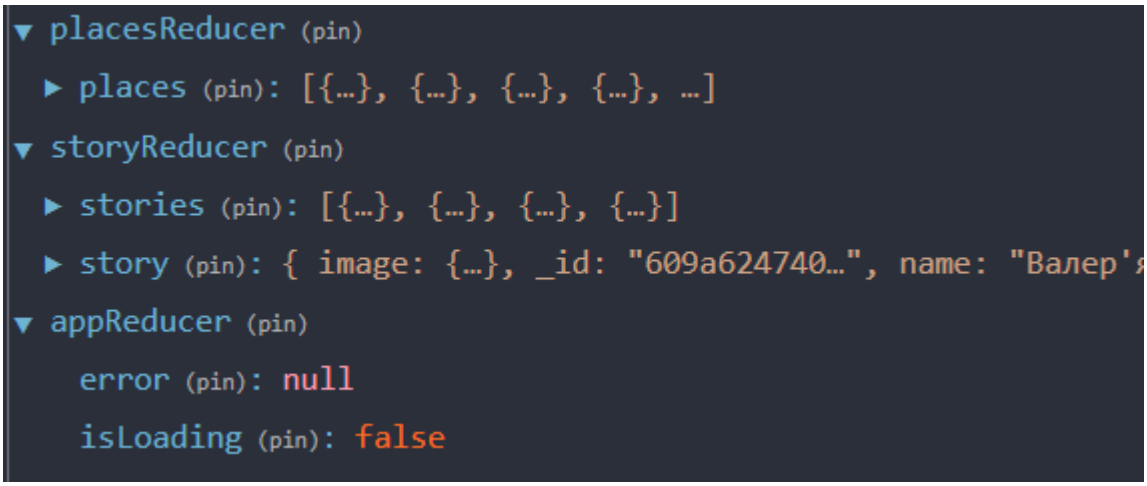


Рисунок 11 – Redux-сховище застосунку

Для збереження глобальних станів застосунку та надсилання запитів на серверну частину застосунку розробимо наступну структуру сховища (redux store) (рисунок 11), де розділимо логіку застосунку на три основні розділи (reducers): місця (placesReducer), історії (storyReducer) і загальна логіка застосунку (appReducer).

За загальну логіку роботи застосунку відповідає appReducer (рисунок 12). Він зберігає такі стани як:

– error – повідомлення про помилку роботи застосунку; для зміни даного стану ред’юсер має дві дії: success та error, кожна з яких відповідає за

видалення будь-яких помилок про помилку у разі успішно відпрацьованого запиту або ж, відповідно, зберігання помилки в сховищі застосунку;

– isLoading – boolean змінна процесу завантаження, для якої ред'юсерд має дві дії: ввімкення "завантаження" та вимкнення; даний стан буде відслідковуватися у окремо створеному компоненті спінера завантаження, якій з'являтиметься у випадках, коли дана змінна набуватиме значення true.

```
export const appReducer = (state = initialState, action) => {
  switch (action.type) {
    case types.ERROR:
      return {
        ...state,
        error: action.payload,
      };
    case types.SUCCESS:
      return {
        ...state,
        error: null,
      };
    case types.START_LOADING:
      return { ...state, isLoading: true };
    case types.STOP_LOADING:
      return { ...state, isLoading: false };
    default:
      return state;
  }
};
```

Рисунок 12 – AppReducer

Для роботи з історіями користувача розробимо storyReducer (рисунок 13), який відповідатиме за наступні стани:

– stories – масив історій користувачів, що будуть завантажуватися за допомогою виклику дії завантаження статей. Дана дія виконується у такій послідовності:

- ввімкнення лоадеру застосунку;
- надсилання на бекенд GET-запиту на отримання всіх історій блогу сайту;
- збереження отриманих статей в сховищі станів;
- вимкнення лоадеру.

– story – відкрита до перегляду історія, яка буде завантажена шляхом отримання ідентифікатора історії зі шляху сайту і надсилання запиту на сервер. Дана дія виконується у послідовності, аналогічній отриманню всіх статей застосунку.

```
export const storyReducer = (state = initialState, action) => {
  switch (action.type) {
    case types.GET_STORIES:
      return {
        ...state,
        stories: action.payload,
      };
    case types.GET_STORY_BY_ID:
      return {
        ...state,
        story: action.payload,
      };
    default:
      return state;
  }
};
```

Рисунок 13 – StoryReducer

Для роботи з інтерактивною картою створимо placesReducer (рисунок 14), який відповідатиме за збереження масиву даних про пункти прийому вторинної сировини у місті Києві та області. Даний ред’юсер має дії для отримання всіх точок, збережених в базі даних, а також на отримання даних про пункти прийому конкретної вторинної сировини.

```
export const placesReducer = (state = initialValues, action) => {
  switch (action.type) {
    case types.GET_PLACES:
      return {
        ...state,
        places: action.payload,
      };
    case types.GET_PLACES_BY_CATEGORY:
      return { ...state, places: action.payload };
    default:
      return state;
  }
};
```

Рисунок 14 – PlacesReducer

Відповідно до створеного ред'юсера розробимо компонент інтерактивної карти (рисунок 15). Для цього скористаємося компонентами Map (MapControl) і TileLayer бібліотеки react-leaflet.js, завдяки яким з'єднаємося з відкритою API OpenStreetMap і отримаємо необхідні нам дані для побудови карти.

```
<MapControl center={center} zoom={14} className={classes.root}>
  <TileLayer
    attribution='&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
    url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
  />
  <Control position="topright">
    <button
      type="button"
      onClick={openSortingMenu}
      className={classes.menuBtn}
    >
      Фільтер
    </button>
  </Control>

  {Array.isArray(places) &&
    places.map((place) => <PlacePopup place={place} key={place._id} />)}
</MapControl>
```

Рисунок 15 – Інтерактивна карта користувача

За допомогою бібліотеки react-leaflet-custom-control.js додамо на карту власний елемент – кнопку "Фільтр", при натиску на яку буде відкриватися бічне меню зі списком фільтрації пунктів прийому вторинної сировини.

При завантаженні сторінки "Карта" сайту буде відпрацьовувати дія отримання та додавання до сховища даних про пункти прийому вторинної сировини. При виборі однієї із запропонованих категорій до фільтрації буде відпрацьовувати дія отримання відповідних даних з серверу і зміни даних про місця у сховищі, завдяки чому дані, що відображатимуть на інтерактивній карті будуть змінюватися.

Для розміщення на карті міток з пунктами прийому вторинної сировини скористаємося компонентами Marker і Popup бібліотеки react-leaflet.js, завдяки яким зможемо динамічно розміщувати маркери місць на карті. Кожен об'єкт даних пункту прийому вторинної сировини містить свої географічні координати, завдяки яким маркер займатиме необхідне положення на карті. При натисканні на маркер буде відкриватися спливаюче вікно (Popup) з детальною інформацією

про пункт прийому вторинної сировини, а саме: назва, адреса, опис, номер телефону, робочі години і список іконок сировин для переробки, кожна з яких є посиланням на інформаційну сторінку з даними про те, що приймається в даній категорії, що не приймається, чому необхідно дані сировини переробляти і як вони переробляються.

```
const schema = object().shape({
  name: string().required("Поле ім'я є обов'язковим до заповнення"),
  email: string()
    .required("Залиште адресу електронної пошти для зворотнього зв'язку")
    .email('Перевірте правильність введеного'),
  story: string().required('Залиште свою історію, щоб надихати інших'),
  image: mixed()
    .test(
      'fileSize',
      'Зображення завелике',
      ({ data }) => !data || data.length <= FILE_SIZE
    )
    .test(
      'contentType',
      'Невідомий формат',
      ({ contentType }) =>
        !contentType || SUPPORTED_FORMATS.includes(contentType)
    ),
});
```

Рисунок 16 – Схема валідації форми

Блок “Поділитися власною історією” є блоком з формою зворотнього зв’язку користувача. Для створення даної форми скористаємося спроектованою схемою моделі даних і створимо Yup-схему (рисунок 16), яка в подальшому спростить процес валідації введених користувачем даних.

Скористаємося бібліотекою Formik.js для спрощення процесу отримання та зміни даних форми, її валідації та “відловлювання” помилок вводу. Розглянемо приклад створення власних полів вводу на полі для додавання файлів (рисунок 17). Унікальність даного поля полягає у безпосередній візуальній відсутності поля вводу (завдяки використанню display: none;) і переняті його функціоналу кнопкою.

```
<label htmlFor="image">
  <Button
    component="span"
    color={values.image.data ? 'primary' : 'secondary'}
    variant="contained"
    className={classes.button}
  >
    {values.image.data ? 'Фото додано' : 'Додайте фото'}
  </Button>
  <input
    id="image"
    name="image"
    type="file"
    onChange={(event) => {
      const reader = new FileReader();
      if (event.currentTarget.files[0]) {
        reader.readAsDataURL(event.currentTarget.files[0]);
        reader.onloadend = () => {
          const image = reader.result;
          setFieldValue('image.data', image);
        };
        setFieldValue(
          'image.contentType',
          event.currentTarget.files[0].type
        );
      } else {
        setFieldValue('image.data', null);
        setFieldValue('image.contentType', null);
      }
    }}
    className={classes.field_file}
  />
</label>
{values.image.data && <Avatar src={values.image.data} alt="" />}
```

Рисунок 17 – Поле для завантаження файлу

Першочергово при відсутності завантаженого зображення кнопка є синьою з текстом “Додати фото”, при натиску на неї з’являється вікно додавання файлу. При додаванні файлу в разі його не валідності (невірного формату і/або завеликого розміру) з’явиться відповідне повідомлення. При виборі валідного файлу до об’єкту форми буде додано завантажений за адресою файл і його розширення, кнопка завантаження змінює колір на зелений, а текст на “Фото додане”, поруч з’являється круглий варіант завантаженого зображення.

При натиску на кнопку “Поділитися” відпрацьовує функція відправлення форми, а саме виклик storyReducer дії, яка надсилає дані форми HTTP-запитом на сервер.

На стороні серверу при отриманні запиту на додавання історії відпрацьовує запит до бази даних на створення нової історії (рисунок 18).

```
module.exports.addStory = async (req, res) => {
  const story = req.body;
  await storyDao.postStory(story);
  res.json({ message: 'Story added successfully' });
};
```

```
module.exports.postStory = async (data) => {
  const newStory = new Story(data);
  await newStory.save();
};
```

Рисунок 18 – Додавання історії користувача до БД

Після додавання нової історії в базу даних відбувається переадресація користувача на сторінку “Рекомендації” для перегляду всіх історій блогу сайту. При завантаженні даної сторінки викликається дія отримання та збереження в глобальному сховищі всіх історій користувачів.

Кожна історія блогу відображатиметься у вигляді картки (рисунок 19) з іменем та адресою електронної пошти користувача, що залишив дану історію, зображенням та коротким викладом тексту.

При натиску на картку відбувається переадресація користувача на сторінку історії, при завантаженні якої викликається дія на отримання даної історії. На даній сторінці користувач має безпосередню можливість ознайомитися з усім текстом історії.

```
return (
  <Card className={classes.container}>
    <CardActionArea onClick={openArticle}>
      <CardHeader
        title={story.name}
        subheader={story.email}
        avatar={<Avatar>{story.name[0]}</Avatar>}
      />
      {story.image.data && (
        <>
          <CardMedia
            image={arrayBufferToBase64(story.image.data.data)}
            className={classes.media}
          />
          <img
            src={arrayBufferToBase64(story.image.data.data)}
            alt=""
            className={classes.media}
          />
        </>
      )}
      <CardContent>
        <Typography variant="body1">{story.story}</Typography>
      </CardContent>
    </CardActionArea>
  </Card>
);
```

Рисунок 19 – Компонент картки історії

3.3 Огляд функціоналу застосунку

Розглянемо результат процесу розробки веб-застосунку Carpe Diem.

Веб-застосунок для сортування сміття отримав шість логічних функціональних сторінок та дві сторінки інформаційного спрямування. Розроблений веб-застосунок відповідає представленій UML-діаграмі діяльностей (рисунок 20).

При вході на сайт користувач потрапляє на головну сторінку (додаток Л), яка являє собою візитівку веб-застосунку. Дана сторінка містить карусель зображень з мотиваційними банерами, міні-статті, що коротко розкривають основну ідею кожного інформаційного блоку застосунку, а також додаткову навігацію у вигляді фото-блоків, що являють собою посилання на відповідні сторінки сайту.

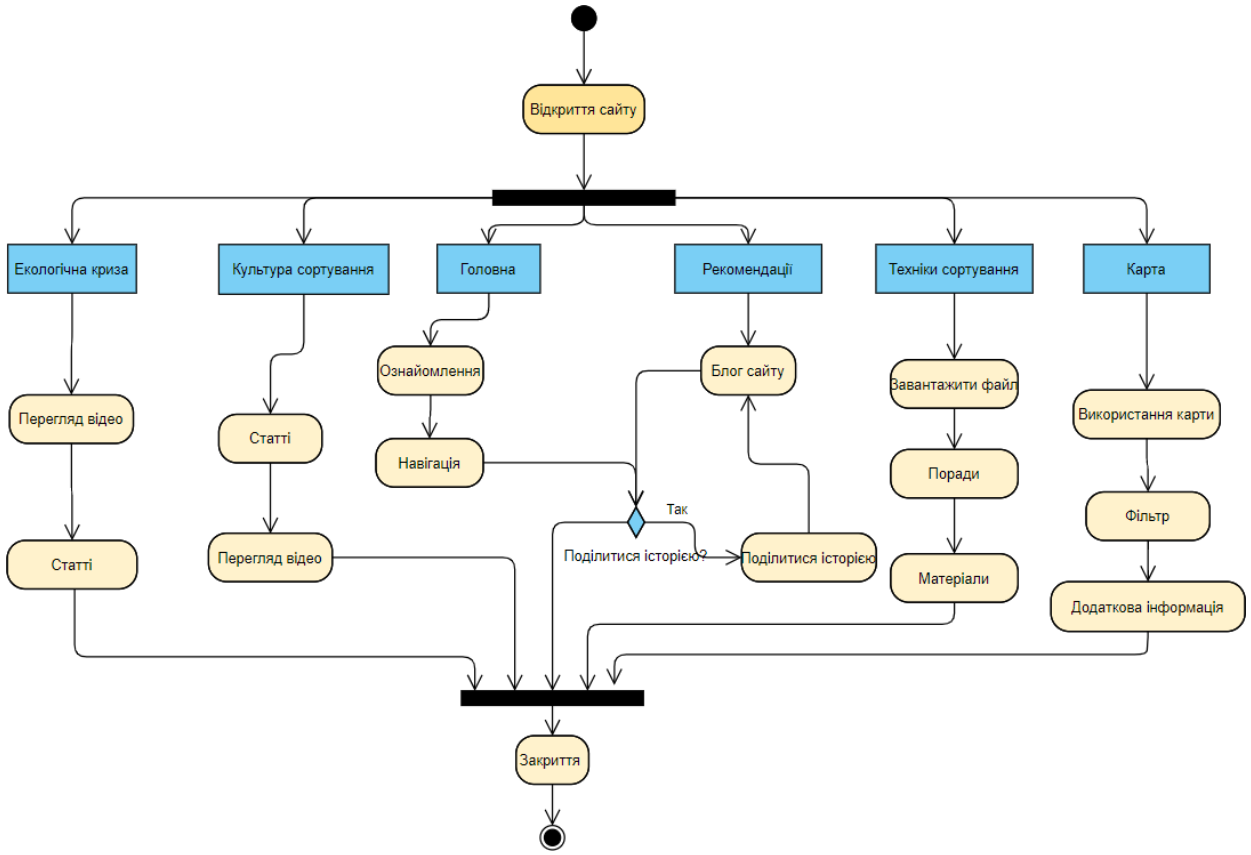


Рисунок 20 – Діаграма діяльності

Заключним блоком даної сторінки є компонент «Поділитися власною історією», що є формою зворотного зв’язку користувача. Даний компонент дає користувачу можливість долучитися до мотиваційно-інформаційної мети сайту і залишити свою історію на сторінці блогу сайту.

Відповідно до поставленого завдання форма містить обов’язкові поля для даних користувача та його історії і додаткове поле для завантаження користувацького зображення. Після заповнення форми відбувається автоматичне переадресування користувача на сторінку блогу сайту.

Відповідно до спроектованих макетів було розроблено головне навігаційне меню сайту, що складається із горизонтального блоку, що містить в собі посилання на всі логічні структурні елементи веб-застосунку. В тому числі, на блоці, відображається емблема веб-застосунку із назвою Carpe Diem, що також слугує за функціональний компонент для повернення на головну сторінку застосунку.

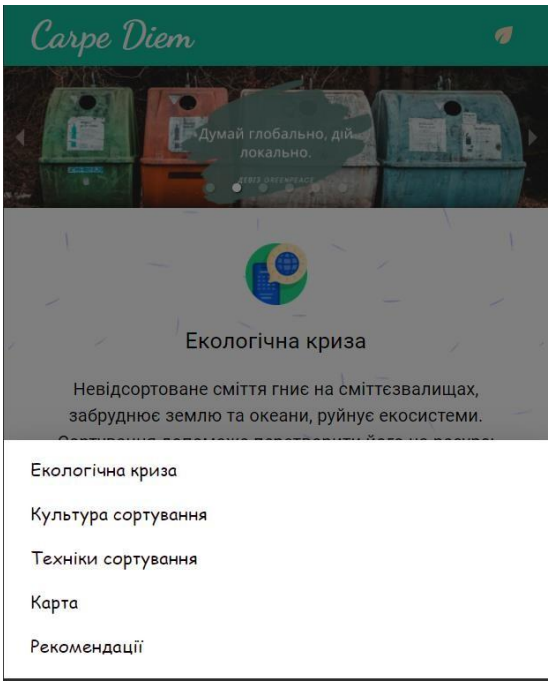


Рисунок 21 – Верхнє навігаційне меню сайту (мобільна версія)

Відповідно до вимог кросбраузерності та доступності з будь-якого пристрою на рисунку 21 продемонстровано трансформацію меню застосунку для відображення на мобільних пристроях. Трансформація перетворює елементи меню у кнопку, групуючи в ній всі посилання, що відображаються при натиску. Такий підхід економить місце на обмеженій площі екрану та відповідає найкращим практикам досвіду користувача.

Заключним блоком як головної сторінки сайту, так і будь-якої іншої сторінки сайту є нижнє навігаційне меню сайту (footer). Даний блок містить в собі логотип сайту, короткий слоган та додатковий блок з внутрішньою навігацією веб-застосунку.

Наступною логічною сторінкою сайту є «Екологічна криза» (додаток М), що має на меті висвітлення екологічної проблеми забруднення планети. Відповідно до поставленого завдання та спроектованого макету дана сторінка містить набір міні-статей і відео матеріалів.

Сторінка «Культура сортування» (додаток М) присвячена висвітленню досвіду сортування сміття країн Європи була розробленою відповідно до поставленого завдання і спроектованого макету і містить відповідний набір фото- та відеоматеріалів, а також міні-статей.

«Техніки сортування» (додаток П) є сторінкою веб-застосунку, присвяченій сортування сміття. Даний компонент був розроблений відповідно до макету і містить додатковий інформаційний блок з банером для завантаження файлу рекомендацій; мотиваційний блок-список необхідний пунктів для сортування вже сьогодні; набір додаткових інформаційних блоків, присвячених конкретним матеріалам (пластик, папір, скло тощо) з додатковою інформацією про їх види і необхідні кроки для подальшого сортування.

Сторінка «Карта» (додаток Р) містить інтерактивну мапу, з нанесеною на нею маркерами пунктів прийому вторинної сировини. Відповідно до поставленого задання карта має додатковий функціонал для фільтрації пунктів прийому за вторинною сировиною. Також дана сторінка має набір додаткових інформаційних сторінок, на які можна перейти, натиснувши на іконку вторинної сировини на віконці пункту прийому.

Останньою сторінкою сайту є «Рекомендації» (додаток С) – мотиваційний-блог веб-застосунку, де користувач може ознайомитися з вже наявними історіями інших користувачів і додати власну.

Висновки до розділу 3

У даному розділі, відповідно до попередніх досліджень, обґрунтування вибору стеку технологій, мною висвітлено процес розроблення веб-застосунку для сортування сміття.

Для проектування веб-застосунку було проведено аналіз поняття досвіду користувача, використано інструмент для підбору кольорової гамми проекту, обрано назву проекту та розроблено логотип, що відповідає кольоровій гаммі та стилістиці додатку.

Задля мінімізації необхідності перероблення фінального продукту перед безпосередньою розробкою було спроектовано каркаси-чернетки усіх необхідних сторінок, представлено їх графічний вид, опис компонентів та планованого функціоналу.

Безпосередньо в процесі розробки веб-застосунку було складено UML-діаграму компонентів, опрацьовано питання роутингу сторінок на принципі односторінкового застосунку. Обраними в розділі 2 інструментами розроблено сторінки веб-застосунку, створено їх функціонал, а самі сторінки об’єднано в логічно-працюючу схему. Проведено роботу для фіксації станів компонентів сторінки, підключено базу даних, а в подальшому, для ефективної роботи, її розгорнуто на хмарному середовищі Google Clound.

З огляду на необхідність кросбраузерності та вимог досвіду користувача розроблено графічний дизайн та стилі проекту. Веб-застосунок має повну сумісність з усіма актуальними браузерами та побудований на принципі адаптивності для правильного відображення на мобільних пристроях.

З усього представленого можна зробити висновок про те, що проектування та розроблення веб-застосунку були успішними, увесь оголошений функціонал реалізовано, користування застосунком відповідає усім сучасними вимогам, стандартам сумісності та користувацького досвіду.

ВИСНОВКИ

У поданій дипломній роботі мною було проведено огляд проблематики екологічної кризи в Україні та світі, досліджено звітні данні Організації Об’єднаних Націй, Світової Організації Охорони Здоров’я, Фонду Еллен Макартур та Міністерства розвитку громад та територій України. Дані з цих ресурсів було використано для аналізу стану проблеми та пошуку наявних програмних аналогів в Україні, що поширюють інформацію про боротьбу з екологічною кризою у вигляді надмірного продукування відходів людської діяльності та відсутності механізмів ефективної переробки таких відходів.

Було проведено аналіз сучасних підходів до розробки веб-застосунків, досліджено принципи односторінкового та багатосторінкового проектування. Проаналізовано переваги та недоліки таких принципів.

На підставі дослідження наявних в українському інформаційному просторі веб-застосунків та світових аналогів мною було проведено вибір ефективних технологій та формування стеку розробки для проектування та розроблення власного веб-застосунку. Для реалізації веб-системиу підсумку було використано вісімнадцять інструментів веб-розробки, бібліотек та фреймворків.

На базі обраного стеку, спочатку, створені каркаси, підібрана кольорова гамма, застосовано принципи UI/UX для адаптивності застосунку. Після проектування в процесі розробки було в повному обсязі реалізовано функціонал, що описано в технічному завданні. Веб-застосунок протестовано як на адаптивність під різні пристрої так і на кросбраузерність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. UN’s mission to keep plastics out of oceans and marine life. UN News. URL: https://news.un.org/en/story/2017/04/556132-feature-uns-mission-keep-plastics-out-oceans-and-marine-life. (дата звернення: 12.02.2021).
2. World Environment Day 2018: a call to “Beat plastic pollution”. Regional Centre for Environmental Health Action. World Health Organization. URL: http://www.emro.who.int/ceha/ceha-news/world-environment-day-2018-a-call-to-beat-plastic-pollution.html. (дата звернення: 12.02.2021).
3. Study confirms need for urgent transition to a circular economy for plastic. Ellen Macarthur Foundation. URL: https://www.ellenmacarthurfoundation.org/news/new-study-confirms-need-for-urgent-transition-to-a-circular-economy-for-plastic. (дата звернення: 12.02.2021).
4. Microplastic in drinking-water. World Health Organization. URL: https://apps.who.int/iris/bitstream/handle/10665/326499/9789241516198-eng.pdf. (дата звернення: 14.02.2021).
5. Five things you should know about disposable masks and plastic pollution. UN News. URL: https://news.un.org/en/story/2020/07/1069151. (дата звернення: 15.02.2021).
6. Стан сфери поводження з побутовими відходами в Україні за 2019 рік. Офіційний веб-сайт Міністерства розвитку громад та територій України. URL: https://www.minregion.gov.ua/napryamki-diyalnosti/zhkh/terretory/stan-sfery-povodzhennya-z-pobutovymy-vi/. (дата звернення: 15.02.2021).
7. В Україні значно виросла інтернет-пенетрація. Інтернет Асоціація України. URL: https://inau.ua/news/v-ukrayini-znachno-vyrosla-internet-penetraciya. (дата звернення: 15.02.2021).
8. Поводження з відходами. URL: http://solvetpv.lviv.ua/. (дата звернення: 19.02.2021).
9. Україна без сміття. URL: https://nowaste.com.ua/. (дата звернення: 23.02.2021).

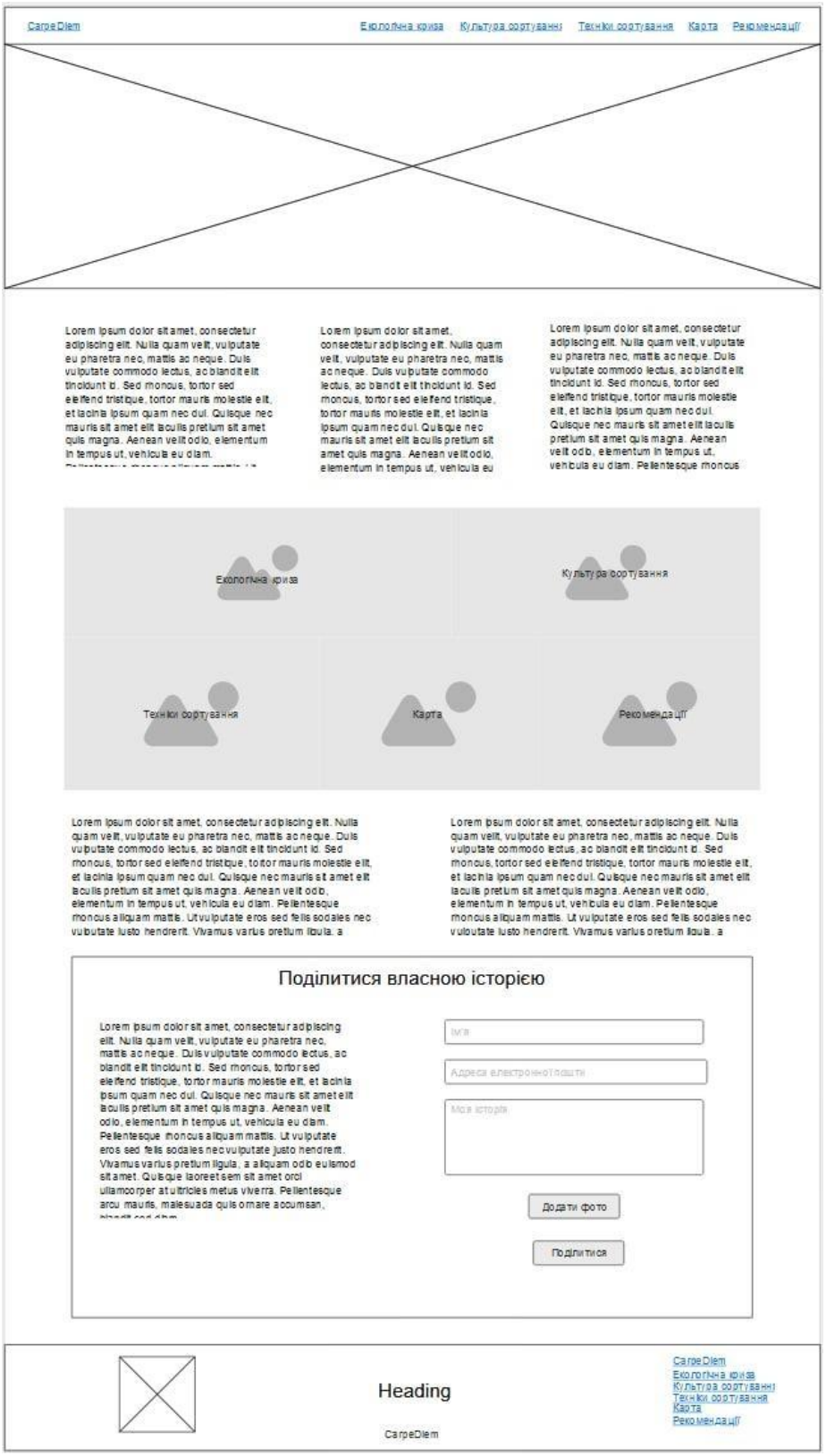
10. RecycleMap. URL: https://recyclemap.org/. (дата звернення: 03.03.2021).
11. Офіційний портал Міністерства захисту довкілля та природних ресурсів України. URL: https://mepr.gov.ua/. (дата звернення:03.03.2021).
12. Believe Earth. URL: https://believe.earth/en/. (дата звернення: 05.03.2021).
13. Developer Survey 2020. Stack Overflow. URL: https://insights.stackoverflow.com/survey/2020#technology-web-frameworks. (дата звернення: 06.03.2021).
14. jQuery. URL: https://jquery.com/. (дата звернення: 09.03.2021).
15. Single Page Application (SPA) vs Multi Page Application (MPA) – Two Development Approaches. ASPER BROTHERS. URL: https://asperbrothers.com/blog/spa-vs-mpa/. (дата звернення: 10.03.2021).
16. HTML5 Developer guides. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5. (дата звернення: 11.03.2021).
17. CSS: Cascading Style Sheets Developer guides. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/CSS. (дата звернення: 11.03.2021).
18. Choose Between Traditional Web Apps and Single Page Apps (SPAs). Microsoft. URL: https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps. (дата звернення: 12.03.2021).
19. React. Facebook Inc. URL: https://uk.reactjs.org/. (дата звернення: 12.03.2021).
20. React Router. React Training. URL: https://reactrouter.com/web/guides/quick-start. (дата звернення: 14.03.2021).
21. Вступ до JSX. Facebook Inc. URL: https://uk.reactjs.org/docs/introducing-jsx.html. (дата звернення: 14.03.2021).
22. Material. Google LLC. URL: https://material.io/design/introduction. (дата звернення:14.03.2021).
23. Material-UI. URL: https://material-ui.com/. (дата звернення: 14.03.2021).

24. Форми. Facebook Inc. URL: https://uk.reactjs.org/docs/forms.html. (дата звернення: 15.03.2021).
25. Formik. Formium, Inc. URL:https://formik.org/. (дата звернення: 17.03.2021).
26. Yup. URL: https://github.com/jquense/yup. (дата звернення: 17.03.2021).
27. Vladimir Agafonkin. Leaflet. OpenStreetMap. URL: https://leafletjs.com/. (дата звернення: 17.03.2021).
28. Paul Le Cam. React Leaflet. URL: https://react-leaflet.js.org/. (дата звернення: 17.03.2021).
29. OpenStreetMap. URL: https://www.openstreetmap.org/. (дата звернення: 17.03.2021).
30. react-leaflet-control. URL: https://github.com/LiveBy/react-leaflet-control. (дата звернення: 17.03.2021).
31. Dan Abramov. React Redux. URL: https://react-redux.js.org/. (дата звернення: 20.03.2021).
32. MongoDB. MongoDB, Inc. URL: https://www.mongodb.com/. (дата звернення: 21.03.2021).
33. SQL to Aggregation Mapping Chart. MongoDB Manual. MongoDB, Inc. URL: https://docs.mongodb.com/v4.0/reference/sql-aggregation-comparison/. (дата звернення: 22.03.2021).
34. Fabrizio Bianchi. Coolors. URL: https://coolors.co/. (дата звернення: 22.03.2021).
35. Canva. URL: https://www.canva.com/. (дата звернення: 25.03.2021).

ДОДАТКИ

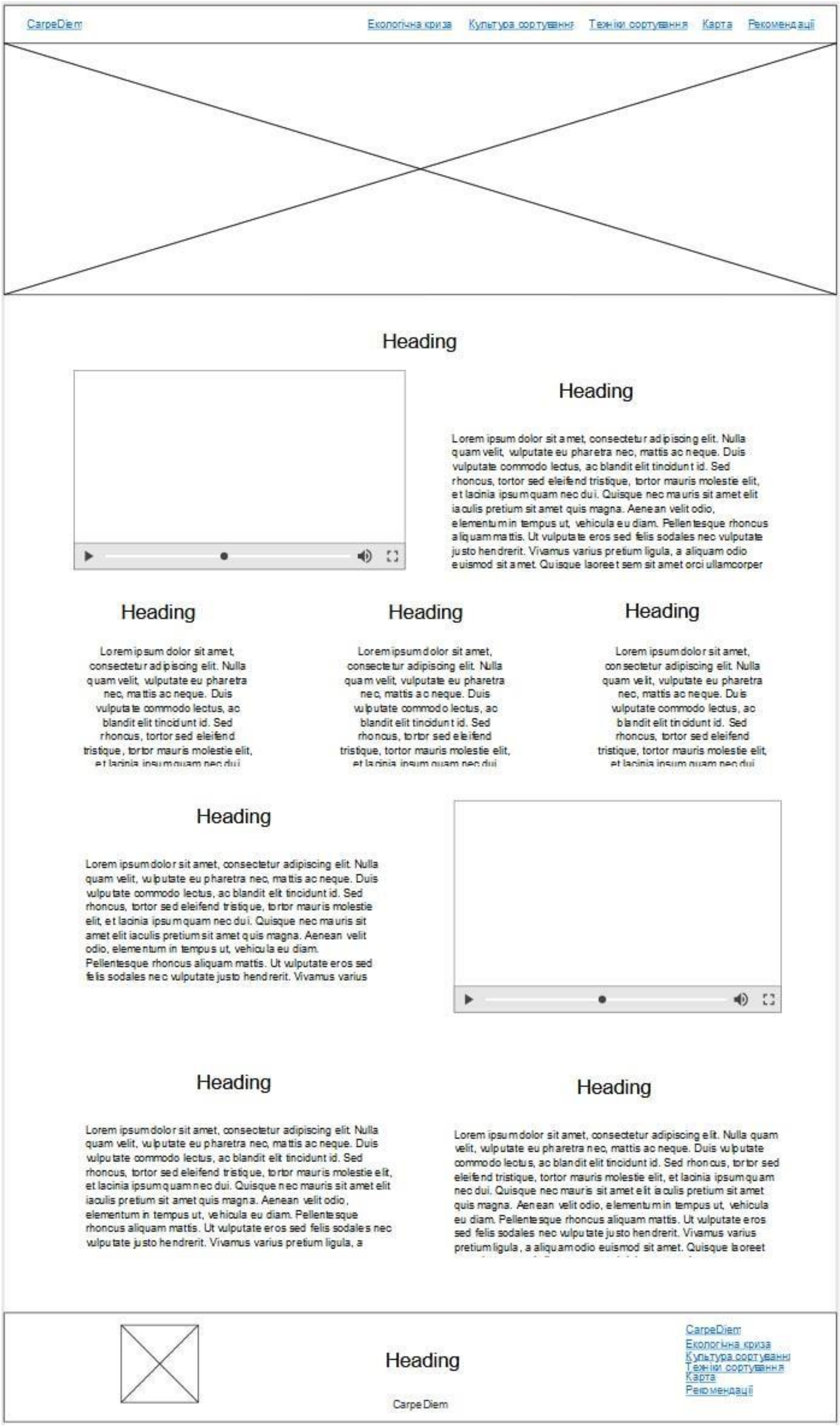
Додаток А

Каркас головної сторінки веб-застосунку



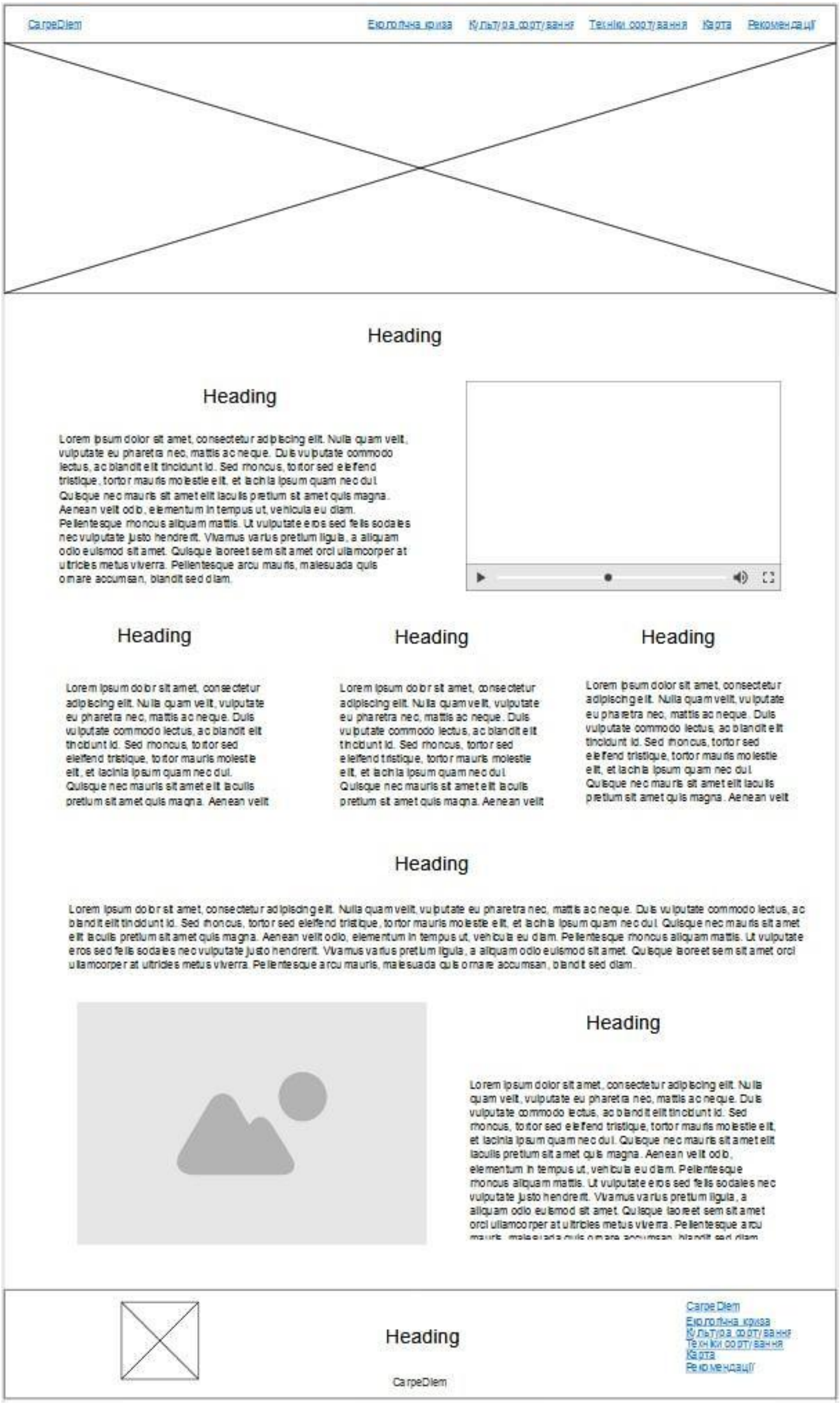
Додаток Б

Каркас сторінки “Екологічна криза”



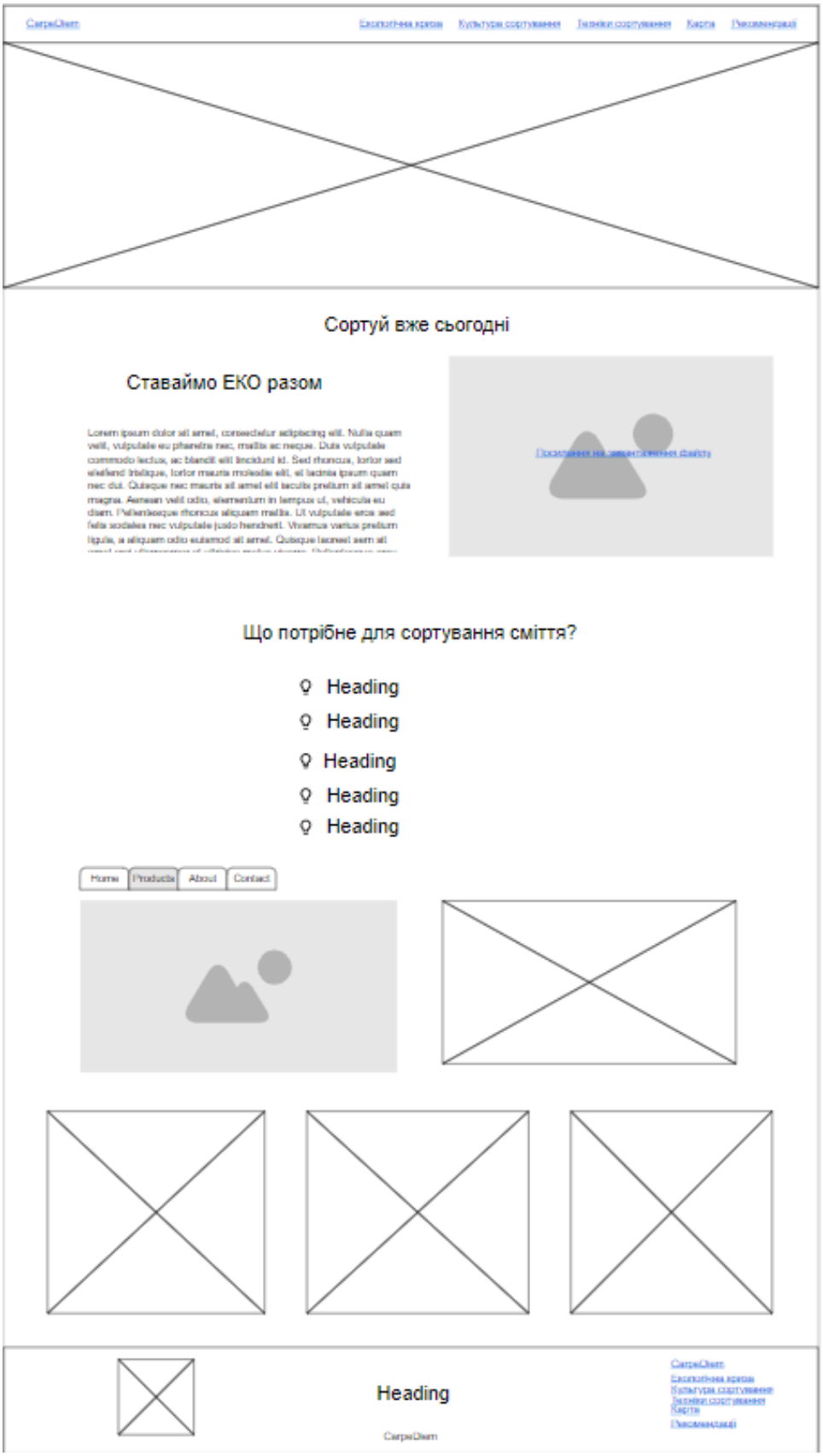
Додаток В

Каркас сторінки “Культура сортування”



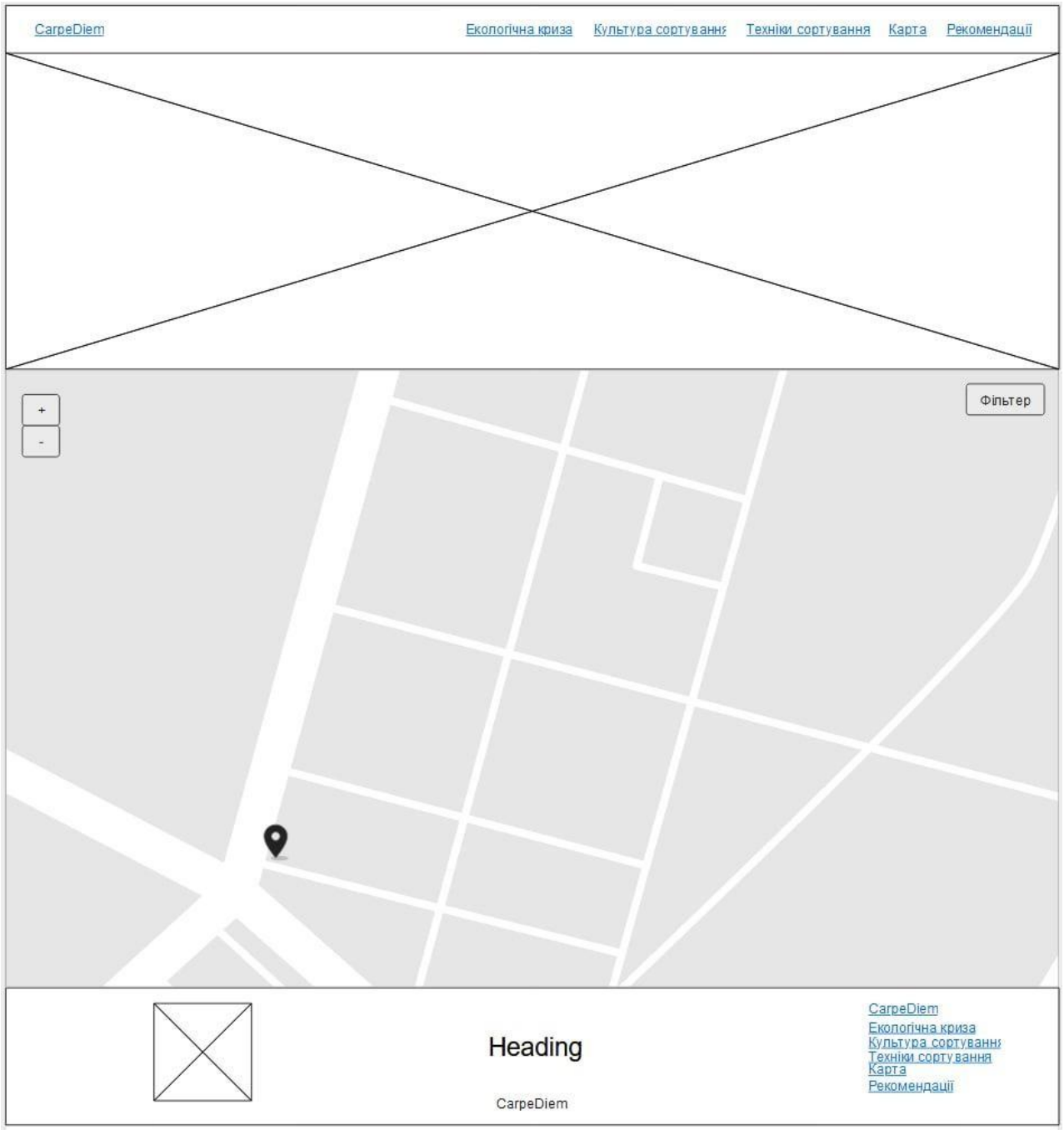
Додаток Г

Каркас сторінки “Техніки сортування”



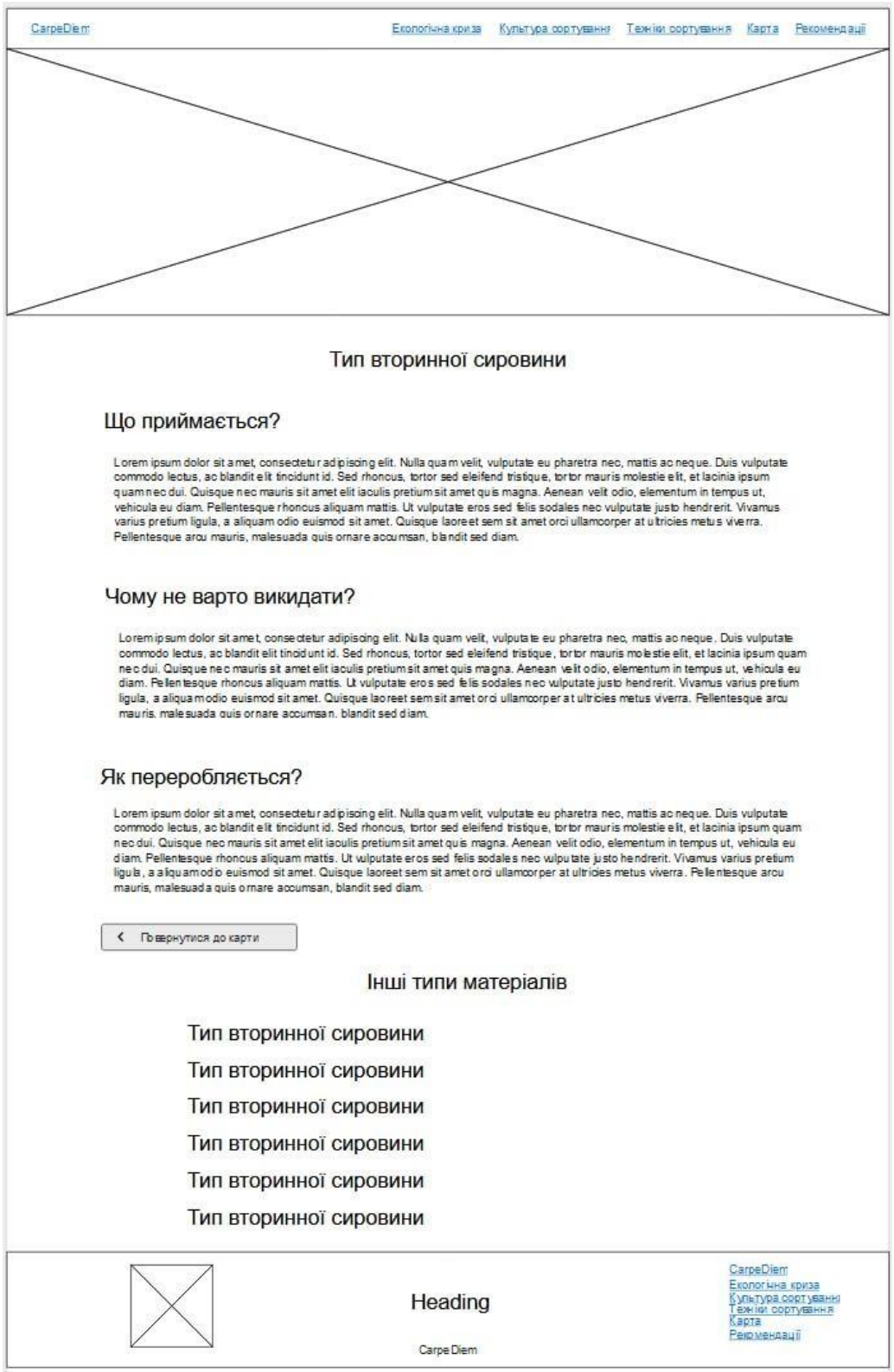
Додаток Д

Каркас сторінки “Карта”



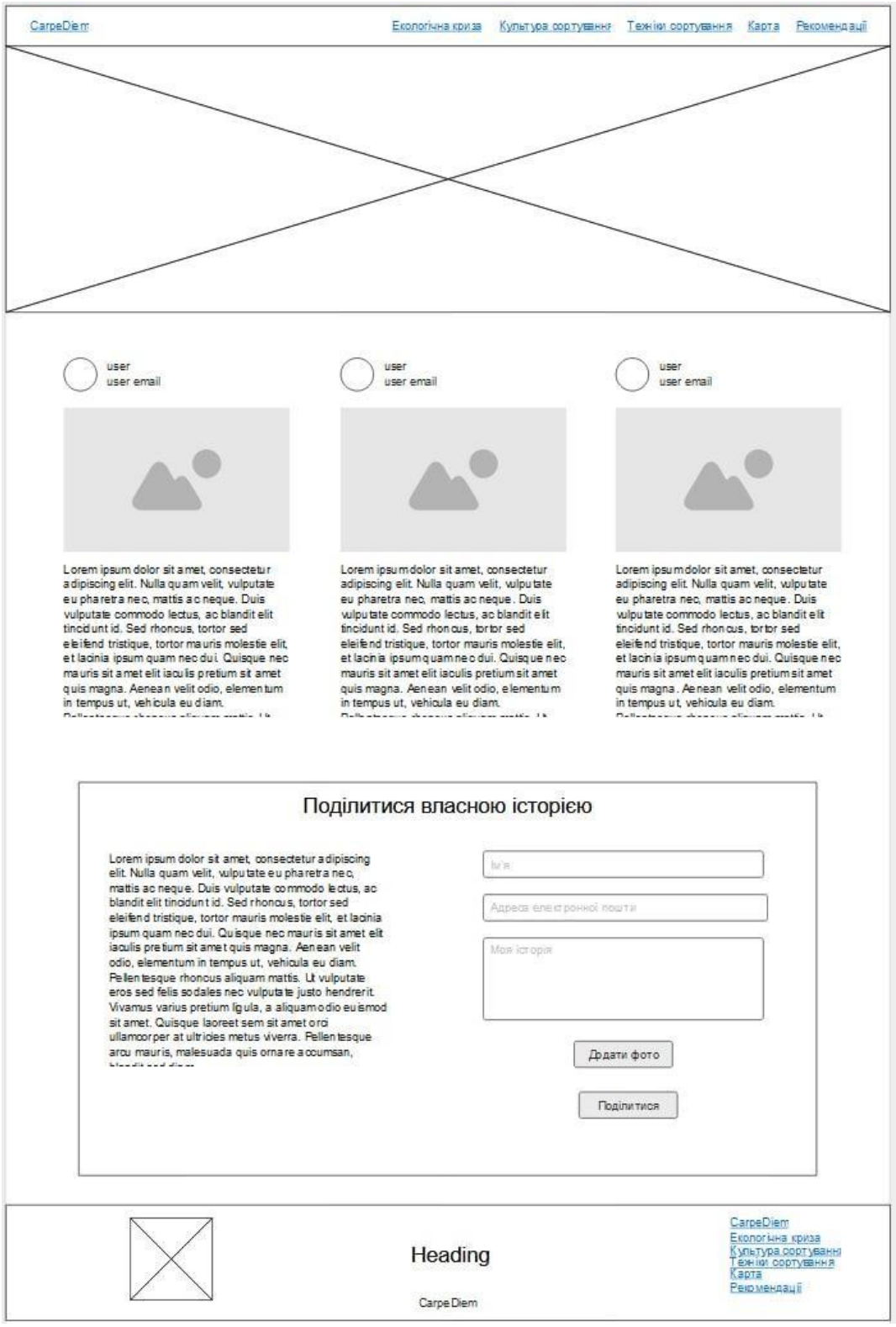
Додаток Е

Каркас інформаційної сторінки вторинної сировини



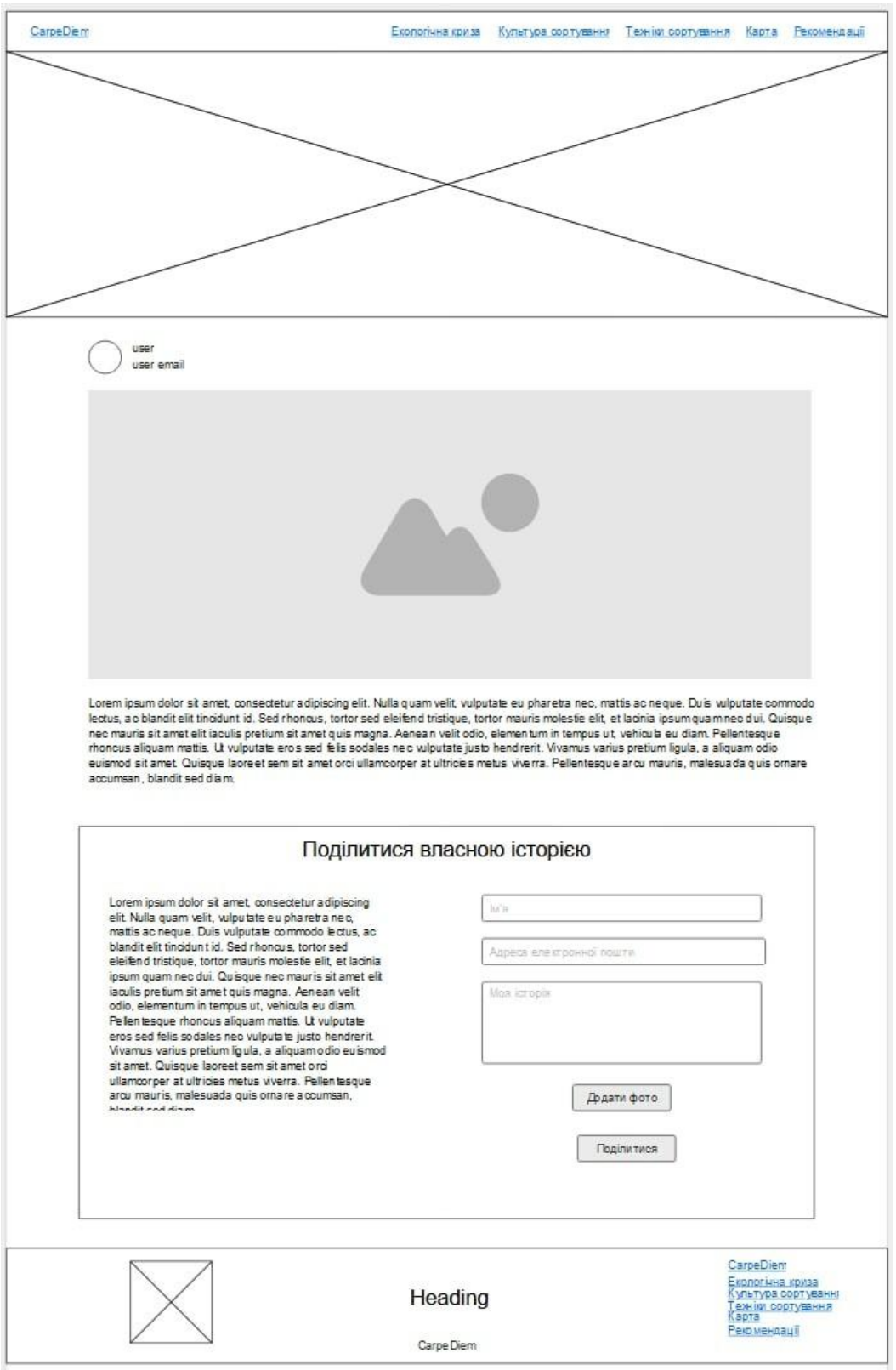
Додаток Ж

Каркас сторінки “Рекомендації”



Додаток И

Каркас сторінки історії користувача



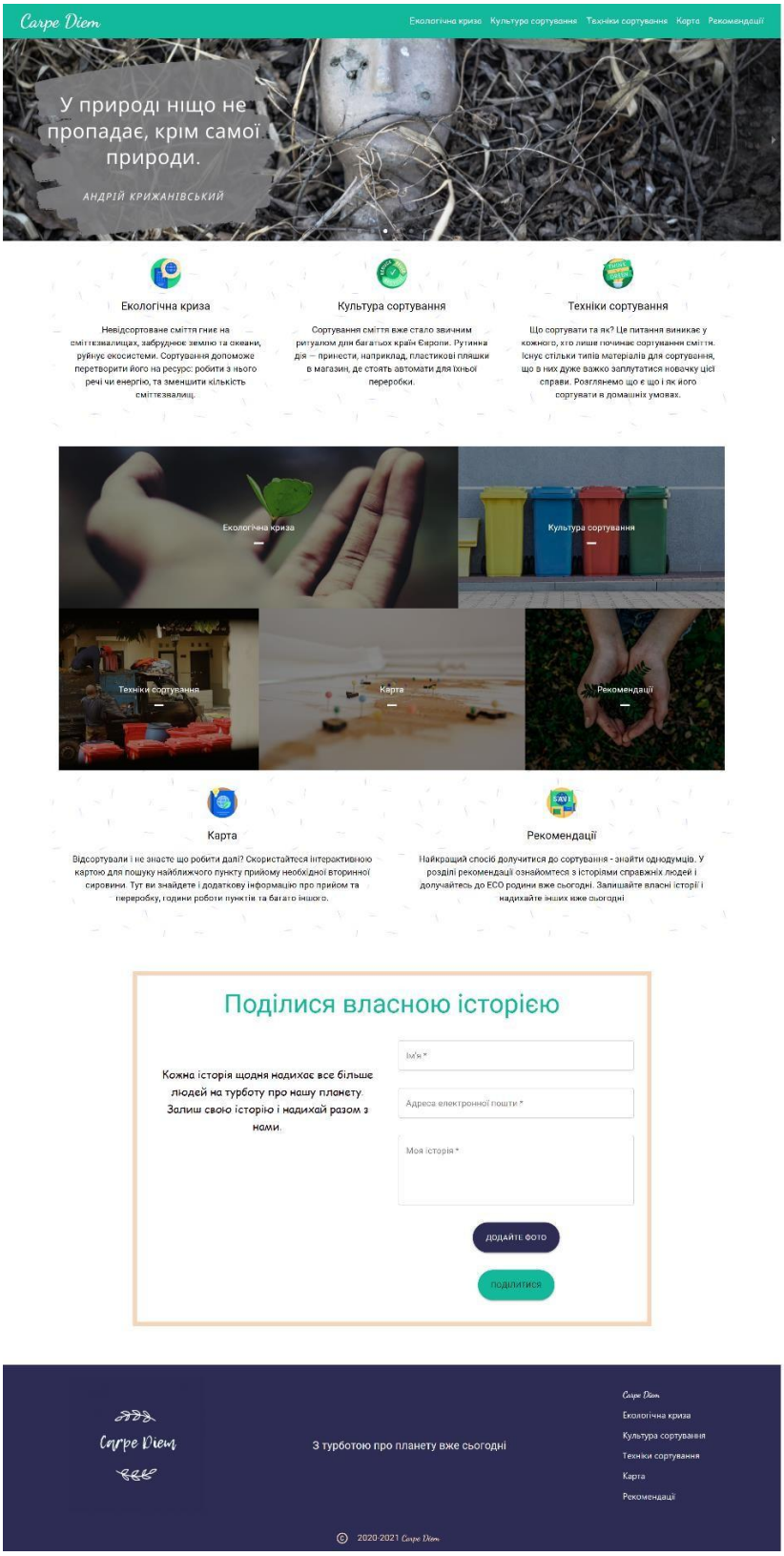
Додаток К

Код компоненту верхнього меню навігації

```
export const HeaderNavigation = () => {
  const classes = useStyles();
  const [showMobMenu, setShowMobMenu] = useState(false);
  return (
    <AppBar position="static">
      <Toolbar className={classes.header}>
        <Link to={routes.home} className={classes.header__link}>
          <Typography variant="h4" className={classes.header__text}>
            Carpe Diem
          </Typography>
        </Link>
        <Hidden smDown>
          <Box className={classes.header__hidden}>
            {headerList.map(({ key, text, path }) => (
              <Link key={key} to={path} className={classes.header__link}>
                <Typography variant="body1" className={classes.header__text}>
                  {text}
                </Typography>
              </Link>
            ))}
          </Box>
        </Hidden>
        <Hidden mdUp>
          <IconButton
            className={classes.header__link}
            onClick={() => setShowMobMenu(!showMobMenu)}
          >
            <EcoIcon />
          </IconButton>
          <Drawer
            anchor="bottom"
            open={showMobMenu}
            onClose={() => setShowMobMenu(false)}
          >
            <List>
              {headerList.map(({ key, text, path }) => (
                <ListItem key={key}>
                  <Link to={path} className={classes.header__link_mob}>
                    <Typography
                      variant="body1"
                      className={classes.header__text}
                    >
                      {text}
                    </Typography>
                  </Link>
                </ListItem>
              ))}
            </List>
          </Drawer>
        </Hidden>
      </Toolbar>
    </AppBar>
  );
};
```

Додаток Л

Головна сторінка сайту

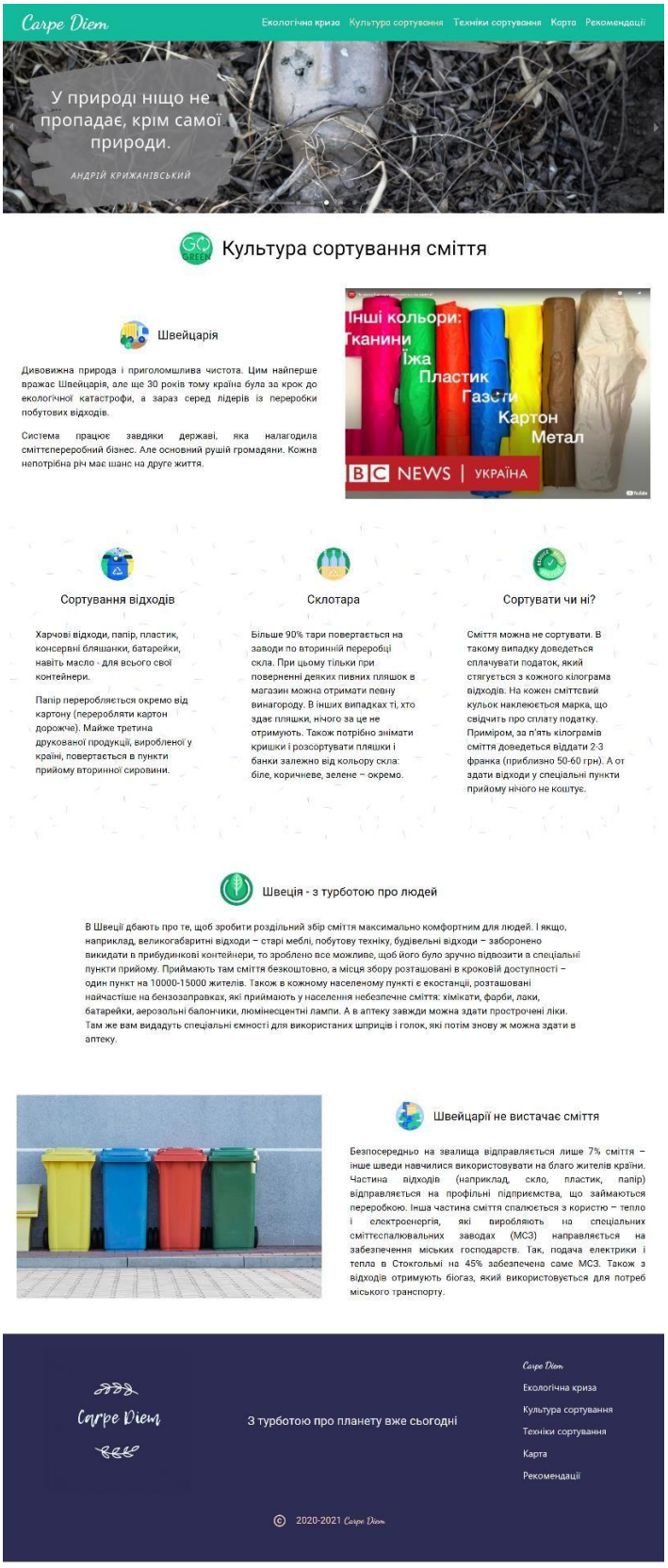


Додаток М

Сторінка «Екологічна криза»

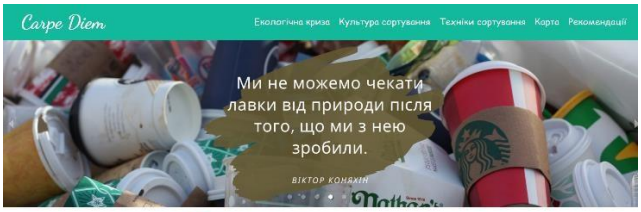
Додаток Н

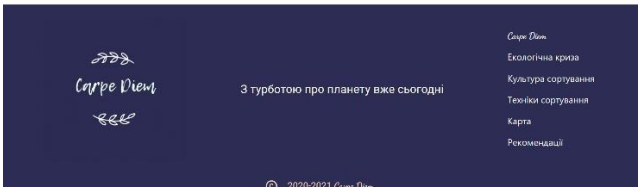
Сторінка «Техніки сортування»



Додаток П

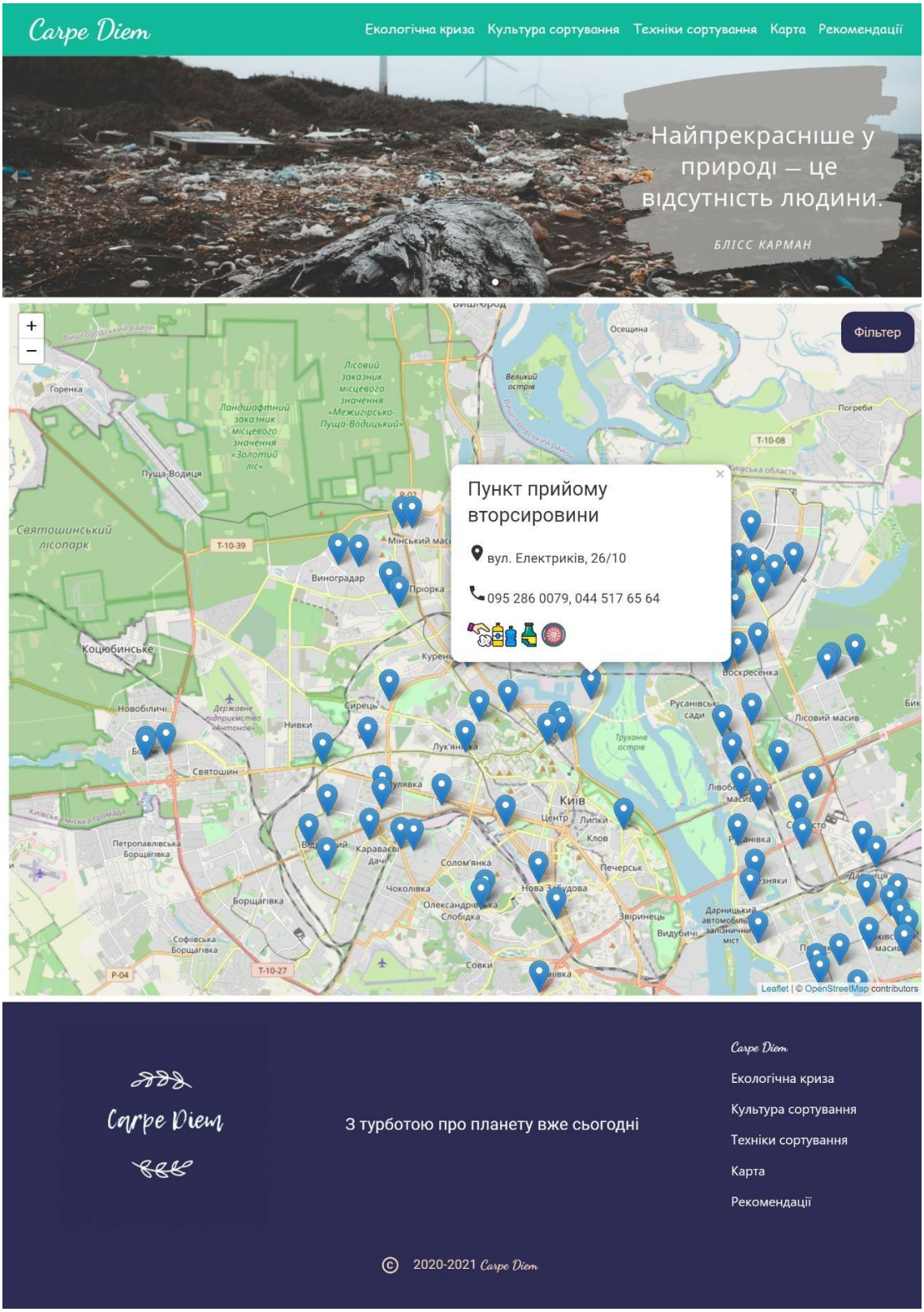
Сторінка «Техніки сортування»





Додаток Р

Сторінка «Карта»



Додаток С

Сторінка «Рекомендації»