МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

КОЗАК Аліна Андріївна

Веб-орієнтована інформаційна система надання волонтерської допомоги у період війни / Web-oriented information system for the provision of volunteer assistance during the war

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконала студентка групи КН-42
А.А. Козак

Науковий керівник
к.т.н., професор М.П. Комар

Кваліфікаційну роботу допущено до захисту
«___»_____20___р.

Завідувач кафедри
_____М.П. Комар

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТКИ
КОЗАК Аліна Андріївна
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Веб-орієнтована інформаційна система надання волонтерської допомоги в період війни/ Web-Oriented Information System for Providing Volunteer Assistance During a War

керівник роботи Комар Мирослав Петрович, д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 грудня 2023 р. № 753.

2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- провести аналіз та опис предметної області;
- провести огляд існуючих веб-орієнтованих інформаційних систем для надання волонтерської допомоги;
- зробити постановку задачі проектування;
- забезпечити інформаційне наповнення системи;
- проектування архітектури системи, структурно-функціональне моделювання;
- проектування бази даних для зберігання інформації даних користувача;
- програмна реалізація;
- розробка інтерфейсу користувача;

5. Перелік графічного матеріалу в роботі:
- аналіз та порівняння продуктів-аналогів;
- схеми алгоритмів процесу;
- схема структури бази даних;
- схема архітектури веб-орієнтованої інформаційної системи.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент_____ А.А. Козак
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ М.П. Комар_____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра: «Веб-орієнтована інформаційна система надання волонтерської допомоги». Вступ, три розділи, висновки та список використаних джерел із 26 найменувань складають пояснювальну записку. Пояснювальна записка складається з 84 сторінок, включаючи 37 сторінки основного тексту, 3 сторінки списку використаних джерел і 44 сторінок додатків.

Метою дослідження є розробка та впровадження веб-орієнтованої системи для ефективної координації та надання волонтерської допомоги в умовах воєнного стану.

У даному дослідженні використовуються кілька методів для досягнення поставленої мети. Теоретичний аналіз літератури та існуючих рішень у сфері волонтерської діяльності та інформаційних систем дозволив сформувати базу знань та визначити ключові напрями роботи. Для розробки структурно-функціональної моделі застосовано методи структурного та об'єктно-орієнтованого моделювання, зокрема IDEF0. Алгоритм роботи системи був розроблений за допомогою методів алгоритмізації та процесного моделювання. Для проектування бази даних використовувалися методи реляційного моделювання даних. Архітектура та програмна реалізація веб-орієнтованої інформаційної системи базувалися на сучасних принципах розробки програмного забезпечення, таких як MVC (Model-View-Controller). Інтерфейс користувача був створений з урахуванням принципів UX/UI дизайну для забезпечення зручності використання.

Запропонована система дозволить значно покращити організацію та управління волонтерськими ініціативами, забезпечуючи швидку мобілізацію ресурсів та оперативну реакцію на запити про допомогу.

Ключові слова: WEB-ДОДАТОК, БАЗА ДАНИХ, ВОЛОНТЕРСТВО, ІНФОРМАЦІЙНА СИСТЕМА, ОГОЛОШЕННЯ, ДОПОМОГА, ГРОШОВИЙ ЗБІР, РОЗРОБКА.

ANNOTATION

Bachelor's Qualification Work Topic: "Web-Oriented Information System for Providing Volunteer Assistance." The explanatory note comprises the introduction, three chapters, conclusions, and a list of 26 sources. The explanatory note consists of 84 pages, including 37 pages of the main text, 3 pages of the list of sources, and 44 pages of appendices.

The aim of the research is to develop and implement a web-oriented system for the effective coordination and provision of volunteer assistance in wartime conditions.

In this study, several methods are used to achieve the set goal. Theoretical analysis of literature and existing solutions in the field of volunteer activities and information systems allowed the formation of a knowledge base and the determination of key directions for work. For the development of the structural-functional model, methods of structural and object-oriented modeling, particularly IDEF0, were applied. The system's workflow algorithm was developed using methods of algorithmization and process modeling. Methods of relational data modeling were used for database design. The architecture and software implementation of the web application were based on modern software development principles, such as MVC (Model-View-Controller). The user interface was created considering UX/UI design principles to ensure ease of use.

The proposed system will significantly improve the organization and management of volunteer initiatives, ensuring the rapid mobilization of resources and prompt response to assistance requests.

Keywords: WEB APPLICATION, DATABASE, VOLUNTEERING, INFORMATION SYSTEM, ANNOUNCEMENT, ASSISTANCE, FUNDRAISING, DEVELOPMENT.

ЗМІСТ

ВСТУП

Волонтерська допомога набуває особливої ваги в сучасних умовах, коли світова спільнота стикається з безпрецедентними викликами та кризами. Враховуючи поточну ситуацію в нашій країні, люди повинні максимально допомагати один одному та оточуючим. Тисячі волонтерів активно залучаються до збору грошей на закупівлю зброї, транспортних засобів, обладнання для військових, автомобілів і допомоги тим, чиї домівки були втрачені через атаки, а також інших предметів, необхідних для захисту України.

Наразі волонтерство активно розвивається, тому Україна потрапила до семи найкращих країн Європи. За останній рік більше 12 000 людей допомагали спільноті. Крім того, кількість волонтерів і ті, хто допомагає, зростають щороку. Через появу великої кількості неструктурованих зборів ідея їх групування та підпорядкування стає все більш актуальною.

Веб-орієнтовані інформаційні системи дозволяють швидко та точно обробляти дані, організовувати діяльність волонтерів і гарантувати прозорість у використанні ресурсів. Вони допомагають у створенні платформи, яка об’єднує зусилля різних волонтерських груп і організацій, щоб уникнути дублювання завдань і зробити процес надання допомоги більш ефективним.

Крім того, важливо пам’ятати, що люди, які втратили своє житло, майно або розшукують своїх близьких, мають пройти складний шлях, щоб повідомити про допомогу, навіть якщо вони не потребують матеріальної допомоги. Об’єднує всі можливості волонтерства в країні в одному місці, служить мостом між ефективними ініціативами та волонтерами, допомагає окремим особам чи цілим організаціям знайти активістів для допомоги.

Розроблена система значно підвищить ефективність роботи волонтерських організацій, зробить використання ресурсів прозорим і підзвітним, і дозволить швидко реагувати на потреби постраждалих. Зрештою, це збільшить загальну підтримку та допомогу, що надається в умовах військових дій.

Метою дослідження є розробка та впровадження веб-орієнтованої системи для ефективної координації та надання волонтерської допомоги в умовах воєнного стану.

Для досягнення мети необхідно вирішити наступні проблеми:

1. Вивчити теоретичні основи волонтерської діяльності в умовах воєнного стану для розуміння ключових аспектів та викликів цієї діяльності.
2. Провести огляд існуючих рішень у сфері волонтерської допомоги з метою визначення їх переваг і недоліків.
3. Розробити структурно-функціональну модель системи для ефективної організації та управління бізнес-процесами волонтерської допомоги.
4. Створити алгоритм роботи системи, що забезпечить чітке виконання всіх етапів процесу надання волонтерської допомоги.
5. Спроектувати модель бази даних для зберігання та управління інформацією про користувачів, запити на допомогу та ресурси.
6. Розробити архітектуру веб-орієнтованої інформаційної системи, що забезпечить його стабільну та ефективну роботу.
7. Реалізувати програмну частину веб-орієнтованої інформаційної системи з використанням сучасних технологій та фреймворків.
8. Створити інтуїтивно зрозумілий інтерфейс користувача, який забезпечить зручний доступ до всіх функцій системи для волонтерів та користувачів..

Об'єктом дослідження є процес надання волонтерської допомоги в умовах воєнного стану.

Предметом дослідження є розробка та впровадження веб-орієнтованої системи для координації та управління волонтерською діяльністю.

Методи дослідження. У даному дослідженні використовуються кілька методів для досягнення поставленої мети. Теоретичний аналіз літератури та існуючих рішень у сфері волонтерської діяльності та інформаційних систем дозволив сформувати базу знань та визначити ключові напрями роботи. Для розробки структурно-функціональної моделі застосовано методи структурного та об’єктно-орієнтованого моделювання, зокрема IDEF0. Алгоритм роботи системи був розроблений за

допомогою методів алгоритмізації та процесного моделювання. Для проектування бази даних використовувалися методи реляційного моделювання даних. Архітектура та програмна реалізація веб-орієнтованої інформаційної системи базувалися на сучасних принципах розробки програмного забезпечення, таких як MVC (Model-View-Controller). Інтерфейс користувача був створений з урахуванням принципів UX/UI дизайну для забезпечення зручності використання.

Практичне значення даного дослідження полягає у створенні ефективної веб-орієнтованої системи для координації волонтерської допомоги під час воєнного стану. Запропонована система дозволить значно покращити організацію та управління волонтерськими ініціативами, забезпечуючи швидку мобілізацію ресурсів та оперативну реакцію на запити про допомогу. Впровадження цієї системи сприятиме підвищенню прозорості та підзвітності волонтерської діяльності, що в свою чергу зміцнить довіру до волонтерських організацій та підвищить їх ефективність. Крім того, дана система може бути адаптована та використана в інших сферах, де потрібна координація великої кількості учасників та управління ресурсами, що робить її універсальним інструментом для багатьох організацій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Теоретичні основи волонтерської діяльності в умовах воєнного стану

Волонтерська діяльність – це безоплатна, добровільна допомога, що надається громадянами для підтримки та покращення життя інших людей, громад або організацій. Волонтерство включає широкий спектр активностей, таких як соціальна допомога, охорона здоров'я, екологічні проекти, культурні та освітні програми.

Волонтерська діяльність може бути класифікована за кількома критеріями: За сферою діяльності:

- Соціальна допомога (підтримка малозабезпечених, людей з обмеженими можливостями, бездомних)
- Медична допомога (підтримка медичних закладів, донорство, допомога під час епідемій)
- Екологічні проекти (прибирання територій, посадка дерев, просвітницька діяльність)

За формою участі:

- Індивідуальне волонтерство (особисті ініціативи)
- Групове волонтерство (діяльність в рамках організацій чи команд)

За тривалістю:

- Постійне волонтерство (регулярна допомога)
- Тимчасове волонтерство (одноразові акції чи проекти)

За місцем здійснення:

- Локальне волонтерство (допомога в межах свого міста чи регіону)
- Міжнародне волонтерство (допомога в інших країнах)

Волонтерська діяльність набуває особливого значення та стикається з особливими труднощами під час воєнного стану. Потреба в допомозі зростає в багатьох напрямках, включаючи підтримку біженців, медичну допомогу, гуманітарну допомогу, психологічну допомогу та відновлення зруйнованої інфраструктури.

Організаційні характеристики:

- Координація та комунікація: включають створення прозорої системи для спілкування між волонтерами, організаціями та державними структурами.
    - Безпека: включає навчання безпеці, надання засобів індивідуального захисту та інструктаж щодо дій в екстрених ситуаціях.
    - Гнучкість та адаптивність: Волонтерська діяльність повинна бути гнучкою, щоб швидко адаптуватися до змінних обставин і потреб.
    - Підтримка ресурсами: ефективна діяльність залежить від того, щоб волонтери були забезпечені необхідними ресурсами, такими як продукти, ліки та транспорт.
    - Психологічна підтримка: Оскільки волонтери часто стикаються з емоційним виснаженням і високим рівнем стресу, важливо надавати їм психологічну підтримку та консультації.

Використання веб-орієнтованих систем значно полегшує організацію та координацію волонтерських заходів. Онлайн-ресурси, доступні волонтерським організаціям, дозволяють відстежувати прогрес волонтерів, централізовано керувати заходами та розподіляти завдання серед волонтерів. Це запобігає дублюванню роботи та підвищує ефективність використання ресурсів.

Волонтерство стає більш доступним завдяки онлайн-системам. Потрібні волонтери можуть легко знайти інформацію про доступні проекти та зареєструватися для участі завдяки мобільним додаткам та онлайн-платформам. Це сприяє залученню більшої кількості учасників і кращому розподілу допомоги.

Для успіху будь-якої волонтерської ініціативи необхідно ефективно спілкуватися. Використовуючи веб-орієнтовані системи, волонтери, координатори та бенефіціари можуть швидко та легко обмінюватися інформацією. Вони дозволяють швидко повідомляти про зміни в планах, надсилати нагадування та отримувати зворотний зв’язок.

Онлайн-платформи роблять волонтерську діяльність більш прозорою. Організації можуть продемонструвати свою надійність і відповідальність,

публікуючи звіти про проекти, фінансові звіти та відгуки учасників. Це підвищує довіру з боку волонтерів і донорів.

Веб-орієнтовані системи дозволяють збирати та аналізувати велику кількість інформації про волонтерську роботу. Це дозволяє знайти найефективніші практики, знайти проблемні місця та приймати розумні рішення, щоб покращити роботу компанії. Використання аналітичних інструментів сприяє оптимізації процесів і підвищує ефективність волонтерських проектів.

Таким чином, висновки дослідження показують, що інтеграція сучасних веб-орієнтованих інформаційних систем у волонтерську роботу є важливою частиною підвищення ефективності, особливо в умовах кризи. Це свідчить про те, наскільки важливо продовжувати розвивати та впроваджувати такі технології, щоб підтримувати та оптимізувати роботу волонтерів.

1.2 Огляд існуючих рішень

Волонтерство піднялося на новий рівень після початку широкомасштабної війни в Україні. Навіть це стало нормою. Допомога армії та громадянам, які постраждали від боїв, наближає нашу перемогу та доводить, що українці незламні.

Джерело [1] розглядає волонтерство під час війни. Наголошується на тому, що такі добровольці є потужним двигуном громадянського суспільства, який сприяє наближенню перемоги України. Бо велика справа складається з маленьких вчинків, зазначається, що кожен, хто хоче, може долучитися та допомогти.

Крім того, джерело [2] зазначає, що Україна входить до десятки країн, які розвивають благодійність. Згідно з рейтингом «Всесвітнього індексу благодійності» (WGI), опублікованим у 2022 році, країна демонструє один із найвищих темпів зростання за останні роки. Незважаючи на це, особлива увага приділяється саме питанню організації волонтерської діяльності. У перші місяці війни ця благодійна діяльність була хаотична та потребувала планування.

Отже, виходячи з усього вищезазначеного, можна зробити висновок, що розробка веб-орієнтованої інформаційної системи для підтримки волонтерської діяльності є дійсно актуальною. Сьогодні, у цей важкий час, забезпечення потреб населення України є надзвичайно важливим завданням. Застосування цього веб-додатку дозволить збільшити обізнаність про волонтерську організацію та її діяльність. Це дозволить їй охопити більше людей і, можливо, залучити більше підтримки. Використання такого веб-орієнтованої інформаційної системи може викликати ентузіазм і зацікавленість у діяльності організації, надихаючи інших взяти участь і допомогти більшій кількості людей, демонструючи місію, цінності та досягнення організації.

Пошук і аналіз веб-орієнтованої інформаційної системи-аналогів волонтерських організацій необхідні перед розробкою прототипу вищевказаного веб-орієнтованої інформаційної системи.

Основними критеріями аналізу є такі:

- дизайн (дизайн веб-орієнтованої інформаційної системи, підбір кольорів, фонових зображень, читабельність тексту);
- дотримання стилю; зручність навігації;
- якість і доцільність ілюстрованого матеріалу.

«Гуманітарна та медична допомога Україні» — це перший приклад онлайн-додатка для волонтерства [3]. У цьому веб-ресурсі для волонтерів є веб-додаток, який дозволяє створювати донати, переглядати звітність організації, отримувати контактну інформацію, ознайомлюватися з новинами та багато іншого. Відвідувач також може дізнатися, як допомогти іншій країні. Згідно з рис. 1.1, дизайн цього веб-додатку було правильно розроблено. Використовувалися чотири різні шрифти. Веб-додаток використовує жовтий, синій, білий і чорний кольори для фону. Чорний, білий і червоний забарвлення можна використовувати для оформлення гами тексту. Кольори чудово поєднуються. Переваги цього веб-додатку полягають у тому, що він повністю відповідає тематиці, його оформлення є сучасним, зображення підібрано професійно та стиль витримано відповідно до теми. Основними недоліками програми є

неправильна навігація на веб-додатку, а стрічка новин значно виділяється зі стилю основної частини, що робить її незручною для користувача (рис. 1.2).

Рисунок 1.1 – Головна сторінка веб-додатку helpukraine.center

Рисунок 1.2 – Сторінка новин веб-додатку helpukraine.center

«Повернись живим» — веб-додаток організації [4]. Цей веб-додаток є фондом, який надає допомогу військовим. Рисунок 1.3 показує його головну сторінка. У цьому місці ви можете ознайомитися з новинами з фронту, переглянути звітність фонду, дізнатися більше про нього та про те, як він працює. Є можливість направити фінансування на армію. Веб-додаток був розроблений за допомогою п’яти різних шрифтів, які були гармонійно поєднані. Кольори були підібрані відповідно до тематики війни. Структура та стиль оформлення даного онлайн-ресурсу відповідають

тематиці, шрифти вдало підібрані, навігація дозволяє знайти приховану інформацію тощо. Найбільшим недоліком веб-додатку є низька якість ілюстрацій у деяких новинах.

Рисунок 1.3 – Головна сторінка веб-додатку savelife.in.ua

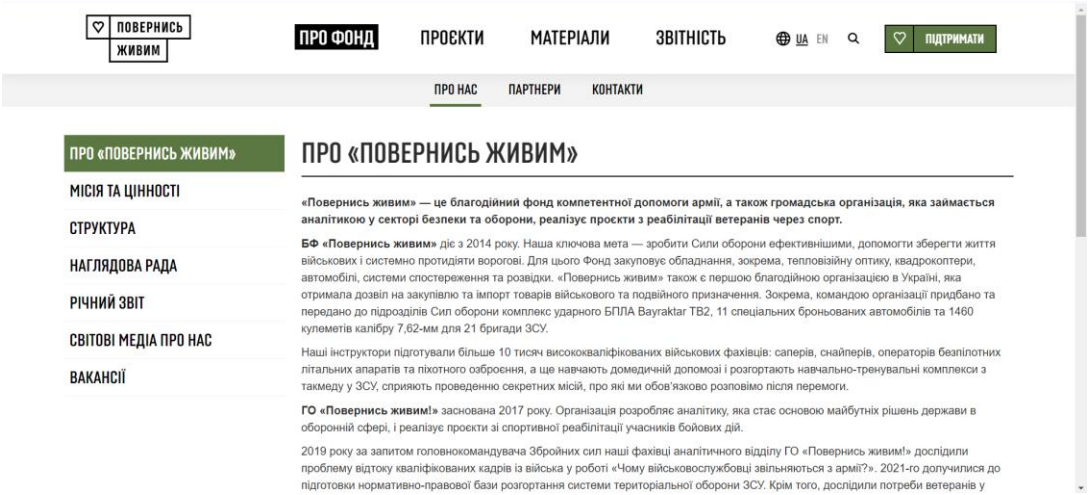

Рисунок 1.4 – Інформаційна сторінка веб-додатку savelife.in.ua

«ARMY SOS» — це третій приклад веб-додатка благодійної організації, яка допомагає армії країни [5]. Це невелика інформаційна платформа, на якій ви маєте можливість допомогти в закупівлі сучасного обладнання та обладнання. Рисунок 1.5 показує головну сторінку веб-додатку. Крім того, ви можете переглянути звіти фонду, а також дізнатися, хто працює в цій команді та як зв'язатися з конкретною організацією. Створення сторінок є простим, але інформативним. Вдало підібрана

палітра кольорів Рис. 1.5 показує, що головна сторінка містить лише основні дані. Щоб знайти більше інформації, ви можете використовувати меню, яке знаходиться вище. Таким чином, користувачам досить зручно, що немає нагромадження непотрібних даних відразу. Інформаційність і кольорова гамма цього веб-додатку є перевагами (рис. 1.6). Його головний недолік — це навігація, оскільки зверху не вистачає пунктів, хоча всю інформацію про переміщення можна знайти у футері цього веб-додатку.

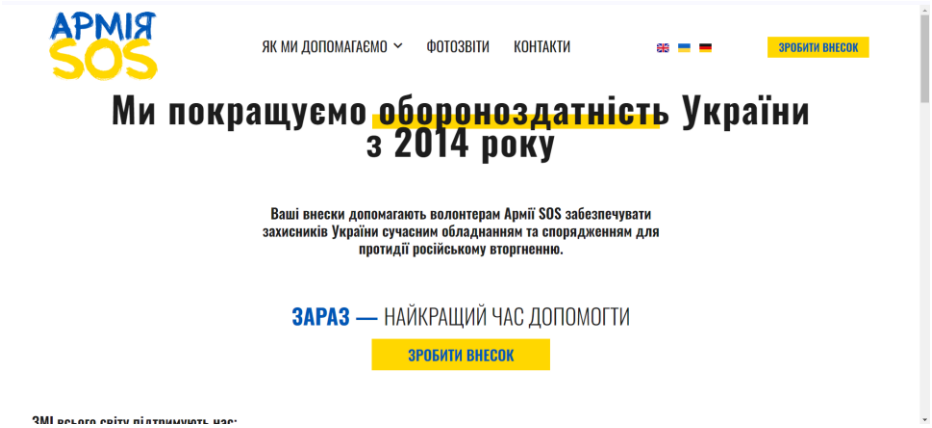

Рисунок 1.5 – Головна сторінка та меню веб-додатку armysos.com.ua

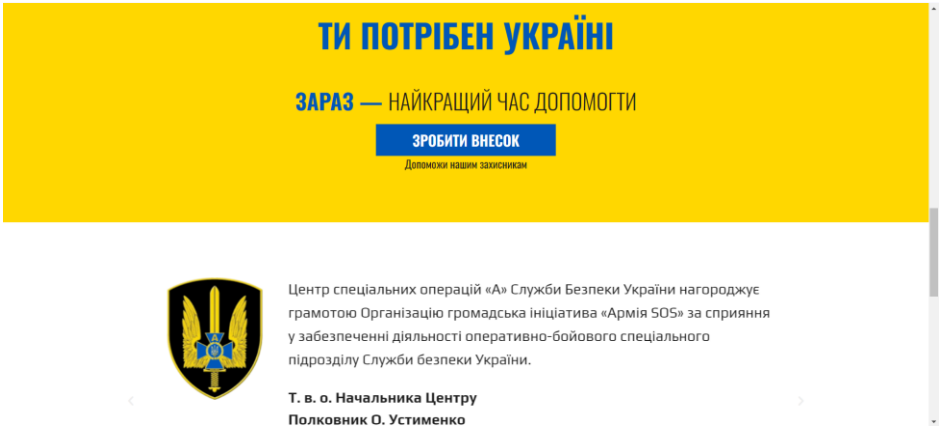

Рисунок 1.6 – Інформаційна сторінка веб-додатку armysos.com.ua

«Мрія вільних людей» — це веб-додаток громадської організації [6]. На рис. 1.7 показано веб-додаток одеської волонтерської організації. В більшості він інформаційний і соціальний. Він містить відеофайли. На рис. 1.8 наведено короткий опис роботи організації, яку вона виконує, яку допомогу надає населенню та

військовим, а також на що зараз збирається матеріальна допомога (рис. 1.8). Крім того, його веб-додаток містить корисну інформацію від самого волонтерського центру. Цей веб-додаток був розроблений спеціалістами. Три різні шрифти використовуються. Кольорова палітра добре підібрана. Оформлення та читабельність тексту є двома його головними перевагами. Основним недоліком веб-додатку є велика кількість незадіяного простору. Крім того, як показано на рис. 1.9, кольорове оформлення новин не відповідає основній палітрі кольорів онлайн ресурсу.

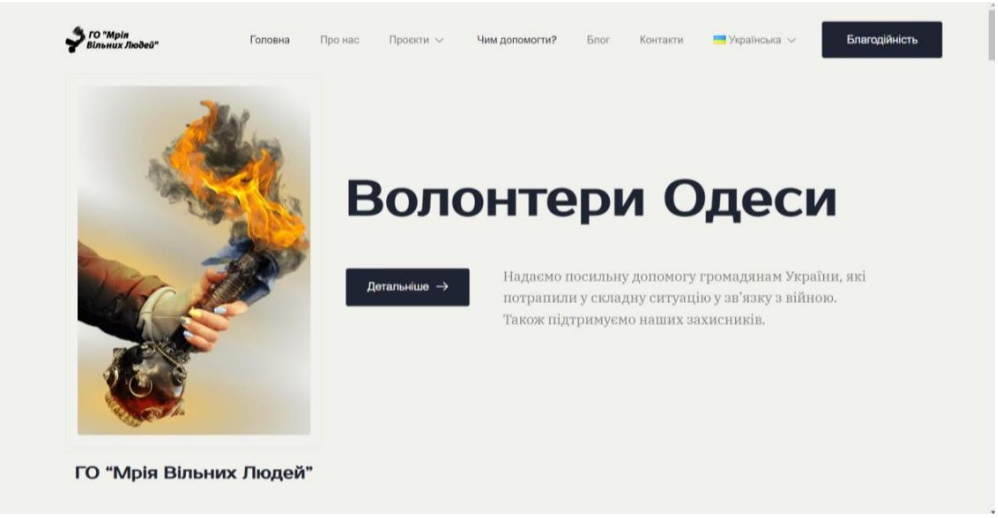

Рисунок 1.7 – Головна сторінка веб-додатку mriya.od.ua

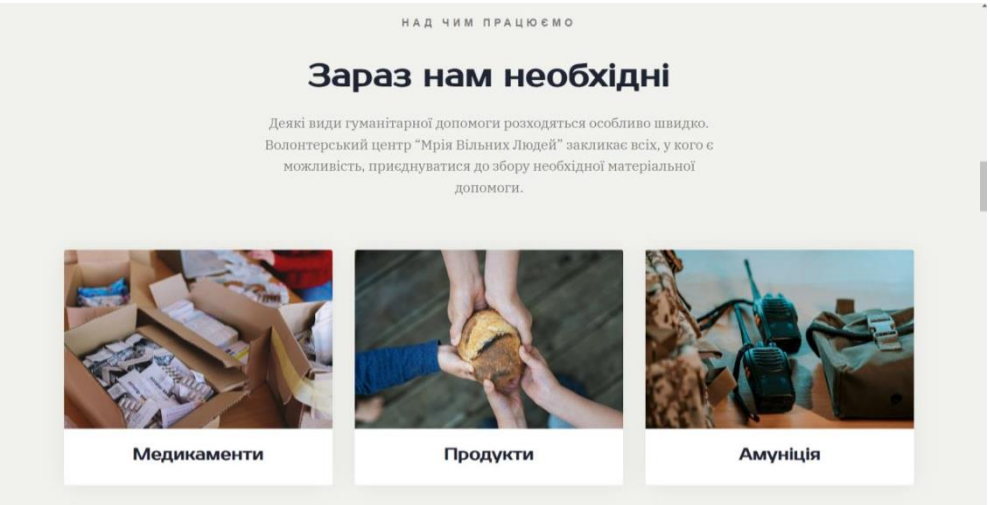

Рисунок 1.8 – Інформаційна сторінка веб-додатку mriya.od.ua

Було визначено основні переваги та недоліки веб-додатків-аналогів після їх ретельного аналізу. Його результати представлені в таблиці 1.1.

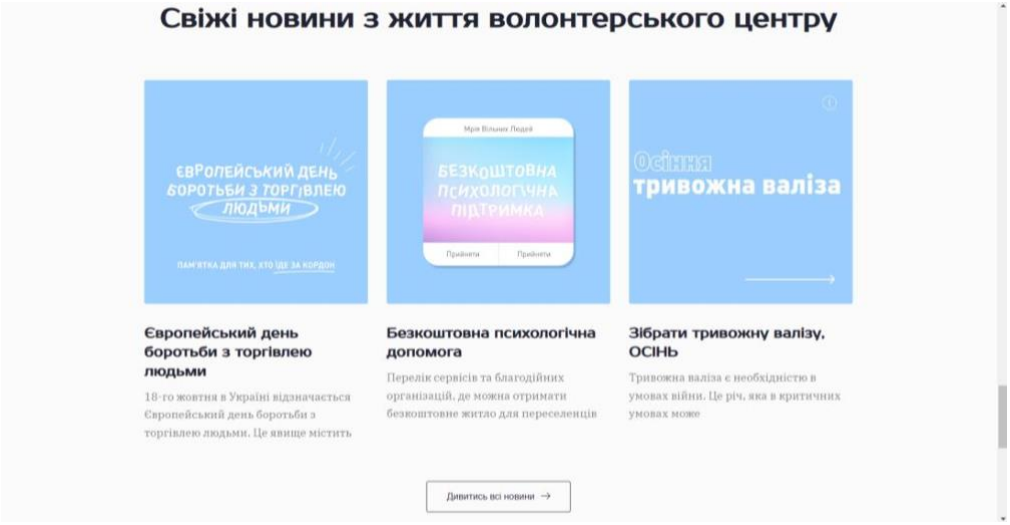

Рисунок 1.9 – Сторінка новин веб-додатку mriya.od.ua

Таблиця 1.1 – Порівняльна таблиця характеристик web-додатків-аналогів

Характеристика/ Онлайн ресурс	helpukraine.center	savelife.in.ua	armysos.com.ua	mriya.od.ua
Зручний інтерфейс	+	+	+	+
Сучасний дизайн	-	+	+	-
Функціональність	+	-	+	+
Інтерактивність	-	-	+	+
Навігація	+	+	+	+
Реєстрація користувачів	-	-	-	-
Інформаційна сторінка	+	+	+	+
Сторінка новин	+	+	+	+
Наявність відгуків	+	-	-	-

Після аналізу аналогів можна зробити висновки. Більшість вищезазначених онлайн-ресурсів містять багато інформації. Але для того, щоб створити програмний продукт, який буде подобатися користувачам, дизайн має бути якісно розроблений. Інформаційну складову, структуру, кольорову гамму та ілюстративний матеріал можна взяти з аналогів, доповнюючи власну веб-орієнтовану інформаційну систему новими функціями, що робить розроблюваний продукт унікальним і дозволить подолати вищезазначені недоліки.

1.3 Постановка задачі проектування

В умовах воєнного стану потреба у волонтерській допомозі значно зростає, оскільки багато людей і організацій стикаються з гострою нестачею ресурсів, фінансових та матеріальних засобів. Забезпечення оперативної та ефективної координації волонтерських дій стає критично важливим для підтримки постраждалих, організації допомоги та забезпечення базових потреб населення. У цьому контексті веб-орієнтована система надання волонтерської допомоги є незамінним інструментом для організації, управління та моніторингу волонтерських ініціатив.

Сучасні інформаційні технології дозволяють створити потужні веб-орієнтовані платформи, які забезпечують інтерактивність, доступність та зручність у використанні. Такі системи можуть інтегрувати численні функції, починаючи від реєстрації волонтерів і закінчуючи відстеженням виконання завдань та звітністю. В умовах воєнного стану це дозволяє максимально швидко мобілізувати ресурси, координувати дії волонтерів та забезпечити прозорість і підзвітність кожного етапу процесу надання допомоги.

Веб-орієнтована система надання волонтерської допомоги підвищує ефективність комунікації між волонтерами, благодійними організаціями та тими, хто потребує допомоги. Вона спрощує процеси запиту та надання допомоги, дозволяє зберігати та аналізувати дані про потреби та ресурси, що в свою чергу сприяє більш точному і своєчасному реагуванню на запити. Крім того, така система може включати механізми контролю та верифікації, що мінімізує ризик зловживань та підвищує довіру до волонтерських ініціатив.

У підсумку, впровадження веб-орієнтованої системи надання волонтерської допомоги під час воєнного стану є надзвичайно актуальним та необхідним кроком. Це дозволяє не лише оптимізувати процеси надання допомоги, але й забезпечити сталий розвиток волонтерського руху, підвищити його ефективність та прозорість,

що в кінцевому результаті сприяє зменшенню негативних наслідків воєнного стану для населення.

Метою дослідження є розробка та впровадження веб-орієнтованої системи для ефективної координації та надання волонтерської допомоги в умовах воєнного стану.

Для досягнення мети необхідно вирішити наступні проблеми:

1. Вивчити теоретичні основи волонтерської діяльності в умовах воєнного стану для розуміння ключових аспектів та викликів цієї діяльності.
2. Провести огляд існуючих рішень у сфері волонтерської допомоги з метою визначення їх переваг і недоліків.
3. Розробити структурно-функціональну модель системи для ефективної організації та управління бізнес-процесами волонтерської допомоги.
4. Створити алгоритм роботи системи, що забезпечить чітке виконання всіх етапів процесу надання волонтерської допомоги.
5. Спроектувати модель бази даних для зберігання та управління інформацією про користувачів, запити на допомогу та ресурси.
6. Розробити архітектуру веб-орієнтованої інформаційної системи, що забезпечить його стабільну та ефективну роботу.
7. Реалізувати програмну частину веб-орієнтованої інформаційної системи з використанням сучасних технологій та фреймворків.
8. Створити інтуїтивно зрозумілий інтерфейс користувача, який забезпечить зручний доступ до всіх функцій системи для волонтерів та користувачів.

Здійснення цих завдань дозволить створити ефективну інформаційну систему, орієнтовану на веб, яка допоможе волонтерам у воєнному стані. Такий підхід дозволить постраждалим отримувати оперативну та скоординовану підтримку, підвищить ефективність управління ресурсами та координації волонтерської діяльності, а також зробить використання ресурсів більш прозорим і звітним. У результаті це покращить здатність волонтерських організацій реагувати на надзвичайні ситуації та надавати постраждалим необхідну допомогу.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Структурно-функціональне моделювання

Для моделювання функцій системи та організації бізнес-процесів використовуються структуровані та об’єктно-орієнтовані методи. Одним із найбільш поширених інструментів для цього є IDEF0 [7]. Це метод, який описує системи та процеси організації як набір взаємопов’язаних функцій на графіку. Він дозволяє вивчати останні без прив’язки їх до місць, де вони виконуються. З іншого боку, це досить зручно.

У стандарті IDEF0 управління відображає об’єкти, інформаційні та матеріальні потоки, трансформовані в бізнес-процеси.

Граф контексту містить такі дані: вхідні дані: запит на публікацію оголошення про допомогу; управління: правила реєстрації та авторизації користувачів, інструкції публікації оголошень, положення про доброчесність; механізми: веб-додатки, апаратне забезпечення та база даних; і вихідні дані: розміщене оголошення про допомогу.

На рисунку 2.1 представлено функціональну діаграму IDEF0 виконання проектів волонтерською організацією.

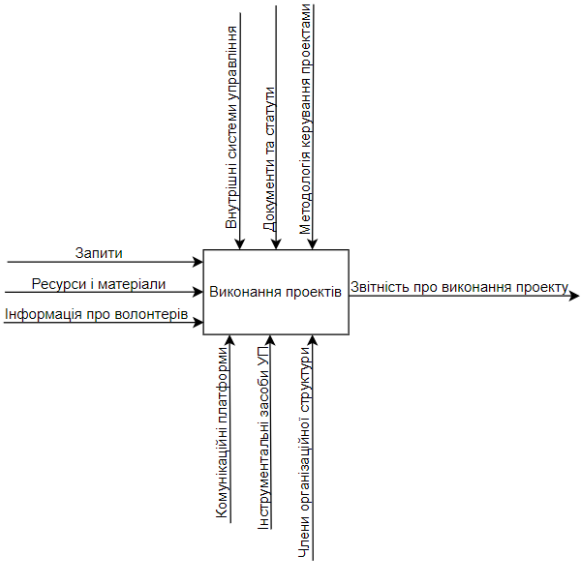

Рисунок 2.1 - IDEF0-діаграма виконання проектів

Більш детально про стрілки та їх опис написано в таблиці (див. Таблиця А, Додаток А).

Таким чином, використання цього підходу дозволяє ефективно організувати процеси публікації оголошень про допомогу, забезпечуючи чіткі правила та інструменти для досягнення кінцевого результату.

2.2 Алгоритм роботи системи

На рисунку 2.2 представлено схему алгоритмів роботи системи з вказанням стандартних носіїв і засобів збереження та відображення інформації.

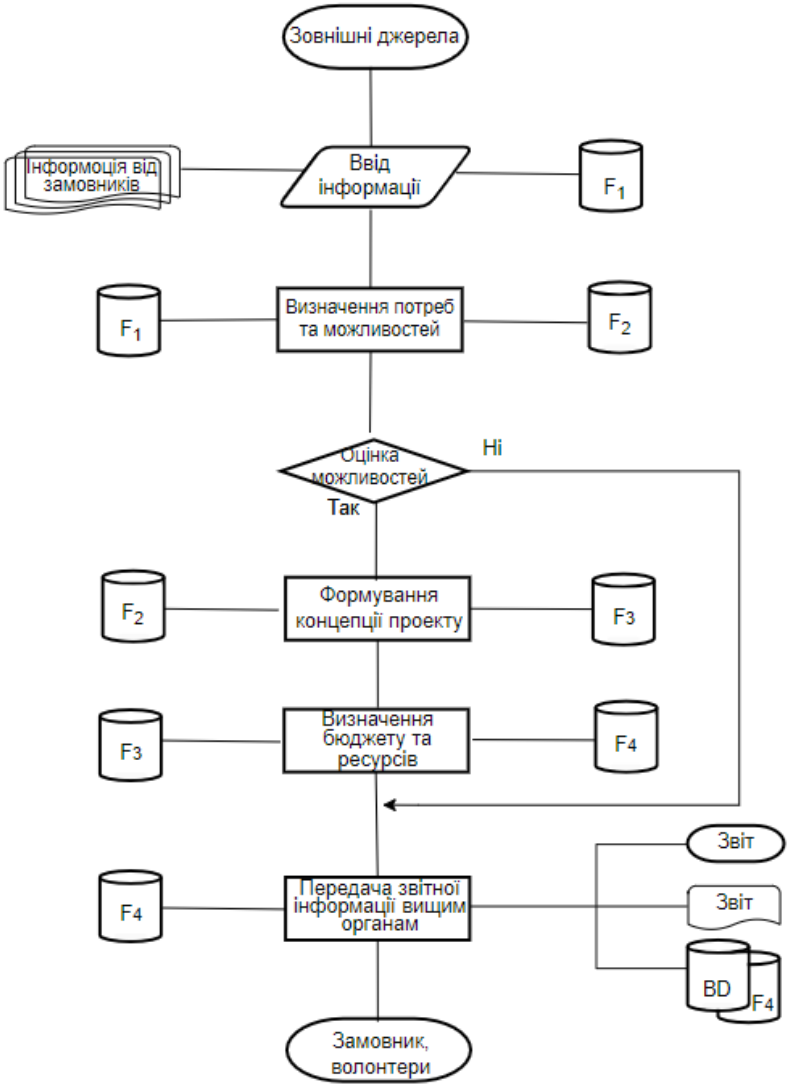

Рисунок 2.2 – Схема алгоритму процесу формування початкової інформації про проект

Як показано на рисунку 2.2, загальний алгоритм процесу формування початкової інформації про проект такий:

- Джерелом інформації є попередні дані та дані проекту.
- Наступним кроком є визначення потреб і можливостей, а також того, чи готові волонтери та благодійні організації взяти участь у проекті.
- Визначення можливостей. Волонтери можуть взяти участь у проекті, якщо можливості задовольняють потреби.
- Створення концепції проекту, розробка першої версії та узгодження.
- Визначення бюджету та ресурсів, необхідних для проекту.
- Завершення звітів, які надсилаються волонтерам і замовнику, є останнім етапом.

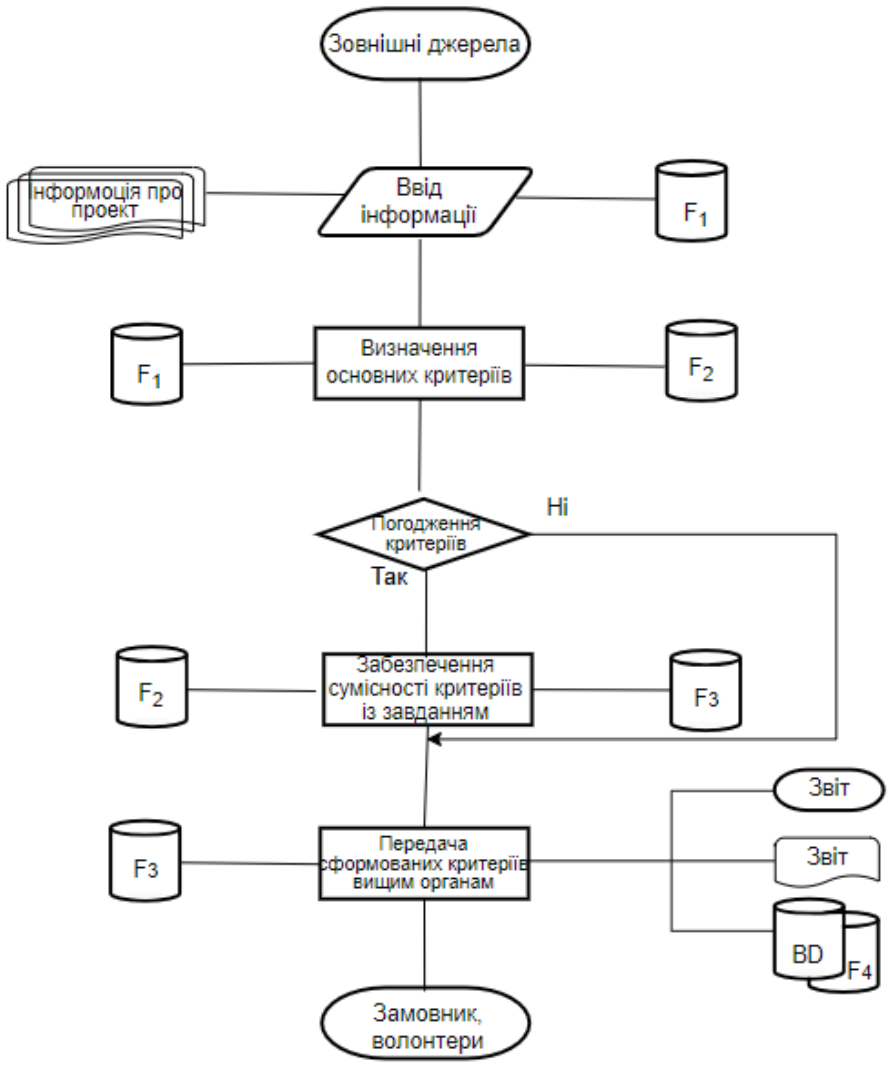

Рисунок 2.3 – Схема алгоритму процесу формування критеріїв проекту

Загальний алгоритм процесу формування критеріїв проекту має таку логіку, як показано на малюнку 2.3:

– Дані отримані з документів вводу.
– Визначення основних критеріїв, які впливають на процес створення проекту.
– Внесення змін до критеріїв і обговорення їх із замовниками
– Перевірка, що критерії відповідають завданню, що вони відповідають завданням і що очікуваний результат відповідає розробленому макету критеріїв.
– Звіт як у паперовому, так і в електронному вигляді, а також внесення зміни в існуючий файл коштів і базу даних.
– Надсилання звітів замовникам і волонтерам.

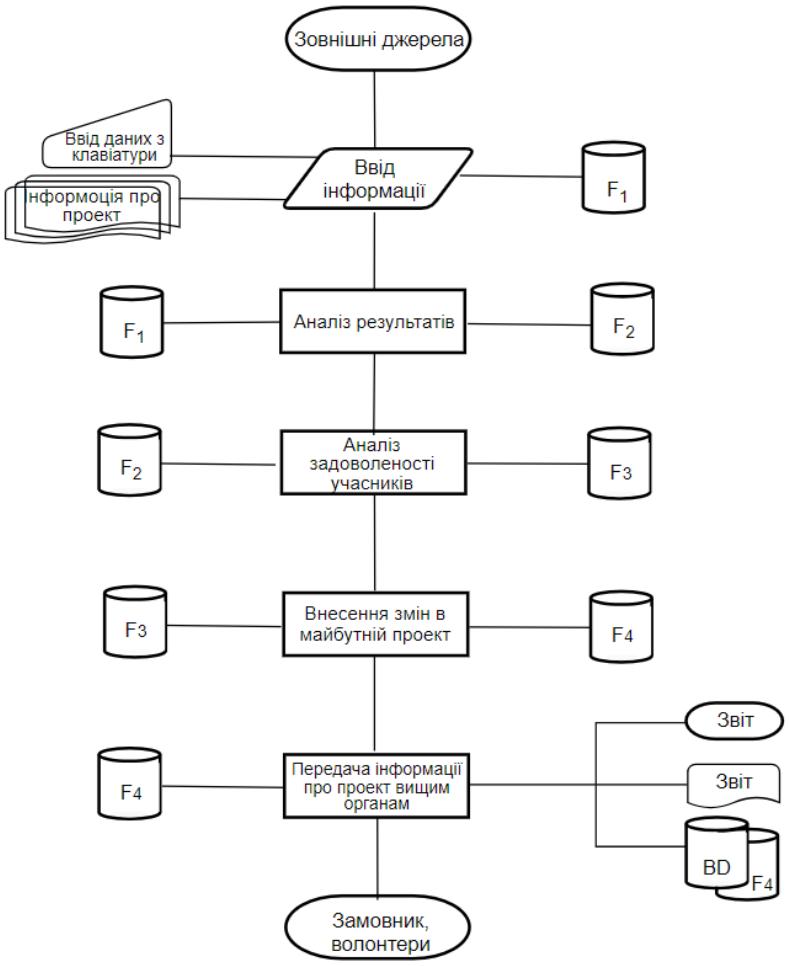

Рисунок 2.4 – Схема алгоритму процесу звітності про виконання проекту

Загальний алгоритм процесу звітності про виконання проекту має таку логіку, як показано на малюнку 2.4:

- Отримання інформації про проект.
- Аналіз результатів, визначення відмінностей від початкового плану та визначення причин цих відмінностей.
- Аналіз задоволеності учасників проекту та збір відгуків.
- Внесення змін до планів проекту та пропозицій щодо покращення стратегій, планів і процесів.
- Передача інформації про проект керівництву, складання звітів і створення бази даних про проект.

2.3 Проектування моделі бази даних

Було вирішено підключити базу даних для роботи з даними веб-орієнтованої системи. База даних буде зберігати дані користувачів, оголошення, грошові збори та інші дані.

На рисунку 2.5 зображено структуру бази даних, яка розділена на п'ять таблиць: category, region, statement, selected, і comment.

Таблиця category:

- id: унікальний ідентифікатор категорії.
- name: назва категорії.

Таблиця region:

- id: унікальний ідентифікатор регіону.
- name: назва регіону.

Таблиця statement:

- id: унікальний ідентифікатор заяви.
- title: заголовок заяви.
- description: опис заяви.
- category_id: зв'язок з таблицею category.

- region_id: зв'язок з таблицею region.
- owner_uuid: ідентифікатор власника (користувача).
- date_created: дата створення заяви.
- date_updated: дата оновлення заяви.

Таблиця selected:

- id: унікальний ідентифікатор вибраної заяви.
- statement_id: зв'язок з таблицею statement.
- user_uuid: ідентифікатор користувача, який вибрав заяву.

Таблиця comment:

- id: унікальний ідентифікатор коментаря.
- statement_id: зв'язок з таблицею statement.
- user_uuid: ідентифікатор користувача, який написав коментар.
- comments: текст коментаря.
- date_created: дата створення коментаря.

Зв'язки між таблицями:

statement -> category: Кожна заява має категорію, зв'язану через category_id.

statement -> region: Кожна заява має регіон, зв'язаний через region_id.

selected -> statement: Таблиця selected відслідковує, які заяви були вибрані користувачами, через statement_id.

comment -> statement: Коментарі прив'язані до заяв через statement_id.

Проектування моделі бази даних передбачає підключення бази даних для ефективної роботи з даними веб-орієнтованої системи, яка буде зберігати інформацію про користувачів, оголошення, грошові збори та інші дані. Структура бази даних складається з п'яти основних таблиць: category, region, statement, selected, і comment. Таблиці забезпечують унікальне ідентифікування категорій, регіонів, заяв, вибраних заяв та коментарів відповідно. Зв'язки між таблицями забезпечують інтеграцію даних, зокрема зв'язок заяв з категоріями та регіонами, відслідковування вибраних заяв

користувачами, а також асоціювання коментарів із заявами. Це проектування забезпечує структуроване та ефективне управління даними у системі.

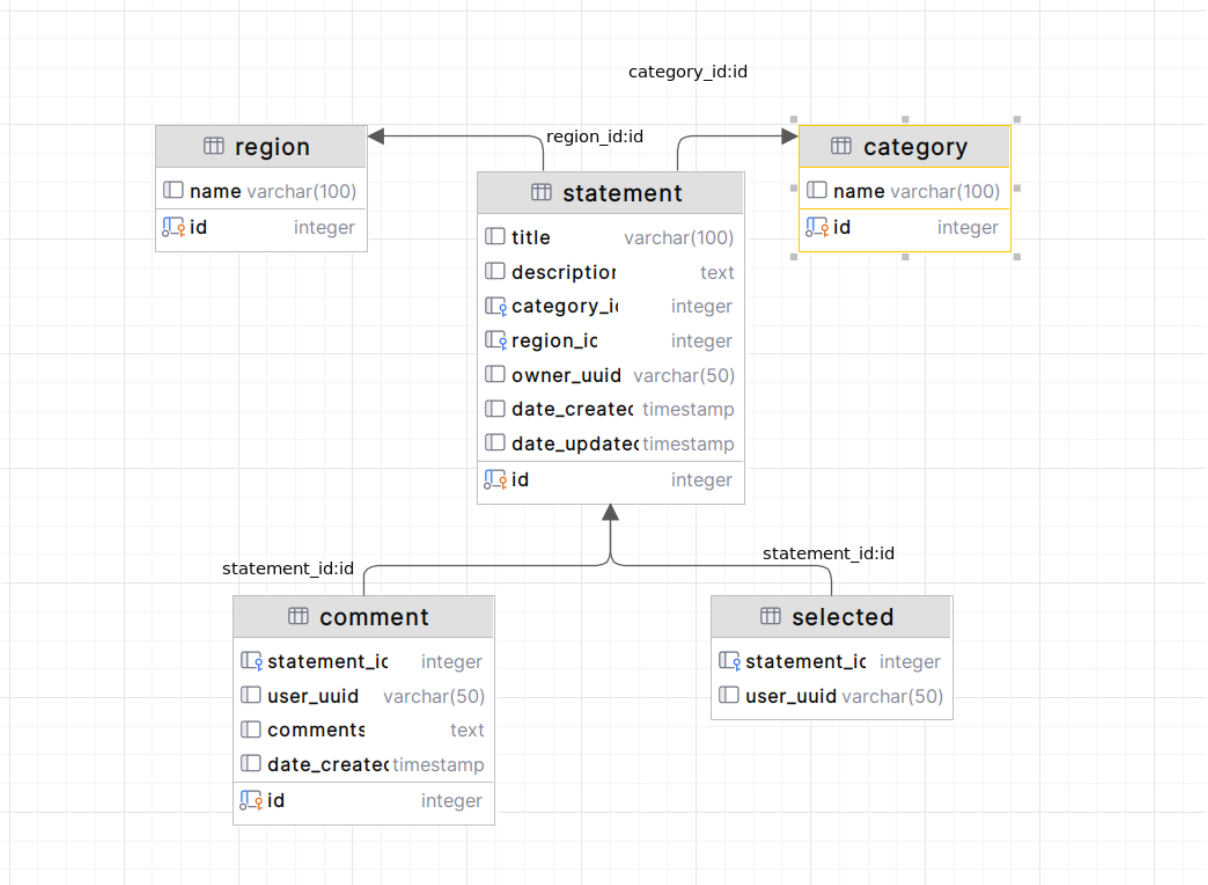

Рисунок 2.5 – Схема структури бази даних

2.4 Архітектура веб-орієнтованої інформаційної системи

У веб-додатках використовується архітектура Model-View-Controller (MVC) [8]. Це дозволяє відокремити їхню логіку від інтерфейсу.

Оскільки цю архітектуру легше оновлювати веб-додатки та кожна сторінка має свій файл View, кожна таблиця в базі даних має модель і кожна функція спільного «об’єкта» має свій контролер. Оскільки структура містить тисячі різних інструментів, це значно спрощує кодування.

При відкритті сторінки процес роботи веб-додатку можна розділити на кілька етапів:

– Посилання дозволяє користувачеві відвідати сторінку.
– Маршрутизатор запускається шляхом ідентифікації URI та виклику методу контролера, який був визначений для запиту.
– Контролер виконує додаткові завдання.
– Можна додавати дані до моделі або заповнювати базу даних ними.
– Користувач може побачити сторінку з дизайном, яка містить базу даних.

Архітектура даного веб-додатку представлена на рисунку 2.6.

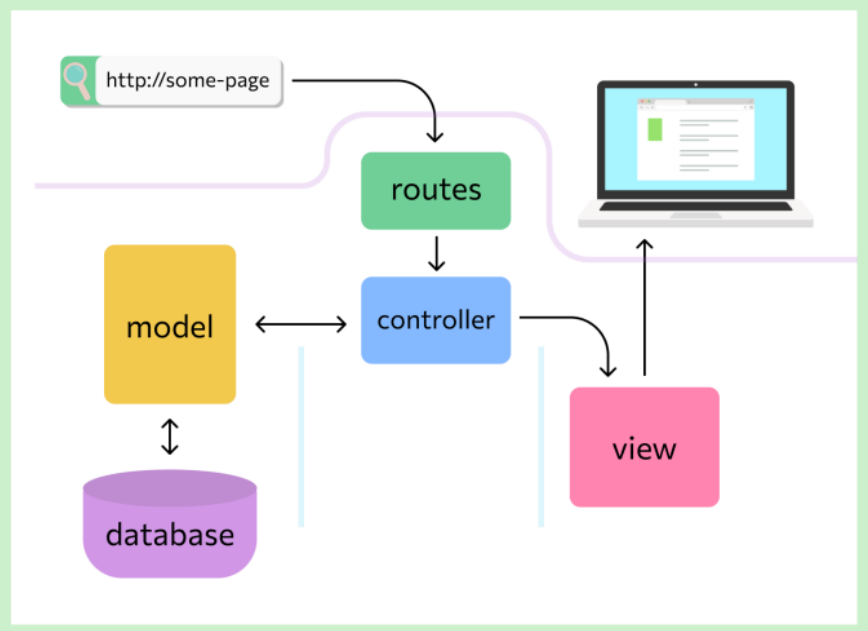

Рисунок 2.6 – Архітектура веб-додатку

Архітектура веб-додатку, побудована на основі Model-View-Controller (MVC), забезпечує чітке розділення логіки та інтерфейсу, що полегшує оновлення та підтримку системи. Кожна сторінка має свій файл View, таблиці бази даних представлені окремими моделями, а функції керуються відповідними контролерами. Така структура, що використовує безліч інструментів, спрощує процес кодування та забезпечує ефективне управління даними. Під час відкриття сторінки маршрутизатор ідентифікує URI, запускає відповідний контролер для виконання завдань, який взаємодіє з моделями та базою даних, в результаті чого користувач бачить сторінку з відповідним дизайном і даними.

3 РЕАЛІЗАЦІЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Програмна реалізація

Перед початком розробки веб-орієнтованої інформаційної системи необхідно вибрати середовище розробки та технологію розробки. Такі технології, як HTML і CSS для створення структури та зовнішнього вигляду, Java для обробки логіки на сервері, Postgres для взаємодії з базою даних, JavaScript та JQuerly для створення інтерактивних елементів, vert.x — фреймворк, який полегшує розробку, Bootstrap — фреймворк CSS, який використовується для респонсивного дизайну, JDBC – для організації зв'язку з базаю даних, AWS Cognito – для реєстрації та логіну користувача, Maven – для збірки проекту та викачування бібліотек, Freemarker – для генерації динамічних веб-сторінок. Було обрано Jet Brains Idea для написання коду [9].

Цей рисунок 3.1 містить налаштування для підключення до бази даних PostgreSQL та налаштування AWS Cognito для автентифікації користувачів.

Для того, щоб створити зв'язок з базою даних потрібний Driver, Username та Password. Так як писати ці значення в коді небезпечно, застосовано конфігураційний файл webapp.yml.

```
pool.pg.url: jdbc:postgresql://localhost:5432/webapp?charSet=UTF-8&tcpKeepAlive=true&targetServerType=master&loadBalanceHosts=true
pool.pg.username: postgres
pool.pg.password: postgres

path.prefix: app/contrib
aws.profile: terraform-sb

aws.region: us-east-1
aws.user.pool.id: us-east-1_37d07576bdb84dcfa84a0ad677e02a06
aws.user.pool.client.id: znhc5q37rpzfnee8sdhiji55sd
```

Рисунок 3.1 – Вміст файлу webapp.yml

Файл pom.xml (додаток А) є конфігураційним файлом для проекту на Maven. Він описує різні аспекти проекту, такі як залежності, плагіни, збірка та інші налаштування.

На рисунку 3.2 показано клас «StaticVariables» який використовується для зберігання статичних змінних та налаштувань, що стосуються конфігурації додатка.

Там є всі статистичні змінні, яким присвоюються значення з файлу webapp.yml (Рис.3.1).

```
package com.webapp;

import com.auth0.jwt.JWTVerifier;
import com.google.inject.Inject;
import com.google.inject.name.Named;
import io.micrometer.core.instrument.util.StringUtils;
import io.vertx.jdbcclient.JDBCPool;
import software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;

import java.util.HashMap;
import java.util.Map;

public class StaticVariables {
    public static String appHome;

    static {
        String path = System.getProperty("app.home");
        if (StringUtils.isBlank(path))
            throw new RuntimeException("The VM option -Dapp.home= is not defined. It should be the home directory of the application");
        StaticVariables.appHome = path.replaceAll("/+$", "");
    }
    @Inject @Named("aws.user.pool.id") public static String awsUserPoolId;
    @Inject @Named("aws.user.pool.client.id") public static String awsUserPoolClientId;
    @Inject(optional = true) public static CognitoIdentityProviderClient cognitoIdentityProviderClient;
    @Inject(optional = true) public static JWTVerifier jwtVerifier;
    static Map<String, String> dataSourceConfig = new HashMap<>();
    @Inject public static JDBCPool dbp;
}
```

Рисунок 3.2 – Вміст файлу StaticVariables.java

Файл config.java (додаток Б) показує використання класу Config для конфігурації залежностей у додатку, який використовує Vert.x та AWS Cognito. Конструктор приймає JsonObject, що містить конфігураційні параметри, і ці параметри потім обробляються в методі configure(). Зокрема, клас налаштовує пул з'єднань до PostgreSQL, створює клієнт для AWS Cognito, а також налаштовує перевірку JWT токенів за допомогою RSA ключів. Після налаштування всі ці залежності ін'єктуються в потрібні частини додатку.

На рисунку 3.3 зображено файл AwsCognitoRSAKeyProvider.java, де показано клас який реалізує інтерфейс RSAKeyProvider для забезпечення верифікації JWT токенів за допомогою публічних ключів, отриманих з AWS Cognito. Конструктор приймає ідентифікатор пулу користувачів AWS Cognito і базовий URL, формуючи повний URL для отримання ключів у форматі JWK.

Файл AwsCognitoService.java (додаток В) показує клас «AwsCognitoService», який забезпечує інтеграцію з Amazon Cognito для управління користувачами. Метод «signUp» реєструє нового користувача, використовуючи надані дані (email, пароль, ім'я, прізвище, телефон). Метод «confirmSignUp» підтверджує реєстрацію

користувача за допомогою коду підтвердження і додає його до групи "users". Метод «signUp» з іншими параметрами ініціює аутентифікацію користувача, а метод «addUserToGroup» додає користувача до групи.

Файл main.java (додаток Г) містить клас «main», який відповідає за запуск і конфігурацію веб-орієнтованої інформаційної системи на базі Vertx. Спочатку завантажується конфігураційний файл webapp.yml і конвертується в об'єкт «JsonObject». Потім створюються об'єкти «VertxOptions» і «Vertx» з відповідними налаштуваннями. Для збору метрик створюється реєстр Prometheus і підключаються метрики для процесорів, аптайму і файлових дескрипторів. Створюється інжектор Guice з конфігурацією «config» яка містить налаштування для різних сервісів, таких як AWS Cognito. Після цього розгортається основний вертикаль «Web» з відповідними опціями розгортання і обробником результатів розгортання.

```
package com.webapp.service;

import com.auth0.jwk.JwkException;
import com.auth0.jwk.JwkProvider;
import com.auth0.jwk.JwkProviderBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.net.MalformedURLException;
import java.net.URL;
import java.security.interfaces.RSAPrivateKey;
import com.auth0.jwt.interfaces.RSAKeyProvider;
import java.security.interfaces.RSAPublicKey;

// JWT verification for Aws cognito api
public class AwsCognitoRSAKeyProvider implements RSAKeyProvider {
    private final Logger log = LoggerFactory.getLogger(this.getClass());
    private URL aws_kid_store_url;
    private final JwkProvider provider;

    public AwsCognitoRSAKeyProvider(String aws_user_pools_id, String urli) {
        String url = String.format(urli + "/%s/.well-known/jwks.json", aws_user_pools_id);
        try {
            aws_kid_store_url = new URL(url);
        } catch (MalformedURLException e) {
            log.error(String.format("Invalid URL provided, URL=%s", url));
        }
        provider = new JwkProviderBuilder(aws_kid_store_url).build();
    }


    @Override
    public RSAPublicKey getPublicKeyById(String kid) {
        try {
            return (RSAPublicKey) provider.get(kid).getPublicKey();
        } catch (JwkException e) {
            throw new RuntimeException(String.format("Failed to get JWT kid=%s from aws_kid_store_url=%s", kid, aws_kid_store_url));
        }
    }

    @Override
    public RSAPrivateKey getPrivateKey() {
        return null;
    }

    @Override
    public String getPrivateKeyId() {
        return null;
    }
}
```

Рисунок 3.3 – Вміст файлу AwsCognitoRSAKeyProvider.java

Файл Web.java (додаток Д) вміщує всю логіку веб-орієнтованої інформаційної системи та описує основні функції. До прикладу функцію, яка приймає запит з браузера, обробляє його і віддає відповідь від сервера, тоді коли сервер повертає браузеру сторінку з даними.

3.2 Інтерфейс користувача

При завантаженні web-додатку волонтерської допомоги, користувач потрапляє на головну сторінку (рис. 3.4). Вона містить інформаційні блоки та зображення, які пояснюють, як працює цей ресурс. За допомогою меню користувач може перейти до наступних розділів з головної сторінки (рис. 3.5).

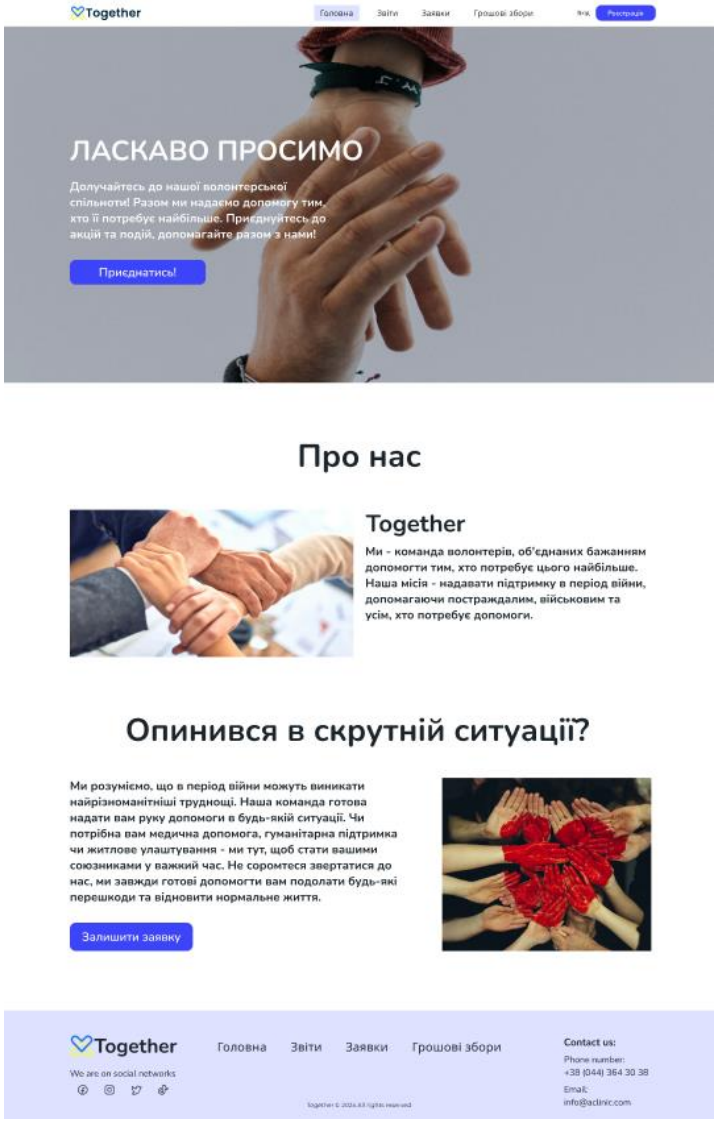

Рисунок 3.4 – Головна сторінка веб-орієнтованої інформаційної системи

Рисунок 3.5 – Меню веб- орієнтованої інформаційної системи

Користувач повинен зареєструватися, щоб отримати доступ до всіх наявних функцій, або здійснити вхід у вже існуючий акаунт. Рисунки 3.6–3.8 показують форми для реєстрації та авторизації.

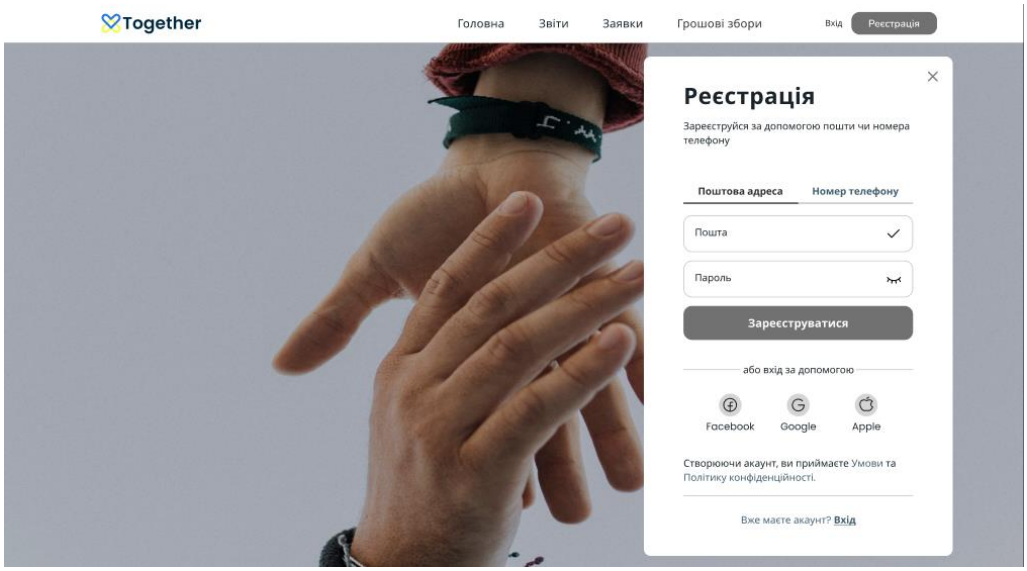

Рисунок 3.6 – Форма реєстрації через поштову адресу

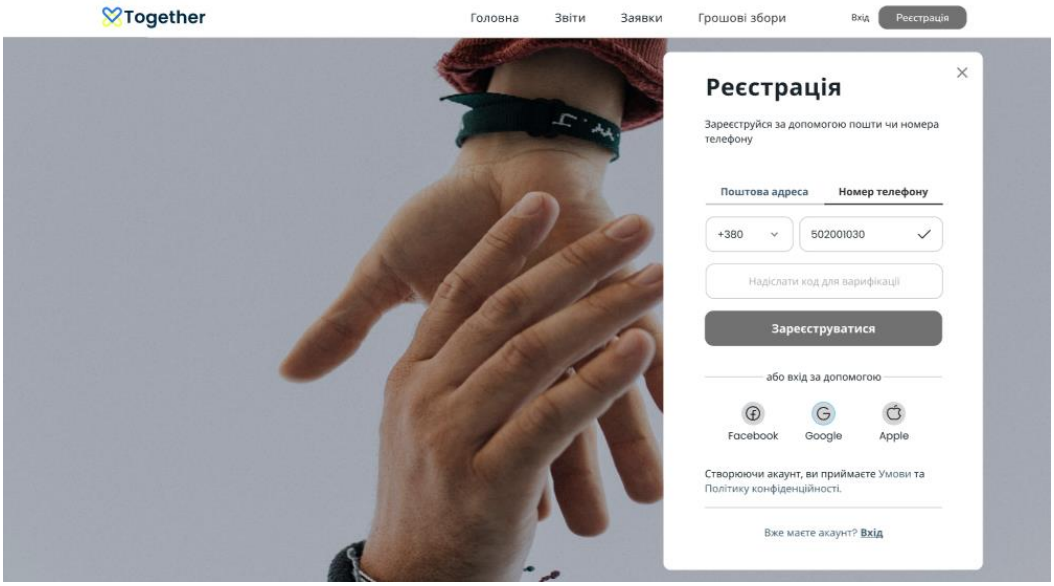

Рисунок 3.7 – Форма реєстрації через номер телефону

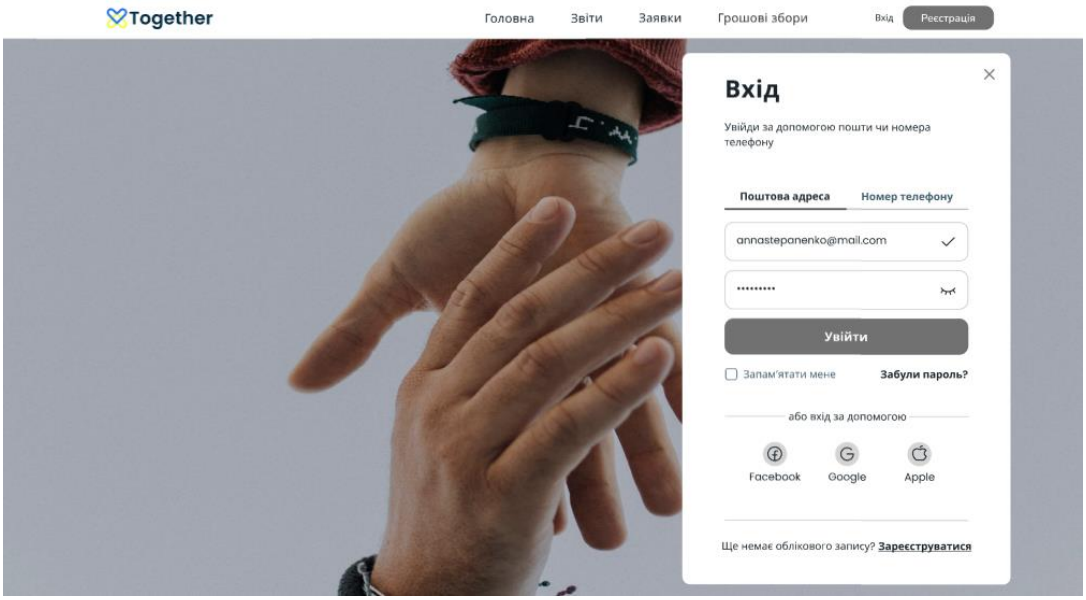

Рисунок 3.8 – Форма авторизації

Наступною, після головної, в меню розміщується сторінка звітності (Рис.3.9), на якій можна побачити таку інформацію як: загальну кількість закритих заявок, кількість скасованих заявок, кількість активних заявок та статистику основних потреб із заявок.

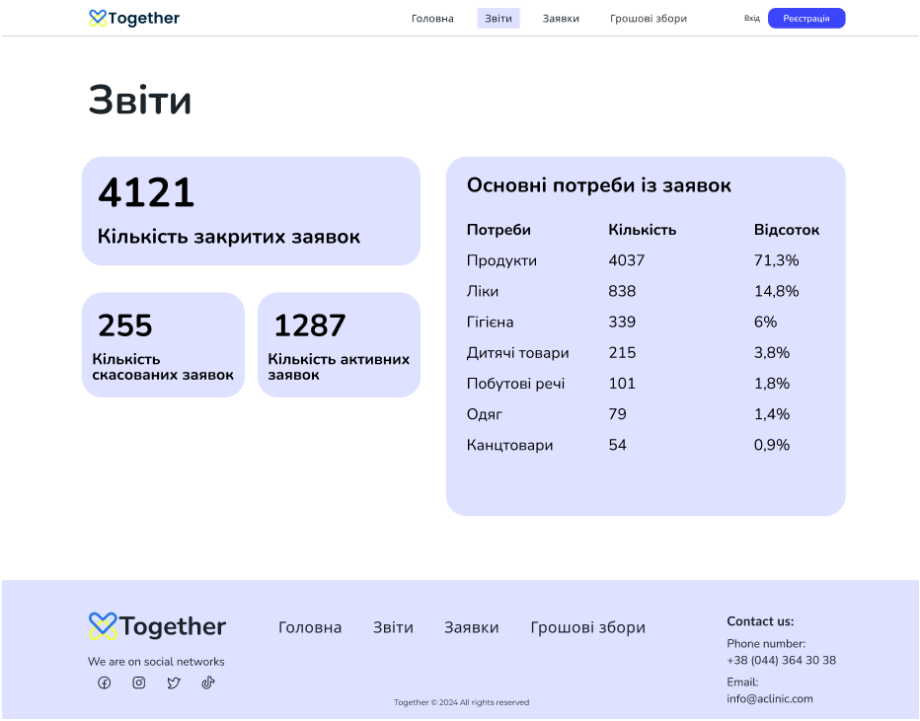

Рисунок 3.9 – Сторінка «Звіти»

«Заявки» є наступним пунктом меню, як показано на рис. 3.10. Це сторінка, яка містить оголошення від осіб, які потребують допомоги.

Присутня фільтрація, щоб полегшити пошук необхідної заявки в онлайн-додатку. Наприклад, за назвою, за категоріями допомоги та місцем розташування автора оголошення.

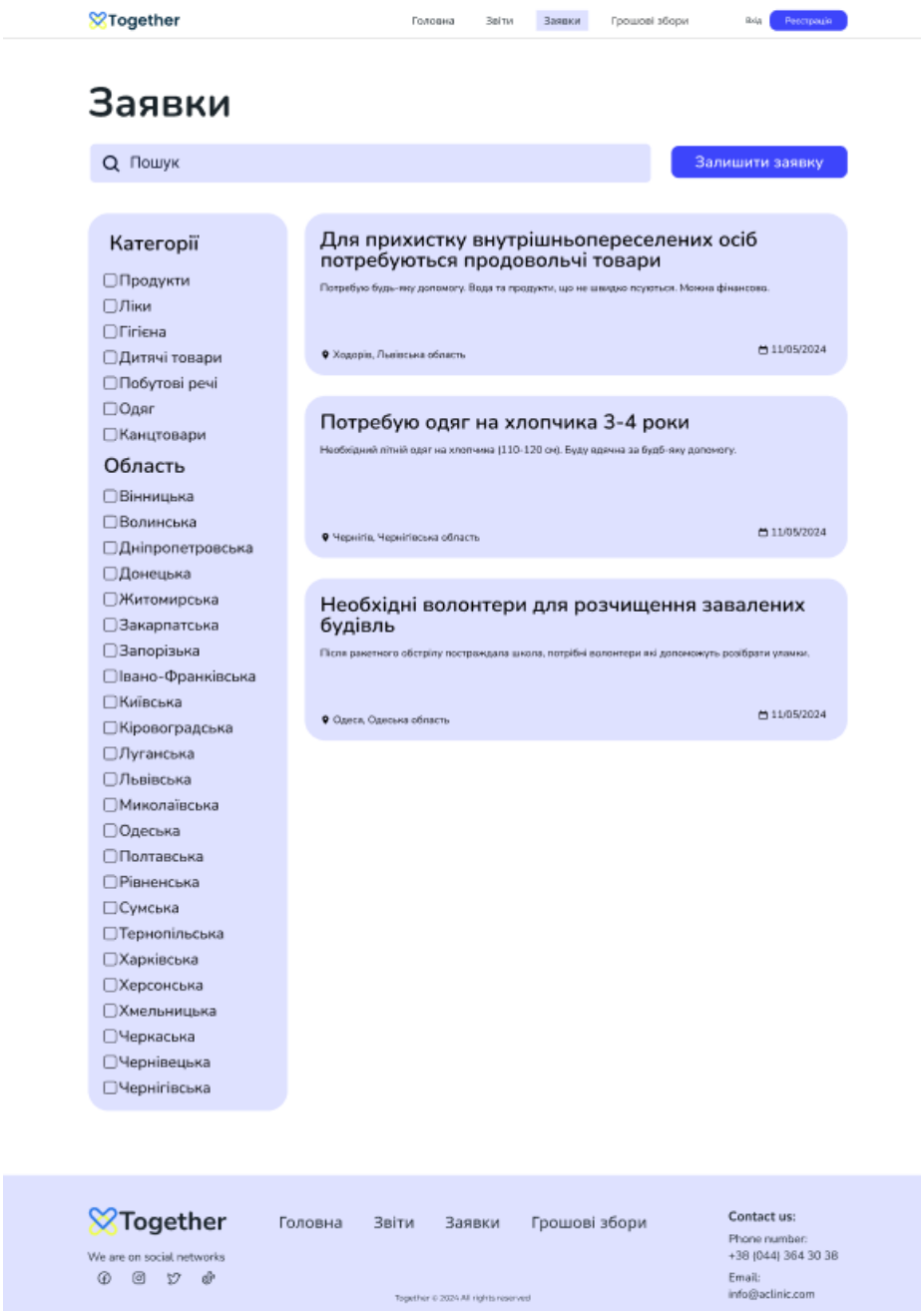

Рисунок 3.10 – Сторінка «Заявки»

На рис. 3.11 показано, що натиснувши на оголошення можна перейти на його окрему сторінку.

Ознайомитись з контактними даними автора та залишити коментар. Доступ до функції публічного листування під оголошеннями обмежений та доступний лише для зареєстрованих користувачів. Волонтер може не тільки написати коментар, але й видалити його за потреби.

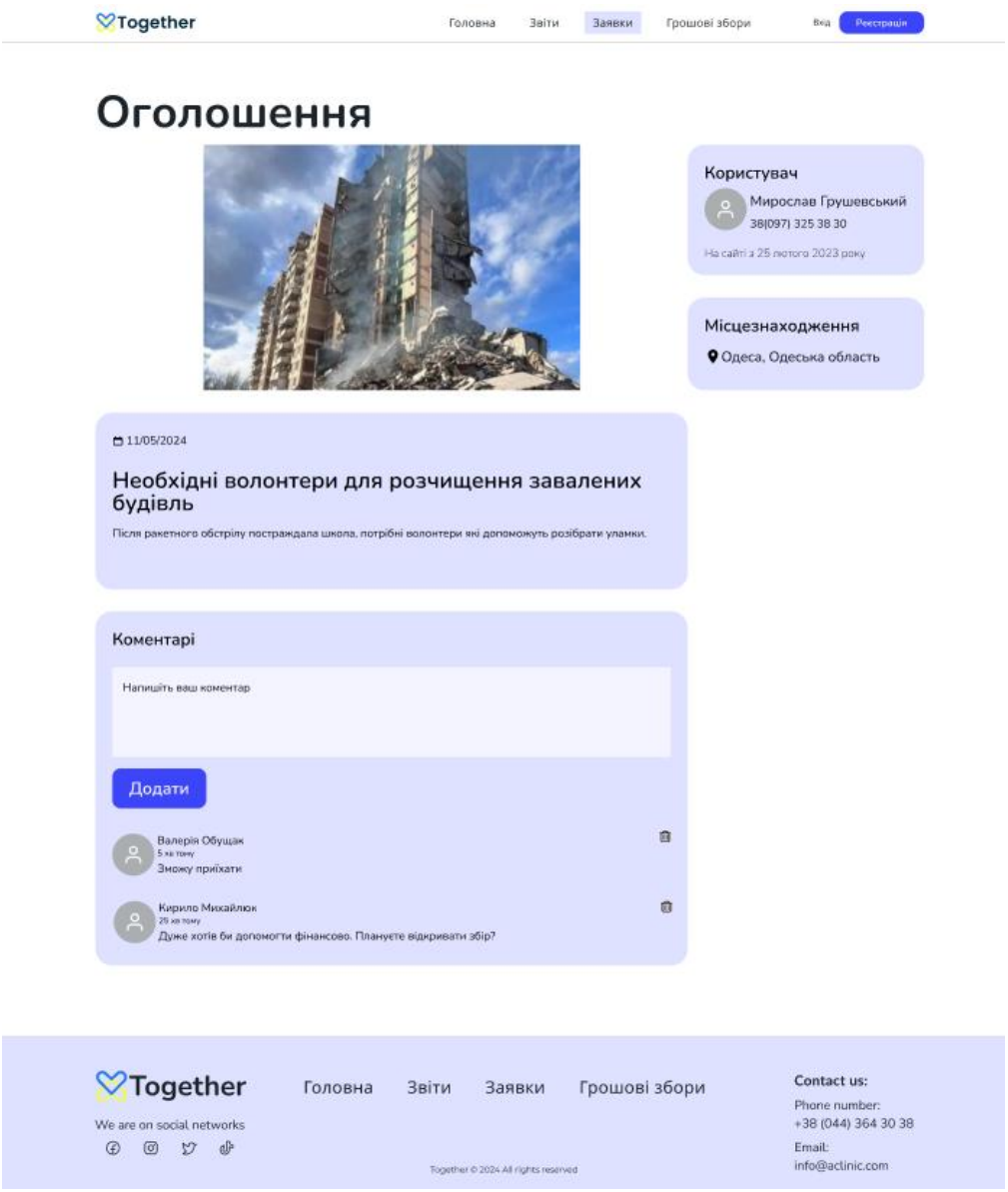

Рисунок 3.11 – Сторінка «Оголошення»

На рисунку 3.11 зображено процес створення заявки та перелік необхідної інформації. Для безпеки та контролю створення заявок, вони проходять перевірку на доцільність та і сама функція доступна лише для авторизованих користувачів.

Модальне вікно «Відхилення заявки» зображено на рисунку 3.12.

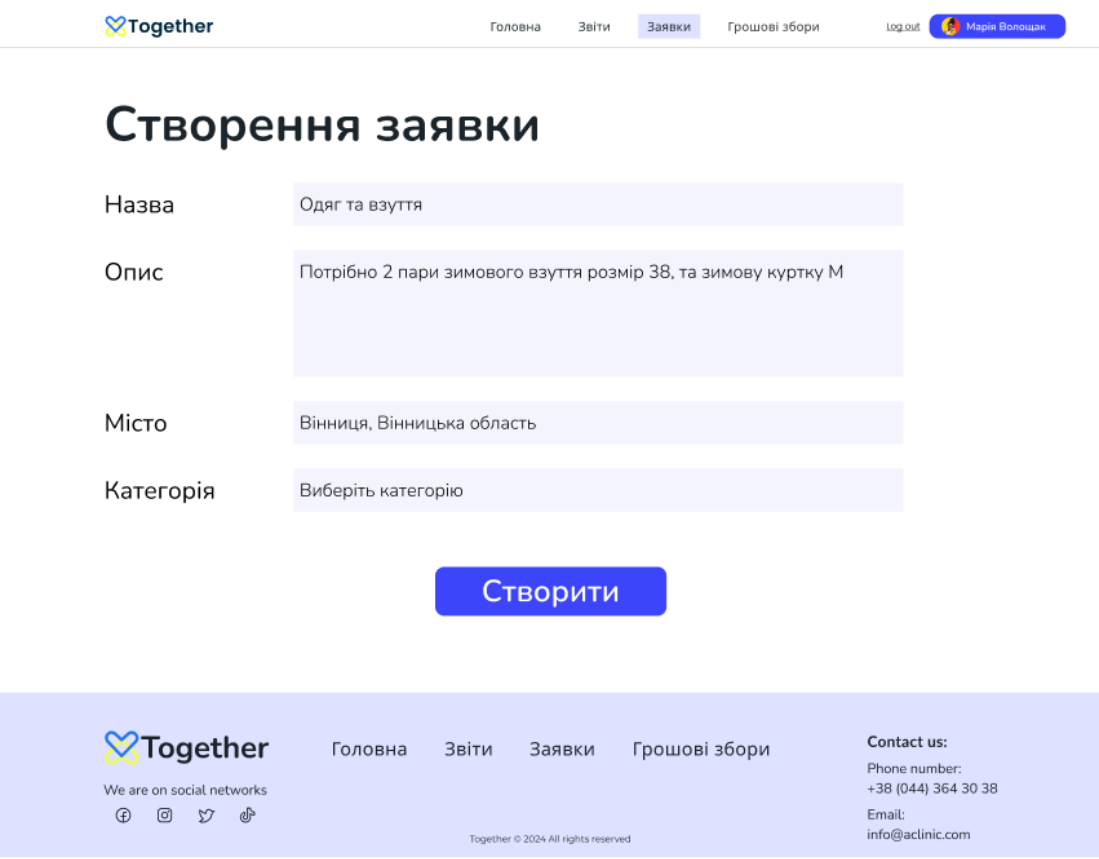

Рисунок 3.12 – Сторінка «Створення заявки»

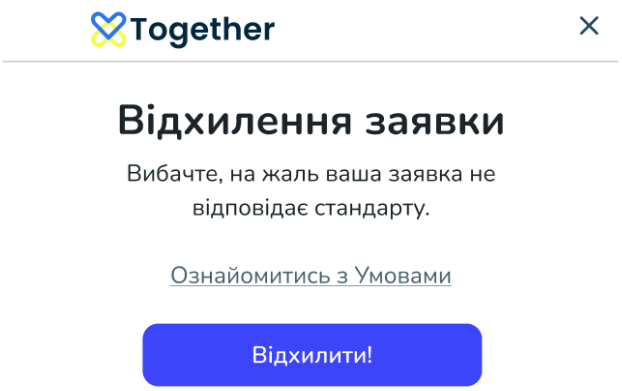

Рисунок 3.13 – Модальне вікно «Відхилення заявки»

Так як, адміністративна панель передбачає створення грошових зборів, можна перейти до сторінки «Грошові збори» з головного меню (рис. 3.14). Симулятор розроблений для переказу коштів. Щоб задонатити, користувач, який зареєструвався, нажимає кнопку «Поповнити» у діалоговому вікні.

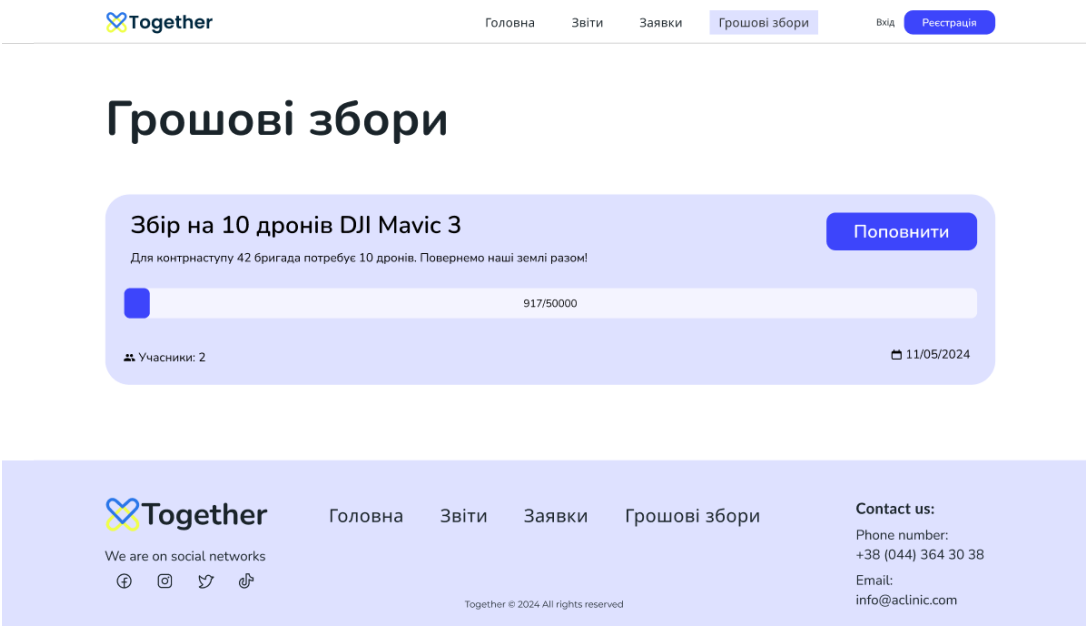

Рисунок 3.14 – Сторінка «Грошові збори»

Натиснувши на значок акаунту в правому верхньому кутку, можна потрапити на сторінку облікового запису (Рис. 3.15). Клікнувши на пункт зліва «Мої рекомендації» зразу видніються фільтри по яких можна їх сортувати.

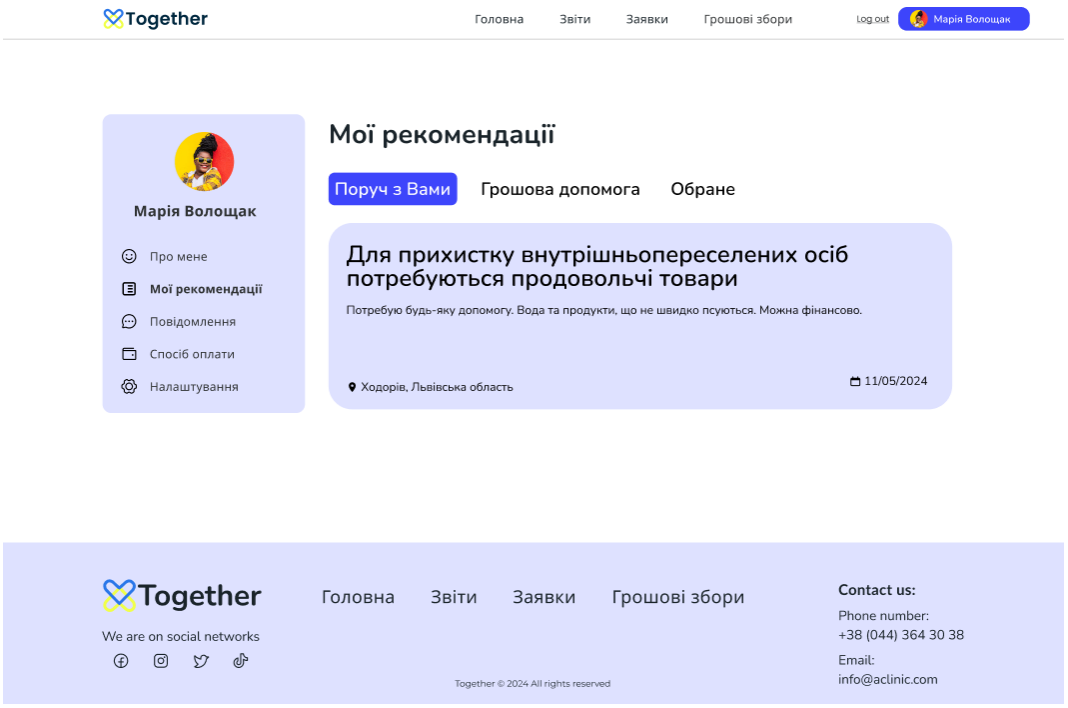

Рисунок 3.15 – Сторінка «Мої рекомендації»

Інтерфейс користувача веб- орієнтованої інформаційної системи волонтерської допомоги забезпечує зручну навігацію та доступ до основних функцій системи. На головній сторінці представлені інформаційні блоки та зображення, які пояснюють принципи роботи ресурсу, а меню дозволяє користувачам переходити до інших розділів, таких як звіти та заявки. Користувачі повинні зареєструватися або увійти до свого облікового запису, щоб мати повний доступ до функцій, включаючи створення та перегляд заявок, залишення коментарів, та участь у грошових зборах. Інтерфейс також включає форми для реєстрації та авторизації, фільтри для зручного пошуку заявок, і сторінку облікового запису для керування особистими рекомендаціями.

ВИСНОВКИ

Дипломний проект «Веб-орієнтована інформаційна система надання волонтерської допомоги» досяг значних результатів, які дозволяють забезпечити ефективне функціонування системи та задоволення потреб користувачів.

На першому етапі роботи було проведено ретельне вивчення сфери волонтерської допомоги. Було виявлено основні проблеми та потреби в цій галузі. Це допомогло отримати розуміння основних завдань, які має вирішувати розроблена інформаційна система, орієнтована на веб. Досліджено існуючі варіанти систем надання волонтерської допомоги, орієнтованих на Інтернет. Було проведено аналіз як їхніх переваг, так і їхніх недоліків, що дозволило визначити основні вимоги до створеної системи. Цей аналіз був основою для визначення функціональних і нефункціональних вимог до системи, а також допоміг уникнути типових помилок, пов’язаних з існуючими рішеннями.

У другій частині дослідження було продемонстровано структурно-функціональне моделювання системи. Було розроблено модель, яка показує основні функціональні частини системи та те, як вони взаємодіють. Це включало опис ролей користувачів, основних бізнес-процесів і їх взаємозв’язків. Було розроблено детальний алгоритм роботи системи. Він містить опис основних процесів і їх логічну послідовність. Контекстна діаграма IDEF0 демонструє взаємодію основних компонентів системи. Крім того, було розроблено схему алгоритму для різних процесів, а також схему структури бази даних. Вони слугуватимуть основою для подальшої реалізації системи.

Базу даних було спроектовано та реалізовано у відповідному середовищі відповідно до розробленої схеми структури бази даних. Для ефективного зберігання та обробки даних використовуються новітні інструменти та технології. Розроблена база даних відповідає всім вимогам проекту та гарантує безпечне зберігання даних.

Java — це мова програмування, яка була використана для реалізації основних функцій системи. Обробка запитів користувачів, управління базою даних і взаємодія

з іншими компонентами системи є функціями серверної частини програми. Для забезпечення високої продуктивності та надійності системи використовуються сучасні бібліотеки та фреймворки. Особлива увага була приділена створенню зручного для користувача інтерфейсу. Система має зручний інтерфейс, який легко зрозуміти, і підходить для різних категорій користувачів. Тестування інтерфейсу було проведено, щоб переконатися, що він відповідає вимогам ергономіки та зручності використання.

Підсумовуючи, завершення цього дипломного проекту дозволило створити ефективну інформаційну систему надання волонтерської допомоги, орієнтовану на веб, яка задовольняє всі вимоги та гарантує високу якість і надійність роботи. Система має значний потенціал для розвитку та вдосконалення, що відкриває нові можливості для волонтерства в сучасному суспільстві.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Здоров’я і суспільство в умовах війни // Збірник наукових статей / Центральноукраїнський інститут розвитку людини Університету «Україна». Кропивницький, 2022. С. 149-151. [Електронний ресурс] – Режим доступу до ресурсу: https://cnej.gov.md/sites/default/files/zbirnik_statey-2022._1.pdf#page=150 (дата звернення: 12.02.2024).
2. Формування соціальної згуртованості територіальної громади шляхом розвитку волонтерської діяльності: Львів, 2023. [Електронний ресурс] – Режим доступу до ресурсу: https://er.ucu.edu.ua/bitstream/handle/1/3714/Tsebenko_mag.pdf?sequence=1 (дата звернення: 12.02.2024).
3. Humanitarian and medical aid to ukraine [Електронний ресурс] – Режим доступу до ресурсу: https://helpukraine.center/ (дата звернення: 16.02.2024).
4. Повернись живим [Електронний ресурс] – Режим доступу до ресурсу: https://savelife.in.ua/ (дата звернення: 16.02.2024).
5. Армія SOS [Електронний ресурс] – Режим доступу до ресурсу: https://armysos.com.ua/uk/ (дата звернення: 16.02.2024).
6. Мрія вільних людей [Електронний ресурс] – Режим доступу до ресурсу: https://mriya.od.ua/ (дата звернення: 16.02.2024).
7. МЕТОДОЛОГІЯ IDEF0 // Stud.com.ua [Електронний ресурс] – Режим доступу до ресурсу: https://stud.com.ua/87184/ekonomika/metodologiya_idef0 (дата звернення: 26.02.2024).
8. MVC Framework - Introduction // Tutorialspoint [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm (дата звернення: 28.02.2024).
9. Jet Brains Idea [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/idea/ (дата звернення: 05.03.2024).

10. HTML Довідник тегів // W3schoolsUA. Українською [Електронний – Режим доступу до ресурсу: https://w3schoolsua.github.io/tags/index.html#gsc.tab=0 (дата звернення: 08.03.2024).
11. CSS Підручник // W3schoolsUA. Українською [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/css/index.html#gsc.tab=0 (дата звернення: 12.03.2024).
12. JavaScript Tutorial // W3schools [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/js/ (дата звернення: 15.03.2024).
13. Java Tutorial// W3schools [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/java/java_intro.asp (дата звернення: 18.03.2024).
14. Maven Tutorial for Beginners // SimpliLearn [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven (дата звернення: 19.03.2024).
15. What is PostgreSQL // PostgreSQL Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/ (дата звернення: 20.03.2024).
16. Introduction to Vert.x // Baeldung [Електронний ресурс] – Режим доступу до ресурсу: https://www.baeldung.com/vertx (дата звернення: 20.03.2024).
17. Amazon Cognito // TechTarget [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/searchaws/definition/Amazon-Cognito (дата звернення: 21.03.2024).
18. Freemarker Common Operations // Baeldung [Електронний ресурс] – Режим доступу до ресурсу: https://www.baeldung.com/freemarker-operations (дата звернення: 21.03.2024).
19. JQuery API // JQuery [Електронний ресурс] – Режим доступу до ресурсу: https://api.jquery.com/ (дата звернення: 21.03.2024).
20. What is JWT // Akana [Електронний ресурс] – Режим доступу до ресурсу: https://www.akana.com/blog/what-is-jwt (дата звернення: 21.03.2024).
21. Introduction to JDBC // Geeksforgeeks [Електронний ресурс] – Режим доступу

до ресурсу: https://www.geeksforgeeks.org/introduction-to-jdbc/ (дата звернення: 21.03.2024).

22. Get started with Bootstrap // Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 21.03.2024).
23. SQL Tutorial // W3schools [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/sql/default.asp (дата звернення: 21.03.2024).
24. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.
25. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.
26. Козак А. Алгоритм роботи веб-орієнтованої інформаційної системи надання волонтерської допомоги у період війни. Матеріали Міжнародної наукової конференції « Проблеми, пріоритети та перспективи розвитку науки, освіти і суспільства в XXI столітті» м. Полтава, Україна, 15 червня 2024 р. URL: https://www.economics.in.ua/2024/06/15.html.

ДОДАТОК А

ОПИС СТРІЛОК З IDEF0-ДІАГРАМИ

Таблиця А.1 – Опис стрілок з IDEF0-діаграми

	Назва стрілки	**Опис**
Вхідна інформація	Запити	– Запит на волонтерську допомогу: Особа або організація може запросити волонтерів для виконання певних проектів або завдань. – Запит на матеріальну допомогу: людина чи організація може звернутися з проханням про матеріальні речі, такі як продукти харчування, одяг і предмети для дому. – Запит на фінансову допомогу: Запит на допомогу для покриття певних витрат, лікування або інших фінансових потреб.
	Інформація про волонтерів	– Персональні дані включають ім'я, прізвище, дату народження, контактні дані (адреса, телефон, електронна пошта) та інші дані. – Досвід та навички: речі, які волонтер може зробити для роботи в організації, наприклад, медичні навички, володіння іноземною мовою та досвід роботи.
	Ресурси і матеріали	– Фінансові ресурси: гроші, які використовуються для фінансування проектів, придбання необхідних матеріалів і оплати витрат, пов’язаних із діяльністю організації. – Матеріальні ресурси: включають одяг, продукти харчування, побутові речовини, ліки та інші речовини, які надаються особам, які потребують допомоги. – Технічні ресурси: це можуть бути обладнання та інструменти, які використовуються для виконання

		проектів, наприклад, автомобілі, інструменти для ремонту та комп'ютери. – Ресурси для лікування: Медичні прилади, ліки, медичний персонал і обладнання можуть бути необхідні для медичних проектів і програм.
Вихідна інформація	Звітність про виконання проекту	– Виконання завдань: Звіт містить інформацію про виконані завдання та активності, а також час, ресурси та робочу силу. – Фінансова звітність: містить інформацію про витрати та доходи проекту. Бюджет проекту, реальні витрати, джерела фінансування та звіт про використання коштів можуть бути включені в звіт. – Результати та досягнення: містить інформацію про те, що було зроблено та що було досягнуто в результаті проекту. Це може включати якісні та кількісні показники, успіхи та невдачі команди проекту.
Управління	Внутрішні системи управління	– Управління ресурсами означає оптимізацію використання організацією фінансових, матеріальних і людських ресурсів. Бюджетування, фінансовий контроль, закупівлі та управління волонтерами є частиною цього. – Процеси та процедури: розробка загальноприйнятих процедур для різних частин діяльності, таких як залучення волонтерів, планування та виконання проектів, фінансовий контроль, комунікація тощо. – Моніторинг та аналіз: створіть системи моніторингу та аналізу, щоб спостерігати за результатами та оцінювати ефективність діяльності організації.
	Документи та статути	– Статут є документом, який визначає юридичний статус організації, її місію, цілі, структуру та принципи її функціонування.

		– Внутрішні правила та політики: Організація може створити внутрішні правила, які регулюють такі речі, як управління волонтерами, фінансовий контроль і безпека. – Договори та угоди: Організація укладає угоди зі спонсорами, партнерами, постачальниками та іншими сторонами щодо фінансової підтримки, співпраці та поставок. – Положення про організаційну структуру: документ, який описує структуру організації, ролі та обов'язки кожного підрозділу та посадової особи.
	Методологія керування проектами	– Планування проектів: цілі, обсяг, ресурси та графік проекту визначаються керівництвом проекту. Складання планів і розробка планів реалізації входять до цього. – Розподіл обов'язків: керівництво проектами визначає, які команди та волонтери будуть відповідати за різні частини проекту. Визначення відповідальних осіб і розподіл відповідальності можуть бути частиною цього процесу. – Моніторинг і контроль: Управління проектами стежить за прогресом проекту, вирішує проблеми та гарантує, що завдання виконуються вчасно. – Звітність та оцінка: Звіт про стан виконання проекту та оцінка результатів виконуються керівництвом проекту. Це допомагає визначити, наскільки проект досягає цілей. – Управління ресурсами: За ефективне використання ресурсів, таких як бюджет та матеріали відповідає керівництво проекту.
Механізми	Комунікаційні платформи	– Електронна пошта: Члени організації можуть використовувати електронну пошту для офіційної комунікації та

		обміну документами та інформацією. – Соціальні мережі: Використання популярних соціальних мереж для спілкування з громадськістю та спільноти волонтерів. Це може включати веб-сайти, такі як Facebook, Twitter і Instagram, серед інших. – Внутрішні форуми та чати: створені форуми або чати в Інтернеті, де члени команди та волонтери можуть спілкуватися, обговорювати проекти та ініціативи разом.
	Інструментальні засоби Управління Проектами	– Системи керування проектами: використання програм для керування проектами, таких як Asana, Trello або Microsoft Project, для створення завдань, відстеження їх прогресу та управління витратами часу та ресурсів. – Системи моніторингу та оцінки результатів (системи моніторингу та оцінки результатів): використання інструментів для збору та аналізу даних про виконання проектів, таких як звіти, опитування та метрики ефективності. – Системи фінансового обліку — це програми для обліку фінансових операцій, бюджету та звітності. – Системи збору та аналізу даних: використання інструментів для збору та аналізу даних про діяльність волонтерів, результати проектів і взаємодію зі стейкхолдерами.
	Члени організаційної структури	– Відділ волонтерства керує волонтерською діяльністю. Це включає набір та підтримку волонтерів, учасників, які надають свою безкоштовну роботу та ресурси для благодійних цілей. – Відділ комплектації відповідає за керування та забезпечення постачання ресурсів і матеріалів, необхідних для виконання проектів і програм. – Облік, бюджетування та нагляд за фінансами знаходяться в підрозділі фінансів. – Організація та реалізація соціальних і благодійних проектів лежить на відділі

		управління проектами. – Відділ зв’язків з громадськістю керує зв’язками з громадськістю, рекламою та публічними відносинами, а також заохочує залучення волонтерів.

ДОДАТОК Б

ВМІСТ ФАЙЛУ POM.XML

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.webapp</groupId>
<artifactId>diploma</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven-compiler-plugin.version>3.8.1</maven-compiler-plugin.version>
<maven-shade-plugin.version>3.4.1</maven-shade-plugin.version>
<maven-surefire-plugin.version>2.22.2</maven-surefire-plugin.version>
<exec-maven-plugin.version>3.0.0</exec-maven-plugin.version>
<vertx.version>4.3.8</vertx.version>
<junit-jupiter.version>5.9.1</junit-jupiter.version>
<main.verticle>com.webapp.Web</main.verticle>
<launcher.class>com.webapp.Main</launcher.class>
<awssdk.version>2.22.12</awssdk.version>
</properties>
<dependencyManagement>
<dependencies>
<dependency>
<groupId>io.vertx</groupId>
<artifactId>vertx-stack-depchain</artifactId>
<version>${vertx.version}</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
<dependencies>
<dependency>
<groupId>io.vertx</groupId>
<artifactId>vertx-web</artifactId>
</dependency>
<dependency>
<groupId>org.apache.commons</groupId>
<artifactId>commons-csv</artifactId>
<version>1.1</version>
</dependency>
<dependency>
<groupId>commons-io</groupId>
<artifactId>commons-io</artifactId>
<version>2.11.0</version>
</dependency>
<dependency>
```

<groupId>org.slf4j</groupId>
<artifactId>slf4j-api</artifactId>
<version>2.0.6</version>
</dependency>
<dependency>
<groupId>ch.qos.logback</groupId>
<artifactId>logback-classic</artifactId>
<version>1.4.5</version>
</dependency>
<dependency>
<groupId>com.google.inject</groupId>
<artifactId>guice</artifactId>
<version>5.1.0</version>
</dependency>
<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-core</artifactId>
<version>2.14.2</version>
</dependency>
<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.14.2</version>
</dependency>
<dependency>
<groupId>com.fasterxml.jackson.dataformat</groupId>
<artifactId>jackson-dataformat-yaml</artifactId>
<version>2.14.2</version>
</dependency>
<dependency>
<groupId>io.vertx</groupId>
<artifactId>vertx-junit5</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.junit.jupiter</groupId>
<artifactId>junit-jupiter-api</artifactId>
<version>${junit-jupiter.version}</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.junit.jupiter</groupId>
<artifactId>junit-jupiter-engine</artifactId>
<version>${junit-jupiter.version}</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.26</version>
<scope>compile</scope>

```
</dependency>
<dependency>
<groupId>com.univocity</groupId>
<artifactId>univocity-parsers</artifactId>
<version>2.9.1</version>
</dependency>
<dependency>
<groupId>io.vertx</groupId>
<artifactId>vertx-config</artifactId>
<version>${vertx.version}</version>
</dependency>
<dependency>
<groupId>io.micrometer</groupId>
<artifactId>micrometer-registry-prometheus</artifactId>
<version>1.9.0</version>
</dependency>
<dependency>
<groupId>com.github.loki4j</groupId>
<artifactId>loki-logback-appender</artifactId>
<version>1.4.0</version>
</dependency>
<dependency>
<groupId>io.vertx</groupId>
<artifactId>vertx-web-templ-freemarker</artifactId>
<version>4.4.2</version>
</dependency>
<dependency>
<groupId>org.apache.commons</groupId>
<artifactId>commons-lang3</artifactId>
<version>3.12.0</version>
</dependency>
<dependency>
<groupId>software.amazon.awssdk</groupId>
<artifactId>cognitoidentityprovider</artifactId>
<version>${awssdk.version}</version>
</dependency>
<dependency>
<groupId>com.auth0</groupId>
<artifactId>java-jwt</artifactId>
<version>4.4.0</version>
</dependency>
<dependency>
<groupId>com.auth0</groupId>
<artifactId>jwks-rsa</artifactId>
<version>0.22.1</version>
</dependency>
<dependency>
<groupId>org.postgresql</groupId>
<artifactId>postgresql</artifactId>
<version>42.6.0</version>
</dependency>
```

```
<dependency>
<groupId>io.vertx</groupId>
<artifactId>vertx-jdbc-client</artifactId>
<version>4.4.6</version>
</dependency>
<dependency>
<groupId>io.vertx</groupId>
<artifactId>vertx-sql-client</artifactId>
<version>4.4.6</version>
</dependency>
<dependency>
<groupId>org.apache.tomcat</groupId>
<artifactId>tomcat-dbcp</artifactId>
<version>9.0.50</version>
<scope>compile</scope>
</dependency>
<dependency>
<groupId>commons-beanutils</groupId>
<artifactId>commons-beanutils</artifactId>
<version>1.9.4</version>
<scope>compile</scope>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
<artifactId>maven-compiler-plugin</artifactId>
<version>${maven-compiler-plugin.version}</version>
<configuration>
<release>17</release>
</configuration>
</plugin>
<plugin>
<artifactId>maven-shade-plugin</artifactId>
<version>${maven-shade-plugin.version}</version>
<executions>
<execution>
<phase>package</phase>
<goals>
<goal>shade</goal>
</goals>
<configuration>
<transformers>
<transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer"
>
<manifestEntries>
<Main-Class>${launcher.class}</Main-Class>
<Main-Verticle>${main.verticle}</Main-Verticle>
</manifestEntries>
</transformer>
```

```
<transformer implementation="org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"/
>
</transformers>
<outputFile>${project.build.directory}/${project.artifactId}-${project.version}.war </outputFile>
<!--            <minimizeJar>true</minimizeJar> -->
</configuration>
</execution>
</executions>
</plugin>
<plugin>
<artifactId>maven-surefire-plugin</artifactId>
<version>${maven-surefire-plugin.version}</version>
</plugin>
<plugin>
<groupId>org.codehaus.mojo</groupId>
<artifactId>exec-maven-plugin</artifactId>
<version>1.4.0</version>
<executions>
<execution>
<id>deploy2repo</id>
<phase>deploy</phase>
<goals>
<goal>exec</goal>
</goals>
<configuration>
<executable>bash</executable>
<arguments>
<argument>deploy-repo.sh</argument>
<argument>${basedir}</argument>
<argument>${project.artifactId}-${project.version}.war</argument>
</arguments>
<workingDirectory>.</workingDirectory>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
<pluginManagement>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-deploy-plugin</artifactId>
<executions>
<execution>
<id>default-deploy</id>
<phase>none</phase>
</execution>
</executions>
</plugin>
</plugins>
</pluginManagement>
```

```
<resources>
<resource>
<directory>src/main/resources</directory>
</resource>
</resources>
</build>
</project>
```

ДОДАТОК В

ВМІСТ ФАЙЛУ CONFIG.JAVA

```
package com.webapp;

import com.auth0.jwt.JWT;
import com.auth0.jwt.JWTVerifier;
import com.auth0.jwt.algorithms.Algorithm;
import com.auth0.jwt.interfaces.RSAKeyProvider;
import com.google.inject.AbstractModule;
import com.google.inject.binder.AnnotatedBindingBuilder;
import com.google.inject.name.Names;
import com.webapp.service.AwsCognitoRSAKeyProvider;
import io.vertx.core.Vertx;
import io.vertx.core.json.JsonArray;
import io.vertx.core.json.JsonObject;
import io.vertx.jdbcclient.JDBCPool;
import lombok.SneakyThrows;
import org.apache.commons.beanutils.BeanUtilsBean;
import org.apache.commons.lang3.StringUtils;
import org.apache.tomcat.dbcp.dbcp2.BasicDataSource;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClientBuilder;

import java.net.URI;
import java.util.List;
import java.util.Map;

import java.util.stream.Collectors;

public class Config extends AbstractModule {
```

```
        private final Logger log = LoggerFactory.getLogger(this.getClass());
        private JsonObject config;

        public Config(JsonObject config) {
            this.config = config;
        }

        @Override
        protected <T> AnnotatedBindingBuilder<T> bind(Class<T> clazz) {
            return super.bind(clazz);
        }

        @SneakyThrows
        @Override
        protected void configure() {
            config.fieldNames().forEach((field) -> {
                Object o = config.getValue(field);
                if (o instanceof JsonArray) {

bind(List.class).annotatedWith(Names.named(field)).toInstance(config.getJsonArray(field).getList());
                } else {

bind(String.class).annotatedWith(Names.named(field)).toInstance(StringUtils.defaultIfBlank(config.getSt
ring(field), ""));
                    if (field.contains("pool.pg")) {
                        StaticVariables.dataSourceConfig.put(StringUtils.substringAfter(field, "pool.pg."),
StringUtils.defaultIfBlank(config.getString(field), ""));
                    }
                }
            });

            // Конфіг бази даних
            BasicDataSource dataSource = new BasicDataSource();
            dataSource.setDriverClassName("org.postgresql.Driver");
```

```
            log.info("pg pool config: " +
StaticVariables.dataSourceConfig.entrySet().stream().collect(Collectors.toMap(Map.Entry::getKey, e ->
e.getKey().equals("password") ? "***" : e.getValue())));
            BeanUtilsBean.getInstance().populate(dataSource, StaticVariables.dataSourceConfig);
            JDBCPool client = JDBCPool.pool(Vertx.vertx(), dataSource);
            bind(JDBCPool.class).toInstance(client);

            // Конфігурація для Когніто

            String region = config.getString("aws.region", "us-east-1");
            String profile = config.getString("aws.profile", "instance");
            CognitoIdentityProviderClientBuilder cipcb = CognitoIdentityProviderClient.builder()
                .region(Region.of(region));
            if (!StringUtils.equals("assume", profile) && !StringUtils.equals("instance", profile))
               cipcb.endpointOverride(new URI("http://127.0.0.1:4566"));
            CognitoIdentityProviderClient cognitoClient = cipcb.build();
            bind(CognitoIdentityProviderClient.class).toInstance(cognitoClient);

            // Конфігурація верифікації і валідації для JWT токена

            RSAKeyProvider keyProvider = new
AwsCognitoRSAKeyProvider(config.getString("aws.user.pool.id"),
                (!StringUtils.equals("assume", profile) && !StringUtils.equals("instance", profile))
                    ? "http://127.0.0.1:4566"
                    : String.format("https://cognito-idp.%s.amazonaws.com", region)
            );
            Algorithm algorithm = Algorithm.RSA256(keyProvider);
            JWTVerifier jwtVerifier = JWT.require(algorithm).build();
            bind(JWTVerifier.class).toInstance(jwtVerifier);

            requestStaticInjection(StaticVariables.class);
        }
    }
```

ДОДАТОК Г

ВМІСТ ФАЙЛУ AWSCOGNITOSERVICE.JAVA

```
package com.webapp.service;

import com.webapp.StaticVariables;
import io.vertx.core.json.JsonObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cognitoidentityprovider.model.*;

import java.util.*;

public class AWSCognitoService {
   private final Logger log = LoggerFactory.getLogger(this.getClass());

   public SignUpResponse signUp(String password, String email, String firstName, String lastName,
String phoneNumber) {
      AttributeType emailAttr = AttributeType.builder().name("email").value(email).build();
      AttributeType firstNameAttr =
AttributeType.builder().name("given_name").value(firstName).build();
      AttributeType lastNameAttr =
AttributeType.builder().name("family_name").value(lastName).build();
      AttributeType phoneAttr =
AttributeType.builder().name("phone_number").value(phoneNumber).build();

      List<AttributeType> userAttrsList = new ArrayList<>();
      userAttrsList.add(emailAttr);
      userAttrsList.add(firstNameAttr);
      userAttrsList.add(lastNameAttr);
      userAttrsList.add(phoneAttr);
      try {
         SignUpRequest signUpRequest = SignUpRequest.builder()
               .clientId(StaticVariables.awsUserPoolClientId)
               .username(email)
```

```
                .userAttributes(userAttrsList)
                .password(password)
                .build();

        return StaticVariables.cognitoIdentityProviderClient.signUp(signUpRequest);
    } catch (CognitoIdentityProviderException e) {
        throw new RuntimeException(e.awsErrorDetails().errorMessage());
    }
}

public ConfirmSignUpResponse confirmSignUp(String code, String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                .clientId(StaticVariables.awsUserPoolClientId)
                .confirmationCode(code)
                .username(userName)
                .build();

        ConfirmSignUpResponse result =
StaticVariables.cognitoIdentityProviderClient.confirmSignUp(signUpRequest);
        if (result.sdkHttpResponse().statusCode() == 200) {
            AdminAddUserToGroupResponse userToGroupResult = addUserToGroup(userName);
            if (userToGroupResult.sdkHttpResponse().statusCode() == 200) {
                return result;
            }
        }
    } catch (CognitoIdentityProviderException e) {
        throw new RuntimeException(e.awsErrorDetails().errorMessage());
    }
    throw new RuntimeException("Incorrect result");
}


public InitiateAuthResponse signUp(String userName, String password) {
```

```
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            InitiateAuthRequest authRequest = InitiateAuthRequest.builder()
                    .clientId(StaticVariables.awsUserPoolClientId)
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.USER_PASSWORD_AUTH)
                    .build();
            return StaticVariables.cognitoIdentityProviderClient.initiateAuth(authRequest);
        } catch (CognitoIdentityProviderException e) {
            log.error(e.getMessage());
        }
        return null;
    }

    public AdminAddUserToGroupResponse addUserToGroup(String username) {
        try {
            AdminAddUserToGroupRequest toGroupRequest = AdminAddUserToGroupRequest.builder()
                    .groupName("users")
                    .username(username)
                    .userPoolId(StaticVariables.awsUserPoolId)
                    .build();

            return StaticVariables.cognitoIdentityProviderClient.adminAddUserToGroup(toGroupRequest);
        } catch (CognitoIdentityProviderException e) {
            throw new RuntimeException(e.awsErrorDetails().errorMessage());
        }
    }

    public ChangePasswordResponse changePassword(String token, String previousPassword, String
newPassword) {
        try {
```

```
            ChangePasswordRequest request = ChangePasswordRequest.builder()
                    .accessToken(token)
                    .previousPassword(previousPassword)
                    .proposedPassword(newPassword)
                    .build();
            return StaticVariables.cognitoIdentityProviderClient.changePassword(request);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }
    }

    public String getUserUuid(String token) {
        String uuid = "";
        try {
            List<AttributeType> userAttr =
StaticVariables.cognitoIdentityProviderClient.getUser(GetUserRequest.builder().accessToken(token).buil
d()).userAttributes();
            for (AttributeType attr : userAttr) {
                String name = attr.name();
                if (name.equals("sub")) uuid = attr.value();
            }
        } catch (Exception e) {
            log.error(e.getMessage());
        }
        return uuid;
    }

    public JsonObject getUserData(String token) {
        JsonObject userData = new JsonObject();
        try {
            List<AttributeType> userAttr =
StaticVariables.cognitoIdentityProviderClient.getUser(GetUserRequest.builder().accessToken(token).buil
d()).userAttributes();
            for (AttributeType attr : userAttr) {
```

```
            if (attr.name().equals("sub")) {
            } else if (attr.name().equals("address")) {
                JsonObject address = new JsonObject(attr.value());
                Iterator<Map.Entry<String, Object>> iterator = address.stream().iterator();
                while (iterator.hasNext()) {
                    Map.Entry<String, Object> entry = iterator.next();
                    userData.put(entry.getKey(), entry.getValue());
                }
            } else {
                userData.put(attr.name(), attr.value());
            }
        }
    } catch (Exception e) {
        log.error(e.getMessage());
    }
    return userData;
}

public void updateUser(String token, String firstName, String middleName, String lastName, String
address, String phoneNumber) {
    AttributeType firstNameAttr =
AttributeType.builder().name("given_name").value(firstName).build();
    AttributeType middleNameAttr =
AttributeType.builder().name("middle_name").value(middleName).build();
    AttributeType lastNameAttr =
AttributeType.builder().name("family_name").value(lastName).build();
    AttributeType addressAttr = AttributeType.builder().name("address").value(address).build();
    AttributeType phoneAttr =
AttributeType.builder().name("phone_number").value(phoneNumber).build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(firstNameAttr);
    userAttrsList.add(lastNameAttr);
    userAttrsList.add(middleNameAttr);
```

```
        userAttrsList.add(addressAttr);
        userAttrsList.add(phoneAttr);

        try {
            UpdateUserAttributesRequest request = UpdateUserAttributesRequest.builder()
                    .accessToken(token)
                    .userAttributes(userAttrsList)
                    .build();
            StaticVariables.cognitoIdentityProviderClient.updateUserAttributes(request);
        } catch (Exception e) {
            log.error(e.getMessage());
        }
    }
}
```

ДОДАТОК Д
ВМІСТ ФАЙЛУ MAIN.JAVA

```
package com.webapp;

import java.io.File;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.dataformat.yaml.YAMLFactory;
import com.google.inject.CreationException;
import com.google.inject.Guice;
import com.google.inject.Injector;
import io.micrometer.core.instrument.Clock;
import io.micrometer.core.instrument.binder.system.FileDescriptorMetrics;
import io.micrometer.core.instrument.binder.system.ProcessorMetrics;
import io.micrometer.core.instrument.binder.system.UptimeMetrics;
import io.micrometer.prometheus.PrometheusConfig;
import io.micrometer.prometheus.PrometheusMeterRegistry;
import io.prometheus.client.CollectorRegistry;
import io.vertx.core.*;
import io.vertx.core.eventbus.EventBus;
import io.vertx.core.json.JsonObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.IOException;

public class Main {
    private final static Logger log = LoggerFactory.getLogger(Main.class);

    public static void main(String[] args) throws IOException {
        try {
            String file = StaticVariables.appHome + "/config/webapp.yml";
```

```
        ObjectMapper yamlReader = new ObjectMapper(new YAMLFactory());
        JsonNode options = yamlReader.readTree(new File(file));
        log.info("config is loaded: " + file);

        JsonObject config = new JsonObject(options.toString());
        VertxOptions vertxOptions = new VertxOptions(config);

        CollectorRegistry prometheusClientRegistry = new CollectorRegistry();
        PrometheusMeterRegistry registry = new
PrometheusMeterRegistry(PrometheusConfig.DEFAULT, prometheusClientRegistry, Clock.SYSTEM);
        new ProcessorMetrics().bindTo(registry);
        new UptimeMetrics().bindTo(registry);
        new FileDescriptorMetrics().bindTo(registry);

        Vertx vertx = Vertx.vertx(vertxOptions);

        Handler<AsyncResult<String>> resultHandler = deployedEvent -> {
           if (!deployedEvent.succeeded()) {
              log.error("Deployment error" + deployedEvent.cause().getMessage());
              vertx.undeploy(deployedEvent.result(), undeployEvent -> {
                 System.exit(-1);
              });
           } else {
              EventBus eb = vertx.eventBus();
              eb.consumer("io.exit", finishedMessage -> {
                 try {
                    Thread.sleep(3000);
                 } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                 }
                 if (finishedMessage.body() != null) log.error(finishedMessage.body().toString());
                 vertx.undeploy(deployedEvent.result(), undeployEvent -> {
                    System.exit(-1);
                 });
```

```
                });
            }
        };

        Injector injector = Guice.createInjector(new Config(config));
        vertx.deployVerticle(Web.class, new DeploymentOptions().setConfig(config), resultHandler);
      } catch (CreationException e) {
        log.error(e.getMessage());
        System.exit(-1);
      }
    }
}
```

ДОДАТОК Е
ВМІСТ ФАЙЛУ WEB.JAVA

```
package com.webapp;

import com.auth0.jwt.interfaces.DecodedJWT;
import com.webapp.service.AWSCognitoService;
import freemarker.template.Configuration;
import freemarker.template.Template;
import io.vertx.core.AbstractVerticle;
import io.vertx.core.Promise;
import io.vertx.core.json.JsonObject;
import io.vertx.ext.web.Router;
import io.vertx.ext.web.Session;
import io.vertx.ext.web.handler.BodyHandler;
import io.vertx.ext.web.handler.SessionHandler;
import io.vertx.ext.web.handler.StaticHandler;
import io.vertx.ext.web.sstore.LocalSessionStore;
import io.vertx.ext.web.sstore.SessionStore;
import io.vertx.sqlclient.Row;
import io.vertx.sqlclient.RowSet;
import io.vertx.sqlclient.Tuple;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cognitoidentityprovider.model.ChangePasswordResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.InitiateAuthResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpResponse;

import java.io.*;
import java.sql.Timestamp;
import java.util.*;
```

```
public class Web extends AbstractVerticle {
    private final Logger log = LoggerFactory.getLogger(this.getClass());
    private final AWSCognitoService cognitoService = new AWSCognitoService();

    @Override
    public void start(Promise<Void> startPromise) throws Exception {
        int port = 8882;
        try {
            port = Integer.valueOf(System.getProperty("server.port")).intValue();
        } catch (Exception e) {
        }

        if (port > 65535 || port <= 0) {
            throw new RuntimeException("Wrong tcp port used");
        }

        Router router = Router.router(vertx);
        Router routerPrivate = Router.router(vertx);

        // Freemarker Config
        Configuration configuration = new Configuration(Configuration.VERSION_2_3_31);
        configuration.setDirectoryForTemplateLoading(new
File("/home/natalia/IdeaProjects/diploma/src/main/resources/"));
        configuration.setDefaultEncoding("UTF-8");
        configuration.setSharedVariable("path", "/");

        // Session handler
        SessionStore store = LocalSessionStore.create(vertx, "sessionMap");
        SessionHandler sessionHandler = SessionHandler.create(store);

        router.route("/private/*")
                .handler(sessionHandler).handler((v) -> {
                    String token = v.session().get("auth");
                    if (token != null) {
```

```
                DecodedJWT jwt;
                try {
                    jwt = StaticVariables.jwtVerifier.verify(token);
                    if (jwt != null) {
                        v.next();
                    }
                } catch (Exception e) {
                    v.redirect("/");
                }
            } else {
                v.redirect("/");
            }
        })
        .subRouter(routerPrivate);


    router.route("/static/*").handler(StaticHandler.create("static"));
    // сторінка де є логін і реєстрація
    router.get("/index").handler(req -> {
        try {
            Map<String, Object> data = new HashMap<>();
            Template template = configuration.getTemplate("web/index.ftl");
            Writer writer = new StringWriter();
            template.process(data, writer);

            req.response().putHeader("content-type", "text/html")
                    .send(writer.toString());
        } catch (Exception e) {
            req.fail(500);
        }
    });
// цим контроллером обробляється запит на реєстрацію
    router.post("/sign-up").handler(BodyHandler.create().setHandleFileUploads(false)).handler(req -> {
        try {
```

```
            String password = req.request().getParam("password");
            String email = req.request().getParam("newemail");
            String firstName = req.request().getParam("firstname");
            String lastName = req.request().getParam("lastname");
            String phoneNumber = req.request().getParam("phone");
            phoneNumber = phoneNumber.replaceAll("[() -]*", "");


            SignUpResponse result = cognitoService.signUp(password, email, firstName, lastName,
phoneNumber);
            if (result.sdkHttpResponse().statusCode() == 200) {
               req.response()
                     .setStatusCode(201)
                     .putHeader("content-type", "application/json")
                     .send(JsonObject.of("message", "Created!").toString());
            } else {
               req.response().setStatusCode(result.sdkHttpResponse().statusCode())
                     .putHeader("content-type", "application/json")
                     .send(JsonObject.of("message", "Oops, something went wrong!").toString());
            }
         } catch (Exception e) {
            log.error(e.getMessage());
            req.response()
                  .setStatusCode(401)
                  .putHeader("content-type", "application/json")
                  .send(JsonObject.of("error", e.getMessage()).toString());
         }
      });
// цей контроллер обробляє підтвердження реєстрації
      router.post("/sign-up-
confirm").handler(BodyHandler.create().setHandleFileUploads(false)).handler(req -> {
         try {
            String email = req.request().getParam("hidemail");
            String code = req.request().getParam("code");
```

```
            ConfirmSignUpResponse result = cognitoService.confirmSignUp(code, email);
            if (result.sdkHttpResponse().statusCode() == 200) {
                req.response().setStatusCode(200)
                        .putHeader("content-type", "application/json")
                        .send(JsonObject.of("message", "User confirmed!").toString());
            } else {
                req.response().setStatusCode(result.sdkHttpResponse().statusCode())
                        .putHeader("content-type", "application/json")
                        .send(JsonObject.of("message", "Oops, something went wrong!").toString());
            }
        } catch (Exception e) {
            log.error(e.getMessage());
            req.response()
                    .setStatusCode(401)
                    .putHeader("content-type", "application/json")
                    .send(JsonObject.of("error", e.getMessage()).toString());
        }
    });
// цей контроллер відповідає за логін, якщо він успішний іде редірект на борду юзера (де всі його публікації) /board
    router.post("/sign-in").handler(sessionHandler).handler(BodyHandler.create().setHandleFileUploads(false)).handler(req -> {
        try {
            String email = req.request().getParam("email");
            String password = req.request().getParam("password");

            InitiateAuthResponse result = cognitoService.signIn(email, password);
            if (result != null && result.sdkHttpResponse().statusCode() == 200) {
                Session session = req.session();
                String token = result.authenticationResult().accessToken();
                session.put("auth", token);
                String refreshToken = result.authenticationResult().refreshToken();
```

```
            session.put("authRefresh", refreshToken);
            session.put("uuid", cognitoService.getUserUuid(token));
            session.put("email", cognitoService.getUserData(token).getString("email"));
        } else {
            throw new RuntimeException("Something went wrong");
        }
        req.response()
                .setStatusCode(200)
                .putHeader("content-type", "application/json")
                .send(JsonObject.of("redirect", "private/board").toString());
    } catch (Exception e) {
        log.error(e.getMessage(), e);
        req.response()
                .setStatusCode(401)
                .putHeader("content-type", "application/json")
                .send(JsonObject.of("error", e.getMessage()).toString());
    }
});

// сторінка зі всіма публікаціями юзера якщо такі є
routerPrivate.get("/board").handler(sessionHandler).handler(req -> {
    String uuid = req.session().get("uuid");
    List<JsonObject> statements = new ArrayList<>();
    try {
        RowSet<Row> rs = StaticVariables.dbp.preparedQuery("SELECT * from statement where
owner_id = ?").execute(Tuple.of(uuid)).toCompletionStage().toCompletableFuture().get();
        for (Row row : rs) {
            statements.add(row.toJson());
        }
    } catch (Exception e) {
        log.error("Can not get statements");
    }
    try {
        Map<String, Object> data = new HashMap<>();
```

```
            data.put("statements", statements);
            Template template = configuration.getTemplate("web/board.ftl");
            Writer writer = new StringWriter();
            template.process(data, writer);

            req.response().putHeader("content-type", "text/html")
                    .send(writer.toString());
        } catch (Exception e) {
            req.fail(500);
        }
    });

    // сторінка конкретної публікації
    routerPrivate.get("/statement").handler(sessionHandler).handler(req -> {
        String statement_id = req.session().get("statement_id");
        JsonObject statement = new JsonObject();
        try {
            RowSet<Row> rs = StaticVariables.dbp.preparedQuery("SELECT * from statement where id =
?").execute(Tuple.of(statement_id)).toCompletionStage().toCompletableFuture().get();
            for (Row row : rs) {
                statement = row.toJson();
            }
        } catch (Exception e) {
            log.error("Can not get statements");
        }
        try {
            Map<String, Object> data = new HashMap<>();
            data.put("statement", statement.toString());
            Template template = configuration.getTemplate("web/statement.ftl");
            Writer writer = new StringWriter();
            template.process(data, writer);

            req.response().putHeader("content-type", "text/html")
                    .send(writer.toString());
```

```
        } catch (Exception e) {
            req.fail(500);
        }
    });

    // запит на створення поста
    routerPrivate.post("/statement-
create").handler(sessionHandler).handler(BodyHandler.create().setHandleFileUploads(false)).handler(req
-> {
        String uuid = req.session().get("uuid");
        String title = req.request().getParam("title");
        String description = req.request().getParam("description");
        String category = req.request().getParam("category");
        String region = req.request().getParam("region");
        try {
            Map<String, Object> obj = new HashMap<>();
            obj.put("title", title);
            obj.put("description", description);
            obj.put("category", Integer.parseInt(category));
            obj.put("region", Integer.parseInt(region));
            obj.put("owner_uuid", uuid);
            String sql = "INSERT INTO statement (" + StringUtils.join(obj.keySet(), ",") + ") VALUES ("
+ StringUtils.join(obj.keySet().stream().map(v -> "?").toList(), ",") + ")";
            StaticVariables.dbp.preparedQuery(sql).execute(Tuple.from(obj.values().toArray()))
                    .onFailure(f -> log.error("Error during inserting statement"));
            req.response()
                    .setStatusCode(200)
                    .putHeader("content-type", "application/json")
                    .send(JsonObject.of("message", "Created!", "redirect", "private/board").toString());
        } catch (Exception e) {
            log.error(e.getMessage(), e);
            req.response()
                    .setStatusCode(401)
                    .putHeader("content-type", "application/json")
```

```
                .send(JsonObject.of("error", e.getMessage()).toString());
        }
    });

    // запит на редагування поста
    routerPrivate.post("/statement-
update").handler(sessionHandler).handler(BodyHandler.create().setHandleFileUploads(false)).handler(re
q -> {
        String statement_id = req.request().getParam("statement_id");
        String title = req.request().getParam("title");
        String description = req.request().getParam("description");
        String category = req.request().getParam("category");
        String region = req.request().getParam("region");
        try {
            StaticVariables.dbp.preparedQuery("UPDATE statement SET title = ?, description = ?,
category = ?, region = ?, du = ? WHERE owner_id = ?")
                .execute(Tuple.of(title, description, Integer.parseInt(category), Integer.parseInt(region),
new Timestamp(System.currentTimeMillis()), statement_id))
                .onFailure(f -> log.error("Error during updating statement"));
            req.response()
                .setStatusCode(200)
                .putHeader("content-type", "application/json")
                .send(JsonObject.of("message", "Updated!", "redirect", "private/board").toString());
        } catch (Exception e) {
            log.error(e.getMessage(), e);
            req.response()
                .setStatusCode(401)
                .putHeader("content-type", "application/json")
                .send(JsonObject.of("error", e.getMessage()).toString());
        }
    });

    // запит на видалення поста
    routerPrivate.delete("/statement-delete").handler(req -> {
```

```
            String statement_id = req.request().getParam("id");
            StaticVariables.dbp.preparedQuery("DELETE FROM statement WHERE id =
?").execute(Tuple.of(statement_id)).onFailure(f -> log.error("Error during removing statement"));
            req.response().setStatusCode(200).end();
        });

        // запит на створення комента
        routerPrivate.post("/comment-
create").handler(sessionHandler).handler(BodyHandler.create().setHandleFileUploads(false)).handler(req
-> {
            String uuid = req.session().get("uuid");
            String statement_id = req.request().getParam("statement_id");
            String comment = req.request().getParam("comment");
            try {
                Map<String, Object> obj = new HashMap<>();
                obj.put("comment", comment);
                obj.put("statement_id", Integer.parseInt(statement_id));
                obj.put("user_uuid", uuid);
                String sql = "INSERT INTO comment (" + StringUtils.join(obj.keySet(), ",") + ") VALUES ("
+ StringUtils.join(obj.keySet().stream().map(v -> "?").toList(), ",") + ")";
                StaticVariables.dbp.preparedQuery(sql).execute(Tuple.from(obj.values().toArray()))
                        .onFailure(f -> log.error("Error during inserting comment"));
                req.response()
                        .setStatusCode(200)
                        .putHeader("content-type", "application/json")
                        .send(JsonObject.of("message", "Added!", "redirect", "private/statement").toString());
            } catch (Exception e) {
                log.error(e.getMessage(), e);
                req.response()
                        .setStatusCode(401)
                        .putHeader("content-type", "application/json")
                        .send(JsonObject.of("error", e.getMessage()).toString());
            }
        });
```

```
        // запит на редагування комента
        routerPrivate.post("/comment-
update").handler(sessionHandler).handler(BodyHandler.create().setHandleFileUploads(false)).handler(re
q -> {
            String comment_id = req.request().getParam("comment_id");
            String comment = req.request().getParam("comment");
            try {
                StaticVariables.dbp.preparedQuery("UPDATE comment SET comment = ? WHERE id = ?")
                        .execute(Tuple.of(comment, comment_id))
                        .onFailure(f -> log.error("Error during updating comment"));
                req.response()
                        .setStatusCode(200)
                        .putHeader("content-type", "application/json")
                        .send(JsonObject.of("message", "Updated!", "redirect", "private/statement").toString());
            } catch (Exception e) {
                log.error(e.getMessage(), e);
                req.response()
                        .setStatusCode(401)
                        .putHeader("content-type", "application/json")
                        .send(JsonObject.of("error", e.getMessage()).toString());
            }
        });

        // запит для отримання коментарів для конткретного поста
        routerPrivate.get("/comment-get").handler(sessionHandler).handler(req -> {
            String statement_id = req.session().get("statement_id");
            try {
                JsonObject statement = new JsonObject();
                try {
                    RowSet<Row> rs = StaticVariables.dbp.preparedQuery("SELECT * from comment where
statement_id = ?").execute(Tuple.of(statement_id)).toCompletionStage().toCompletableFuture().get();
                    for (Row row : rs) {
                        statement = row.toJson();
```

```
            }
        } catch (Exception e) {
            log.error("Can not get comments");
        }
        req.response()
                .setStatusCode(200)
                .putHeader("content-type", "application/json")
                .send(JsonObject.of("result", statement.toString()).toString());
    } catch (Exception e) {
        log.error(e.getMessage(), e);
        req.response()
                .setStatusCode(401)
                .putHeader("content-type", "application/json")
                .send(JsonObject.of("error", e.getMessage()).toString());
    }
});

// запит на видалення комента
routerPrivate.delete("/comment-delete").handler(req -> {
    String comment_id = req.request().getParam("comment_id");
    StaticVariables.dbp.preparedQuery("DELETE FROM comment WHERE id =
?").execute(Tuple.of(comment_id)).onFailure(f -> log.error("Error during removing comment"));
    req.response().setStatusCode(200).end();
});

// запит на додавання лайку
routerPrivate.post("/like-
add").handler(sessionHandler).handler(BodyHandler.create().setHandleFileUploads(false)).handler(req ->
{
    String uuid = req.session().get("uuid");
    String statement_id = req.request().getParam("statement_id");
    try {
        Map<String, Object> obj = new HashMap<>();
        obj.put("statement_id", Integer.parseInt(statement_id));
```

```
            obj.put("user_uuid", uuid);
            String sql = "INSERT INTO selected (" + StringUtils.join(obj.keySet(), ",") + ") VALUES (" +
StringUtils.join(obj.keySet().stream().map(v -> "?").toList(), ",") + ")";
            StaticVariables.dbp.preparedQuery(sql).execute(Tuple.from(obj.values().toArray()))
                    .onFailure(f -> log.error("Error during inserting selected"));
            req.response().setStatusCode(200);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
            req.response()
                    .setStatusCode(401)
                    .putHeader("content-type", "application/json")
                    .send(JsonObject.of("error", e.getMessage()).toString());
        }
    });

    // запит на видалення лайку
    routerPrivate.delete("/like-remove").handler(req -> {
        String uuid = req.session().get("uuid");
        String statement_id = req.request().getParam("statement_id");
        StaticVariables.dbp.preparedQuery("DELETE FROM selected WHERE statement_id = ? and
user_uuid = ?").execute(Tuple.of(statement_id, uuid)).onFailure(f -> log.error("Error during removing
comment"));
        req.response().setStatusCode(200).end();
    });

    // сторінка "поруч з вами"
    routerPrivate.get("/nearby").handler(sessionHandler).handler(req -> {
        String token = req.session().get("auth");
        // дістаємо дані юзера з когніто
        JsonObject userData = cognitoService.getUserData(token);
        // витягуємо регіон
        String userRegion = userData.getString("region");
        //робимо запит до бд
        List<JsonObject> statements = new ArrayList<>();
```

```
        try {
            RowSet<Row> rs = StaticVariables.dbp.preparedQuery("select * from statement inner join public.region r on r.id = statement.region_id where r.name = ?").execute(Tuple.of(userRegion)).toCompletionStage().toCompletableFuture().get();
            for (Row row : rs) {
                // отримані результати записуємо в колекцію яку вертаємо на фронтенд
                statements.add(row.toJson());
            }
        } catch (Exception e) {
            log.error("Can not get statements");
        }
        try {
            Map<String, Object> data = new HashMap<>();
            data.put("statements", statements);
            Template template = configuration.getTemplate("web/nearby.ftl");
            Writer writer = new StringWriter();
            template.process(data, writer);

            req.response().putHeader("content-type", "text/html")
                    .send(writer.toString());
        } catch (Exception e) {
            req.fail(500);
        }
    });

    // сторінка "грошова допомога"
    routerPrivate.get("/money-help").handler(sessionHandler).handler(req -> {
        // категорія грошової допомоги в нас має id 1. нема сенсу передавати її з фронта, оскільки категорії в нас статичні (не міняються), тому тут я просто захардкоджу кверю
        //робимо запит до бд
        List<JsonObject> statements = new ArrayList<>();
        try {
            RowSet<Row> rs = StaticVariables.dbp.query("select * from statement where region_id = 1").execute().toCompletionStage().toCompletableFuture().get();
```

```
            for (Row row : rs) {
                // отримані результати записуємо в колекцію яку вертаємо на фронтенд
                statements.add(row.toJson());
            }
        } catch (Exception e) {
            log.error("Can not get statements");
        }
        try {
            Map<String, Object> data = new HashMap<>();
            data.put("statements", statements);
            Template template = configuration.getTemplate("web/money-help.ftl");
            Writer writer = new StringWriter();
            template.process(data, writer);

            req.response().putHeader("content-type", "text/html")
                    .send(writer.toString());
        } catch (Exception e) {
            req.fail(500);
        }
    });

    // сторінка "обране"
    routerPrivate.get("/selected").handler(sessionHandler).handler(req -> {
        String uuid = req.session().get("uuid");
        //робимо запит до бд (шукаємо всі лайки конкретного юзера)
        List<Integer> statementsIds = new ArrayList<>();
        try {
            RowSet<Row> rs = StaticVariables.dbp.preparedQuery("select statement_id from selected
where user_uuid = ?").execute(Tuple.of(uuid)).toCompletionStage().toCompletableFuture().get();
            for (Row row : rs) {
                // отримані результати записуємо в колекцію
                statementsIds.add(row.getInteger(0));
            }
        } catch (Exception e) {
```

```
            log.error("Can not get statements ids");
        }
        // шукаємо всі пости які юзер лайкнув по id які ми знайшли у запиті раніше і які записані в колекцію statementsIds і записуємо їх в колекцію (то те саме шо масив)
        List<JsonObject> statements = new ArrayList<>();
        for (Integer id : statementsIds) {
            try {
                RowSet<Row> rs = StaticVariables.dbp.preparedQuery("select * from statement where id = ?").execute(Tuple.of(id)).toCompletionStage().toCompletableFuture().get();
                for (Row row : rs) {
                    // отримані результати записуємо в колекцію statements
                    statements.add(row.toJson());
                }
            } catch (Exception e) {
                log.error("Can not get statements");
            }
        }
        try {
            Map<String, Object> data = new HashMap<>();
            data.put("statements", statements);
            Template template = configuration.getTemplate("web/selected.ftl");
            Writer writer = new StringWriter();
            template.process(data, writer);

            req.response().putHeader("content-type", "text/html")
                    .send(writer.toString());
        } catch (Exception e) {
            req.fail(500);
        }
    });

    // сторінка юзера де юзер може переглянути свої дані
    routerPrivate.get("/personal-info").handler(sessionHandler).handler(req -> {
        String token = req.session().get("auth");
```

```
        try {
            Map<String, Object> data = new HashMap<>();
            data.put("user", cognitoService.getUserData(token).toString());
            Template template = configuration.getTemplate("web/personal-info.ftl");
            Writer writer = new StringWriter();
            template.process(data, writer);

            req.response().putHeader("content-type", "text/html")
                    .send(writer.toString());
        } catch (Exception e) {
            req.fail(500);
        }
    });
    // запит який відповідає за оновлення даних для юзера
    routerPrivate.post("/update-personal-
info").handler(sessionHandler).handler(BodyHandler.create().setHandleFileUploads(false)).handler(req -
> {
        String token = req.session().get("auth");
        try {
            String givenName = req.request().getParam("given_name");
            String middleName = req.request().getParam("middle_name");
            String familyName = req.request().getParam("family_name");
            String address1 = req.request().getParam("address1");
            String address2 = req.request().getParam("address2");
            String city = req.request().getParam("city");
            String region = req.request().getParam("region");
            String zip = req.request().getParam("zip");
            String phoneNumber = req.request().getParam("phone_number");
            phoneNumber = phoneNumber.replaceAll("[() -]*", "");

            JsonObject address = new JsonObject();
            address.put("address1", address1);
            address.put("address2", address2);
            address.put("city", city);
```

```
            address.put("region", region);
            address.put("zip", zip);

            cognitoService.updateUser(token, givenName, middleName, familyName, address.toString(),
phoneNumber);
            req.response()
                    .setStatusCode(200)
                    .putHeader("content-type", "application/json")
                    .send(JsonObject.of("message", "Updated!").toString());

        } catch (Exception e) {
            log.error(e.getMessage(), e);
            req.response()
                    .setStatusCode(401)
                    .putHeader("content-type", "application/json")
                    .send(JsonObject.of("error", e.getMessage()).toString());
        }
    });
    // запит який відповідає за зміну пароля юзера
    routerPrivate.post("/change-
password").handler(sessionHandler).handler(BodyHandler.create().setHandleFileUploads(false)).handler(
req -> {
        String token = req.session().get("auth");
        try {
            String old_password = req.request().getParam("old_password");
            String new_password = req.request().getParam("new_password");
            ChangePasswordResponse response = cognitoService.changePassword(token, old_password,
new_password);
            if (response.sdkHttpResponse().statusCode() == 200) {
                req.response()
                        .setStatusCode(200)
                        .putHeader("content-type", "application/json")
                        .send(JsonObject.of("message", "Updated!").toString());
            }
```

```
            } catch (Exception e) {
                log.error(e.getMessage(), e);
                req.response()
                        .setStatusCode(401)
                        .putHeader("content-type", "application/json")
                        .send(JsonObject.of("error", e.getMessage()).toString());
            }
        });

        int finalPort = port;
        vertx.createHttpServer().requestHandler(router).listen(port, http -> {
            if (http.succeeded()) {
                startPromise.complete();
                log.info("server started on port " + finalPort);
            } else {
                startPromise.fail(http.cause());
            }
        });
    }
}
```

ДОДАТОК Є

АПРОБАЦІЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Секція: Інформаційні технології

Козак А.А.,

здобувач першого (бакалаврського) рівня вищої освіти

Західноукраїнського національного університету

керівник: ***Комар М.П.,***

кандидат технічних наук, професор кафедри інформаційно-обчислювальних систем і управління Західноукраїнського національного університету

Алгоритм роботи веб-орієнтованої інформаційної системи надання волонтерської допомоги в період війни

Волонтерська допомога набуває особливої ваги в сучасних умовах, коли світова спільнота стикається з безпрецедентними викликами та кризами. Враховуючи поточну ситуацію в нашій країні, люди повинні максимально допомагати один одному та оточуючим. Тисячі волонтерів активно залучаються до збору грошей на закупівлю зброї, транспортних засобів, обладнання для військових, автомобілів і допомоги тим, чиї домівки були втрачені через атаки, а також інших предметів, необхідних для захисту України.

Алгоритм роботи системи:

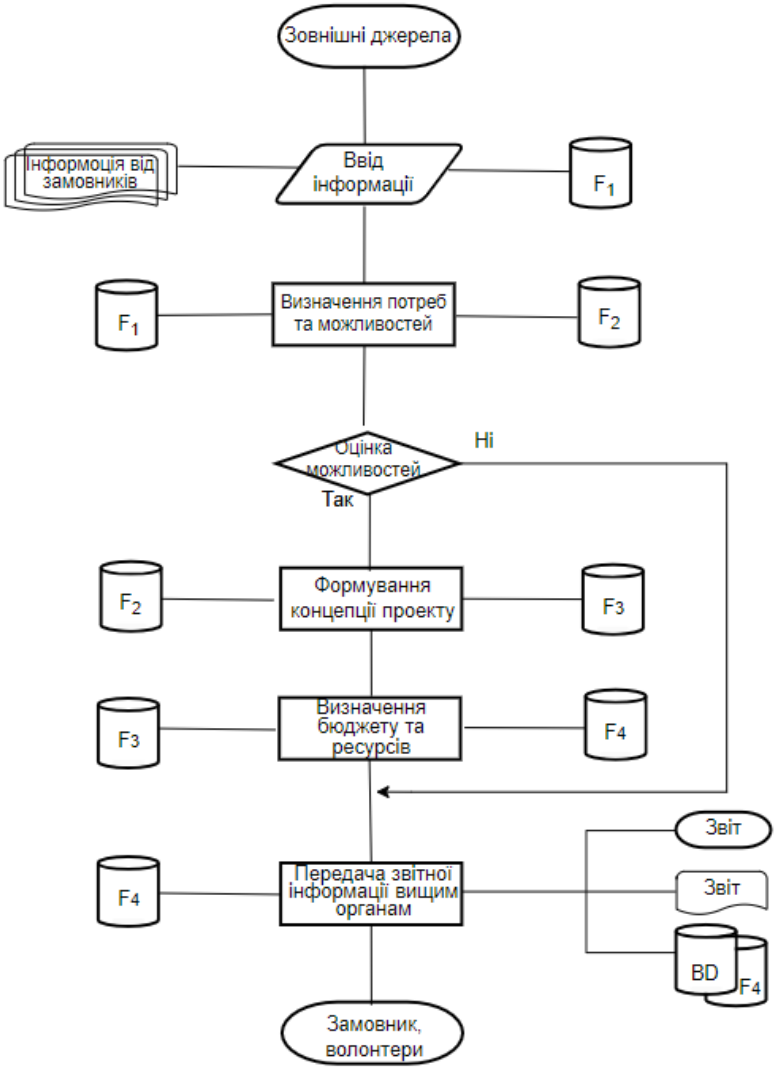

Рисунок 1 – Схема алгоритму процесу формування початкової інформації про проект

Як показано на рисунку 1, загальний алгоритм процесу формування початкової інформації про проект такий:

- Джерелом інформації є попередні дані та дані проекту.
- Наступним кроком є визначення потреб і можливостей.
- Визначення можливостей. Волонтери можуть взяти участь у проекті, якщо можливості задовольняють потреби.
- Створення концепції проекту, розробка першої версії та узгодження.
- Визначення бюджету та ресурсів, необхідних для проекту.
- Завершення звітів, які надсилаються волонтерам і замовнику, є останнім етапом.

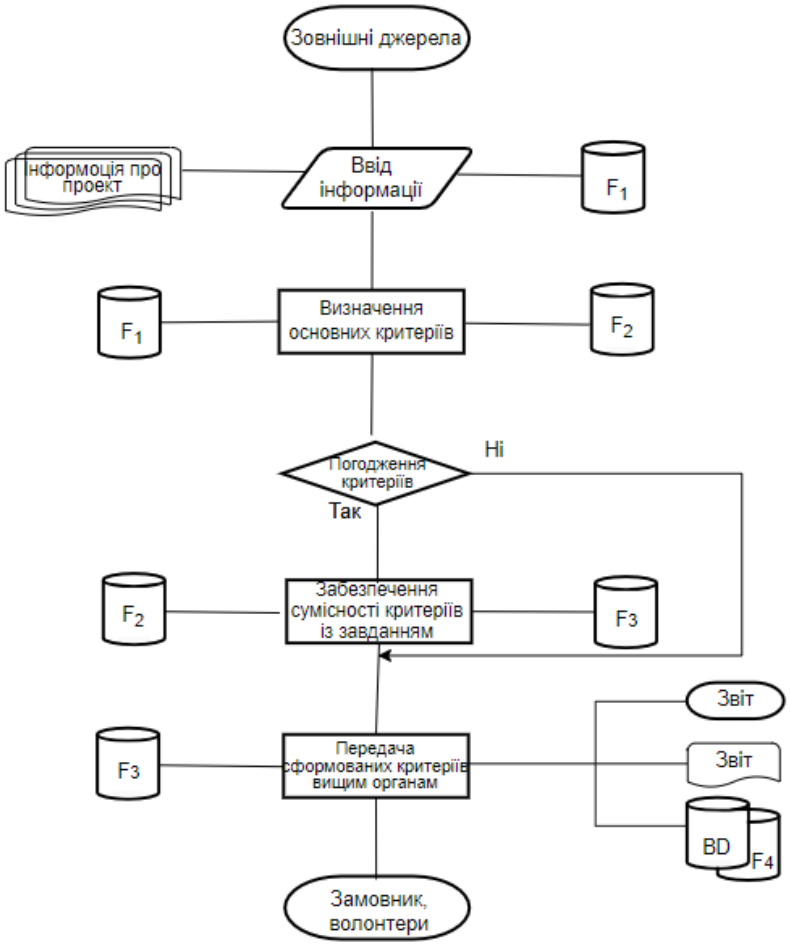

Рисунок 2 – Схема алгоритму процесу формування критеріїв проекту

Загальний алгоритм процесу формування критеріїв проекту має таку логіку, як показано на рисунку 2:

- Дані отримані з документів вводу.
- Визначення основних критеріїв, які впливають на процес створення проекту.
- Внесення змін до критеріїв і обговорення їх із замовниками
- Перевірка, що критерії відповідають завданню.
- Звіт як у паперовому, так і в електронному вигляді, а також внесення зміни в існуючий файл коштів і базу даних.
- Надсилання звітів замовникам і волонтерам.

Список літератури:

27. Здоров’я і суспільство в умовах війни // Збірник наукових статей / Центральноукраїнський інститут розвитку людини Університету «Україна». Кропивницький, 2022. С. 149-151. [Електронний ресурс] – Режим доступу до ресурсу: https://cnej.gov.md/sites/default/files/zbirnik_statey-2022._1.pdf#page=150 (дата звернення: 12.02.2024).
28. Формування соціальної згуртованості територіальної громади шляхом розвитку волонтерської діяльності: Львів, 2023. [Електронний ресурс] – Режим доступу до ресурсу: https://er.ucu.edu.ua/bitstream/handle/1/3714/Tsebenko_mag.pdf?sequence=1 (дата звернення: 12.02.2024).