Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Юрій КРАВЧЕНКО
«_____»_____2022 року

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»
за спеціальністю 172 «Телекомунікації та радіотехніка»
освітньо-професійна програма «Мережеві та інтернет технологій»
на тему:

РОЗРОБКА MVC ДОДАТКУ ДЛЯ ВИРІШЕННЯ ЕКОЛОГІЧНИХ ПРОБЛЕМ ТА ІНТЕГРАЦІЄЮ GOOGLE MAP API

Виконав: студент групи МІТ -41

Олександр НЕВМЕРЖИЦЬКИЙ _____
(ім'я та ПРІЗВИЩЕ) (підпис)

Керівник:

к.т.н. Олександр МАХОВИЧ _____
(науковий ступень, вчене звання, (підпис)
ім'я та ПРІЗВИЩЕ)

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____Юрій КРАВЧЕНКО
«____»_____2023 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти Невмержицький Олександр Михайлович
(прізвище, ім'я, по батькові)

1. Тема роботи:
Розробка MVC додатку для вирішення екологічних проблем та інтеграцією Google map API

затверджена на засіданні кафедри МІТ « » грудня 2022 р.
протокол №

2. Термін здачі закінченої роботи «30» травня 2023 р

3. Вихідні дані до проекту (роботи)

- Розробка MVC додатку для вирішення екологічних
- Інтеграція з google map api

4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-63 стор.

Вступ

1. Дослідження інформаційних технологій як засобу подолання проблем переробки побутових відходів в Україні
1.1 Проблеми переробки побутових відходів в Україні
1.2 Стратегії покращення переробки побутових відходів в Україні
1.3 Постановка задачі
2 Пошук технологій та дослідження вимог для розробки веб додатку «Be Green»
2.1 Дослідження потреб користувачів та функціональних вимог до додатку
2.2 Дослідження вимог до користувацького інтерфейсу додатку
2.3 Забезпечення безпеки додатку

2.4 Дослідження ефективного алгоритму розподілу балів серед осіб, що зробили внесок у контейнер
2.5 Підготовка до розгортання додатку в хмарі
3. Розробка програмного додатку «Be Green»
3.1 Аналіз досліджених технологій для розробки додатку «Be Green»
3.2 Реалізація програмного додатку «Be Green»
3.3 Демонстрація додатку «Be Green»

5. Перелік графічного матеріалу 8-10 слайдів.

Дата видачі завдання _____

Керівник роботи _____ к.т.н. Олександр МАХОВИЧ

(підпис) (посада, ім'я, ПРІЗВИЩЕ)

Завдання прийняв до виконання Олександр НЕВМЕРЖИЦЬКИЙ

(підпис) (ім'я, ПРІЗВИЩЕ)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	20.01.2023	Вик.
2	Розділ 1	17.02.2023	Вик.
3	Розділ 2	24.03.2023	Вик.
4	Розділ 3	21.04.2023	Вик.
5	Доповідь та слайди	20.05.2023	Вик.
6	Пояснювальна записка	30.05.2023	Вик.

Здобувач вищої освіти _____Олександр НЕВМЕРЖИЦЬКИЙ

(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник _____Олександр МАХОВИЧ

(підпис) (ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка: 62 с., 44 рис., 5 додатоків, 20 джерел.

Об'єкт дослідження: Проблеми переробки побутових відходів в Україні.

Предмет дослідження: Розробка веб-додатку "Be Green" для покращення переробки побутових відходів в Україні.

Мета роботи (проекту): Розробити веб-додаток "Be Green", який сприятиме покращенню переробки побутових відходів в Україні шляхом інформаційної підтримки та стимулювання активності користувачів у сфері екології.

Методи дослідження: Системний підхід, методи порівняння, індексний метод, структурний аналіз, кореляційно-регресійний аналіз.

У спеціальній частині дана характеристика проблем переробки побутових відходів в Україні та огляд стратегій покращення цього процесу, таких як організація освітніх заходів, співпраця з місцевими органами влади, залучення громади та розміщення контейнерів для сортування відходів.

В роботі проведено аналіз сучасного стану переробки побутових відходів в Україні, виявлено проблеми та розглянуто стратегії для покращення цього процесу.

Запропоновано розробку веб-додатку "Be Green", який надасть інформаційну підтримку щодо переробки побутових відходів, сприятиме залученню громади та створить стимули для активності користувачів у цій сфері.

Розроблено алгоритм розподілу балів між особами, які зробили вклад у контейнер, з використанням алгоритму Nash Bargaining Solution, що дозволяє оптимізувати розподіл ресурсів у справедливий спосіб.

Побудовано веб-додаток "Be Green", що включає функції відображення карт, реєстрації користувачів, відстеження внесків до контейнерів, розподілу балів.

Розгорнуто додаток "Be Green" в хмарному середовищі, що забезпечує його доступність та масштабованість для користувачів.

Практичне значення роботи полягає у сприянні покращенню переробки побутових відходів в Україні шляхом надання інформаційної підтримки, створення стимулів для участі громади та оптимізації розподілу ресурсів.

Результати здійснених у дипломному проекті досліджень можуть бути використані органами влади, екологічними організаціями та розробниками програмного забезпечення для покращення екологічної ситуації в Україні та інших країнах.

Напрямки подальших досліджень можуть включати розширення функціональності додатку "Be Green", впровадження додаткових методів стимулювання участі користувачів та покращення алгоритму розподілу ресурсів.

Ключові слова: ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, ПЕРЕРОБКА ПОБУТОВИХ ВІДХОДІВ, ВЕБ-ДОДАТОК, ЕКОЛОГІЯ, СТИМУЛЮВАННЯ УЧАСТІ, ОПТИМІЗАЦІЯ РОЗПОДІЛУ РЕСУРСІВ, ХМАРНІ ТЕХНОЛОГІЇ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CSS – Cascading Style Sheets (каскадні таблиці стилів)
HTTP – Hypertext Transfer Protocol (протокол передачі гіпертексту)
HTML – HyperText Markup Language (мова розмітки гіпертексту)
IDEA – Integrated Development Environment for Applications (інтегроване середовище розробки програм)
MVC – Model-View-Controller (архітектурний шаблон для розробки програмного забезпечення)
ORM – Object-Relational Mapping (технологія, яка встановлює зв'язок між об'єктно-орієнтованою моделлю програми та реляційною базою даних)
PWC – PriceWaterhouseCoopers (компанія з аудиту та консалтингу)
URL – Uniform Resource Locator (уніфікований локатор ресурсів, веб-адреса)

ВСТУП

Сучасне суспільство щодалі більше стикається з численними екологічними проблемами. Одна з таких проблем – забруднення планети побутовими відходами, котрі виникають у процесі життєдіяльності самої людини. Нині кадрами островів пластику в океані, загиблих тварин, гір сміття в райських куточках планети нікого не здивувати. Це зворотній бік комфортного способу життя.

На жаль, проблема забруднення навколишнього середовища побутовими відходами є актуальною і для України. Величезні гори сміття на сміттєзвалищах бачив кожен із нас. До прикладу, міжнародна аудиторська компанія PWC у 2020 році оприлюднила дослідження, яке показало: на одного українця щороку припадає 230-330 кг побутових відходів. На всю країну це близько 10 млн тон сміття. Переробка сміття стає ключовим фактором уникнення забруднень побутовими відходами.

Переробка сміття розпочинається з його сортування. У Києві, наприклад, розміщені різні контейнери, куди можна помістити папір, скло, пластик. Проте обізнаність населення щодо правил первинного сортування побутових відходів залишається низькою, про що часто свідчать, наприклад, пластикова чи скляна тара, викинута у звичайні контейнери, коли поряд стоять спеціалізовані. Хоча саме правильне поводження з побутовими відходами на етапі сортування дозволяє у подальшому відправити їх на переробку. Якщо людство хоче, щоб планета була чиста, то кожен має докласти зусиль до цього і намагатися якомога більше сміття відправити на переробку. Переробка сміття зараз не популярність чи мода, а необхідність.

У даному проекті пропонується використовувати програмні рішення, а саме веб додаток, для підвищення обізнаності щодо розташування пунктів роздільного збору сміття, а також мотивувати населення до сортування сміття шляхом нарахування балів. Розподіл балів між особами, що наповнювали спеціалізований контейнер, ґрунтується на рівні довіри до користувача.

1 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ЯК ЗАСІБ ПОДОЛАННЯ ПРОБЛЕМ ПЕРЕРОБКИ ПОБУТОВИХ ВІДХОДІВ В УКРАЇНІ

1.1 Проблеми переробки побутових відходів в Україні

Управління побутовими відходами та їх переробка є актуальною проблемою не лише в Україні, але й у багатьох інших країнах світу. Забруднення довкілля та негативний вплив відходів на здоров'я людей ставлять під загрозу сталість та екологічну безпеку суспільства. Це стимулює наукове та практичне дослідження в галузі аналізу побутових відходів та пошук ефективних шляхів їх переробки.

Згідно звіту міжнародної аудиторської компанії PWC було виділено наступні завдання успішного впровадження реформи з управління відходами в Україні (подається у порядку пріоритетності)[1]:

1. впровадження обов'язкового роздільного збирання побутових відходів;
2. впровадження системи розширеної відповідальності виробника;
3. підвищення рівня обізнаності населення, просвітницька діяльність;
4. підвищення контролю та відповідальності у сфері поводження з відходами;
5. розробка карт поводження за напрямами відходів;
6. запобігання виникненню відходів та повторне використання;
7. розбудова та оновлення інфраструктури поводження з відходами;
8. забезпечення прозорості даних у сфері поводження з відходами.

Переробка сміття розпочинається з його сортування і роздільного збирання. Поміщати на переробку треба чисту тару, не забруднену органічними речовинами. На контейнерах, як правило, наводиться інформація, що можна здати на переробку, яке має бути маркування у такої тари.

Основними групами, на які можна розділити побутові відходи, є: пластик, папір/картон, скло, метал, біовідходи, одяг/взуття, змішане сміття (залишкове

сміття, яке не відноситься до попередніх фракцій). Окремо слід збирати небезпечні відходи.

Згідно [2], в Україні діють 17 підприємств із переробки макулатури, 39 – із переробки полімерів, 19 – із переробки ПЕТФ-сировини, 16 – із переробки склобою. Проте потужність підприємств із переробки полімерів та ПЕТ-пляшок, наприклад, завантажена тільки на 65%. Картонно-паперові заводи, склозаводи завантажені на 70%. Аби завантажити переробні підприємства хоча б на дві третини, бізнес змушений купувати сировину за кордоном.

За п'ять місяців 2021 року Україна імпортувала вторсировини на майже 1,2 млрд грн. Переробники зазначають, що продукція із вторинної сировини є дешевшою, ніж та, яка виробляється із первинної. Українській промисловості з переробки побутових відходів не вистачає сировини, а в той час мільйони тон її власного побутового сміття щороку вивозяться на полігони. Проблема полягає у тому, що в Україні немає постачальників, які на постійній основі можуть постачати вторсировину відповідної якості. За різними оцінками 30%-50% сміття на українських полігонах можна повторно переробити. Однак ставити сортувальні лінії на сміттєзвалищах нерентабельно, бо так відсортувати можна лише 10-15%. Якщо не забезпечити роздільний збір сміття, то полімери та папір стають непридатними для переробки. Закордонна сировина є зачасту кращої якості, оскільки там впроваджено більш ретельне сортування побутового сміття.

Станом на 2020 рік в Україні у 28 населених пунктах працювало 34 сортувальні лінії, у 17 населених пунктах будувалися сортувальні комплекси.

Лише у 1462 населених пунктах України впроваджується роздільне збирання побутових відходів, при кількості всіх населених пунктів близько 30000.

Отже, проблема переробки побутових відходів в Україні відрізняється низьким рівнем роздільного збору та переробки, а також недостатньою якістю вторинної сировини, що обмежує можливості повторного використання цих матеріалів. Розвиток проблеми пов'язаний зі зростанням обсягів сміття та відсутністю ефективних механізмів його управління. Протягом історії України було прийнято кілька нормативно-правових актів, спрямованих на вирішення цієї

проблеми. Одним із пріоритетних завдань є впровадження обов'язкового роздільного збирання побутових відходів, що передбачає забезпечення належної інфраструктури для збору та сортування різних відходів

Як можна бачити, однією з основних проблем переробки побутових відходів в Україні є низький відсоток відходів, які спрямовуються на переробку, та низька якість цієї сировини у зв’язку з відсутністю знань у населення щодо правил роздільного збору вторинної сировини. Переробки побутових відходів – це важливе соціально-екологічне явище, яке потребує негайних заходів та реформ для досягнення сталого та ефективного управління відходами. Велика кількість побутових відходів утворюється в нашому суспільстві, і неефективне їх управління призводить до серйозних негативних наслідків для навколишнього середовища та здоров'я людей.

1.2 Стратегії покращення переробки побутових відходів в Україні

1.2.1 Організація освітніх заходів та інформаційних кампаній для підвищення свідомості громади про переробку побутових відходів

Одним із способів підвищення свідомості громади є організація освітніх заходів, таких як семінари та лекції. Ці заходи можуть бути проведені в місцевих школах, університетах, громадських центрах та інших місцях, що залучають велику кількість учасників. Під час семінарів та лекцій фахівці з екології та утилізації відходів мають змогу пояснити мешканцям важливість сортування відходів та наслідки неправильної утилізації. Вони можуть розповісти про переваги переробки, такі як збереження природних ресурсів та зменшення негативного впливу на навколишнє середовище. Крім того, для ефективного поширення інформації про переробку відходів використовуються різноманітні інформаційні матеріали. Наприклад, розповсюдження брошур, плакатів та листівок, які містять інструкції щодо правильного сортування відходів за категоріями.

В епоху швидкого поширення інформації, також можна використовувати сучасні засоби комунікації для поширення інформації про переробку відходів. Соціальні мережі, веб-сайти та блоги є потужними інструментами для залучення уваги громади до цієї проблеми. Можна створити спеціальні сторінки або групи, де будуть публікуватись корисні поради, інформаційні матеріали, новини з питань переробки відходів. Також варто розглянути можливість співпраці з популярними блогерами або впливовими особистостями в соціальних мережах, щоб вони привертали увагу до проблеми переробки відходів та розповідали про свої особисті досвіди у цій сфері. Надзвичайно важливо створити зрозумілу та доступну інформацію щодо сортування відходів та викидання їх до контейнерів для переробки. Інструкції повинні бути лаконічними, ілюстративними та легко сприйнятними для широкого загалу. Детально пояснити, які відходи відносяться до паперу, пластику, скла, металу та інших категорій, та навести приклади конкретних предметів, які слід сортувати. Також варто наголосити на значенні чистоти та відсутності забруднень у сортованих відходах, оскільки це має прямий вплив на процес переробки та якість вторинної сировини.

1.2.2 Співпраця з місцевими органами влади для розміщення контейнерів для переробки відходів

Один із ключових аспектів співпраці з місцевими органами влади щодо розміщення контейнерів для переробки відходів полягає в забезпеченні можливості відслідковувати та мати актуальну інформацію про їх стан. Для досягнення цієї мети можна запропонувати використання сучасних технологій, наприклад, "розумних" контейнерів.

Розумні контейнери обладнані датчиками, які моніторять рівень заповненості відходами. Ця інформація передається безпосередньо до місцевої адміністрації, що дозволяє їм відстежувати стан контейнерів у режимі реального часу. Такий підхід має декілька переваг. По-перше, це дозволяє оптимізувати процеси збирання відходів, так як контейнери будуть порожнюватись за необхідністю, а не

за графіком. По-друге, це сприяє покращенню якості обслуговування, оскільки контейнери будуть порожнюватись своєчасно, запобігаючи їх переповненню.

Додатково, можна запропонувати встановлення системи сповіщень для мешканців, що допоможе забезпечити їх активну участь у сортуванні відходів. Наприклад, можна створити мобільний додаток, який нагадуватиме жителям про правильне сортування та розподілятиме інформацію про розміщення ближчих контейнерів для переробки. Такий підхід стимулюватиме активну участь громади у зборі та переробці відходів, а також дозволить місцевим органам влади отримувати фідбек від мешканців щодо якості обслуговування.

1.2.3 Залучення громади до переробки відходів через використання реклами та стимулів

Важливим аспектом успішної системи переробки відходів є залучення більшої кількості мешканців до активної участі у цьому процесі. Один із способів досягнення цієї мети - використання реклами та стимулів, які допоможуть залучити увагу та зацікавленість мешканців у справі переробки відходів.

Перший крок - це розміщення інформаційних плакатів, білбордів, оголошень у соціальних мережах та місцевих ЗМІ. Ці канали комунікації допоможуть поширити свідомість громади про важливість переробки відходів. На рекламних матеріалах підкресліть екологічні переваги переробки та позитивний вплив, який цей процес має на середовище. Використовуйте прості та зрозумілі повідомлення, щоб мешканці зрозуміли важливість своєї ролі в сортуванні та переробці відходів.

Додатково, встановлення системи стимулів для мешканців, які активно залучаються до сортування відходів. Це може бути програма нагородження, де громадськість має можливість отримувати призи або вигоди за свою активну участь у переробці відходів. Наприклад, розіграші призів серед учасників, знижки на комунальні платежі або інші форми винагороди. Це стимулюватиме мешканців до дотримання правил сортування та переробки відходів, а також підвищить рівень участі громади у цьому процесі

1.2.4 Розміщення контейнерів для сортування відходів та їх моніторинг

Для успішної системи переробки відходів важливо забезпечити належне розміщення контейнерів для сортування відходів у різних доступних місцях. Це дозволить зробити процес сортування відходів зручним для мешканців та підвищить рівень їх участі у цьому процесі. При розподілі контейнерів варто враховувати такі фактори, як доступність для громадськості та наявність великого потоку людей. Зокрема, важливо розміщувати контейнери для сортування відходів у місцях, які є популярними серед мешканців, таких як парки, площі, торгові центри та об'єкти загального користування. Це стимулюватиме мешканців до активної участі у сортуванні відходів, оскільки контейнери будуть легкодоступними та зручними для використання.

Для ефективного моніторингу контейнерів та забезпечення актуальної інформації про їх стан, можна використовувати онлайн-додаток. Цей додаток дозволить мешканцям відслідковувати стан контейнерів та взаємодіяти з ними. Він забезпечить можливість моніторингу заповненості контейнерів, їх регулярного обслуговування та вивезення. Громадяни зможуть за допомогою додатку перевіряти, чи є контейнери вільні для сортування відходів або чи потрібно їх вивозити та очищати. Такий моніторинг дозволить місцевим органам влади мати актуальну інформацію про стан контейнерів та оперативно реагувати на потреби громади.

Онлайн-додаток може бути доступним на мобільних платформах, що забезпечить його зручний доступ для мешканців. Крім того, додаток може містити корисну інформацію про правила сортування відходів, перелік прийнятних матеріалів та поради щодо ефективного утилізації відходів. Це стимулюватиме мешканців до активного використання додатку та дотримання правил сортування.

Застосування онлайн-додатку для моніторингу контейнерів та взаємодії з ними не лише полегшить процес сортування відходів для мешканців, але й допоможе місцевим органам влади збирати цінну інформацію. На підставі зібраних даних вони зможуть аналізувати ефективність системи переробки

відходів, виявляти проблемні ділянки та розробляти стратегії для подальшого покращення системи.

Узагалі, використання реклами та стимулів, а також впровадження онлайн-додатку для моніторингу контейнерів є важливими кроками у залученні громади до активної участі у сортуванні та переробці відходів.

1.3 Постановка задачі

Розповсюдженість смартфонів дозволяє забезпечити доступ кожному бажаючому до інформації щодо правил роздільного збору та місць розташування відповідних контейнерів з геолокацією, а нарахування певних бонусів чи балів, які у подальшому можуть бути певним чином обміняні або використані на товари чи знижки, дає можливість залучити та мотивувати населення до виконання певних дій. Отже, розробка програмного веб додатку з відповідними функціями дозволить суттєво покращити стан справ у цій сфері. Веб додаток — це взаємодія різного роду технологій на різних рівнях, які в результаті формують інформаційну систему для отримання та керування інформацією.

Метою роботи є розробка інформаційної системи «Be Green» для зберігання інформації про фізичне місцезнаходження контейнерів та зручного відображення її на карті, нарахування балів користувачам системи за наповнення відповідних контейнерів для збору вторсировини з метою залучити їх та мотивувати правильно поводитися з побутовими відходами. В інформаційній системі можливе редагування та різного роду активності з інформацією у залежності від типу користувача. Додаток буде розрізняти тип юзера (звичайний користувач, власник контейнера) та надавати різний функціонал та алгоритми дій відповідно. Додаток повинен надавати можливість реєстрації та авторизації користувачів, внесення інформації про контейнери, пошук контейнерів та відображення їх на карті, нарахування балів кінцевому користувачу, який зробив вклад у наповнення контейнера.

2 ПОШУК ТЕХНОЛОГІЙ ТА ДОСЛІДЖЕННЯ ВИМОГ ДЛЯ РОЗРОБКИ ВЕБ ДОДАТКУ «BE GREEN»

2.1 Дослідження потреб користувачів та функціональних вимог до додатку

Додаток "Be Green" має надавати можливість користувачам зареєструватися і створити обліковий запис для використання всіх функціональних можливостей. Після реєстрації користувачі зможуть авторизуватися в системі, використовуючи свої облікові дані. Цей процес забезпечує безпеку та конфіденційність даних користувачів та буде найбільш доречним для додатку. В майбутньому при зростанню попиту на додаток буде доцільно вдосконалити системи реєстрації та авторизації, включаючи використання біометричних даних, таких як відбиток пальця або розпізнавання обличчя, для ще більшої безпеки та зручності користувачів.

Користувачі додатку "Be Green" мають мати можливість додавати нову інформацію про контейнери для збору вторсировини. Вони зможуть ввести деталі, такі як місцезнаходження, розмір контейнера та інші важливі параметри. Це дозволить створити повний каталог контейнерів для користувачів з мінімальною інформацією, що спростить процес розробки. В майбутньому при відповідній підтримці можливе впровадження системи повідомлень та сповіщень для користувачів, яка буде надсилати інформацію про нові контейнери, оновлення даних та спеціальні пропозиції.

Додаток "Be Green" має надавати функціонал пошуку контейнерів для зручного використання користувачами. Вони зможуть шукати контейнери за місцезнаходження. Знайдені контейнери будуть відображені на карті, що допоможе користувачам знайти найближчі та найзручніші контейнери для здачі вторсировини.

Система "Be Green" має надавати різні функціональні можливості та алгоритми дій, залежно від типу користувача (звичайний користувач, власник контейнера). Звичайному користувачу буде доступний пошук контейнерів,

внесення вторсировини та отримання балів за свої внески. Власникам контейнерів надається додатковий функціонал для керування своїми контейнерами, такий як оновлення інформації, перевірка наповненості контейнера [3].

2.2 Дослідження вимог до користувацького інтерфейсу додатку

Дослідження вимог до користувацького інтерфейсу додатку є важливим етапом розробки, оскільки успішний дизайн інтерфейсу визначає зручність взаємодії користувача з програмним забезпеченням. У цьому процесі використання фреймворків може значно полегшити розробку та впровадження вимог до інтерфейсу. Є декілька фреймворків для розробки користувацького інтерфейсу: Bootstrap, Foundation, Materialize. Однак, Bootstrap здобув велику популярність і визнання завдяки своїм перевагам.

Bootstrap є одним з найпопулярніших фреймворків для розробки користувацького інтерфейсу [4]. Він надає набір готових компонентів, шаблонів та стилів, які допомагають створювати сучасний та привабливий дизайн. Bootstrap забезпечує респонсивний дизайн, що означає, що інтерфейс автоматично адаптується до різних розмірів екранів, що забезпечує оптимальний вигляд на різних пристроях. Використання Bootstrap дозволяє ефективно впроваджувати вимоги до зовнішнього вигляду, компонентів, розміщення елементів та стилізації інтерфейсу.

Ринок шаблоних двигунів пропунує на вибір такі двигуни: Thymeleaf, Freemarker, Mustache. Однак, Thymeleaf має свої унікальні переваги, які виправдовують його вибір.

Thymeleaf - це шаблонний двигун для розробки веб-інтерфейсу в рамках Java-проектів [5]. Він надає можливості динамічного генерування веб-сторінок та легку інтеграцію з фронтендом. Thymeleaf дозволяє використовувати HTML-шаблони з додатковими атрибутами та директивами, що розширюють можливості стандартного HTML. Це дозволяє зручно вбудовувати змінні, умови, цикли та

інші динамічні елементи в шаблони сторінок. Використання Thymeleaf спрощує впровадження вимог до динамічного контенту, даних з сервера.

Використання фреймворків Bootstrap і Thymeleaf сприяє швидкій і зручній розробці інтерфейсу додатку "Be Green" та дозволяє задовольнити вимоги до його зовнішнього вигляду, компонентів та динамічного контенту. Це сприяє поліпшенню взаємодії користувача з додатком і створює приємний та зручний користувацький досвід.

2.3 Забезпечення безпеки додатку

Дослідження вимог до забезпечення безпеки додатку є критичним для захисту конфіденційності, цілісності та доступності даних. На ринку існує кілька фреймворків, які пропонують рішення для забезпечення безпеки додатків. Проаналізувавши його, було прийнято рішення обрати фреймворк Spring Security для допомоги у виконанні цих вимог [6].

Apache Shiro, що є іншим популярним фреймворком безпеки для Java-додатків та має механізми аутентифікації, авторизації, керування сеансами та шифрування даних. Однак, Spring Security має переваги у розширюваності, інтеграції з іншими фреймворками Spring та широкою підтримкою спільноти розробників.

Також було розглянуто протокол авторизації OAuth, який використовується для зовнішньої авторизації та дозволу доступу до ресурсів. Хоча Spring Security також підтримує протокол OAuth, він має більше можливостей і розширюваності у встановленні рівнів доступу, управлінні ролями та правами, а також інтеграції з іншими механізмами безпеки.

У порівнянні з конкурентами, Spring Security виграє завдяки своїм перевагам. Spring Security надає повноцінні механізми аутентифікації, авторизації, керування сеансами, захисту від атак та інших аспектів безпеки. Він дозволяє гнучко налаштовувати правила безпеки та інтегруватись з різними фронтенд і бекенд технологіями та ідеально поєднується з іншими фреймворками Spring, такими як

Spring MVC, Spring Boot та іншими. Це забезпечує єдиний стек технологій для розробки та забезпечення безпеки додатку.

Враховуючи ці переваги, Spring Security є вигідним вибором для додатку "Be Green".

2.4 Дослідження ефективного алгоритму розподілу балів серед осіб, що зробили внесок у контейнер

Досліди ефективного алгоритму розподілу балів серед осіб, які зробили внесок у контейнер, можуть включати використання принципів алгоритму Nash Bargaining Solution. Nash Bargaining Solution є концепцією з теорії ігор та економічної теорії, яка допомагає вирішити проблему розподілу ресурсів або вигоди між учасниками.

Цей алгоритм дозволяє визначити справедливий розподіл, забезпечуючи оптимальний результат для всіх сторін. В контексті розподілу балів за внесок у контейнер, Nash Bargaining Solution може враховувати такі фактори, як кількість внесків, якість внесків, час, витрачений на внесок та інші параметри, які впливають на визначення ваги кожного внеску [7].

Використання Nash Bargaining Solution допомагає забезпечити справедливий розподіл балів, зважаючи на внески та вартість, яку кожна особа принесла в контейнер. Алгоритм дозволяє досягти ефективного рішення, яке задовольняє інтереси всіх сторін та сприяє мотивації для подальшого внеску у контейнер.

Таким чином, дослідження та використання алгоритму Nash Bargaining Solution у контексті розподілу балів допомагає вирішити проблему справедливого розподілу інцентивів серед осіб, які вносять внесок у контейнер. Це дозволяє створити стимули для активної участі користувачів та підтримує ефективну роботу додатку «Be Green».

2.5 Підготовка до розгортання додатку в хмарі

На ринку для розгортання веб-додатків існують декілька хмарних платформ.

Amazon Web Services (AWS) Elastic Beanstalk: Elastic Beanstalk є хмарною платформою, яка дозволяє легко розгортати, масштабувати та керувати веб-додатками. Вона підтримує різні мови програмування і надає багато інструментів для автоматизації та керування інфраструктурою. AWS Elastic Beanstalk має більш широкий спектр можливостей і інтеграцій з іншими сервісами AWS.

Google Cloud Platform (GCP) App Engine: App Engine є сервісом хмарного розгортання, який дозволяє розробникам створювати і масштабувати веб-додатки безпосередньо на інфраструктурі Google Cloud. Він надає велику гнучкість і автоматичне масштабування ресурсів, а також підтримує різні мови програмування. GCP App Engine має сильну інтеграцію з іншими сервісами Google Cloud.

Кожна з цих хмарних платформ має свої переваги і можливості, і вибір залежить від потреб розробника і особливостей проекту. Однак, Heroku відзначається деякими особливостями, які видають його конкурентну перевагу

Heroku – хмарна PaaS-платформа, що підтримує низку мов програмування. Heroku, одна з перших хмарних платформ, з'явилася в червні 2007 року. На серверах Heroku використовують операційні системи Debian або Ubuntu.

Heroku надає зручний спосіб розгортання веб-додатків. Можливе використання Git для надсилання свого коду на сервери Heroku, і вони автоматично збережуть, збудують і розгорнуть додаток.

Heroku дозволяє легко масштабувати додатки в залежності від потреб. Можливе збільшення або зменшення кількість ресурсів, таких як робочі процеси та бази даних, для забезпечення оптимальної продуктивності додатка.

Heroku надає інтуїтивний інтерфейс користувача, що спрощує налаштування і управління додатками. Через веб-консоль Heroku можливе керування додатками, переглядання журналів, налаштування змінних середовища та багато іншого.

Heroku надає різноманітні додаткові можливості для покращення додатку. Це включає можливість додавати аддони, такі як бази даних, кешування, логування, моніторинг тощо. Також можливе використання розширених інструментів, такі як Heroku CLI, для автоматизації задач і розробки в локальному середовищі.

Heroku надає сильну підтримку Git [8]. Можливе підключення репозиторію Git до Heroku та автоматичне розгортання зміни з кожним комітом. Це дозволяє швидко впроваджувати зміни та підтримувати неперервну доставку.

Heroku надає можливості автоматичного масштабування, які дозволяють динамічно збільшувати або зменшувати кількість ресурсів в залежності від обсягу трафіку та навантаження на додаток. Це допомагає забезпечити стабільну продуктивність та швидке реагування на змінні обсяги роботи.

Загалом, Heroku є потужним та зручним хмарним сервісом для розгортання та керування веб-додатками. Він надає широкі можливості інтеграції, масштабування та автоматизації, що допомагають розробникам швидко впроваджувати та керувати своїми додатками. [9]

ЗРОЗРОБКА ПРОГРАМНОГО ДОДАТКУ «BE GREEN»

3.1 Аналіз досліджених технологій для розробки додатку «be green»

3.1.1 Шаблонний двигун Thymeleaf

Thymeleaf є потужним інструментом для розробки веб-інтерфейсу в Java-проектах. Він надає можливості генерувати HTML-код з використанням шаблонів та інтегрується з логікою додатку. Ось кілька прикладів використання Thymeleaf для розробки інтерфейсу (Рисунок 3.1-3.4).

:

```
<p>
    Welcome, <span th:text="${user.name}"></span>!
</p>
```

Рисунок 3.1 – Вставка значень змінних

У цьому прикладі значення змінної user.name вставляється в тег . Під час відображення сторінки, значення буде підставлено в HTML.

```
<div th:if="${user.loggedIn}">
    <p>Welcome,
        <span th:text="${user.name}"></span>!
    </p>
</div>.
```

Рисунок 3.2 – Умовна видимість елементів

У цьому прикладі блок буде відображено тільки тоді, коли значення user.loggedIn дорівнює true. Це дозволяє контролювати видимість елементів в залежності від умов.

:

```
<ul>
    <li th:each="product : ${products}">
        <span th:text="${product.name}"></span>
    </li>
</ul>
```

.

Рисунок 3.3 – Цикли для повторення елементів

В цьому прикладі кожен елемент product зі списку products буде використаний для генерації окремого <li> елемента. Значення product.name вставляється в .

:

```
<a
        th:href="@{/products/{id}(id=${product.id})}"
        th:text="${product.name}"
></a>
```

.

Рисунок 3.4 – Використання атрибутів

У цьому прикладі атрибут th:href встановлює посилання на сторінку /products/{id} з параметром id, який береться з product.id. Значення product.name використовується як текст посилання.

3.1.2 Bootstrap фреймворк стилізація

Bootstrap є потужним фреймворком для стилізації інтерфейсу веб-додатків. Він надає готові компоненти, класи стилів та JavaScript-плагіни, які допомагають швидко та легко створювати привабливий та респонсивний дизайн. Ось кілька прикладів використання Bootstrap для стилізації інтерфейсу (Рисунок 3.5-3.6).

```
<button
        type="button"
        class="btn btn-primary"
>Click me</button>
```

Рисунок 3.5 – Створення кнопки

У цьому прикладі кнопка створюється за допомогою класів btn і btn-primary. Це надає кнопці стилізацію Bootstrap та встановлює колір фону.

```
<div class="row">
    <div class="col-md-6">Left column</div>
    <div class="col-md-6">Right column</div>
</div>.
```

Рисунок 3.6 – Використання сітки для розміщення елементів

У цьому прикладі два елементи розміщуються у рядок (row). Вони займають по половині ширини рядка на середньому (md) розмірі екрану. Сітка Bootstrap допомагає створити респонсивну розмітку для різних розмірів екранів.

3.1.3 Google API інтеграція

Для інтеграції карт у додаток використовуватиметься Google API. Google Maps API надає набір інструментів для відображення карт, маркерів, маршрутів та багато іншого. Ось кілька прикладів використання Google API для інтеграції карт (Рисунок 3.7-3.8) [10].

```
<div id="map"></div>
<script
        src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"
        async
        defer
></script>
<script> function initMap() {
    var mapOptions = {
        center: { lat: 40.712776, lng: -74.005974 }, zoom: 10
    };
    var map = new google.maps.Map(document.getElementById('map'), mapOptions); }
</script>
```

Рисунок 3.7 – Відображення карти

У цьому прикладі створено контейнер з ідентифікатором map, який буде використовуватись для відображення карти. Зовнішній JavaScript-файл, який завантажується з Google API, ініціалізує карту, встановлюючи центр і масштаб.

```
<script>
    function initMap() {
        var mapOptions = {
            center: { lat: 40.712776, lng: -74.005974 }, zoom: 10
        };
        var map = new google.maps.Map(
            document.getElementById('map'),
            mapOptions
        );
        var marker = new google.maps.Marker({
            position: { lat: 40.712776, lng: -74.005974 },
            map: map,
            title: 'New York City'
        });
    }
</script>
```

Рисунок 3.8 – Додавання маркерів

У цьому прикладі додається маркер на карту, встановлюючи його положення (position) і назву (title). Маркер буде відображений на карті.

3.1.4 Spring Boot сервера додатку

Spring Boot є фреймворком для розробки серверних додатків на мові Java. Він надає багато корисних інструментів та утиліт, які спрощують процес розробки та підтримки серверного коду. Ось кілька прикладів того, як Spring Boot допомагає з розробкою сервера додатку [11]:

Швидке створення сервера: Завдяки конфігурації за замовчуванням та автоматичному конфігуруванню, Spring Boot дозволяє швидко створити сервер додатку. Для створення сервера потрібно лише декілька рядків коду, наприклад (Рисунок 3.9).

```
3   import org.springframework.boot.SpringApplication;
4   import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6   @SpringBootApplication
7   public class MyApp {
8       public static void main(String[] args) {
9           SpringApplication.run(MyApp.class, args);
10      }
11  }
```

Рисунок 3.9 – Опис сервера

У цьому прикладі анотація @SpringBootApplication вказує Spring Boot на те, що цей клас є основним класом додатку. Метод main запускає додаток за допомогою SpringApplication.run.

Spring Boot має вбудований веб-сервер, що дозволяє запускати додаток безпосередньо без необхідності налаштування та розгортання зовнішнього веб-сервера. Можливе використовування вбудованого Tomcat, Jetty або Undertow. Це спрощує процес розробки та тестування серверного коду (Рисунок 3.10).

```java
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @GetMapping("/hello")
    public String hello() {
        return "Hello, World!";
    }
}
```

Рисунок 3.10 – Опис контроллера

У цьому прикладі створюється простий контролер, що містить метод hello(). Анотація @RestController вказує, що цей клас є контролером, який обробляє HTTP-запити. Анотація @GetMapping("/hello") вказує, що цей метод обробляє GET-запити на шлях "/hello". Метод просто повертає рядок "Hello, World!" як відповідь на запит.

3.1.5 ORM Hibernate фреймворк

Робота з базою даних за допомогою ORM (Object-Relational Mapping) дозволяє спростити і зручно взаємодіяти з базою даних без прямого написання SQL-запитів. ORM перетворює об'єкти програми на відповідні записи в базі даних і забезпечує доступ до цих записів через об'єктно-орієнтований інтерфейс[12]. Одним з популярних ORM-фреймворків для Java є Hibernate. Він надає потужні інструменти для взаємодії з реляційними базами даних.

Уникнення прямих SQL-запитів: Можлива робота з базою даних в термінах об'єктів і методів, що робить код більш зрозумілим та об'єктно-орієнтованим.

Автоматична генерація SQL: ORM-фреймворки автоматично генерують SQL-запити на основі моделей даних, що дозволяє уникнути ручного написання складних SQL-запитів.

Підтримка різних баз даних: ORM-фреймворки зазвичай підтримують різні типи баз даних, такі як MySQL, PostgreSQL, Oracle і багато інших. Це дозволяє легко змінювати тип бази даних, не змінюючи код додатку.

Кешування і оптимізація запитів: ORM-фреймворки зазвичай надають можливість кешування запитів і автоматичної оптимізації взаємодії з базою даних, що покращує продуктивність додатку.

3.1.6 Фреймворк безпеки додатку Spring Security

Spring Security фреймворк потрібний щоб додати бар'єр, який змусить відвідувача увійти до системи. Робиться це завдяки налаштуванням Spring Security у додатку. Spring Security автоматично захищає всі кінцеві точки HTTP за допомогою базової аутентифікації. Однак, є можливість додатково налаштувати параметри безпеки. За допомогою Maven потрібно додати два додаткові записи (одна для програми та один для тестування) в <dependency> в pom.xml (Рисунок 3.11).

```
95      <dependency>
96          <groupId>org.springframework.boot</groupId>
97          <artifactId>spring-boot-starter-test</artifactId>
98          <scope>test</scope>
99      </dependency>
100     <dependency>
101         <groupId>org.springframework.security</groupId>
102         <artifactId>spring-security-test</artifactId>
103         <scope>test</scope>
104     </dependency>
```

Рисунок 3.11 – Зразок опису залежностей в pom.xml

Потім створений клас WebSecurityConfig.java, який безпосередньо і є файлом, який буде надавати доступ (додаток А).

Використана анотацію @EnableWebSecurity, щоб включити підтримку веб безпеки Spring Security та забезпечити інтеграцію Spring MVC. Він також

розширює WebSecurityConfigurerAdapter і перевизначає кілька своїх методів, щоб встановити деякі особливості конфігурації веб безпеки.

Метод configure(HttpSecurity) визначає, які шляхи URL мають бути захищені, а які ні. Зокрема, шляхи "/app/auth/registration", "/static", "/app/auth/activate/*", "/css/**", "/js/**". налаштовані так, щоб не вимагати автентифікації. Всі інші шляхи мають бути автентифіковані.

Метод configure(AuthenticationManagerBuilder auth) налаштовує сховище користувача в пам'яті та зашифровує пароль юзера за допомогою passwordEncoder.

3.1.7 Алгоритм Nash Bargaining Solution

Алгоритм розподілу балів між учасниками доцільно побудувати на принципах, на яких ґрунтується справедливий поділ «пирога» теорії ігор. У процесі роботи над проектом було проаналізовано декілька алгоритмів розподілу теорії ігор [13, 14, 15, 16].

Для прикладу коротко розглянемо Nash Bargaining Solution. Nash Bargaining Solution – це концепція теорії ігор, яка визначає спосіб розподілу ресурсів між двома або більше особами або групами. Теорію розробив Джон Неш, який отримав Нобелівську премію з економіки за свій внесок до теорії ігор.

Nash Bargaining Solution передбачає що кожна окрема особа або група, яка бере участь у процесі розподілу, прагне максимізувати власну вигоду чи отримання виграшу. Вона надає рішення, яке є справедливим для всіх сторін.

Основна ідея Nash Bargaining Solution – це спосіб знайти угоду між двома сторонами в кооперативній грі. Вона передбачає, що обидві сторони хочуть максимізувати власну вигоду і намагається знайти результат, з яким обидві сторони можуть погодитися.

Nash Bargaining Solution базується на ідеї продукту Неша, який є продуктом комунальних послуг обох сторін. Nash Bargaining Solution — це результат, який максимізує добуток відмінностей між корисностями сторін та їхніми

відповідними розбіжностями. Пункт розбіжностей — це користь, яку отримує кожна сторона, якщо згоди не буде досягнуто.

Щоб скористатися Nash Bargaining Solution, сторони повинні узгодити свої відповідні згоди та свої розбіжності. Потім вони можуть використовувати формулу переговорного рішення Неша, щоб знайти результат.

Переговорне рішення Неша можна використовувати в багатьох сценаріях, таких як трудові переговори, ділові угоди та політичні переговори. Це дає змогу двом сторонам знайти справедливу угоду, яка максимізує їхні відповідні корисності. Однак для цього потрібно, щоб обидві сторони добре розуміли свої переваги та точки розбіжностей, які в деяких випадках може бути важко визначити.

3.1.8 Хмарні технології Heroku

Щоб підготувати та розгорнути Spring Boot додаток на хмарному сервісі Heroku за допомогою Git, знадобиться виконати кілька кроків.

Потрібно мати обліковий запис на Heroku. Якщо його немає, перейти на веб-сайт Heroku та зареєструйтесь. Потрібно переконайтеся, що на комп'ютері встановлений Git. Якщо ні, то завантажити Git з офіційного сайту Git. Опісля можна створити новий Git-репозиторій для Spring Boot проекту. Це можна зробити, виконавши команди git init у кореневій папці проекту. Необхідно налаштувати файл gitignore, створивши файл .gitignore у кореневій папці проекту і вказавши файли та папки, які потрібно ігнорувати при коміті коду до Git. Деякі типові рекомендовані елементи .gitignore для Spring Boot проектів можуть включати файли .classpath, .project, .settings/, .idea/, target/ та інші згенеровані файли та папки. Після цього можна комітити код до Git за допомогою команд ‘git add .’, та ‘git commit -m "Перший коміт"’.

Після цього можна увійті до облікового запису Heroku та створити новий додаток (app). Вибрати регіон та ім'я додатку. Обравши відповідний додаток (app) перейти до вкладки "Deploy" (Розгортання) у верхньому меню. У розділі

"Deployment method" (Спосіб розгортання) обрати опцію "GitHub" або "Git". Потім зв'язати облікові записи GitHub з Heroku та обрати репозиторій з проектом. Після вибору способу розгортання Heroku попросить надати доступ до репозиторію Git (наприклад, авторизація GitHub або введення облікових даних Git). Після успішного підключення репозиторію Heroku дозволить налаштувати спосіб розгортання. В розділі "Manual deploy" (Ручне розгортання) можна обрати гілку, яку потрібно розгорнути, натиснувши кнопку "Deploy Branch" (Розгорнути гілку).

Heroku розпочне процес розгортання Spring Boot додатку. Можливе спостерігання за процесом розгортання, переглядання журналів, помилки та іншу інформацію. Після успішного розгортання додаток буде доступний за URL, наданим Heroku.

3.2 Реалізація програмного додатку «Be Green»

3.2.1 Розробка шаблонів інтерфейсу для додатку «Be Green»

Було реалізовано сторінку використовуючи потужності Thymeleaf, який дозволяє використовувати динамічні дані та вирази у шаблонах. Основні використання Thymeleaf це вирази, які включаються в атрибути тегів з префіксом th:. Наприклад, вираз th:text="${message}" (Рисунок 3.12) вставляє значення змінної message у текстовий контент елемента <p>. Аналогічно, th:text="${userPoints}" та th:text="${userCurrency}" вставляють значення змінних userPoints і userCurrency в текст посилань.

```
18      <nav class="my-2 my-md-0 mr-md-3">
19          <p th:if="${message}" class="p-2 text-dark" th:text="${message}"></p>
20      </nav>
```

Рисунок 3.12 – Зразок вставки значення змінної message

Також були використані умовні вирази, тобто умовні перевірки та вставки вмісту в залежності від умови. Наприклад, вираз sec:authorize="hasAuthority('COMPANY')", який використовує атрибут

sec:authorize, перевіряє, чи має поточний користувач роль "COMPANY" (Рисунок 3.13). Залежно від результату перевірки, відповідний елемент <a> буде включений або виключений.

```
26      <nav sec:authorize="hasAuthority('COMPANY')">
27          <a class="p-2 text-dark" th:href="@{/app/currency/add}"><span th:text="${userCurrency}"></span>$</a>
28      </nav>
```

Рисунок 3.13 – Зразок використання ролі користувача

Thymeleaf дозволяє використовувати інтерполяцію значень, тобто вставляти значення змінних у різні атрибути тегів. Наприклад, th:href="@{/app/home}" вставляє значення шляху, яке буде залежати від значення змінної.

```
22      <nav>
23          <a class="p-2 text-dark" th:href="@{/app/home}"><span th:text="${userPoints}"></span> Points</a>
24      </nav>
```

Рисунок 3.14 – Зразок вставки посилань

Використання фрагментів: Thymeleaf надає можливість використовувати фрагменти коду, що можна включати в різні місця сторінки. Наприклад, у коді присутній фрагмент <script th:src="@{/js/button-spin.js}"></script>, який вставляє шлях до скрипту button-spin.js (Рисунок 3.15).

```
129     <script th:src="@{/js/button-spin.js}"></script>
130     <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
131             integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
132             crossorigin="anonymous"></script>
133     <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"
```

Рисунок 3.15 – Зразок використання бібліотек

3.2.2 Стилізація сторінок додатку «Be Green»

У реалізованому коді HTML використовується фреймворк Bootstrap для стилізації інтерфейсу. У коді використовуються класи CSS в різних елементах HTML, такі як d-flex, flex-column, align-items-center, mb-3, bg-white, border-bottom, box-shadow, my-0, font-weight-normal, px-md-4, card, card-header, card-body, list-unstyled, btn, btn-success, btn-lg (Рисунок 3.16).

```
<div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom box-shadow">
    <h5 class="my-0 mr-md-auto font-weight-normal">Be Green</h5>
    <nav class="my-2 my-md-0 mr-md-3">
        <p th:if="${message}" class="p-2 text-dark" th:text="${message}"></p>
    </nav>
```

Рисунок 3.16 – Зразок застосування стилів

Ці класи надають стилізацію елементам і визначають їх розташування, вигляд і поведінку. Також використовується елемент <nav> з Bootstrap для створення навігаційного меню. Класи my-2, my-md-0, mr-md-3, p-2, text-dark визначають вигляд та розташування елементів меню. Використовуються кнопки Bootstrap з класами btn, btn-outline-success, btn-block, to-spin. Вони надають стилізацію кнопкам та визначають їхню поведінку.

Використовується сітка Bootstrap з класами container, row, col-12, col-md, яка розташовує елементи на сторінці відповідно до заданої сітки (Рисунок 3.17). Bootstrap дозволяє швидко та зручно стилізувати елементи інтерфейсу, використовуючи готові класи CSS та компоненти Bootstrap.

```
    <div class="row">
        <div class="col-12 col-md">
            <img class="mb-2" th:src="@{/images/app-icon.png}" alt="" width="24"
                 height="24">
            <small class="d-block mb-3 text-muted">© 2022</small>
        </div>
        <div class="col-6 col-md">
            <h5>Features</h5>
```

Рисунок 3.17 – Зразок застосування розмітки через класи

3.2.3 Відображення карт в додатку «Be Green»

Був реалізований Java Script код, який використовує Google API для роботи з картами та геокодуванням. Створена фунуція initMap (Рисунок 3.18), яка ініціалізує карту Google з використанням google.maps.Map. Вона встановлює початкову позицію та зум карти при налаштуваннях center і zoom. Також вона викликає інші функції для додавання кнопок та маркерів на карту.

```
11  function initMap() {
12      let loading_message = document.getElementById( elementId: "loading");
13      let spinner = document.getElementById( elementId: 'spinner');
14      loading_message.textContent = "loading your location...";
15      spinner.style.display = 'block';
16
17      map = new google.maps.Map(document.getElementById( elementId: "map"), {
18          center: start_map_position,
19          zoom: 3,
20      });
21
22      geocoder = new google.maps.Geocoder();
23
24      addShowMeButton();
25
26      createNewMarker();
27
28      showMarkersByUrl( urlPath: '/map/marker/all/visible');
29
30      showMe();
31
32  }
```

Рисунок 3.18 – Зразок функії init map

Також була реалізована функція geocodeLatLng (Рисунок 3.19). Ця функція використовує google.maps.Geocoder для отримання адреси на основі вибраної позиції маркера. Вона викликає метод geocode() та обробляє результат, щоб отримати форматовану адресу, яка потім встановлюється в поле з ідентифікатором "address".

```
34  function geocodeLatLng() {
35      let latLng = selected_marker_position
36
37
38      geocoder
39          .geocode({location: latLng})
40          .then((response) => {
41              if (response.results[0]) {
42                  document.getElementById( elementId: 'address').value = response.results[0].formatted_address;
43              } else {
44                  console.log("No results found");
45              }
46          })
47          .catch((e) => console.log("Geocoder failed due to: " + e));
48
49  }
```

Рисунок 3.19 – Зразок функції geocodeLatLng

Також була реалізована функція createNewMarker() (Рисунок 3.20). Ця функція створює новий маркер google.maps.Marker та приховує його. Вона також додає обробник подій click на карту, який відображає маркер на вибраній позиції та створює кнопку "Add new" через google.maps.InfoWindow. moveMarker(): Ця функція переміщає маркер на обрану позицію та викликає setLatLng() для оновлення координат маркера. Вона також закриває відкрите вікно google.maps.InfoWindow.

```
function createNewMarker() {
    new_marker = new google.maps.Marker({
        position: start_map_position,
        map: map
    });
    new_marker.setVisible(false);

    map.addListener( listener: "click", (e) => {
        new_marker.setVisible(true);
        closeInfoWindow(create_button);
        moveMarker(e.latLng, map, new_marker)
        cleanErrorMessage();

        create_button = new google.maps.InfoWindow({
            content: "<button class=\"btn btn-success to-spin\" onclick=\"addNewMarker()\">Add new</button>",
        });
        create_button.open(map, new_marker);
    });
}
```

Рисунок 3.20 – Зразок функції createNewMarker

Була реалізована асинхронна функція getById виконує запит до сервера за допомогою fetch() для отримання додаткової інформації про контейнер на основі переданого ідентифікатора. Отримані дані використовуються для створення HTML-коду, який додається до відповідного блоку.

Також була реалізована функція showMarkersByUrl (Рисунок 3.21). Ця асинхронна функція виконує запит до сервера за допомогою fetch() для отримання списку маркерів. Отримані дані використовуються для створення маркерів google.maps.Marker на карті.

```
async function showMarkersByUrl(urlPath) {
    let more_info
    const url = ctx + urlPath;

    try {
        const response = await fetch(url);
        const data = await response.json();
        const markers = [];

        Array.from(data).forEach(
            function createMarker( markerEntity ) {
                let position = new google.maps.LatLng(markerEntity.lat, markerEntity.lng);
                let title = markerEntity.name;
                let containerId = markerEntity.containerId;
                let address = markerEntity.address;
                let des = markerEntity.description;

                const marker = new google.maps.Marker({
                    position,
                    title
                });

                marker.addListener( listener: "click", () => {
                    closeInfoWindow(more_info);

                    more_info = new google.maps.InfoWindow({
                        content: "<div class=\"card\" style=\"width: 18rem;\">\n" +
                            "  <div class=\"card-body\">\n" +
                            "    <h5 class=\"card-title\">" + marker.getTitle() + "</h5>\n" +
                            "    <h6 class=\"card-subtitle mb-2 text-muted\">" + address + "</h6>\n" +
                            "    <p class=\"card-text\">" + des + "</p>\n" +
                            "<button class=\"btn btn-success to-spin\" onclick=\"getById('" + containerId + "')\">More info</button>" +
                            "  </div>\n" +
                            "</div>",
                    });
                    more_info.open(map, marker);
                });
                markers.push(marker)
            }
        );
        new MarkerClusterer(map, markers);
    } catch (error) {
        console.log(error)
    }
}
```

Рисунок 3.21 – Зразок функції show marker by url

Також була реалізована addNewMarker (Рисунок 3.22) функція. Ця функція викликається при натисканні кнопки "Add new" в google.maps.InfoWindow. Вона відправляє форму з ідентифікатором "set-marker", яка містить координати нового маркера, на сервер для обробки.

```
function addNewMarker() {
    let form = document.querySelector( selectors: '#set-marker');
    form.submit();
}
```

Рисунок 3.22 – Зразок функції add new marker

Також була реалізована showMe (Рисунок 3.23) функція. Ця функція викликається при натисканні кнопки "Show me" і використовує

navigator.geolocation для отримання поточної локації користувача. Якщо локація доступна, вона встановлює центр карти відповідно до отриманої локації та відкриває вікно google.maps.InfoWindow з повідомленням "Location found".

```
function showMe() {
    const spinner = document.getElementById( elementId: 'spinner');
    spinner.style.display = 'block';
    let infoWindow = new google.maps.InfoWindow();
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(
            successCallback: (position : GeolocationPosition ) => {
                const pos = {
                    lat: position.coords.latitude,
                    lng: position.coords.longitude,
                };

                infoWindow.setPosition(pos);
                infoWindow.setContent("Location found.");
                infoWindow.open(map);
                map.setCenter(pos);
                map.setZoom(16);
                let loading_message = document.getElementById( elementId: "loading");
                loading_message.textContent = "";
                const spinner = document.getElementById( elementId: 'spinner');
                spinner.style.display = 'none';
            },
            errorCallback: () => {
                handleLocationError( browserHasGeolocation: true, infoWindow, map.getCenter());
            }
        );
    } else {
        handleLocationError( browserHasGeolocation: false, infoWindow, map.getCenter());
    }
}
```

Рисунок 3.23 – Зразок функції show me

Також була реалізована функція addShowMeButton (Рисунок 3.24). Ця функція додає кнопку "Show me" на карту. Вона прислуховується до події "click" на кнопці та викликає функцію showMe(). handleLocationError(): Ця функція викликається, якщо виникає помилка при отриманні локації користувача. Вона встановлює повідомлення про помилку в google.maps.InfoWindow і відкриває його на карті.

```
237  function addShowMeButton() {
238      const locationButton = document.getElementById( elementId: "show-me");
239
240      locationButton.addEventListener( type: "click",  listener: () => {
241          showMe(map)
242      });
243  }
```

Рисунок 3.24 – Зразок функції add show me button

У коді також є використання інших функцій та об'єктів звичайного JavaScript, таких як getElementById(), addEventListener(), innerHTML, textContent, fetch(), json(), а також використання HTML-елементів та атрибутів для відображення та взаємодії з веб-сторінкою. Код використовує Google API для створення та взаємодії з картами Google, отримання адреси на основі координат, відображення маркерів на карті, обробки подій та виконання запитів до сервера для отримання та відображення додаткової інформації про контейнери.

3.2.3 Розробка серверної сторони додатку «Be Green»

Серверна сторона додатку реалізована за допомогою фреймворка Spring. Автори Spring вирішили надати розробникам деякі утиліти, які автоматизують процедуру налаштування та прискорюють процес створення та розгортання Spring-додатків, під загальною назвою Spring Boot. Spring Boot - це корисний проект, метою якого є спрощення створення програм на основі Spring. Spring Boot має значний функціонал, але його найбільш значущими особливостями є управління залежностями, автоматична конфігурація та вбудовані контейнери сервлетів. Щоб прискорити процес управління залежностями, Spring Boot неявно пакує необхідні сторонні залежності для кожного типу програми на основі Spring і надає їх розробнику за допомогою так званих starter-пакетів (spring-boot-starter-web, spring-boot-starter-data-jpa і т.д. .Д.)

3.2.4 Опис сутностей та взаємодія з базою даних для додатку «Be Green»

Для початку було створено схему реляційної бази данних для надання можливості зберігання всієї необхідної інформації в рамках поставлених потреб (Рисунок 3.25).

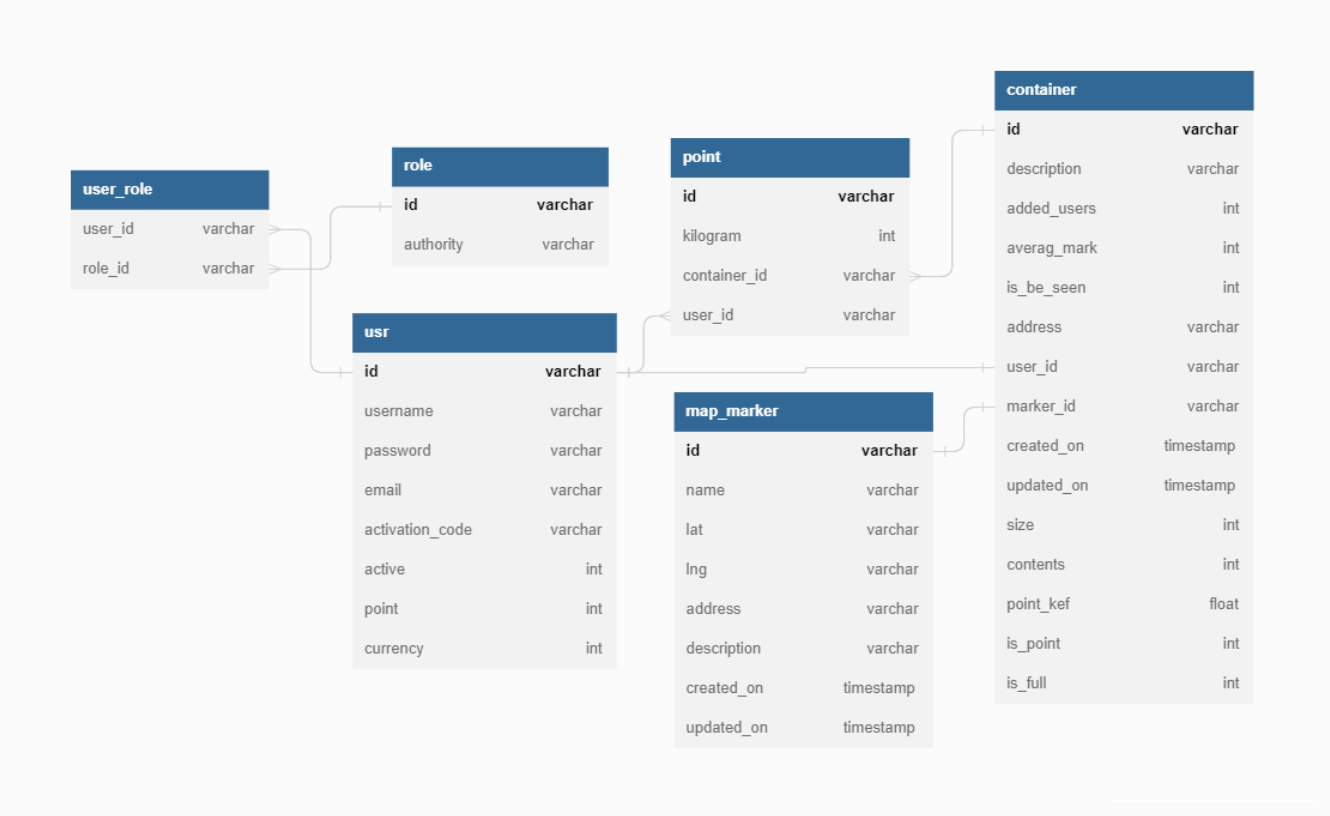

Рисунок 3.25 – Схема розробленої реляційної бази данних

Потім було описано декілька Entity використовуючи анотацію @Entity. Зв’язано таблицю з назвою usr з класом UserEntity за допомогою анотації @Table. Клас UserEntity має поля id з анотацією @Id та генераціями в анотаціях @GeneratedValue та @GenericGenerator Зв’язано з однойменною колонкою id з таблиці usr анотацією @Column. Поля username(string), password(string), email(string), activationCode(string) та active(bool). Set<ContainerEntity> containers який використовує анотацію @OneToMany. Це означає, що одному об’єкту user буде відповідати декільком контейнерам. Set<Role> roles з ролями з анотаціями @ElementCollection де вказано об’єкт нашої колекції Role. @CollectionTable що вказує на зв’язок з таблицею з назвою user_role та в параметрах анотація

@JoinColumn з назвою зв’язаної колонки user_id @Enumerated де вказаний тип перерахування (додаток Б) [17].

Створено MarkerEntity. Зв’язано таблицю з назвою map_marker з класом MarkerEntity за допомогою анотації @Table. Клас MarkerEntity має поля id з анотацією @Id та генераціями в анотаціях @GeneratedValue та @GenericGenerator. Поля name(string) зв’язані з однойменними колонками, lat(String)-широта, lng(string)-довгота, address(string), description(Integer). ContainerEntity container з анотацією @OneToOne що означає кожен об’єкт класу MarkerEntity має зв’язок з одним об’єктом ContainerEntity. Дві змінні для зберігання часу: коли було створено маркер та коли було останній раз модифіковано-createdOn, updatedOn (додаток В).

Створено ContainerEntity. Зв’язано таблицю з назвою container з нашим класом ContainerEntity за допомогою анотації @Table. Клас ContainerEntity має поля id з анотацією @Id та генераціями в анотаціях @GeneratedValue та @GenericGenerator. Поля name(string), description(string), addedUsers(string), activationCode(string), averageMark(Integer), isBeSeen(boolean), address(String).

Дві змінні для зберігання часу коли було створено контейнер та коли було останній раз модифіковано (createdOn, updatedOn). Зв’язано поле MarketEntity marker з колонкою marker_id (додаток Г).

Репозиторій для маркерів MarkerRepository з анотацією @Repository який доповнює реалізацію інтерфейсу JpaRepository<MarkerEntity, String> з об’єктами типу MarkerEntity (Рисунок 3.26).

```
249     @Repository
250     public interface MarkerRepository extends JpaRepository<MarkerEntity, String> {
251     }
252
```

Рисунок 3.26 – Файл MarkerRepository.java

Репозиторій для контейнерів ContaineRerository з анотацією @Repository, який доповнює реалізацію інтерфейсу JpaRepository<ContainerEntity, String> з об’єктами типу ContainerEntity з методами findByAddress(String address) та findByIsBeSeenTrue() (Рисунок 3.27).

```
253     @Repository
254 public interface ContainerRepository extends JpaRepository<ContainerEntity, String> {
255     Optional<ContainerEntity> findByAddress(String address);
256     List<ContainerEntity> findByIsBeSeenTrue();
257 }
```

Рисунок 3.27 – Файл ContainerRepository.java

3.2.5 Зберігання користувачів та обробка аутентифікації в додатку «Be Green»

Зазвичай Spring Security співпрацює з юзерами через пам'ять, але для цієї реалізації було прийнято рішення написати сервіс, який буде працювати з юзерами через базу даних. Для цього було створено клас UserServiceImpl (додаток Ґ).

В коді змінено взаємодію з юзерами, а саме витягування з бази даних, пошук юзерів через юзернейм, код активації, додавання нового юзеру, видалення профілю, його ж модифікація, збереження юзера, а також описаний метод, який буде надсилати листівку запрошення на пошту під час реєстрації. Також UserRepository відповідає за зв'язок з базою даних, а саме він витягує поля юзера з бази, і дозволяє взаємодіяти з цими полями іншим сервісам. Також використовую інтерфейс UserRepository, він є наслідником JpaRepository, методи findByUsername та findByActivationCode якого використовується для пошуку потрібних нам юзерів (Рисунок 3.28).

```
42 public interface UserRepository extends JpaRepository<UserEntity, String> {
43     Optional<UserEntity> findByUsername(String username);
44     Optional<UserEntity> findByActivationCode(String code);
45 }
```

Рисунок 3.28 – Файл UserRepository.java

3.2.6 Розробка розподілу балів між особами, які зробили вклад у контейнер в додатку «Be Green».

Для початку було реалізовано функцію [18] (Рисунок 3.29), яка отримує три параметри: цілочисельний масив playerScores, що представляє порядність

кожного користувача, цілочисельний масив playerKgs, що представляє вагу зданих кілограм кожного користувача, та ціле число numPlayers, що представляє кількість користувачів у грі. Функція повертає цілочисельний масив allocation, що представляє розподіл для гри у одиницях. Він обчислюється множенням порядності кожного користувача на вагу, яку було здано. Функція спочатку ініціалізує масив allocation нулями за допомогою методу Arrays.fill(). Потім вона проходиться по кожному користувачу і обчислює їхній розподіл за допомогою вищезгаданої формули. У кінці функція повертає отриманий масив allocation:

```
17 @    public static int[] findNashLikeAllocation(
18              int[] playerScores,
19              int[] playerKgs,
20              int numPlayers
21      ) {
22          int[] allocation = new int[numPlayers];
23          Arrays.fill(allocation, val: 0);
24          for (int i = 0; i < numPlayers; i++) {
25              allocation[i] += playerScores[i] * playerKgs[i];
26          }
27          return allocation;
28
29      }
```

Рисунок 3.29 – Зразок функції find nash like allocation

Також було реалізовано метод під назвою divideByNashLikeAllocation (Рисунок 3.30), Цей метод приймає ціле число для ділення (число балів) та масив розподілів, який обчислюється попередньою функцією. Він обчислює відсоток загального розподілу для кожного користувача та використовує цей відсоток, щоб розділити ціле число на різні частини для кожного користувача на основі їх розподілу. Зазначений метод повертає масив цілих чисел, який містить розділене ціле число для кожного користувача на основі їх розподілу. Спершу він ініціалізує цілочисельний масив під назвою 'dividedArray' такої ж довжини, як і масив розподілу, сформований. Потім він обчислює суму всіх розподілів користувачів за допомогою циклу. Потім для кожного розподілу обчислюється відсоток від загального розподілу для цього користувача за формулою:

розподіл/сума розподілів. Потім він множить ціле число балів, яке потрібно розділити на цей відсоток, щоб отримати бал для призначення користувачу на основі його розподілу. У кінці округлюється результат до найближчого цілого числа за допомогою методу Math.round() [19] і зберігається по відповідному індексу у dividedArray. Метод повертає dividedArray:

```
31 @    public static int[] divideByNashLikeAllocation(int integerToDivide, int[] NashLikeAllocation) {
32          int[] dividedArray = new int[NashLikeAllocation.length];
33          int sumOfAllocations = 0;
34          for (int allocation : NashLikeAllocation) {
35              sumOfAllocations += allocation;
36          }
37          for (int i = 0; i < NashLikeAllocation.length; i++) {
38              double percentage = (double) NashLikeAllocation[i] / sumOfAllocations;
39              dividedArray[i] = (int) Math.round(integerToDivide * percentage);
40          }
41          return dividedArray;
42      }
```

Рисунок 3.30 – Зразок функції divide by nash like allocation

Таким чином, даний масив містить бали, розподілені від загальної суми для кожного користувача відповідно до їх внеску у контейнер та рівня порядності.

3.2.7 Розгортання додатку «Be Green» в хмарному середовищі

Використовано хмарну базу даних PostgreSQL [20] та завантажано додаток на Heroku. Після розгортання Heroku перекидає на сторінку додатку (Рисунок 3.31)

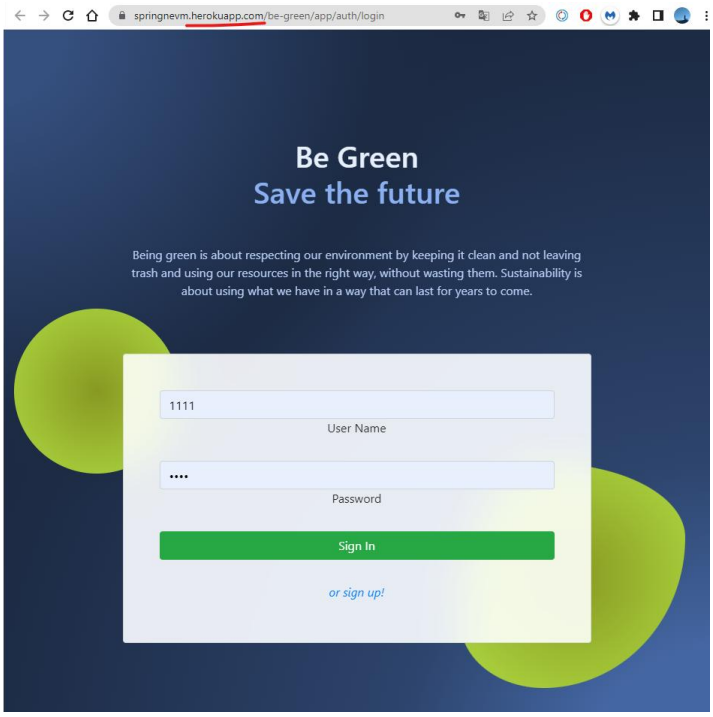

Рисунок 3.31 – Домашня сторінка додатку, розгорнутого на Heroku

3.3 Демонстрація додатку «Be Green»

Додаток має сторінку реєстрації (Рисунок 3.32), авторизації (Рисунок 3.33), головне меню, особистий кабінет та мапу з контейнерами. Коли користувач потрапляє на сайт, йому одразу пропонують зареєструватись або увійти вже як користувачу. На сторінці реєстрації є такі поля як ім`я, email, пароль та роль адміністратора/користувача. Всі поля треба заповнити обов`язково. Після цього на пошту прийде код підтвердження реєстрації.

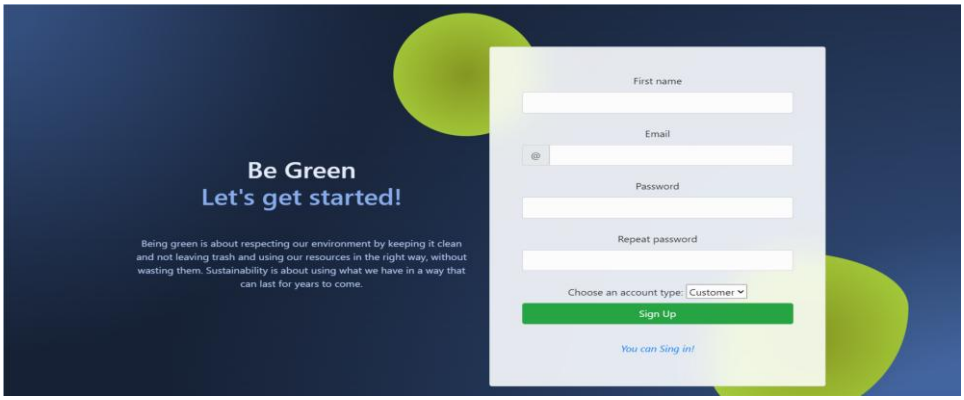

Рисунок 3.32 – Зразок сторінки реєстрації

Зареєстрованим користувачам треба виконати вхід, ввести лише зареєстроване ім`я та пароль. Після входу користувач опиняється на головній сторінці (Рисунок 3.34), де може побачити привітання, та має можливість переглянути вже існуючі контейнери або додати новий (для власників контейнерів). Якщо натиснути кнопку «переглянути контейнери», то здійсниться перехід на сторінку, де виведеться список усіх існуючих контейнерів (Рисунок 3.35). Власник має право вимкнути свій контейнер, редагувати інформацію про нього, а також з цієї сторінки може додати ще один контейнер на мапу.

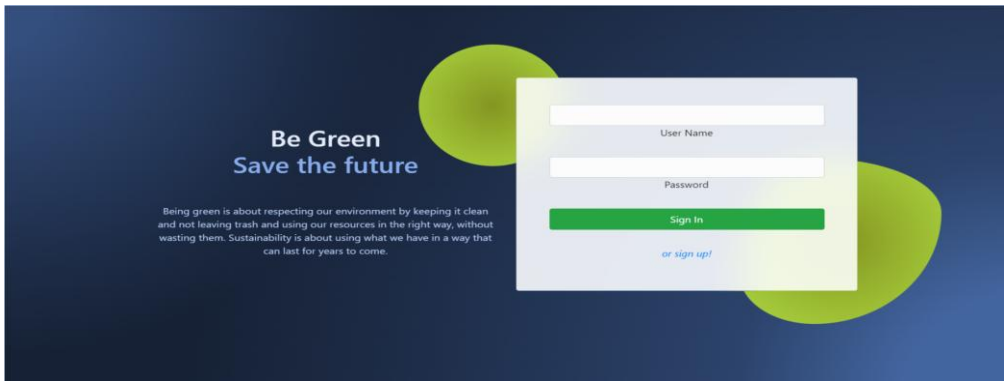

Рисунок 3.33 – Зразок сторінки авторизації

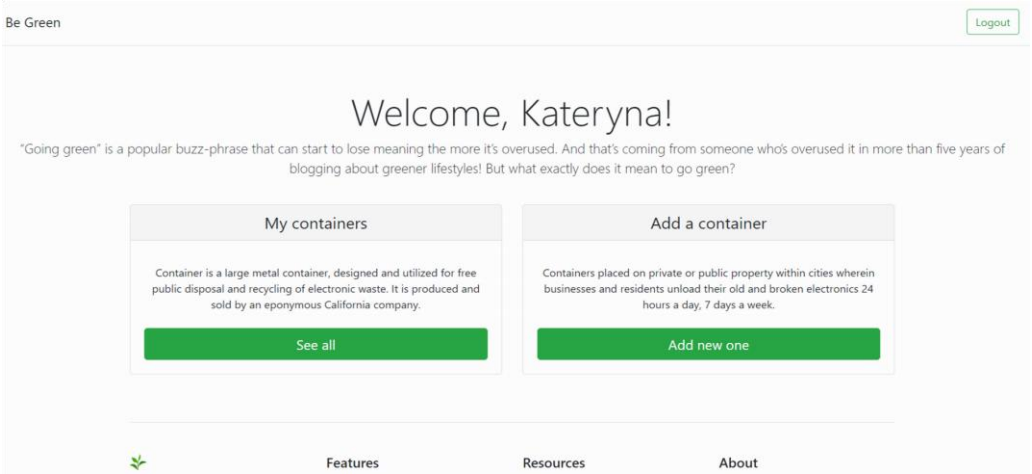

Рисунок 3.34 – Зразок головної сторінки

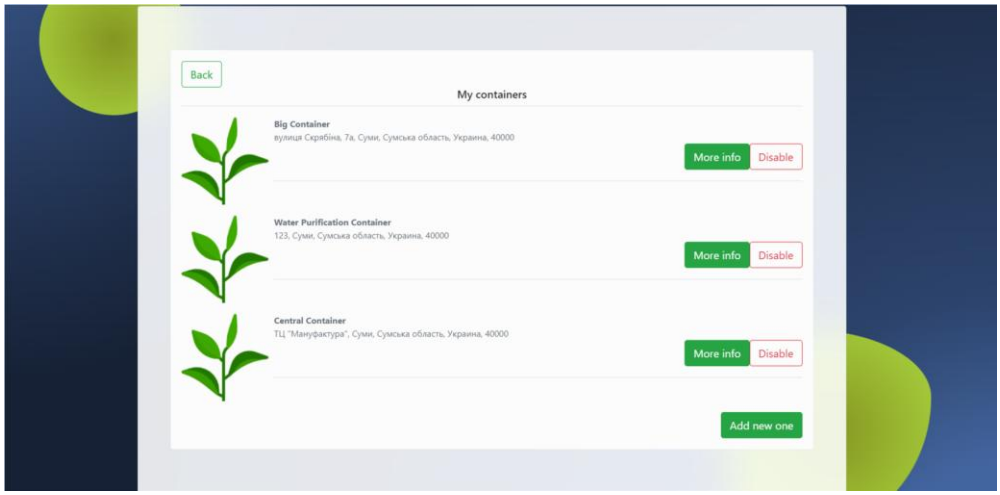

Рисунок 3.35 – Зразок сторінки з контейнерами

Коли користувач хоче додати контейнер, він переходить на сторінку з мапою (Рисунок 3.36). Необхідно обрати або дозволити автоматично обрати локацію користувача, щоб додати контейнер у своєму місті. На сторінці також є можливість пошуку контейнера по назві та можливість переглянути розташування власних контейнерів.

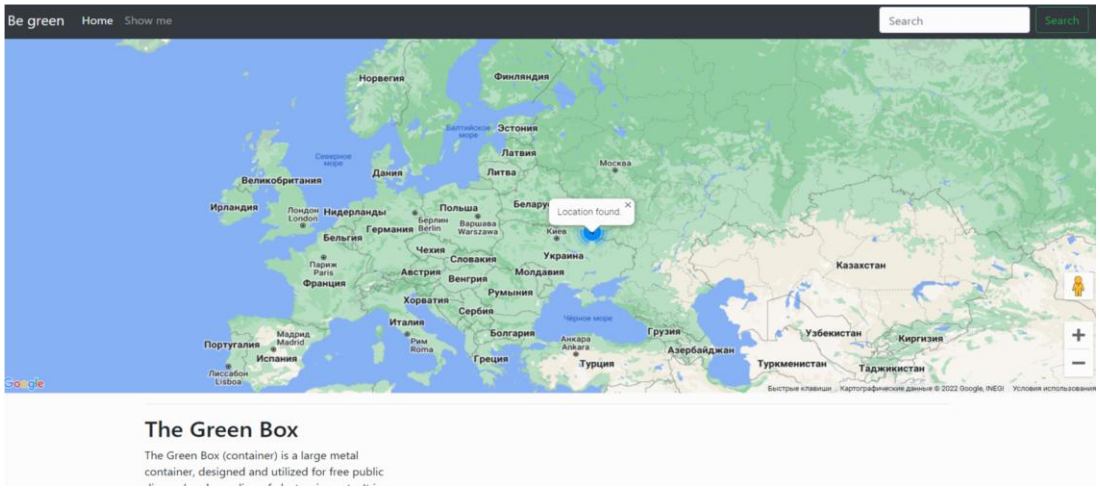

Рисунок 3.36 – Зразок сторінки з картою

В обраному місці з`явиться відмітка, щоб додати новий контейнер. Натиснувши кнопку «додати новий контейнер», з`явиться сторінка, де потрібно ввести дані про нього, а саме назву, розташування та опис (Рисунок 3.37). Натиснути «додати» і новий контейнер буде доступний та з`явиться на мапі.

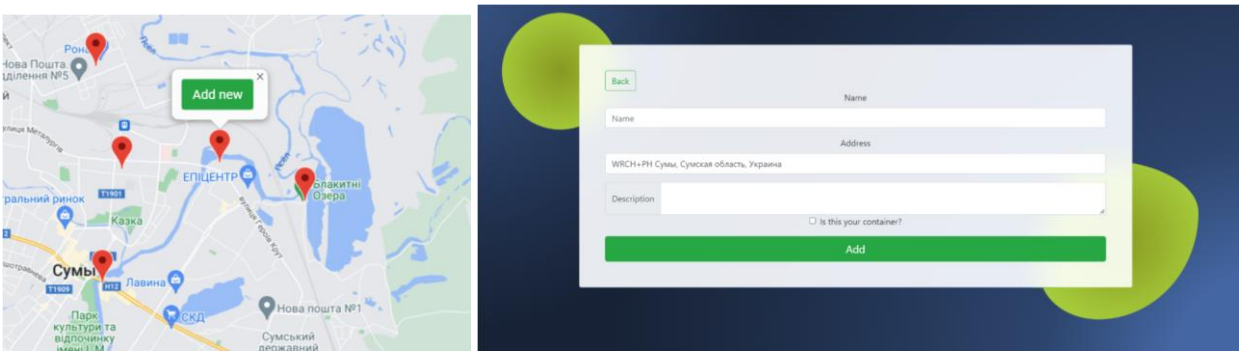

Рисунок 3.37 – Зразок створення контейнера

Також у проєкті за допомогою Java Script було реалізовано саму мапу. Вона пов`язана с Google API і доєднання до неї здійснюється за допомогою бібліотеки. Мапа в html файлі знаходить своє місце та розкривається на заданій частині сторінки. Також методи діставання контейнерів реалізовані за допомгою Java Script. Також реалізована геолокація.

Було реалізовано функціонал, де власники контейнерів мають можливість підключити змагання до контейнери за реальні гроші або бали. Для цього використано алгоритм, розглянутий у попередньому розділі. На початку змагання власнику видається QR-код, за яким кожен юзер має змогу прийняти участь у змаганні. Змагання полягає в тому, що юзер отримає певну кількість балів за кожен вкладений ним кілограм, кількість балів залежить від суми/загальної кількості балів, яку готовий заплатити власник контейнера за змагання. Розглянемо на прикладі процес змагання. Спочатку створюється контейнер із заохоченням (Рисунок 3.38).

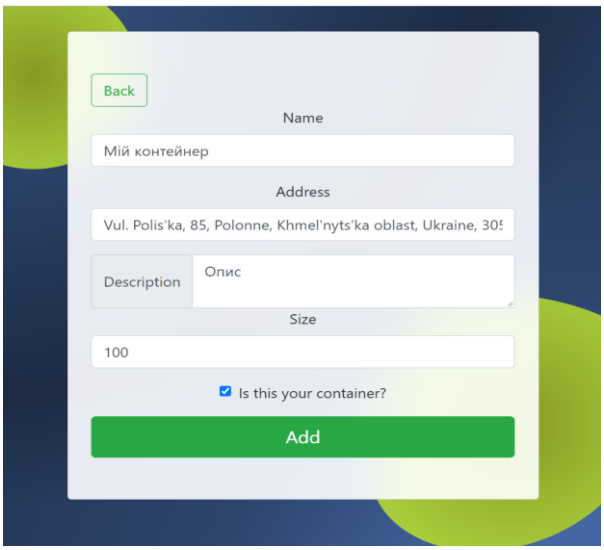

Рисунок 3.38 – Створення контейнеру із заохоченням

Тепер потрібно знайти цей контейнер серед своїх і натиснути на кнопку «Pointing» для відкриття сторінки, де можна почати змагання (Рисунок 3.39).

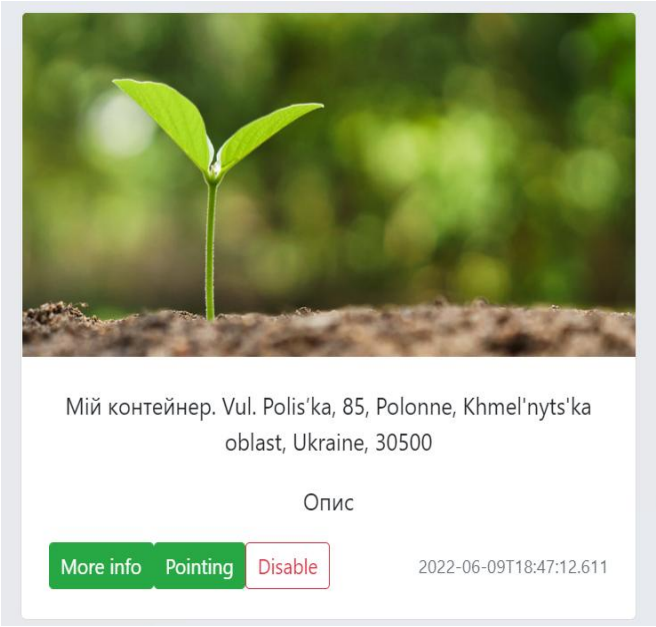

Рисунок 3.39 – Контейнер у головному меню

Після цього потрібно задати коефіцієнт чим він більший тим більше потрібно грошей витратити для власника й тим більше користувачі отримують балів за участь.

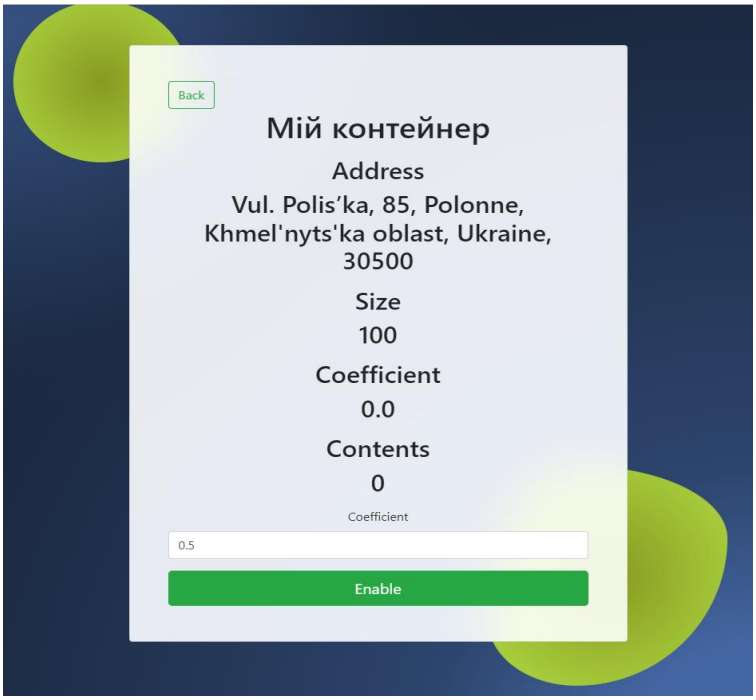

Рисунок 3.36 – Зразок сторінки для початку змагання

Після цього буде наданий QR-код для участі в змаганні. Також на сторінці можна буде обрати чи данні сходяться з вказаними в системі для того щоб контролювати порядність користувачів (Рисунок 3.41).

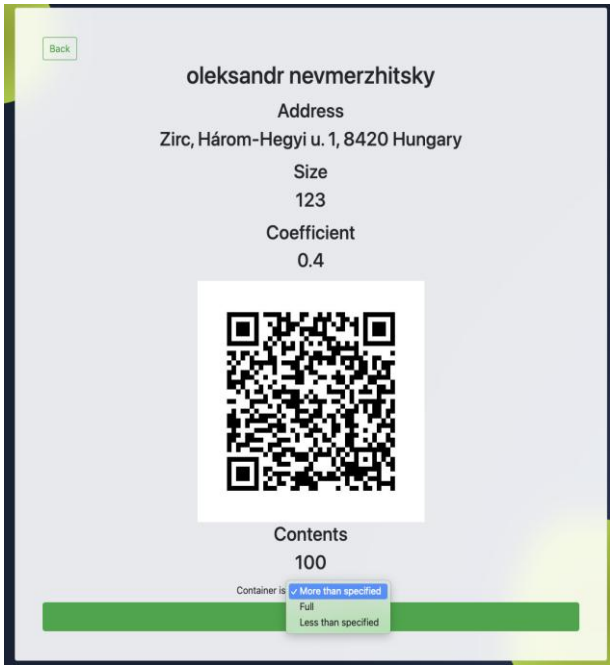

Рисунок 3.41 – Зразок сторінкі з початим змаганням

Тепер, якщо перейти по коду, відкриється сторінка учасника (Рисунок 3.42).

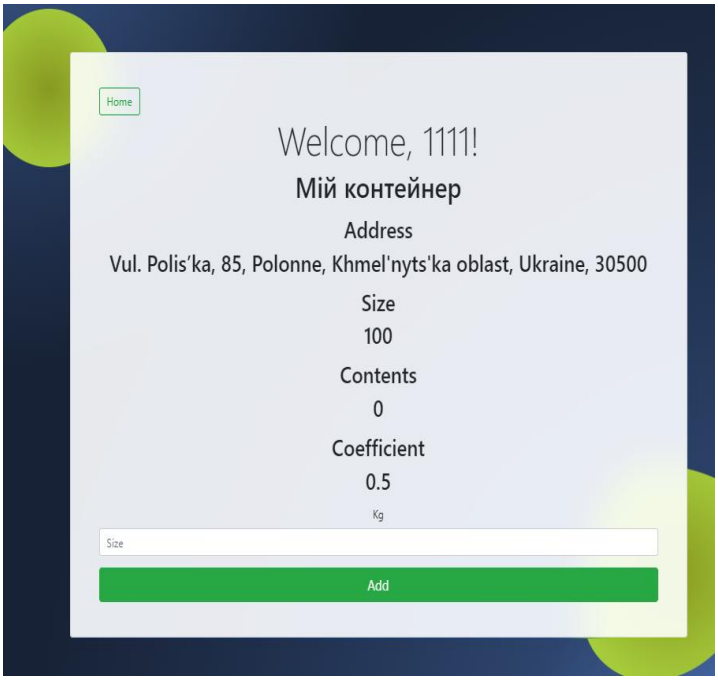

Рисунок 3.42 – Зразок сторінки учасника

Після заповнення даних юзер побачить повідомлення про успішну участь (Рисунок 3.43).

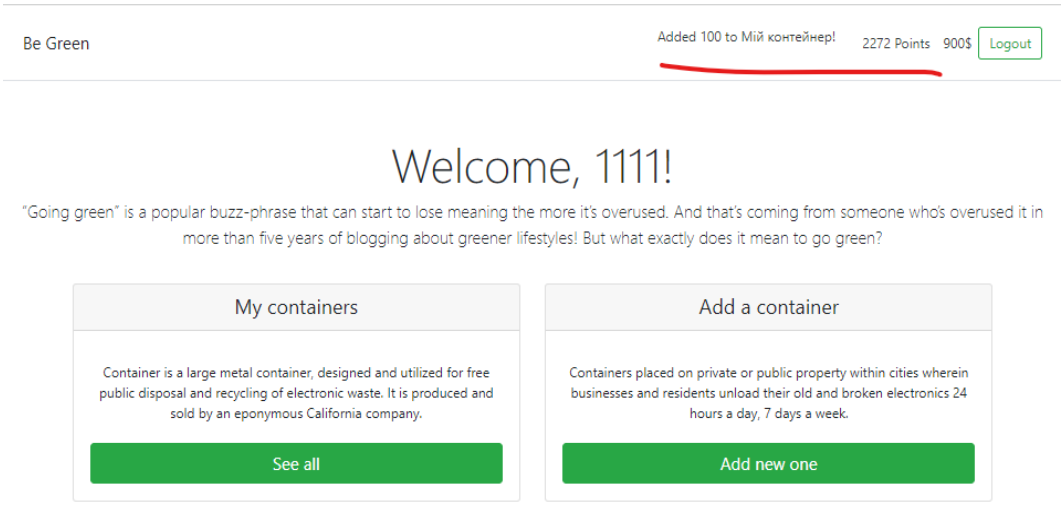

Рисунок 3.43 – Зразок головної сторінки з повідомленням

Коли власник закінчить змагання, кожен учасник отримує бали за свої додані кілограми з урахуванням його порядності (Рисунок 3.44).

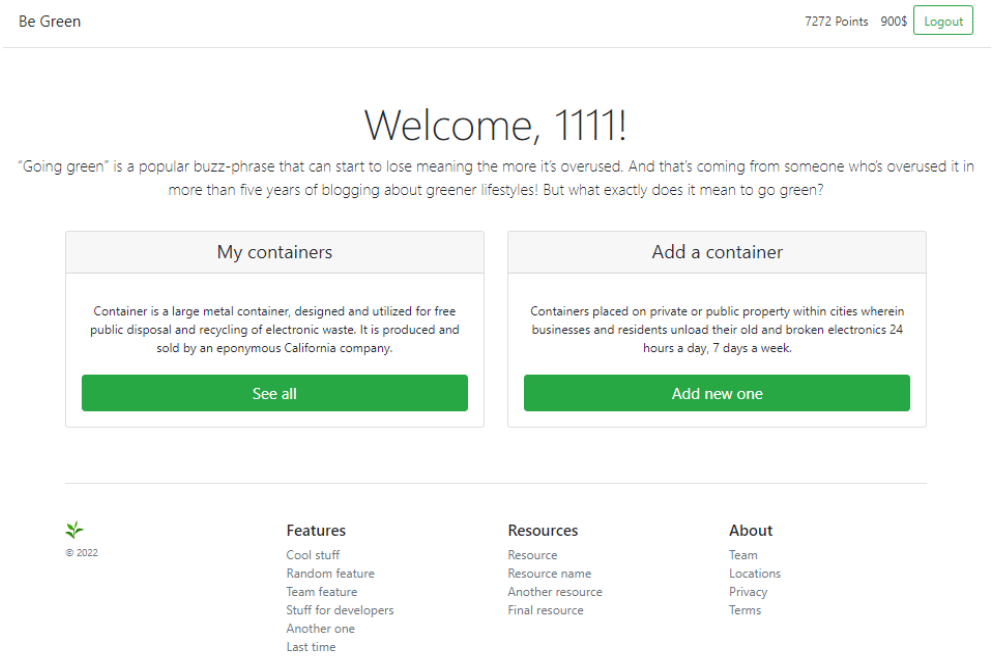

Рисунок 3.44 – Зразок головної сторінки з отриманими балами

ВИСНОВКИ

Розроблений програмний додаток "Be Green" забезпечує доступну інформацію та мотивацію населення сортувати й роздільно збирати побутові відходи. Цей додаток є ефективною інформаційною системою, що зберігає дані про місцезнаходження контейнерів і графічно відображає їх на карті за допомогою Google API. Застосування алгоритму розподілу балів між користувачами, що вносять відходи до контейнера, сприяє мотивації та співпраці. Цей алгоритм враховує загальну кількість балів, виділених власником контейнера для заохочення користувачів, а також рівень порядності та вклад кожного користувача.

Забезпечення доступною інформацією та мотивацією для сортування й роздільного збирання побутових відходів через додаток "Be Green" внесе значний вклад у вирішення проблеми забруднення довкілля. Подальший розвиток додатку може включати вдосконалення алгоритму розподілу балів, що враховуватиме динаміку зміни рівня порядності у часі. Також можна розглядати додавання елементів гейміфікації для привернення уваги дитячої аудиторії і формування правильних привичок щодо поводження з побутовими відходами з самого дитинства.

Загальною метою розробленого додатку є об'єднання людей для спільної мети - зробити світ чистішим. Використання інформаційних технологій та ефективних алгоритмів розподілу ресурсів сприяє досягненню цієї мети та забезпечує сталу систему управління побутовими відходами. Застосування додатку "Be Green" сприяє не лише покращенню екологічної ситуації, але й формує свідоме ставлення до довкілля серед користувачів.

Один з напрямків подальшого розвитку цього додатку полягає у вдосконаленні алгоритму розподілу балів. Розширення функціоналу алгоритму шляхом врахування динаміки зміни рівня порядності з часом дозволить більш точно визначати внесок кожного користувача і стимулювати постійне поліпшення їхньої екологічної активності.

Додавання елементів гейміфікації в додаток "Be Green" є ще одним перспективним напрямом розвитку. Це може включати в себе створення віртуальних досягнень, рейтингів, лідерських таблиць та інших механізмів, що мотивують користувачів до активного участі у сортуванні й роздільному збиранні відходів. Такий підхід може особливо зацікавити дитячу аудиторію, що дозволить формувати екологічні звички вже змалку.

В цілому, використання додатку "Be Green" разом із забезпеченням доступною інформацією та мотивацією для сортування й роздільного збирання побутових відходів допоможе вирішити проблему забруднення довкілля. Залучення населення до спільних зусиль і створення ефективної системи управління дозволять зробити значний вклад у стале збереження природних ресурсів та забезпечення чистого і здорового довкілля для майбутніх поколінь.

ПЕРЕЛІК ПОСИЛАНЬ

1. Із третього світу в перший. Реформа управління відходами в Україні. URL: https://www.pwc.com/ua/en/survey/2020/waste-management.pdf (дата звернення: 21.05.2023). *(електронний ресурс)*
2. Рихліцький В. Україна імпортує відходи з інших країн на мільярди. Чому так та як у нас працює бізнес з переробки сміття? URL: https://www.epravda.com.ua/publications/2021/06/18/675131/ (дата звернення: 21.05.2023). *(електронний ресурс)*
3. Karl Wiegers and Joy Beatty “Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle” Microsoft Press March 23, 2013. *(книга, 2 автори)*
4. Jake Spurlock "Bootstrap: Responsive Web Development" Packt Publishing February 28, 2019. *(книга, один автор)*
5. Greg Turnquist "Thymeleaf: Simple, Rich, and Powerful Templating for Java" Packt March 30, 2016. *(книга, один автор)*
6. Mick Knutson, Robert Winch, and Peter Mularien "Spring Security - Third Edition" Packt November 30, 2020. *(книга, 3 автори)*
7. Abhinay Muthoo Bargaining Theory with Applications, Cambridge University Press March 19, 1999. *(книга, один автор)*
8. Scott Chacon and Ben Straub "Pro Git" Apress November 3, 2014. *(книга, 2 автори)*
9. Richard Schneeman "Heroku: Up and Running" O'Reilly Media September 27, 2013. *(книга, один автор)*
10. Офіційна сторінка ознайомлення з Maps JavaScript API. URL: https://developers.google.com/maps/documentation/javascript (дата звернення: 21.05.2023). *(електронний ресурс)*
11. Craig Walls "Spring Boot in Action" Manning Publications Co. May 4, 2016. *(книга, один автор)*
12. Julia Lerman and Rowan Miller, "Programming Entity Framework", O'Reilly Media, August 31, 2010. *(книга, 2 автори)*

13. Jap S. D. “Pie Sharing” Journal of Marketing Research, 2001. *(книга, один автор)*

14. Kim J., Putterman L., Zhang X. Trust, Beliefs and Cooperation: Excavating a Foundation of Strong Economies. European Economic Review, 2022. – Vol. 147, pp. 104-166. URL: https://reader.elsevier.com/reader/sd/pii/S0014292122000940?token=9E3CB00668C08BF94E60A809D6ED3AD2E214CB3568CA4B997A6D108863C6AA54AF3CAF96D9D3CF1CACB7C7FBCA0225FC&originRegion=eu-west-1&originCreation=20230315162731 (дата звернення: 21.05.2023). *(електронний ресурс)*

15. Zivan R. Can trust increase the efficiency of cake cutting algorithms? Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, 2011. – Vol. 3 (AAMAS '11). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC. – pp. 1145–1146. *(книга, один автор)*

16. Arunachaleswaran E.R., Barman S., Kumar R., Rathi N. Fair and Efficient Cake Division with Connected Pieces. In: Caragiannis, I., Mirrokni, V., Nikolova, E. (eds) Web and Internet Economics. WINE 2019. Lecture Notes in Computer Science(), 2019. – Vol 11920. – Springer, Cham. URL: https://doi.org/10.1007/978-3-030-35389-6_5 (дата звернення: 21.05.2023). *(електронний ресурс)*

17. Christian Bauer and Gavin King "Hibernate in Action" Manning Publications Co. April 1 2004. *(книга, 2 автори)*

18. Robert C. Martin “Clean Code: A Handbook of Agile Software Craftsmanship” Prentice Hall August 11, 2008. *(книга, один автор)*

19. Herbert Schildt “Java: The Complete Reference” McGraw-Hill Education December 10, 2018. *(книга, один автор)*

20. Regina Obe and Leo Hsu "PostgreSQL: Up and Running" O'Reilly Media October 20, 2017. *(книга, 2 автори)*

ДОДАТОК А

```java
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
@RequiredArgsConstructor
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    private final UserServiceImpl userServiceImpl;
    private final PasswordEncoder passwordEncoder;
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/app/auth/registration", "/static", "/app/auth/activate/*", "/css/**", "/js/**").permitAll()
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/app/auth/login")
                .loginProcessingUrl("/app/auth/login")
                .defaultSuccessUrl("/", true)
                .permitAll()
                .and()
                .rememberMe()
                .and()
                .logout()
                .permitAll();
    }
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userServiceImpl)
                .passwordEncoder(passwordEncoder);
    }
}
```

Файл WebSecurityConfig.java

ДОДАТОК Б

```java
@Entity
@Table(name = "usr")
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@EqualsAndHashCode(exclude = {"containers"})
@ToString(exclude = {"containers"})
public class UserEntity {
    @Id
    @GeneratedValue(generator = "uuid")
    @GenericGenerator(name = "uuid", strategy = "uuid2")
    @Column(name = "id")
    private String id;
    private String username;
    private String password;
    private String email;
    private String activationCode;
    private boolean active;
    @OneToMany(mappedBy="user")
    private Set<ContainerEntity> containers;
    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name ="user_role", joinColumns = @JoinColumn(name = "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;
    private Integer point;
    private Integer currency;
}
```

Файл UserEntity.java

ДОДАТОК В

```java
@Data
@Table(name = "map_marker")
@Entity
@NoArgsConstructor
@Builder
@AllArgsConstructor
@EqualsAndHashCode(exclude = {"container"})
@ToString(exclude = {"container"})
public class MarkerEntity {
    @Id
    @GeneratedValue(generator = "uuid")
    @GenericGenerator(name = "uuid", strategy = "uuid2")
    private String id;
    @Column(name = "name", length = 128)
    private String name;
    @Column(name = "lat")
    private String lat;
    @Column(name = "lng")
    private String lng;
    private String address;
    private String description;
    @OneToOne(mappedBy = "marker")
    private ContainerEntity container;
    @CreationTimestamp
    @Column(name = "created_on")
    private LocalDateTime createdOn;
    @UpdateTimestamp
    @Column(name = "updated_on")
    private LocalDateTime updatedOn;
}
```

Файл MarkerEntity.java

ДОДАТОК Г

```
@Data
@Table(name = "container")
@Entity
@NoArgsConstructor
@Builder
@AllArgsConstructor
@EqualsAndHashCode(exclude = {"marker", "user"})
@ToString(exclude = {"marker", "user"})
public class ContainerEntity {
    @Id
    @GeneratedValue(generator = "uuid")
    @GenericGenerator(name = "uuid", strategy = "uuid2")
    private String id;
    private String name;
    private String description;
    private Integer addedUsers;
    @ManyToOne
    @JoinColumn(name="user_id")
    private UserEntity user;
    private Integer averageMark;
    boolean isBeSeen;
    private String address;
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "marker_id", referencedColumnName = "id")
    private MarkerEntity marker;
    @CreationTimestamp
    @Column(name = "created_on")
    private LocalDateTime createdOn;
    @UpdateTimestamp
    @Column(name = "updated_on")
    private LocalDateTime updatedOn;
    @Column(nullable = false)
    private Integer size;
    @Column(nullable = false, columnDefinition = "int default 0")
    private int contents;
    @Column(nullable = false, columnDefinition = "FLOAT(8) default 0.0")
    private double pointKef;
    @Column(nullable = false, columnDefinition = "boolean default false")
    private boolean isPoint;
    @Column(nullable = false, columnDefinition = "boolean default false")
    private boolean isFull;
}
```

Файл ContainerEntity.java

ДОДАТОК Ґ

```java
@Service
@RequiredArgsConstructor
public class UserServiceImpl implements UserDetailsService {
    private final UserRepository userRepository;
    private final MailingService mailingServiceImpl;
    private final PasswordEncoder passwordEncoder;
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        return this.findByUsername(username).orElse(null);
    }
    public UserModel saveUser(UserModel model) {
        return ServiceLayerMapper.I.toUserModel(userRepository.save(ServiceLayerMapper.I.toUserEntity(model)));
    }
    public Optional<UserModel> findByUsername(String userName) {
        return userRepository.findByUsername(userName).map(ServiceLayerMapper.I::toUserModel);
    }
    public Optional<UserModel> findById(String id) {
        return userRepository.findById(id).map(ServiceLayerMapper.I::toUserModel);
    }
    public Optional<UserModel> findByActivationCode(String activationCode) {
        return userRepository.findByActivationCode(activationCode).map(ServiceLayerMapper.I::toUserModel);
    }
    public boolean addUser(UserModel userModel) {
        if (this.findByUsername(userModel.getUsername()).isPresent()) {
            return false;
        }
        userModel.setActive(true);
        userModel.setActivationCode(UUID.randomUUID().toString());
        userModel.setPassword(passwordEncoder.encode(userModel.getPassword()));
        this.saveUser(userModel);
        return true;
    }
    public boolean activateUser(String code) {
        UserModel userModel = this.findByActivationCode(code).orElse(null);
        if (userModel == null) {
            return false;
        }
        userModel.setActivationCode(null);
        this.saveUser(userModel);
        return true;
    }
    private void sendMessage(UserModel userModel) {
        if (!StringUtils.isEmpty(userModel.getEmail())) {
            String message = String.format(
                    "Hello, %s! Welcome to Be Green. Please, visit next link: http://localhost:8080/be-green/app/auth/activate/%s",
                    userModel.getUsername(),
                    userModel.getActivationCode()
            );
            mailingServiceImpl.send(userModel.getEmail(), "Activation code", message);
        }
    }
    public List<ContainerModel> getAllUserContainers(String userName) {
        return userRepository.findByUsername(userName).get().getContainers().stream()
                .map(ServiceLayerMapper.I::toContainerModel).collect(Collectors.toList());
    }
}
```

Файл UserServiceImpl.java