

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ
ГРОМАДСЬКОЇ ОРГАНІЗАЦІЇ
SOFTWARE FOR SUPPORTING THE ACTIVITIES OF A NON-
GOVERNMENTAL ORGANIZATION

Виконав(ла): студент(ка) 4 курсу, групи КНТ-151

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

СІВЕЦЬ Н.Р.

(ПРІЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ О.М.

(ПРІЗВИЩЕ та ініціали)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст.викладач		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Нікіта СІВЕЦЬ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Олександр СТЕПАНЕНКО
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 111 с., 34 табл., 50 рис., 3 дод., 30 джерел.

ГРОМАДСЬКА ОРГАНІЗАЦІЯ, ВОЛОНТЕРСЬКА ДІЯЛЬНІСТЬ, ВЕБСАЙТ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, REACT, NODE.JS, POSTGRESQL.

Об'єкт - процес розроблення програмного забезпечення для підтримки комунікаційної діяльності громадських організацій.

Предмет - методи та програмні засоби створення вебзастосунків для забезпечення ефективної онлайн-присутності громадських організацій.

Мета роботи - підвищення ефективності інформаційної взаємодії громадської організації з цільовою аудиторією шляхом розробки спеціалізованого вебсайту.

Матеріали, методи та технічні засоби: реляційна база даних PostgreSQL, клієнт-серверна архітектура, мікросервісний підхід, мови програмування JavaScript і TypeScript, фреймворки React і Node.js, бібліотеки Redux і Material-UI.

Результати. Створено програмний вебзастосунок для підтримки діяльності громадської організації, який забезпечує реєстрацію та управління профілями волонтерів, створення та управління проєктами, координацію волонтерських груп.

Висновки. Розроблено програмне забезпечення для підтримки діяльності громадської організації, яке відповідає всім визначеним функціональним вимогам, є зручним у використанні, безпечним та масштабованим.

Галузь використання - підтримка діяльності громадських та волонтерських організацій в Україні.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 111 pages, 34 tables, 50 figures, 3 appendixes, 30 sources.

NON-GOVERNMENTAL ORGANIZATION, VOLUNTEER ACTIVITY, WEBSITE, SOFTWARE, REACT, NODE.JS, POSTGRESQL.

Object of study is the process of developing software to support the communication activities of non-governmental organizations.

Subject of study are methods and software tools for creating web applications to ensure effective online presence of non-governmental organizations.

The purpose of the work is to increase the efficiency of information interaction between a non-governmental organization and its target audience by developing a specialized website.

Materials, methods, and tools: PostgreSQL relational database, client-server architecture, microservice approach, JavaScript and TypeScript programming languages, React and Node.js frameworks, Redux and Material-UI libraries, Redis caching system, computer with Intel Core i7 processor running Windows 10 operating system.

Results. A web application has been created to support the activities of a non-governmental organization, which provides registration and management of volunteer profiles, creation and management of projects, coordination of volunteer groups, collection and management of donations, communication between participants, reporting and data analytics.

Conclusions. Software has been developed to support the activities of a non-governmental organization, which meets all the defined functional requirements, is user-friendly, secure, and scalable.

Field of use is supporting the activities of non-governmental and volunteer organizations in Ukraine.

ЗМІСТ

	С.
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Аналіз застосунків для підтримки діяльності громадських організацій.....	10
1.2 Порівняльний аналіз розробки та аналогів існуючих вебрішень	15
1.3 Висновки за розділом 1	23
2 Методи та моделі програмного забезпечення	24
2.1 Аналіз методів розробки	24
2.2 Моделі роботи програмного забезпечення.....	32
2.3 Висновки за розділом 2	41
3 Опис програмного забезпечення	43
3.1 Структура бази даних	43
3.2 Розробка дизайну вебсайту	60
3.3 Визначення порядку роботи програми	65
3.4 Висновки за розділом 3	68
4 Експлуатація та тестування програмного забезпечення	71
4.1 Користувацький інтерфейс програмного забезпечення.....	71
4.2 Оптимізація програмного забезпечення	77
4.3 Висновки за розділом 4	80
Висновки	82
Перелік джерел посилання	84
Додаток А Технічне завдання	87
Додаток Б Фрагмент тексту програми	92
Додаток В Слайди презентації.....	104

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
CMS	– Content Management System;
CSS	– Cascading Style Sheets;
GDPR	– General Data Protection Regulation;
HTML	– HyperText Markup Language;
HTTP	– HyperText Transfer Protocol;
HTTPS	– HyperText Transfer Protocol Secure;
JSON	– JavaScript Object Notation;
JSX	– JavaScript XML;
JWT	– JSON Web Tokens;
MVP	– Minimum Viable Product;
NPM	– Node Package Manager;
ORM	– Object-Relational Mapping;
REST	– Representational State Transfer;
SEO	– Search Engine Optimization;
SPA	– Single Page Application;
SQL	– Structured Query Language;
SSL	– Secure Sockets Layer;
UI	– User Interface.

ВСТУП

У сучасному світі громадські організації відіграють важливу роль у розвитку суспільства, вирішенні соціальних проблем та підтримці різноманітних ініціатив [1]. Ефективна діяльність таких організацій значною мірою залежить від налагодженої комунікації з громадськістю, потенційними донорами та членами організації. З розвитком інформаційних технологій вебсайти стали невід'ємним інструментом для забезпечення видимості, прозорості та ефективності роботи громадських організацій [2]. Вони дозволяють оперативно поширювати інформацію про діяльність, залучати прихильників та ресурси, координувати роботу волонтерів та учасників.

Незважаючи на значний прогрес у розробці вебтехнологій, багато громадських організацій в Україні все ще стикаються з проблемами у створенні та підтримці власних вебресурсів, які б повністю відповідали їхнім специфічним потребам [3]. Існуючі типові рішення не завжди враховують особливості діяльності конкретних організацій, що призводить до зниження ефективності їхньої онлайн-присутності. Розробка спеціалізованого програмного забезпечення для підтримки діяльності громадських організацій є актуальним завданням, що допоможе підвищити їхню ефективність та вплив у суспільстві [4].

Об'єктом дослідження є процес розроблення програмного забезпечення для підтримки комунікаційної діяльності громадських організацій.

Предметом дослідження є методи та програмні засоби створення вебзастосунків для забезпечення ефективної онлайн-присутності громадських організацій.

Мета роботи полягає у підвищенні ефективності інформаційної взаємодії громадської організації з цільовою аудиторією шляхом розробки спеціалізованого вебсайту.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- аналіз існуючих рішень та визначення ключових функціональних вимог до вебсайту громадської організації;
- проєктування архітектури вебзастосунку з урахуванням специфіки діяльності громадських організацій;
- розробка візуального інтерфейсу, що забезпечить зручну взаємодію користувачів із системою;
- реалізація функціональних модулів відповідно до визначених вимог;
- тестування розробленого програмного забезпечення.

Практичне значення роботи полягає у створенні програмного продукту, який може бути використаний громадськими організаціями для покращення їхньої діяльності та комунікації з цільовою аудиторією. Розроблений вебсайт дозволить організаціям ефективніше інформувати суспільство про свою роботу, залучати нових учасників та підтримку, а також координувати свою діяльність [1]. Крім того, запропоновані підходи та рішення можуть бути адаптовані для інших організацій зі схожими потребами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз застосунків для підтримки діяльності громадських організацій

Розглянемо існуючі рішення у сфері програмного забезпечення для підтримки діяльності громадських організацій, зокрема тих, що спеціалізуються на волонтерській діяльності в Україні. Аналіз існуючих застосунків дозволить визначити ключові функціональні можливості, які необхідно реалізувати у власній розробці, а також виявити недоліки наявних рішень, які можна усунути.

Сучасний ринок програмного забезпечення пропонує різноманітні рішення для громадських організацій, які можна умовно поділити на декілька категорій:

- загальні системи управління контентом (CMS), що можуть бути адаптовані для потреб громадських організацій;
- спеціалізовані платформи для некомерційних організацій;
- застосунки для координації волонтерів;
- рішення для збору пожертв та управління ресурсами [1].

Для детального аналізу було обрано шість популярних застосунків, які часто використовуються волонтерськими громадськими організаціями в Україні: «Волонтер», «ДоброДій», «Спільнокошт», NGO Forum, «Мрійдій» та «ВолонтерUA». Порівняння функціональних можливостей цих застосунків представлено у таблиці 1.1.

Як видно з таблиці 1.1, жоден із розглянутих застосунків не надає повного набору функцій, необхідних для ефективної роботи волонтерської громадської організації в Україні. Розглянемо детальніше кожне з рішень.

Таблиця 1.1 - Порівняння функціональних можливостей застосунків для волонтерських громадських організацій

Функціональність	Волонтер	ДоброДій	Спільнокошт	NGO Forum	Мрійдій	ВолонтерUA
Реєстрація волонтерів	Так	Так	Ні	Так	Так	Так
Управління проєктами	Так	Обмежено	Так	Так	Обмежено	Так
Збір пожертв	Так	Так	Так	Ні	Так	Обмежено
Координація волонтерів	Так	Так	Ні	Так	Обмежено	Так
Звітність про діяльність	Обмежено	Обмежено	Так	Так	Ні	Обмежено
Комунікація між учасниками	Так	Так	Обмежено	Так	Так	Так
Інтеграція з соцмережами	Так	Так	Так	Так	Так	Так
Картографування потреб	Так	Ні	Ні	Обмежено	Ні	Так
Мобільний доступ	Так	Так	Так	Обмежено	Так	Так
Аналітика даних	Обмежено	Ні	Так	Так	Ні	Обмежено

«Волонтер» є українською платформою, розробленою спеціально для координації волонтерської діяльності [2]. Застосунок дозволяє реєструвати волонтерів, створювати проєкти, координувати роботу команд та збирати пожертви. Однак, система має обмежені можливості для аналітики та звітності. Архітектура застосунку «Волонтер» базується на мікросервісній моделі, що зображено на рисунку 1.1.

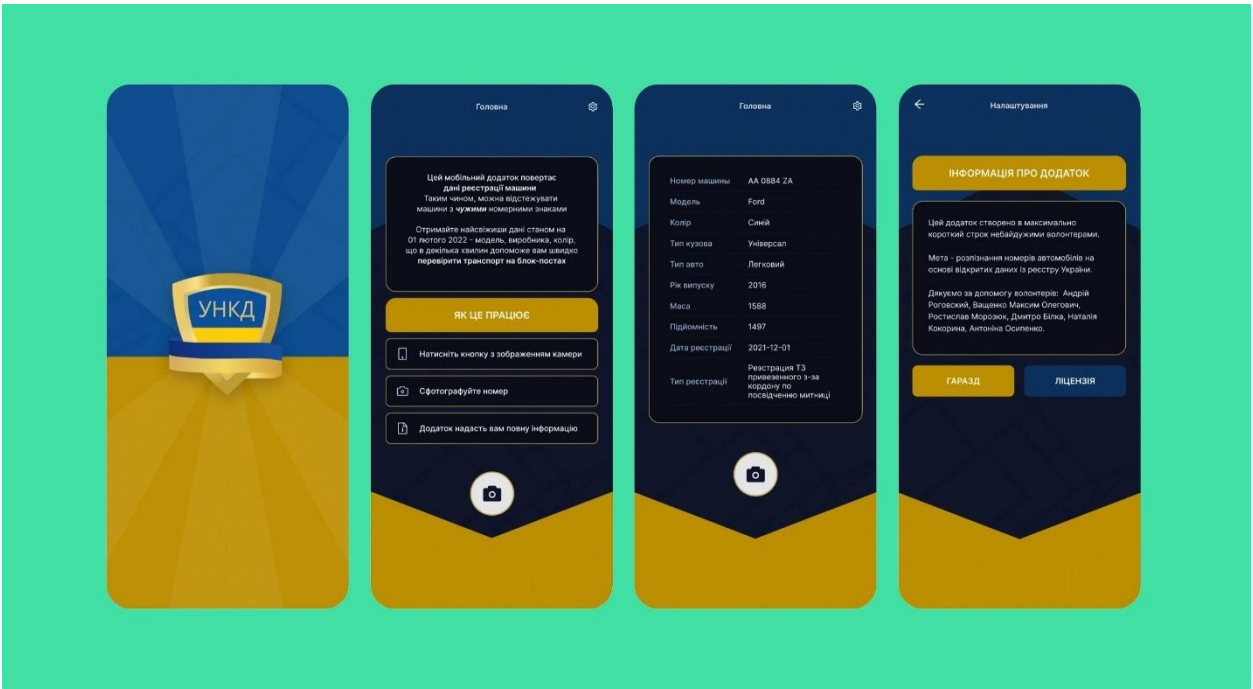


Рисунок 1.1 - Архітектура застосунку «Волонтер»

«ДоброДій» надає інструменти для координації волонтерів та збору пожертв [3]. Платформа має зручний мобільний інтерфейс, але обмежені можливості для управління проєктами та аналітики. Діаграма взаємодії користувачів у системі «ДоброДій» представлена на рисунку 1.2.

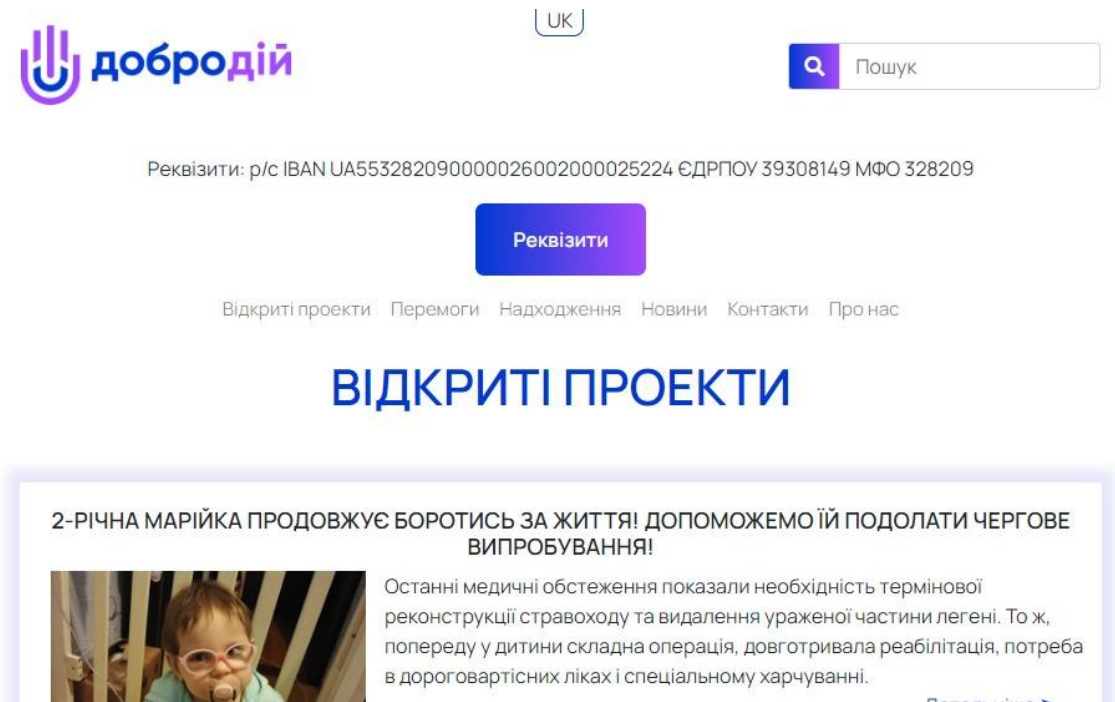


Рисунок 1.2 - Діаграма взаємодії користувачів у системі «ДоброДій»

«Спільнокошт» є українською краудфандинговою платформою, яка дозволяє громадським організаціям збирати кошти на свої проекти [4]. Платформа має потужні інструменти для фінансової звітності, але обмежені можливості для координації волонтерів. Структура даних проєктів у «Спільнокошт» показана на рисунку 1.3.

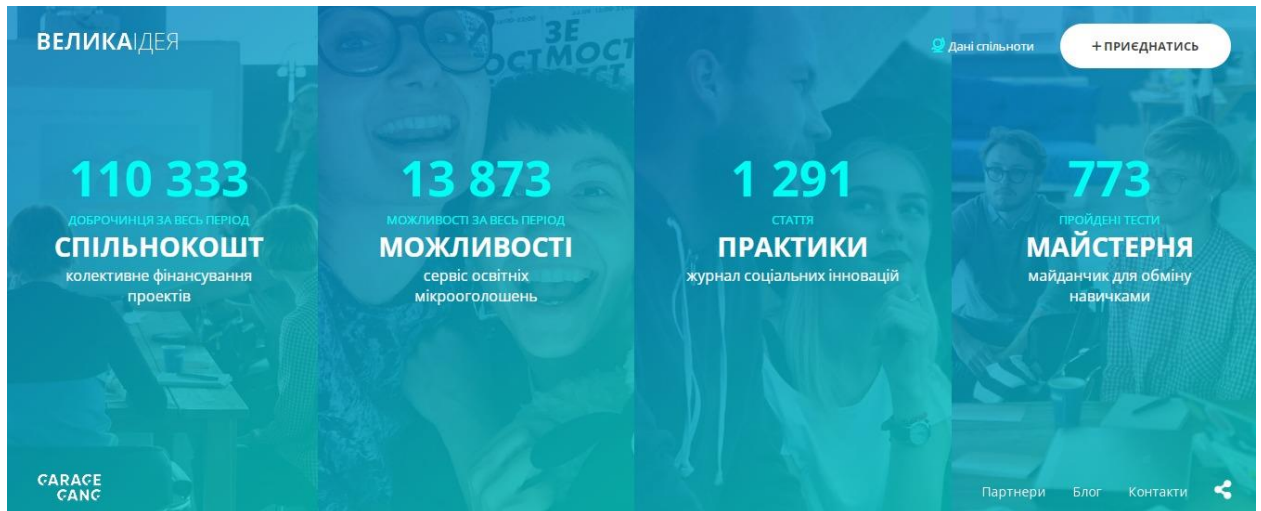


Рисунок 1.3 - Структура даних проєктів у «Спільнокошт»

NGO Forum є платформою для комунікації та співпраці між громадськими організаціями України [5]. Система дозволяє створювати проєкти, координувати волонтерів та генерувати звіти, але не має інтегрованих інструментів для збору пожертв. Архітектура NGO Forum представлена на рисунку 1.4.



Рисунок 1.4 - Архітектура платформи NGO Forum

«Мрійдій» є застосунком для залучення волонтерів до соціальних проєктів [6]. Платформа має зручний мобільний інтерфейс та інтеграцію з соціальними мережами, але обмежені можливості для управління проєктами та звітності [7]. Скріншот API «Мрійдій» показана на рисунку 1.5.

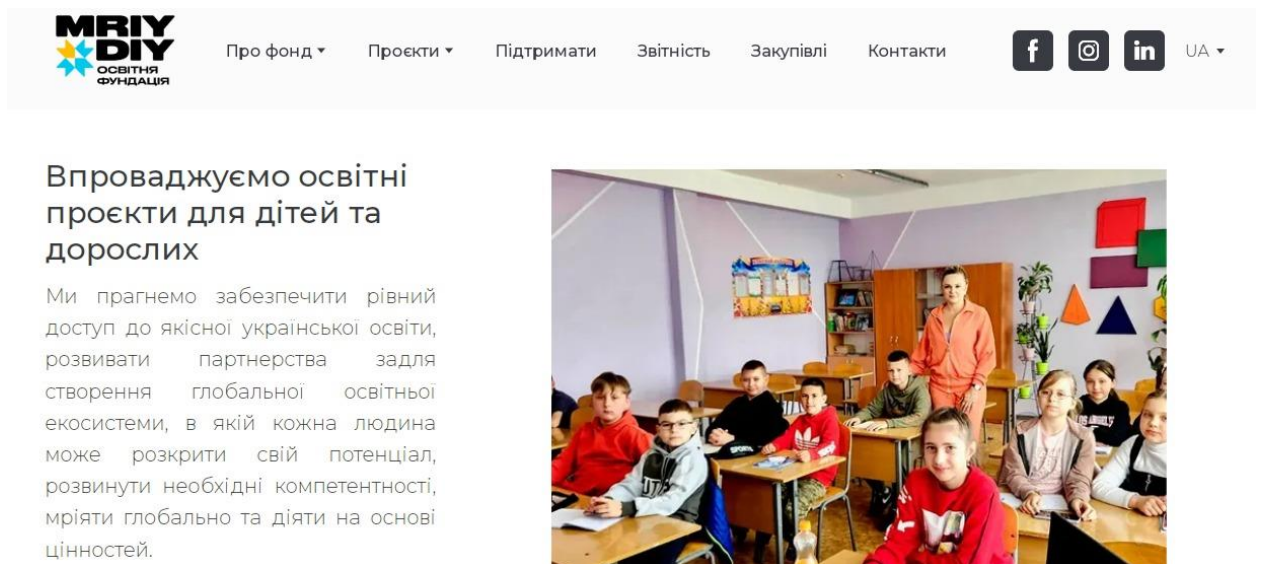


Рисунок 1.5 - Скріншот «Мрійдій»

На основі проведеного аналізу можна визначити ключові функціональні вимоги до вебзастосунку для підтримки діяльності волонтерської громадської організації в Україні:

- реєстрація та управління профілями волонтерів;
- створення та управління проектами;
- координація волонтерських груп;
- збір та управління пожертвами;
- відстеження потреб та їх візуалізація;
- комунікація між учасниками;
- формування звітності;
- інтеграція з соціальними мережами;
- аналітика даних та візуалізація результатів діяльності;
- багатомовний інтерфейс (українська, англійська) [8].

Особливу увагу слід приділити безпеці даних та відповідності законодавству України щодо захисту персональних даних та діяльності громадських організацій [9]. Також важливим аспектом є адаптивність інтерфейсу для різних пристроїв, оскільки значна частина волонтерів використовує мобільні пристрої для комунікації та координації діяльності.

Аналіз існуючих рішень показав, що жоден із розглянутих застосунків не забезпечує всі необхідні функції для ефективної роботи волонтерської громадської організації в Україні. Тому розробка спеціалізованого вебзастосунку, який би враховував усі визначені вимоги та специфіку волонтерської діяльності в українських реаліях, є актуальною задачею. У розробці власного рішення доцільно врахувати сильні сторони існуючих застосунків та уникнути їхніх недоліків, створивши зручну та функціональну платформу для підтримки діяльності волонтерських громадських організацій в Україні [10].

1.2 Порівняльний аналіз розробки та аналогів існуючих вебрішень

Проведемо порівняльний аналіз технологій розробки вебсайтів для громадських організацій та існуючих вебрішень, які представлені на ринку. Це

дозволить обґрунтувати вибір оптимальних технологій та архітектурних рішень для розробки власного програмного забезпечення [11].

Для детального порівняння було обрано найпопулярніші технологічні стеки та рішення, які використовуються для створення вебсайтів громадських організацій: WordPress з плагіном Charitable, Drupal з модулем CiviCRM, React з Node.js, Django, Wix Nonprofit Solutions та Salesforce Nonprofit Cloud. Порівняння ключових характеристик цих технологій представлено у таблиці 1.2.

Таблиця 1.2 - Порівняння технологій для розробки вебсайтів громадських організацій

Характеристика	WordPress + Charitable	Drupal + CiviCRM	React + Node.js	Django	Wix Nonprofit	Salesforce Nonprofit
Швидкість розробки	Висока	Середня	Низька	Середня	Висока	Середня
Гнучкість налаштування	Середня	Висока	Дуже висока	Висока	Низька	Середня
Масштабованість	Обмежена	Висока	Дуже висока	Висока	Обмежена	Висока
Безпека	Середня	Висока	Залежить від реалізації	Висока	Середня	Висока
Складність підтримки	Низька	Висока	Висока	Середня	Низька	Висока
Інтеграційні можливості	Середні	Високі	Дуже високі	Високі	Обмежені	Дуже високі
Вартість розробки	Низька	Середня	Висока	Середня	Низька	Висока
Вартість підтримки	Низька	Середня	Середня	Середня	Низька	Висока
Підтримка мобільних пристроїв	Так	Так	Так	Так	Так	Так

Розглянемо детальніше кожне з технологічних рішень та проаналізуємо їхні архітектурні особливості.

WordPress з плагіном Charitable є найпоширенішим рішенням для створення вебсайтів громадських організацій завдяки низькому порогу входження та широкому набору готових плагінів [12]. Архітектура рішення базується на монолітному ядрі з модульними розширеннями, що зображено на рисунку 1.7.

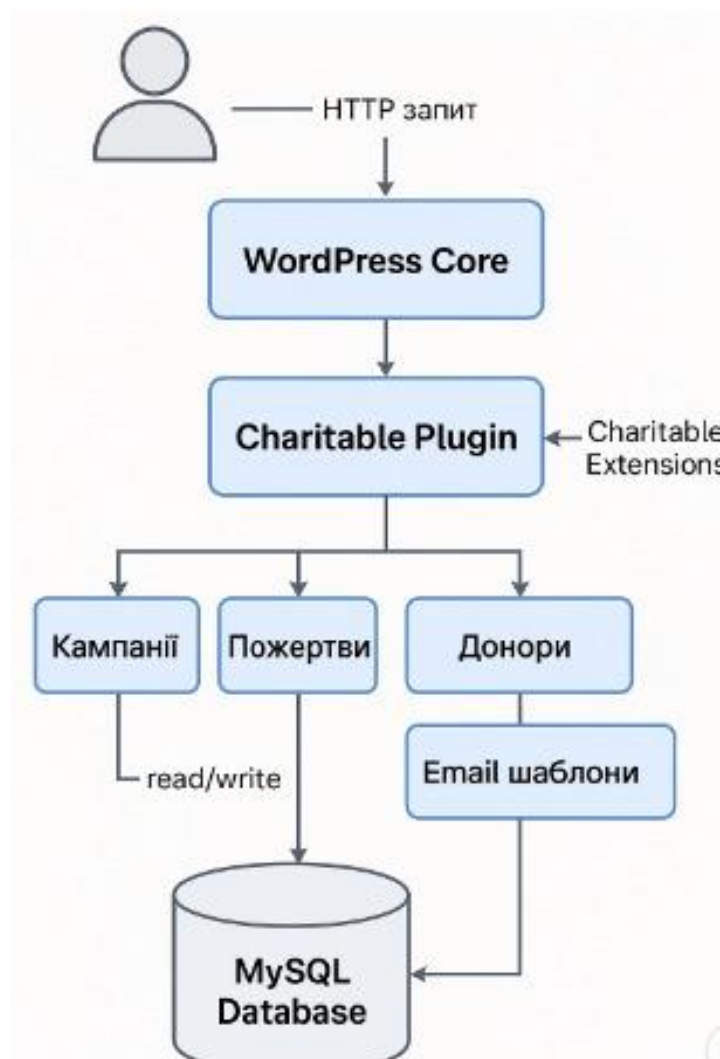


Рисунок 1.7 - Архітектура WordPress з плагіном Charitable

Drupal з модулем CiviCRM є потужним рішенням для великих громадських організацій, яке надає розширені можливості для управління членами, заходами та пожертвами [13]. Система має модульну архітектуру з високим рівнем налаштування, що представлено на рисунку 1.8.

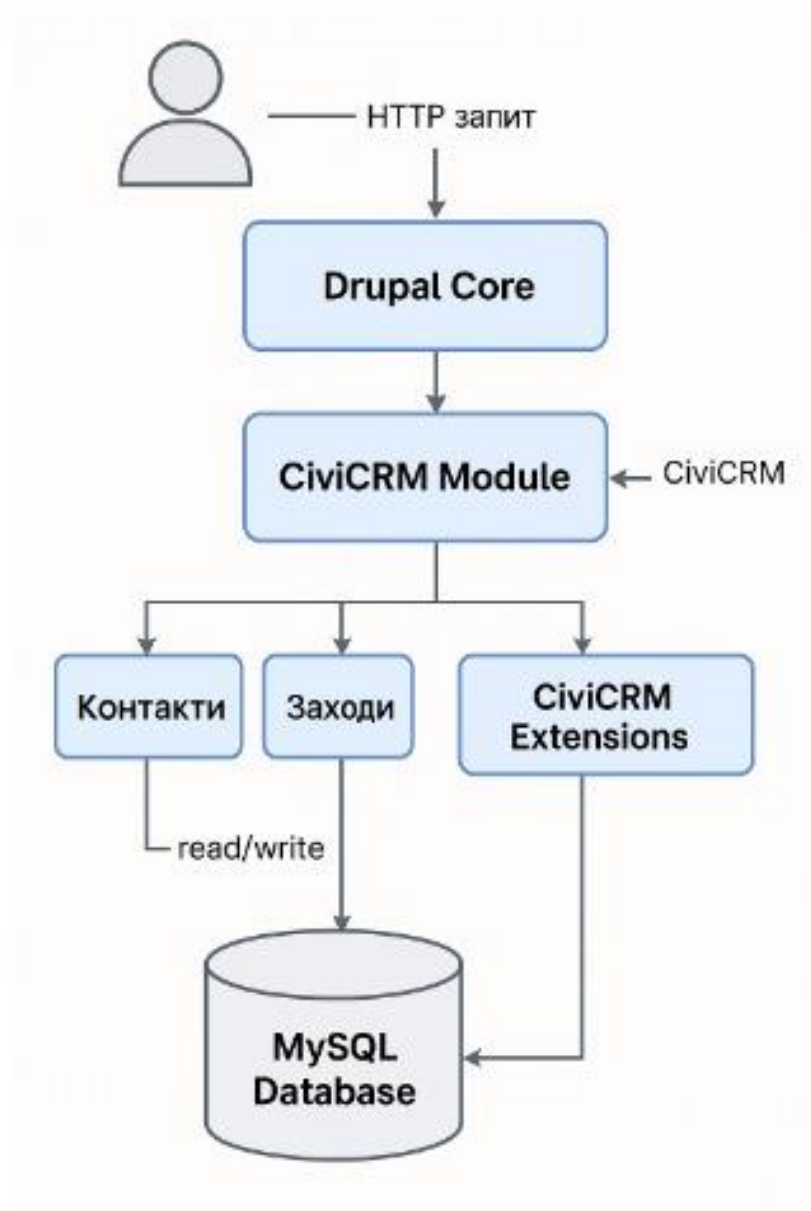


Рисунок 1.8 - Архітектура Drupal з модулем CiviCRM

React з Node.js є сучасним стеком технологій для розробки вебзастосунків з високим рівнем інтерактивності та продуктивності [14]. Архітектура такого рішення базується на принципі Single Page Application (SPA) з відокремленням клієнтської та серверної частин, що зображено на рисунку 1.9.

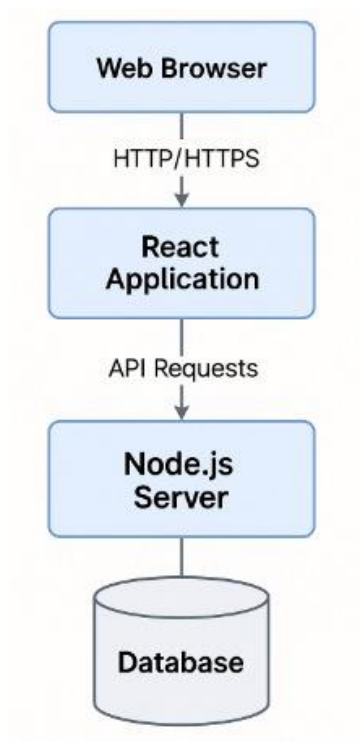


Рисунок 1.9 - Архітектура вебзастосунку на основі React з Node.js

Django є високорівневим Python-фреймворком, який дозволяє швидко розробляти безпечні та масштабовані вебзастосунки [15]. Архітектура Django базується на шаблоні Model-View-Template (MVT), що представлено на рисунку 1.10.

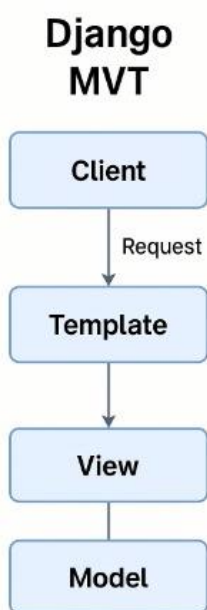


Рисунок 1.10 - Архітектура Django на основі MVT шаблону

Wix Nonprofit Solutions є низькокодовою платформою для створення вебсайтів, яка пропонує спеціалізовані шаблони та інструменти для громадських організацій [16]. Архітектура Wix базується на хмарній інфраструктурі з обмеженими можливостями для налаштування, що зображено на рисунку 1.11.

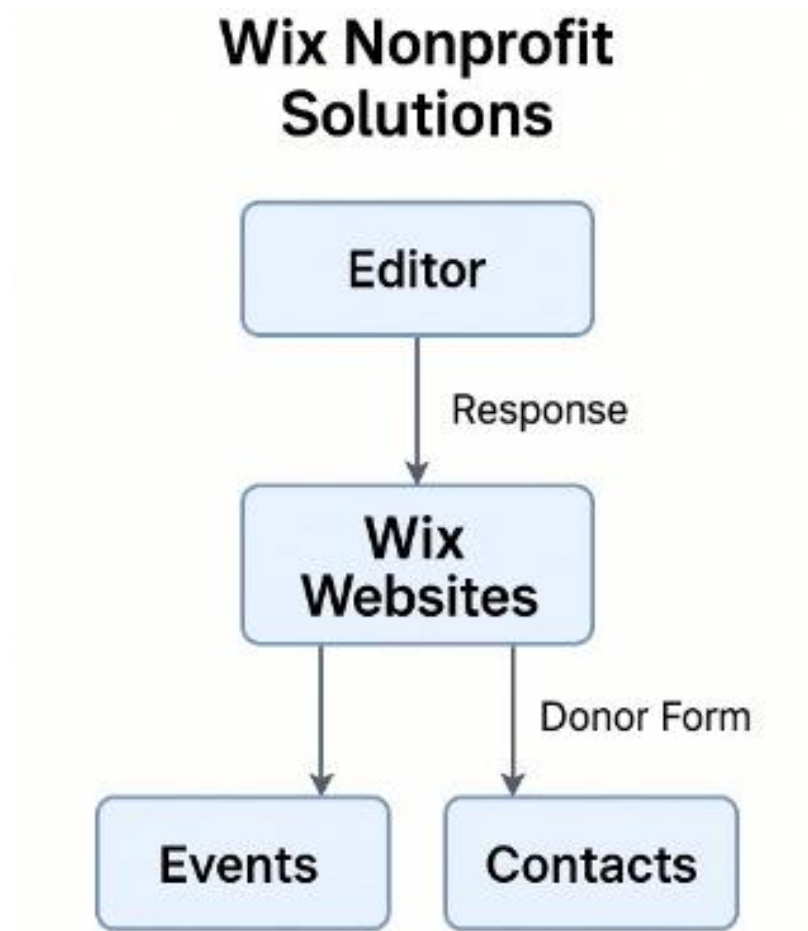


Рисунок 1.11 - Архітектура Wix Nonprofit Solutions

Для порівняння існуючих вебрішень для громадських організацій в Україні було обрано шість популярних вебсайтів: «Повернись живим», «Карітас Україна», «Українська волонтерська служба», «Восток-SOS», «Крила Фенікса» та «Червоний Хрест України». Аналіз функціональних можливостей цих вебсайтів представлено у таблиці 1.3.

Таблиця 1.3 - Порівняння функціональних можливостей вебсайтів українських громадських організацій

Функціональність	Повернись живим	Карітас Україна	Українська волонтерська служба	Восток-SOS	Крила Фенікса	Червоний Хрест України
Інформаційні сторінки	Так	Так	Так	Так	Так	Так
Збір пожертв	Так	Так	Так	Так	Так	Так
Реєстрація волонтерів	Ні	Так	Так	Так	Ні	Так
Календар заходів	Ні	Так	Так	Так	Ні	Так
Звітність про діяльність	Так	Так	Так	Так	Так	Так
Мультимовність	Так	Так	Ні	Так	Ні	Так
Блог/новини	Так	Так	Так	Так	Так	Так
Інтеграція з соцмережами	Так	Так	Так	Так	Так	Так
Карта потреб/проектів	Так	Ні	Ні	Так	Ні	Так
Особистий кабінет	Ні	Ні	Так	Ні	Ні	Так

Технічний аналіз цих вебсайтів за допомогою інструментів Lighthouse та PageSpeed Insights показав значні відмінності у швидкості завантаження, оптимізації мобільного відображення та загальних показниках якості коду [17], [18].

Результати аналізу представлено у таблиці 1.4.

На основі проведеного аналізу можна виділити ключові тенденції у розробці вебсайтів для громадських організацій:

- перехід від статичних до інтерактивних вебзастосунків;
- підвищена увага до мобільної оптимізації;
- інтеграція систем онлайн-пожертв;

- мультимовність;
- прозора звітність про діяльність;
- інтеграція з соціальними мережами;
- простота навігації та інтуїтивно зрозумілий інтерфейс [19].

Таблиця 1.4 - Технічні характеристики вебсайтів українських громадських організацій

Характеристика	Поверни сь живим	Карітас Україна	Українська волонтерська служба	Восток- SOS	Крила Фенікса	Червоний Хрест України
Швидкість завантаження (сек)	2.3	3.5	1.8	4.2	3.1	2.7
PageSpeed Insights (мобільний)	76	62	85	58	65	73
PageSpeed Insights (ПК)	89	75	92	71	81	86
Технологічний стек	WordPress	Drupal	React + Node.js	WordPress	WordPress	Drupal
Адаптивний дизайн	Так	Частков о	Так	Так	Частково	Так
HTTPS	Так	Так	Так	Так	Так	Так
Середній розмір сторінки (МБ)	1.5	2.3	0.9	3.1	2.8	1.7

Також варто відзначити, що найбільш успішні вебсайти громадських організацій мають чітко сформульовану місію, візуально привабливий дизайн, просту форму для збору пожертв, регулярно оновлюваний контент та швидке завантаження сторінок [20].

1.3 Висновки за розділом 1

У результаті аналізу застосунків для підтримки діяльності громадських організацій було встановлено, що на українському ринку представлено декілька рішень, таких як «Волонтер», «ДоброДій», «Спільнокошт», NGO Forum, «Мрійдій» та «ВолонтерUA». Проте жоден із цих застосунків не забезпечує повний набір функцій, необхідних для ефективної роботи волонтерської громадської організації в Україні. Було визначено ключові функціональні вимоги до вебзастосунку: реєстрація та управління профілями волонтерів, створення та управління проектами, координація волонтерських груп, збір та управління пожертвами, відстеження потреб та їх візуалізація, комунікація між учасниками, формування звітності, інтеграція з соціальними мережами, аналітика даних та візуалізація результатів діяльності, багатомовний інтерфейс.

Порівняльний аналіз розробки та аналогів існуючих вебрішень показав, що на ринку представлені різні технологічні стеки для створення вебсайтів громадських організацій: WordPress з плагіном Charitable, Drupal з модулем CiviCRM, React з Node.js, Django, Wix Nonprofit Solutions та Salesforce Nonprofit Cloud. За результатами аналізу, оптимальним підходом для розробки вебсайту для підтримки діяльності волонтерської громадської організації в Україні є використання сучасного фреймворку React з Node.js, що забезпечить високу гнучкість, масштабованість та можливість реалізації всіх необхідних функціональних вимог.

2 МЕТОДИ ТА МОДЕЛІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз методів розробки

У даному підрозділі обґрунтуємо вибір технологій, інструментів та засобів проєктування для розробки вебсайту громадської організації, що спеціалізується на підтримці волонтерської діяльності в Україні. Правильний вибір технологічного стеку є критично важливим для забезпечення функціональності, швидкодії, масштабованості та зручності використання розроблюваного програмного забезпечення.

Порівняльний аналіз основних фреймворків для розробки клієнтської частини представлено у таблиці 2.1.

Таблиця 2.1 - Порівняння фреймворків для розробки клієнтської частини

Характеристика	React	Angular	Vue.js	Svelte
Популярність (GitHub зірки)	180,000+	75,000+	190,000+	60,000+
Розмір бібліотеки	Середній	Великий	Малий	Дуже малий
Крива навчання	Середня	Висока	Низька	Низька
Продуктивність	Висока	Висока	Висока	Дуже висока
Екосистема/спільнота	Дуже велика	Велика	Велика	Середня
TypeScript підтримка	Добра	Вбудована	Добра	Добра
SSR підтримка	Так (Next.js)	Так (Universal)	Так (Nuxt.js)	Так (SvelteKit)
Підтримка великих компаній	Facebook	Google	Незалежний	Незалежний
Гнучкість архітектури	Висока	Середня	Висока	Висока

React дозволяє створювати інтерактивні компоненти інтерфейсу, які ефективно оновлюються при зміні даних. Архітектура React-додатку представлена на рисунку 2.1.

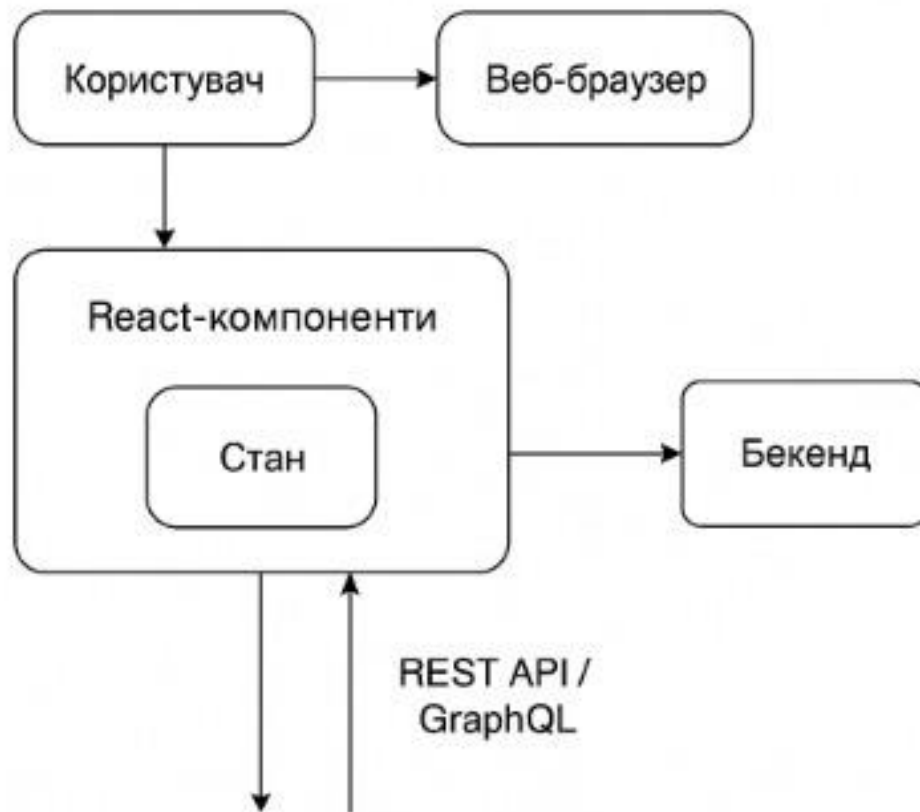


Рисунок 2.1 - Архітектура React-додатку

Додатково було обрано наступні бібліотеки для розширення можливостей React:

- Redux для управління станом додатку;
- React Router для реалізації навігації;
- Material-UI для стандартних компонентів інтерфейсу;
- React Hook Form для обробки форм;
- Axios для HTTP запитів [23].

Діаграма взаємодії основних бібліотек у клієнтській частині представлена на рисунку 2.2.

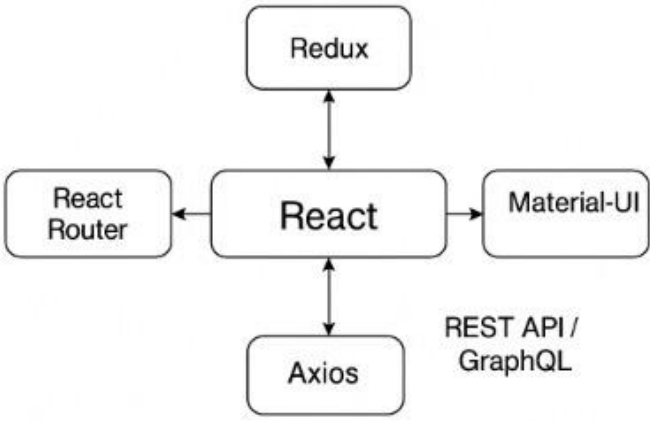


Рисунок 2.2 - Діаграма взаємодії бібліотек у клієнтській частині

Для серверної частини було проведено порівняння популярних технологій, результати якого представлені у таблиці 2.2.

Таблиця 2.2 - Порівняння технологій для розробки серверної частини

Характеристика	Node.js + Express	Django	Ruby on Rails	Laravel	Spring Boot
Мова програмування	JavaScript	Python	Ruby	PHP	Java
Швидкість розробки	Висока	Висока	Дуже висока	Висока	Середня
Продуктивність	Висока	Середня	Середня	Середня	Дуже висока
Асинхронність	Вбудована	Обмежена	Обмежена	Обмежена	Добра
Масштабованість	Висока	Середня	Середня	Середня	Висока
Екосистема/спільнота	Дуже велика	Велика	Велика	Велика	Дуже велика
ORM	Sequelize/Prisma	Вбудовані	Active Record	Eloquent	Hibernate
Мікросервісна підтримка	Відмінна	Обмежена	Обмежена	Обмежена	Відмінна
Типізація	TypeScript	Вбудована	Обмежена	Обмежена	Вбудована

На основі порівняльного аналізу для розробки серверної частини вебсайту було обрано Node.js з Express [24]. Цей вибір обумовлений такими перевагами:

використання єдиної мови програмування (JavaScript) на клієнті та сервері;

- висока продуктивність та асинхронна обробка запитів;
- велика кількість готових пакетів та бібліотек;
- легка інтеграція з React;
- хороша підтримка масштабування;
- можливість використання TypeScript для типізації.

Архітектура серверної частини на основі Node.js та Express представлена на рисунку 2.3.

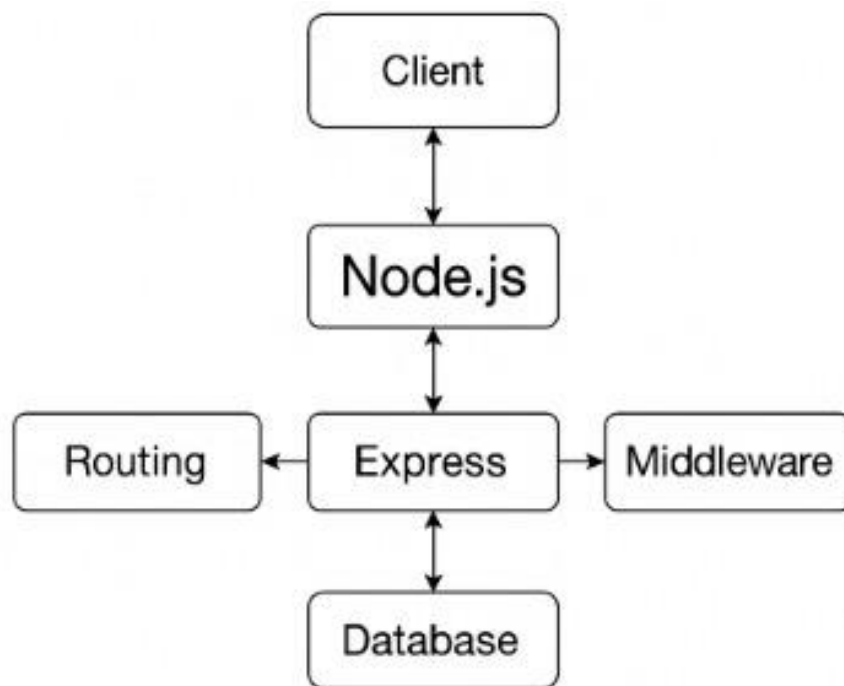


Рисунок 2.3 - Архітектура серверної частини на основі Node.js та Express

Для зберігання даних було проведено порівняння різних систем управління базами даних, результати якого представлені у таблиці 2.3.

Таблиця 2.3 - Порівняння систем управління базами даних

Характеристика	MongoDB	PostgreSQL	MySQL	Firebase	Redis
Тип	NoSQL (документо-орієнтована)	SQL (реляційна)	SQL (реляційна)	NoSQL (хмарна)	NoSQL (ключ-значення)
Масштабованість	Висока	Середня	Середня	Дуже висока	Висока
Продуктивність	Висока	Висока	Висока	Висока	Дуже висока
Підтримка транзакцій	Обмежена	Повна	Повна	Обмежена	Обмежена
Гнучкість схеми	Дуже висока	Середня	Низька	Висока	Висока
Складність налаштування	Низька	Середня	Низька	Дуже низька	Низька
Підтримка спільноти	Висока	Висока	Дуже висока	Висока	Висока
Інтеграція з Node.js	Відмінна	Добра	Добра	Відмінна	Відмінна

Для основного зберігання даних вебсайту громадської організації було обрано PostgreSQL через такі переваги:

- підтримка складних зв'язків між даними, що важливо для управління проєктами, волонтерами та пожертвами;
- надійність та стабільність;
- підтримка транзакцій для забезпечення цілісності даних;
- хороша масштабованість;
- відкритий вихідний код та активна спільнота [25].

Додатково було вирішено використовувати Redis для кешування даних та управління сесіями користувачів, що дозволить підвищити продуктивність системи [26].

Для проєктування інтерфейсу користувача було обрано Figma як основний інструмент дизайну [27]. Цей вибір обумовлений такими перевагами:

- хмарне рішення з можливістю спільної роботи;
- потужні інструменти для створення компонентних дизайн-систем;
- підтримка прототипування;
- хороша інтеграція з іншими інструментами;
- можливість експорту CSS та SVG;
- підтримка адаптивного дизайну.

Приклад процесу проєктування інтерфейсу в Figma представлений на рисунку 2.4.



Рисунок 2.4 - Процес проєктування інтерфейсу в Figma

Для контролю версій та організації робочого процесу було обрано Git з GitHub як платформою для хостингу репозиторію [28]. Цей вибір обумовлений такими перевагами:

- ефективне управління версіями коду;
- можливість паралельної розробки через гілки;
- підтримка code review через pull request;
- інтеграція з CI/CD системами;
- відкритий доступ до репозиторію для волонтерів-розробників.

Схема робочого процесу Git Flow представлена на рисунку 2.5.

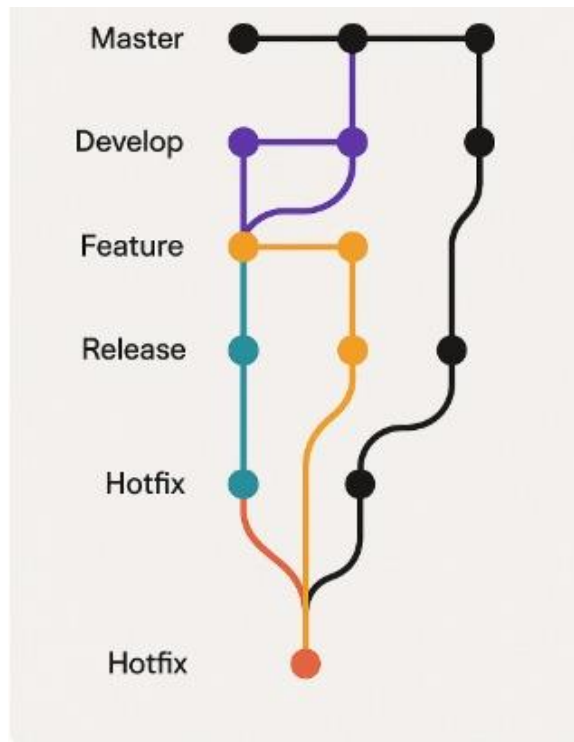


Рисунок 2.5 - Схема робочого процесу Git Flow

Для тестування та розгортання додатку було обрано наступні інструменти:

- Jest та React Testing Library для модульного та інтеграційного тестування клієнтської частини;
- Mocha та Chai для тестування серверної частини;
- Cypress для end-to-end тестування;
- Docker для контейнеризації додатку;
- GitHub Actions для налаштування CI/CD;
- Netlify для хостингу клієнтської частини;
- Heroku для хостингу серверної частини [29].

Схема процесу CI/CD представлена на рисунку 2.6.

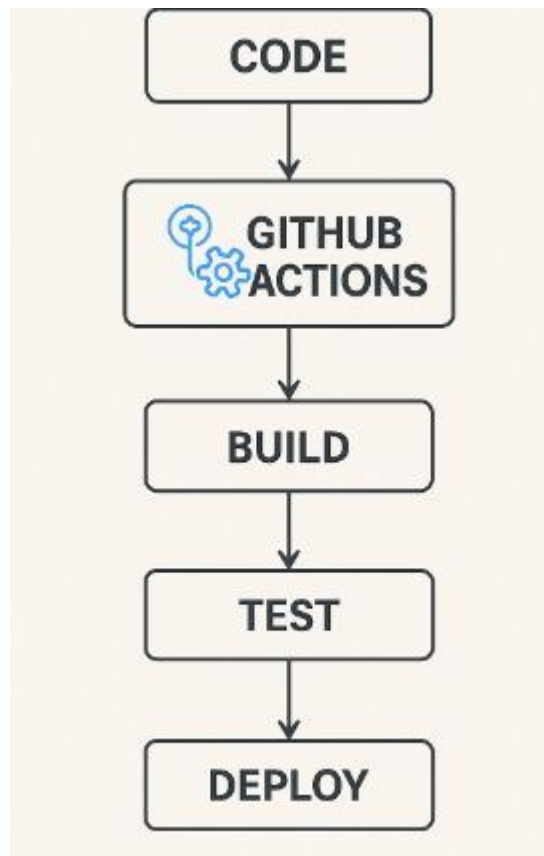


Рисунок 2.6 - Схема процесу CI/CD на основі GitHub Actions

Обрані засоби проєктування та розробки дозволяють створити сучасний, надійний та масштабований вебсайт для громадської організації з урахуванням усіх визначених вимог. Використання JavaScript як єдиної мови програмування для клієнтської та серверної частин спрощує розробку та подальшу підтримку системи. Обрані технології мають активну спільноту розробників, що забезпечує доступ до готових рішень, документації та підтримки [29].

Важливою перевагою обраного технологічного стеку є можливість легкого масштабування системи зі зростанням кількості користувачів та об'єму даних. Крім того, вибрані інструменти дозволяють організувати ефективний процес розробки з використанням сучасних практик, таких як автоматизоване тестування, безперервна інтеграція та розгортання, контроль якості коду [30].

2.2 Моделі роботи програмного забезпечення

У даному підрозділі розглянемо ключові сценарії використання вебсайту для підтримки діяльності волонтерської громадської організації в Україні. Визначення сценаріїв використання є важливим етапом проєктування, що дозволяє чітко окреслити функціональні можливості системи та взаємодію різних типів користувачів із нею [1].

Для ефективної організації сценаріїв використання необхідно визначити основні ролі користувачів, які будуть взаємодіяти з системою. На основі аналізу предметної області та вимог до програмного забезпечення були визначені наступні ролі:

- неавторизований відвідувач (гість);
- зареєстрований волонтер;
- координатор проєкту;
- адміністратор організації;
- донор [3].

Ієрархія ролей користувачів представлена на рисунку 2.7.



Рисунок 2.7 - Ієрархія ролей користувачів системи

Для кожної ролі розроблено відповідні сценарії використання системи, які відображають функціональні можливості, доступні для цієї ролі. Загальна діаграма сценаріїв використання представлена на рисунку 2.8.

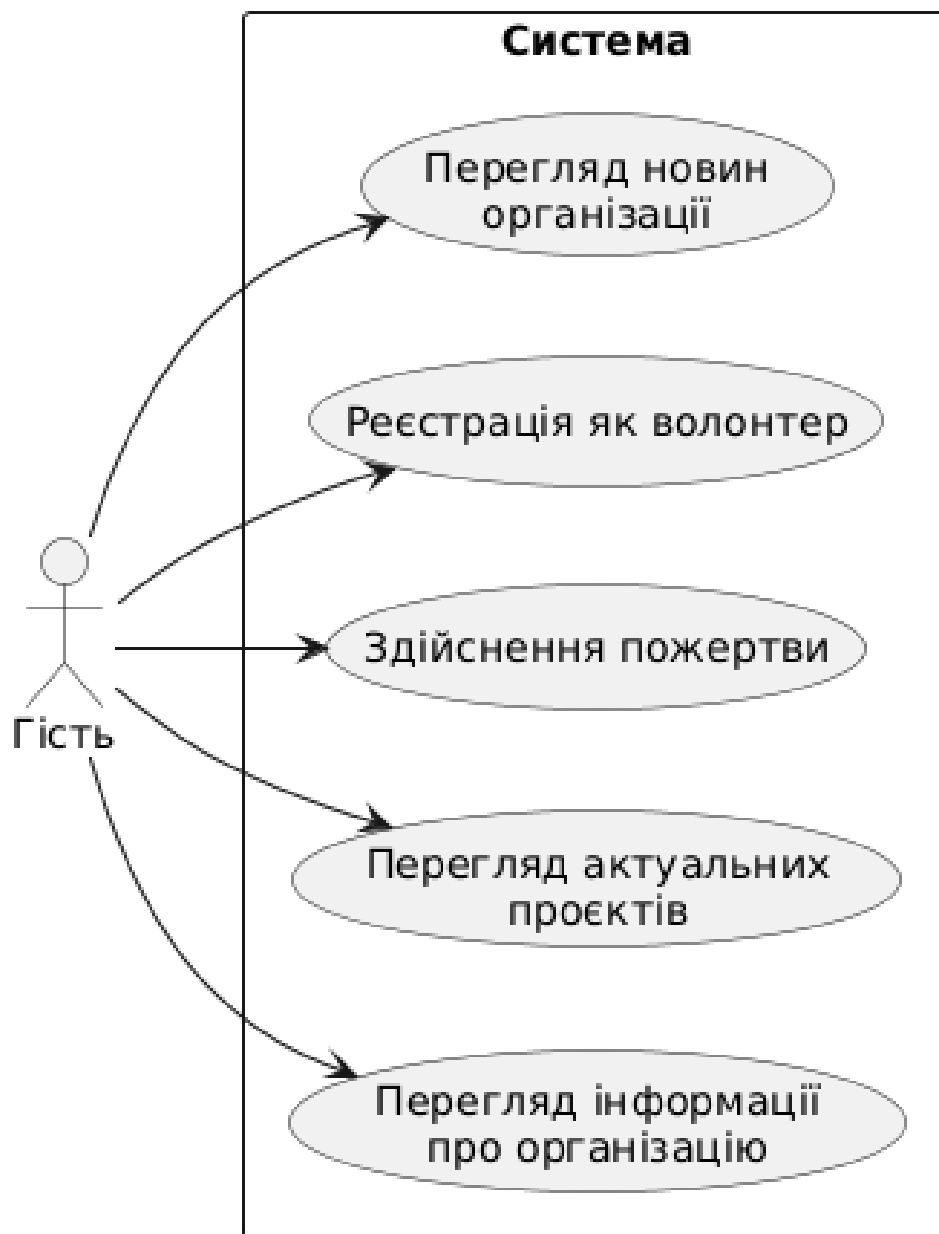


Рисунок 2.8 - Загальна діаграма сценаріїв використання системи

Розглянемо детальніше сценарії використання для кожної з ролей. Сценарії використання для неавторизованого відвідувача (гостя) представлені у таблиці 2.4.

Таблиця 2.4 - Сценарії використання для неавторизованого відвідувача

ID	Назва сценарію	Опис	Передумови	Основний шлях	Альтернативний шлях
CB1	Перегляд інформації про організацію	Гість переглядає загальну інформацію про організацію, її місію та діяльність	Доступ до вебсайту	1. Гість відкриває головну сторінку; 2. Переходить до розділу "Про нас"	Доступ до інформації з будь-якої сторінки через меню
CB2	Перегляд актуальних проєктів	Гість переглядає список активних проєктів організації	Доступ до вебсайту	1. Гість відкриває розділ "Проєкти"; 2. Переглядає список проєктів	Фільтрація проєктів за категоріями
CB3	Здійснення пожертви	Гість робить пожертву на підтримку організації чи конкретного проєкту	Доступ до вебсайту	1. Гість відкриває форму пожертви; 2. Вводить суму та платіжні дані; 3. Підтверджує операцію	Пожертва через зовнішні платіжні системи
CB4	Реєстрація як волонтер	Гість реєструється в системі для участі в волонтерській діяльності	Доступ до вебсайту	1. Гість відкриває форму реєстрації; 2. Заповнює особисті дані; 3. Підтверджує реєстрацію	Реєстрація через соціальні мережі

Для сценарію СВЗ "Здійснення пожертви" розроблено діаграму послідовності, яка відображає взаємодію користувача з системою (рис. 2.9).

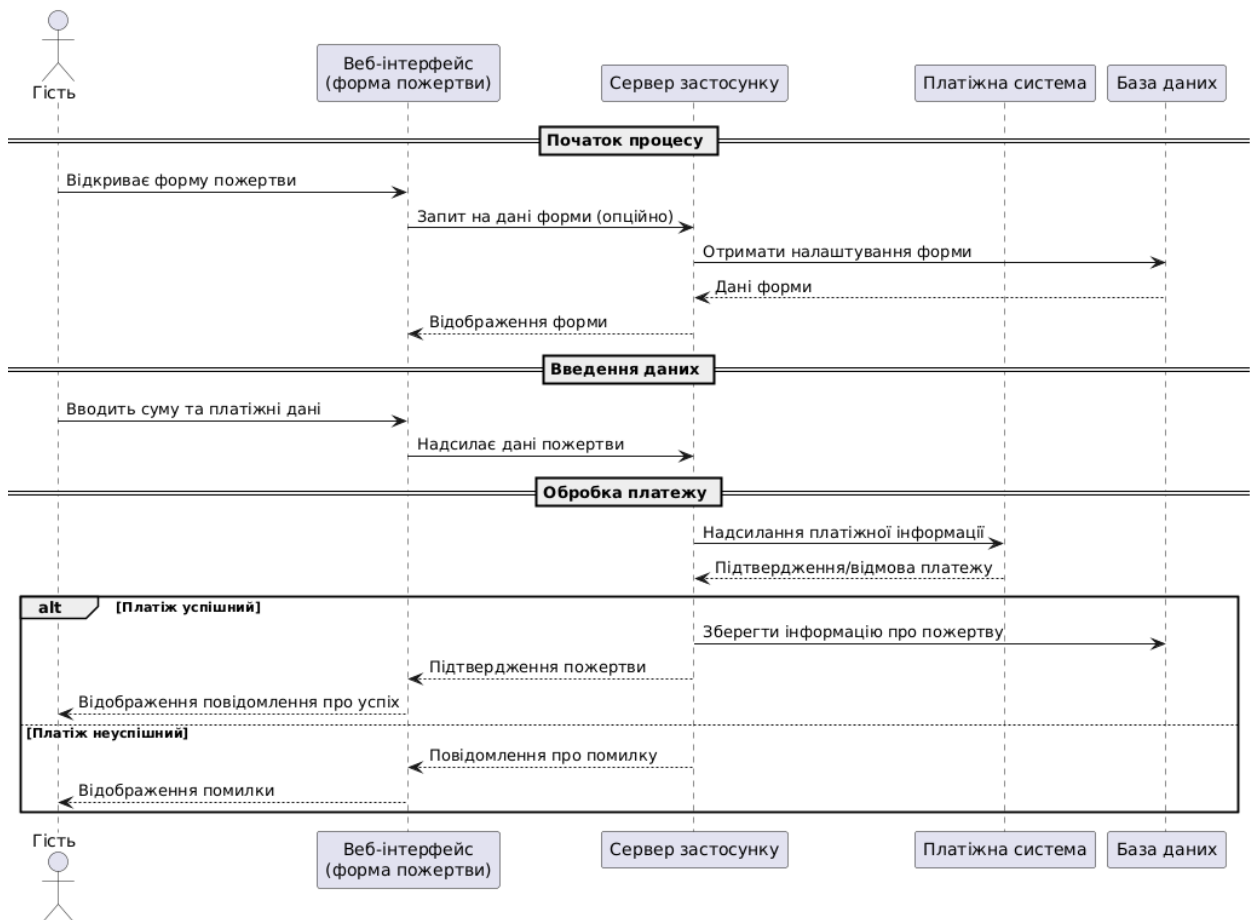


Рисунок 2.9 - Діаграма послідовності для сценарію "Здійснення пожертви"

На діаграмі послідовності для сценарію СВЗ показано, що процес починається з того, що гість відкриває форму пожертви. Система запитує дані про доступні проекти з бази даних та відображає форму з можливістю вибору проекту. Гість вводить суму пожертви та платіжні дані, після чого система валідує введені дані. Якщо валідація успішна, система відправляє запит до платіжного шлюзу для обробки платежу. Платіжний шлюз повертає результат операції, система зберігає інформацію про пожертву в базі даних та відображає користувачу підтвердження операції або повідомлення про помилку.

Сценарії використання для зареєстрованого волонтера розширюють можливості гостя та представлені у таблиці 2.5.

Таблиця 2.5 - Сценарії використання для зареєстрованого волонтера

ID	Назва сценарію	Опис	Передумови	Основний шлях	Альтернативний шлях
CB6	Управління профілем	Волонтер редагує свій профіль, додає навички та досвід	Авторизація в системі	1. Волонтер відкриває свій профіль; 2. Редагує інформацію; 3. Зберігає зміни	Додавання фотографії через завантаження файлу
CB7	Запис на участь у проєкті	Волонтер подає заявку на участь у проєкті	Авторизація в системі	1. Волонтер переглядає проєкти; 2. Обирає проєкт; 3. Подає заявку на участь	Отримання запрошення від координатора проєкту

Для сценарію CB7 "Запис на участь у проєкті" розроблено діаграму взаємодії, яка відображає обмін повідомленнями між компонентами системи (рис. 2.10).

На діаграмі взаємодії для сценарію CB7 (рис. 2.10) показано, що волонтер взаємодіє з інтерфейсом користувача для перегляду списку проєктів. Інтерфейс користувача звертається до сервісу проєктів для отримання доступних проєктів. Після вибору проєкту волонтером, інтерфейс відправляє запит на участь до сервісу заявок. Сервіс заявок зберігає інформацію про заявку в базі даних та повідомляє координатора проєкту через систему сповіщень.

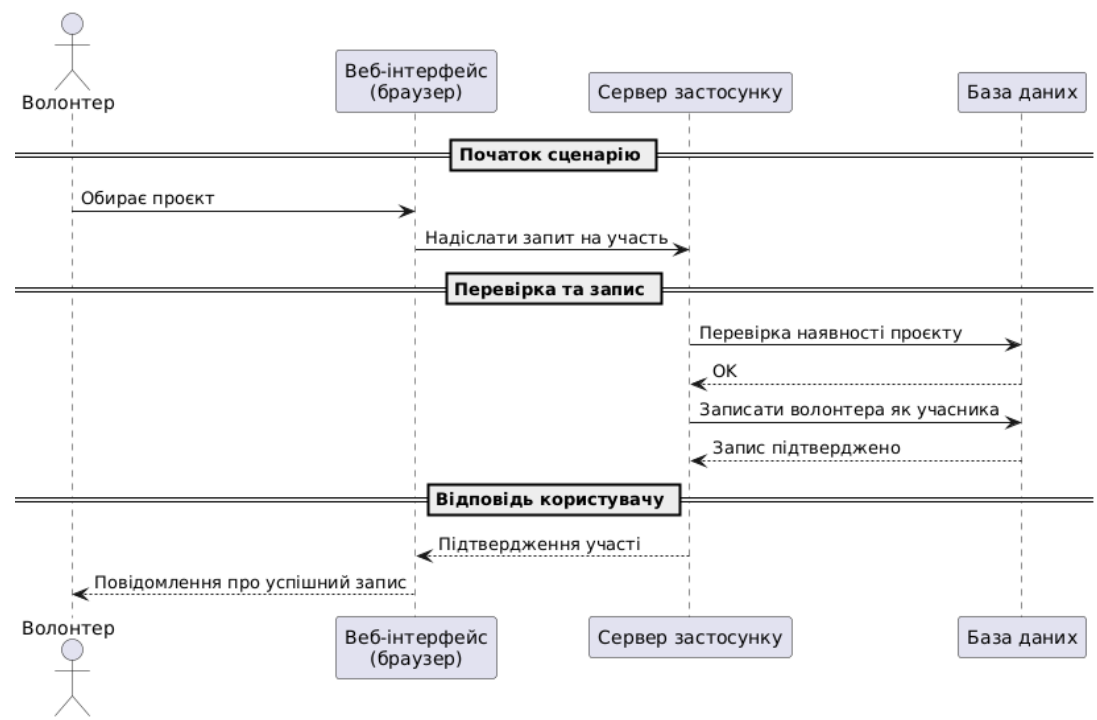


Рисунок 2.10 - Діаграма взаємодії для сценарію "Запис на участь у проєкті"

Для сценарію "Управління волонтерами проєкту" розроблено діаграму послідовності, яка детально відображає процес (рис. 2.11).

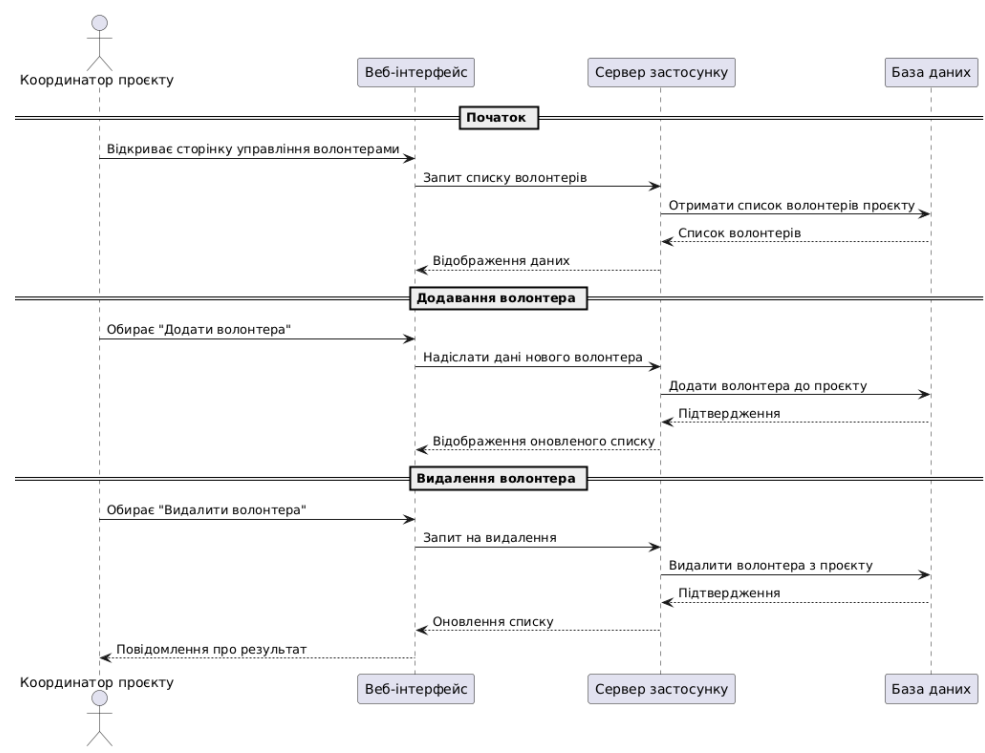


Рисунок 2.11 - Діаграма послідовності для сценарію "Управління волонтерами проєкту"

На діаграмі послідовності показано, що координатор відкриває сторінку управління волонтерами проєкту. Система запитує з бази даних інформацію про поточних учасників проєкту та відображає їх список. Координатор може додати нового волонтера, вказавши його ідентифікатор або електронну пошту. Система перевіряє наявність користувача в базі даних, додає його до проєкту та встановлює відповідні права доступу. Потім система надсилає сповіщення волонтеру про додавання до проєкту та оновлює інтерфейс, відображаючи оновлений список учасників.

Сценарії використання для адміністратора організації охоплюють функції управління всією системою та представлені у таблиці 2.6.

Таблиця 2.6 - Сценарії використання для адміністратора організації

ID	Назва сценарію	Опис	Передумови	Основний шлях	Альтернативний шлях
СВ 8	Управління користувачами	Адміністратор керує обліковими записами користувачів	Авторизація з правами адміністратора	1. Адміністратор відкриває розділ "Користувачі"; 2. Переглядає список; 3. Модифікує права доступу	Блокування порушників правил
СВ 9	Управління контентом сайту	Адміністратор редагує інформаційний контент сайту	Авторизація з правами адміністратора	1. Адміністратор відкриває CMS; 2. Редагує сторінки; 3. Публікує зміни	Планування публікацій на майбутні дати

Сценарії використання для донора представлені у таблиці 2.7.

Таблиця 2.7 - Сценарії використання для донора

ID	Назва сценарію	Опис	Передумови	Основний шлях	Альтернативний шлях
CB10	Реєстрація донора	Користувач реєструється як донор	Доступ до вебсайту	1. Користувач відкриває форму реєстрації; 2. Заповнює дані; 3. Підтверджує реєстрацію	Реєстрація через соціальні мережі або платіжні системи
CB11	Здійснення регулярних платежів	Донор налаштовує автоматичні щомісячні пожертви	Авторизація в системі	1. Донор відкриває розділ "Регулярні платежі"; 2. Встановлює суму та періодичність; 3. Підтверджує	Скасування або зміна параметрів регулярних платежів

Для забезпечення зручності використання системи різними категоріями користувачів було розроблено макети інтерфейсу для ключових сценаріїв використання. Макет головної сторінки вебсайту представлений на рисунку 2.12.

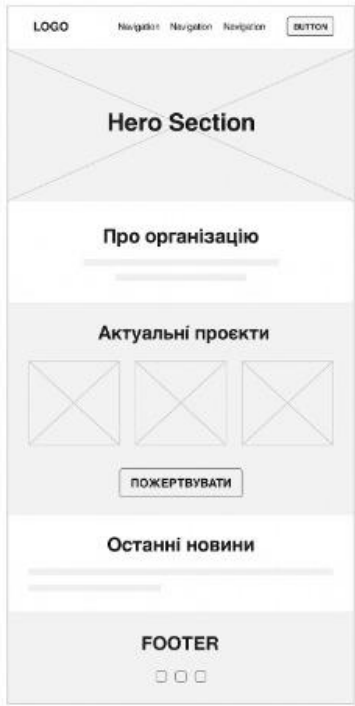


Рисунок 2.12 - Макет головної сторінки вебсайту

Макет сторінки проєкту з можливістю запису волонтерів представлений на рисунку 2.13.



Рисунок 2.13 - Макет сторінки проєкту

Визначені сценарії використання дозволяють чітко окреслити функціональні можливості системи, зрозуміти потреби різних категорій користувачів та спроектувати інтуїтивно зрозумілий інтерфейс. На основі розроблених сценаріїв використання буде побудована архітектура системи та спроектована структура бази даних.

Важливо зазначити, що в процесі розробки та тестування системи сценарії використання можуть уточнюватися та доповнюватися відповідно до зворотного зв'язку від користувачів та зміни вимог до системи. Гнучкий підхід до проєктування дозволить створити максимально зручний та функціональний вебсайт для підтримки діяльності волонтерської громадської організації в Україні.

2.3 Висновки за розділом 2

У другому розділі було проведено детальний аналіз методів розробки та моделей роботи програмного забезпечення для підтримки діяльності волонтерської громадської організації в Україні.

В результаті аналізу методів розробки було обґрунтовано вибір технологій та інструментів проєктування. Для клієнтської частини обрано фреймворк React через його високу популярність, гнучкість, потужну екосистему та відмінну підтримку спільноти. Додатково обрано бібліотеки Redux для управління станом додатку, React Router для навігації, Material-UI для стандартних компонентів інтерфейсу, React Hook Form для обробки форм та Axios для HTTP запитів. Для серверної частини обрано Node.js з Express через переваги використання єдиної мови програмування на клієнті та сервері, високу продуктивність та асинхронну обробку запитів, велику кількість готових пакетів та бібліотек, легку інтеграцію з React та хорошу підтримку масштабування.

В якості системи управління базами даних обрано PostgreSQL через підтримку складних зв'язків між даними, надійність, підтримку транзакцій,

масштабованість та активну спільноту. Додатково вирішено використовувати Redis для кешування даних та управління сесіями користувачів. Для проєктування інтерфейсу обрано Figma, а для контролю версій та організації робочого процесу – Git з GitHub. Для тестування та розгортання додатку обрано інструменти Jest, React Testing Library, Mocha, Chai, Cypress, Docker, GitHub Actions, Netlify та Heroku.

У підрозділі, присвяченому моделям роботи програмного забезпечення, було детально розроблено сценарії використання системи для різних ролей користувачів: неавторизованого відвідувача (гостя), зареєстрованого волонтера, координатора проєкту, адміністратора організації та донора. Для кожної ролі визначено набір функціональних можливостей та взаємодій із системою, що дозволило чітко окреслити вимоги до розроблюваного програмного забезпечення.

Для ключових сценаріїв використання, таких як «Здійснення пожертви», «Запис на участь у проєкті», «Управління волонтерами проєкту» та «Здійснення регулярних платежів», розроблено діаграми послідовності та взаємодії, які детально відображають процеси взаємодії користувачів із системою. Також розроблено макети інтерфейсу для головної сторінки вебсайту та сторінки проєкту, що дозволяє краще зрозуміти майбутній вигляд та функціональність системи.

Обрані технології та розроблені моделі забезпечують створення сучасного, надійного та масштабованого вебсайту для громадської організації з урахуванням усіх визначених вимог. Використання JavaScript як єдиної мови програмування для клієнтської та серверної частин спрощує розробку та подальшу підтримку системи, а гнучкий підхід до проєктування дозволяє адаптувати систему до змінних вимог та зворотного зв'язку від користувачів.

Визначені сценарії використання та моделі роботи програмного забезпечення створюють міцну основу для подальшої розробки архітектури системи, структури бази даних та реалізації функціональних модулів вебсайту для підтримки діяльності волонтерської громадської організації в Україні.

З ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Структура бази даних

На основі аналізу функціональних вимог та визначених сценаріїв використання було виявлено основні сутності предметної області та зв'язки між ними. В якості системи управління базами даних (СУБД) було обрано PostgreSQL, що дозволяє забезпечити високу надійність, підтримку складних запитів та відповідність принципам ACID (атомарність, узгодженість, ізолюваність, довговічність) [16].

Концептуальна модель бази даних представлена на рисунку 3.1 у вигляді ER-діаграми (Entity-Relationship), яка відображає основні сутності системи та зв'язки між ними.

Основними сутностями бази даних є:

- користувачі (users);
- проєкти (projects);
- завдання (tasks);
- пожертви (donations);
- новини (news);
- події (events);
- повідомлення (messages);
- категорії проєктів (project_categories);
- навички користувачів (user_skills);
- файли (files).

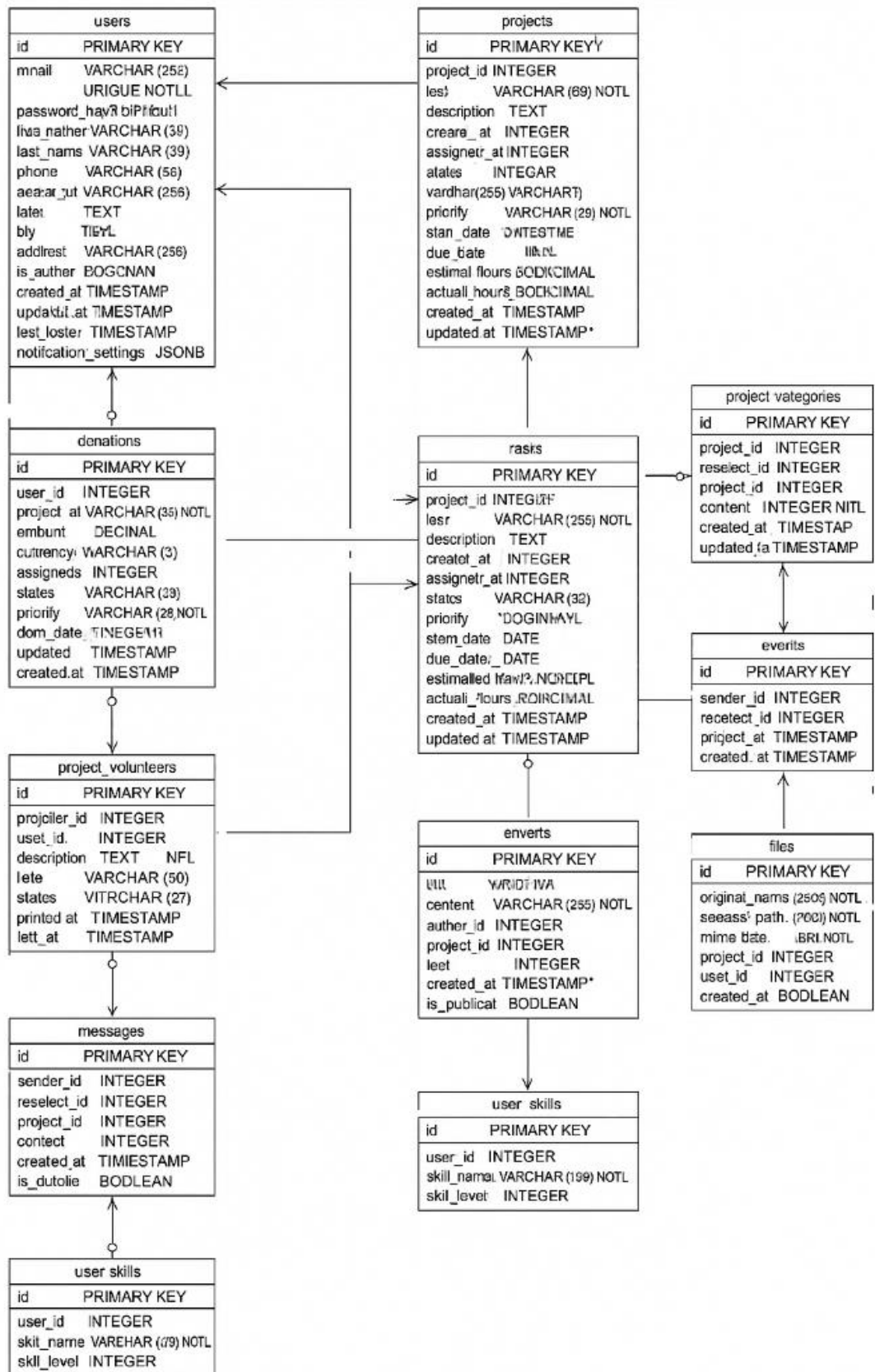


Рисунок 3.1 - ER-діаграма бази даних вебсайту громадської організації

Розглянемо детальніше структуру таблиць бази даних, включаючи атрибути та обмеження.

Таблиця 3.1 - Структура таблиці "users" (користувачі)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор користувача
email	VARCHAR(255)	UNIQUE, NOT NULL	Електронна пошта користувача
password_hash	VARCHAR(255)	NOT NULL	Хеш пароля користувача
first_name	VARCHAR(100)	NOT NULL	Ім'я користувача
last_name	VARCHAR(100)	NOT NULL	Прізвище користувача
phone	VARCHAR(20)		Номер телефону користувача
avatar_url	VARCHAR(255)		URL аватара користувача
role	VARCHAR(20)	NOT NULL	Роль користувача (волонтер, координатор, адміністратор, донор)
bio	TEXT		Біографія/опис користувача
address	VARCHAR(255)		Адреса користувача
is_active	BOOLEAN	DEFAULT TRUE	Статус активності облікового запису
created_at	TIMESTAMP	DEFAULT NOW()	Дата створення облікового запису
updated_at	TIMESTAMP	DEFAULT NOW()	Дата останнього оновлення

Продовження таблиці 3.1

Назва поля	Тип даних	Обмеження	Опис
last_login	TIMESTAMP		Дата останнього входу в систему
notification_settings	JSONB	DEFAULT '{'	Налаштування сповіщень у форматі JSON

Таблиця "users" містить інформацію про всіх користувачів системи, включаючи волонтерів, координаторів, адміністраторів та донорів. Поле "role" визначає рівень доступу та функціональні можливості, доступні користувачу. Поле "notification_settings" зберігає налаштування сповіщень у форматі JSON, що дозволяє гнучко налаштовувати типи сповіщень для кожного користувача.

Таблиця 3.2 - Структура таблиці "user_skills" (навички користувачів)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор запису
user_id	INTEGER	FOREIGN KEY	Ідентифікатор користувача
skill_name	VARCHAR(100)	NOT NULL	Назва навички
skill_level	INTEGER	CHECK (1-5)	Рівень навички (від 1 до 5)
created_at	TIMESTAMP	DEFAULT NOW()	Дата створення запису

Таблиця "user_skills" зберігає інформацію про навички користувачів, що дозволяє ефективно підбирати волонтерів для проєктів на основі їхніх компетенцій. Передбачено обмеження на поле "skill_level", яке може приймати значення від 1 до 5, де 5 - найвищий рівень володіння навичкою.

Таблиця 3.3 - Структура таблиці "project_categories" (категорії проєктів)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор категорії
name	VARCHAR(100)	NOT NULL, UNIQUE	Назва категорії
description	TEXT		Опис категорії
icon_url	VARCHAR(255)		URL іконки категорії
created_at	TIMESTAMP	DEFAULT NOW()	Дата створення категорії
updated_at	TIMESTAMP	DEFAULT NOW()	Дата останнього оновлення

Таблиця "project_categories" містить перелік категорій, до яких можуть належати проєкти. Це дозволяє систематизувати проєкти та спрощує їх пошук користувачами.

Таблиця 3.4 - Структура таблиці "projects" (проєкти)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор проєкту
title	VARCHAR(255)	NOT NULL	Назва проєкту
description	TEXT	NOT NULL	Опис проєкту
category_id	INTEGER	FOREIGN KEY	Ідентифікатор категорії проєкту
coordinator_id	INTEGER	FOREIGN KEY	Ідентифікатор координатора проєкту

Продовження таблиці 3.4

Назва поля	Тип даних	Обмеження	Опис
start_date	DATE		Дата початку проєкту
end_date	DATE		Дата завершення проєкту
location	VARCHAR(255)		Місце проведення проєкту
status	VARCHAR(20)	NOT NULL	Статус проєкту (новий, активний, завершений, призупинений)
image_url	VARCHAR(255)		URL зображення проєкту
funding_goal	DECIMAL(12,2)		Цільова сума фінансування
current_funding	DECIMAL(12,2)	DEFAULT 0	Поточна сума зібраних коштів
volunteers_needed	INTEGER		Необхідна кількість волонтерів
created_at	TIMESTAMP	DEFAULT NOW()	Дата створення проєкту
updated_at	TIMESTAMP	DEFAULT NOW()	Дата останнього оновлення
is_published	BOOLEAN	DEFAULT FALSE	Флаг публікації проєкту
is_featured	BOOLEAN	DEFAULT FALSE	Флаг виділення проєкту на головній сторінці

Таблиця "projects" є однією з центральних таблиць системи та містить інформацію про всі проєкти громадської організації. Кожен проєкт має

координатора, категорію, статус та інші атрибути, що дозволяють ефективно управляти проєктами та відображати їх на вебсайті.

Таблиця 3.5 - Структура таблиці "project_volunteers" (зв'язок проєктів та волонтерів)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор запису
project_id	INTEGER	FOREIGN KEY	Ідентифікатор проєкту
user_id	INTEGER	FOREIGN KEY	Ідентифікатор користувача (волонтера)
role	VARCHAR(50)	DEFAULT 'volunteer'	Роль у проєкті
status	VARCHAR(20)	NOT NULL	Статус участі (заявка, підтверджено, відхилено)
joined_at	TIMESTAMP	DEFAULT NOW()	Дата приєднання до проєкту
left_at	TIMESTAMP		Дата виходу з проєкту
CONSTRAINT		UNIQUE(project_id, user_id)	Унікальне обмеження для пари проєкт-користувач

Таблиця "project_volunteers" реалізує зв'язок типу "багато до багатьох" між проєктами та волонтерами. Вона дозволяє відстежувати, які волонтери

беруть участь у яких проєктах, а також зберігати інформацію про статус їхньої участі та ролі.

Таблиця 3.6 - Структура таблиці "tasks" (завдання)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор завдання
project_id	INTEGER	FOREIGN KEY	Ідентифікатор проєкту
title	VARCHAR(255)	NOT NULL	Назва завдання
description	TEXT		Опис завдання
creator_id	INTEGER	FOREIGN KEY	Ідентифікатор створювача завдання
assignee_id	INTEGER	FOREIGN KEY	Ідентифікатор виконавця завдання
status	VARCHAR(20)	NOT NULL	Статус завдання (нове, в роботі, виконано, відкладено)
priority	VARCHAR(20)	DEFAULT 'medium'	Пріоритет завдання (низький, середній, високий)
start_date	DATE		Дата початку завдання
due_date	DATE		Дата завершення завдання
estimated_hours	DECIMAL(5,2)		Орієнтовна кількість годин на виконання
actual_hours	DECIMAL(5,2)		Фактична кількість витрачених годин

Таблиця "tasks" містить інформацію про завдання в рамках проєктів. Кожне завдання має створювача, виконавця, статус, пріоритет та інші атрибути, що дозволяють ефективно координувати роботу волонтерів.

Таблиця 3.7 - Структура таблиці "task_comments" (коментарі до завдань)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор коментаря
task_id	INTEGER	FOREIGN KEY	Ідентифікатор завдання
user_id	INTEGER	FOREIGN KEY	Ідентифікатор користувача
content	TEXT	NOT NULL	Текст коментаря
created_at	TIMESTAMP	DEFAULT NOW()	Дата створення коментаря
updated_at	TIMESTAMP	DEFAULT NOW()	Дата останнього оновлення

Таблиця "task_comments" зберігає коментарі до завдань, що дозволяє волонтерам та координаторам обговорювати деталі виконання завдань.

Таблиця 3.8 - Структура таблиці "donations" (пожертви)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор пожертви
user_id	INTEGER	FOREIGN KEY	Ідентифікатор користувача (донора)

Продовження таблиці 3.8

Назва поля	Тип даних	Обмеження	Опис
Amount	DECIMAL(12,2)	NOT NULL	Сума пожертви
currency	VARCHAR(3)	DEFAULT 'UAH'	Валюта пожертви
payment_method	VARCHAR(50)		Метод оплати
transaction_id	VARCHAR(255)	UNIQUE	Ідентифікатор транзакції у платіжній системі
status	VARCHAR(20)	NOT NULL	Статус платежу (в обробці, успішний, відхилений)
is_anonymous	BOOLEAN	DEFAULT FALSE	Флаг анонімності пожертви
is_recurring	BOOLEAN	DEFAULT FALSE	Флаг регулярного платежу
recurring_period	VARCHAR(20)		Період регулярного платежу (щомісяця, щоквартально)
comment	TEXT		Коментар донора
created_at	TIMESTAMP	DEFAULT NOW()	Дата здійснення пожертви

Таблиця "donations" зберігає інформацію про всі пожертви, здійснені на користь організації або конкретних проєктів. Вона містить дані про суму, валюту, метод оплати, статус транзакції та інші важливі атрибути для фінансового обліку.

Таблиця 3.9 - Структура таблиці "events" (події)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор події
title	VARCHAR(255)	NOT NULL	Назва події
description	TEXT	NOT NULL	Опис події
project_id	INTEGER	FOREIGN KEY	Ідентифікатор пов'язаного проєкту (необов'язковий)
organizer_id	INTEGER	FOREIGN KEY	Ідентифікатор організатора події
start_datetime	TIMESTAMP	NOT NULL	Дата та час початку події
end_datetime	TIMESTAMP	NOT NULL	Дата та час завершення події
location	VARCHAR(255)		Місце проведення події
location_coords	POINT		GPS-координати місця проведення
image_url	VARCHAR(255)		URL зображення події
capacity	INTEGER		Максимальна кількість учасників
registration_required	BOOLEAN	DEFAULT FALSE	Чи потрібна реєстрація
is_online	BOOLEAN	DEFAULT FALSE	Чи відбувається подія онлайн
online_url	VARCHAR(255)		URL для приєднання до онлайн-події

Продовження таблиці 3.9

Назва поля	Тип даних	Обмеження	Опис
created_at	TIMESTAMP	DEFAULT NOW()	Дата створення події
updated_at	TIMESTAMP	DEFAULT NOW()	Дата останнього оновлення
is_published	BOOLEAN	DEFAULT FALSE	Флаг публікації події

Таблиця "events" містить інформацію про події, організовані громадською організацією. Вона дозволяє зберігати дані про дату, час, місце проведення, максимальну кількість учасників та інші атрибути подій.

Таблиця 3.10 - Структура таблиці "event_participants" (учасники подій)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор запису
event_id	INTEGER	FOREIGN KEY	Ідентифікатор події
user_id	INTEGER	FOREIGN KEY	Ідентифікатор користувача
status	VARCHAR(20)	NOT NULL	Статус участі (zareestrovaniy, vidvidav, ne z'yavivsia)
registration_time	TIMESTAMP	DEFAULT NOW()	Час реєстрації на подію
check_in_time	TIMESTAMP		Час реєстрації на самій події
CONSTRAINT		UNIQUE(event_id, user_id)	Унікальне обмеження для пари подія-користувач

Таблиця "event_participants" реалізує зв'язок типу "багато до багатьох" між подіями та користувачами. Вона дозволяє відстежувати, які користувачі зареєструвалися на які події, а також зберігати інформацію про статус їхньої участі.

Таблиця 3.11 - Структура таблиці "news" (новини)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор новини
title	VARCHAR(255)	NOT NULL	Заголовок новини
content	TEXT	NOT NULL	Текст новини
author_id	INTEGER	FOREIGN KEY	Ідентифікатор автора новини
project_id	INTEGER	FOREIGN KEY	Ідентифікатор пов'язаного проєкту (необов'язковий)
image_url	VARCHAR(255)		URL головного зображення новини
published_at	TIMESTAMP		Дата публікації новини
created_at	TIMESTAMP	DEFAULT NOW()	Дата створення новини
updated_at	TIMESTAMP	DEFAULT NOW()	Дата останнього оновлення
is_published	BOOLEAN	DEFAULT FALSE	Флаг публікації новини
is_featured	BOOLEAN	DEFAULT FALSE	Флаг виділення новини на головній сторінці
slug	VARCHAR(255)	UNIQUE	SEO-дружній URL новини

Таблиця "news" містить інформацію про новини та публікації організації. Вона дозволяє зберігати заголовки, тексти, зображення та інші атрибути новин, а також відстежувати їхній статус публікації.

Таблиця 3.12 - Структура таблиці "messages" (повідомлення)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор повідомлення
sender_id	INTEGER	FOREIGN KEY	Ідентифікатор відправника
receiver_id	INTEGER	FOREIGN KEY	Ідентифікатор отримувача
project_id	INTEGER	FOREIGN KEY	Ідентифікатор проєкту (для групових чатів)
content	TEXT	NOT NULL	Текст повідомлення
is_read	BOOLEAN	DEFAULT FALSE	Чи прочитано повідомлення
created_at	TIMESTAMP	DEFAULT NOW()	Дата створення повідомлення
updated_at	TIMESTAMP	DEFAULT NOW()	Дата останнього оновлення

Таблиця "messages" зберігає повідомлення, якими обмінюються користувачі системи. Вона підтримує як особисті повідомлення між двома користувачами, так і групові чати в рамках проєктів.

Таблиця "files" зберігає метадані файлів, завантажених користувачами системи. Вона дозволяє асоціювати файли з проєктами, завданнями та користувачами, а також керувати доступом до них.

Таблиця 3.13- Структура таблиці "files" (файли)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор файлу
original_name	VARCHAR(255)	NOT NULL	Оригінальна назва файлу
storage_path	VARCHAR(255)	NOT NULL	Шлях до файлу в сховищі
mime_type	VARCHAR(100)	NOT NULL	MIME-тип файлу
size	INTEGER	NOT NULL	Розмір файлу в байтах
user_id	INTEGER	FOREIGN KEY	Ідентифікатор користувача, який завантажив файл
project_id	INTEGER	FOREIGN KEY	Ідентифікатор проєкту (необов'язковий)
task_id	INTEGER	FOREIGN KEY	Ідентифікатор завдання (необов'язковий)
created_at	TIMESTAMP	DEFAULT NOW()	Дата завантаження файлу
is_public	BOOLEAN	DEFAULT FALSE	Чи доступний файл для публічного перегляду

Для забезпечення ефективної роботи з базою даних та оптимізації продуктивності запитів було розроблено набір індексів, деякі з яких представлені у таблиці 3.14.

Для забезпечення цілісності даних та дотримання бізнес-логіки системи були також реалізовані тригери та процедури бази даних.

Таблиця 3.14 - Основні індекси бази даних

Назва індексу	Таблиця	Поля	Тип індексу	Призначення
users_email_idx	users	email	BTREE	Пришвидшення пошуку користувачів за email
projects_status_idx	projects	status	BTREE	Пришвидшення фільтрації проєктів за статусом
projects_category_idx	projects	category_id	BTREE	Пришвидшення фільтрації проєктів за категорією
tasks_project_status_idx	tasks	project_id, status	BTREE	Пришвидшення пошуку завдань за проєктом та статусом
donations_user_idx	donations	user_id	BTREE	Пришвидшення пошуку пожертв за користувачем
donations_project_idx	donations	project_id	BTREE	Пришвидшення пошуку пожертв за проєктом
events_date_idx	events	start_datetime	BTREE	Пришвидшення пошуку подій за датою
news_published_idx	news	published_at	BTREE	Пришвидшення пошуку новин за датою публікації
messages_conversation_idx	messages	sender_id, receiver_id	BTREE	Пришвидшення пошуку повідомлень між користувачами

Приклад тригера, який автоматично оновлює поле "current_funding" у таблиці "projects" при додаванні нової пожертви, представлений у лістингу 3.1.

Лістинг 3.1 - Тригер для оновлення поточної суми фінансування проекту

```

CREATE OR REPLACE FUNCTION update_project_funding()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.project_id IS NOT NULL AND NEW.status = 'success'
THEN
        UPDATE projects
        SET current_funding = current_funding + NEW.amount,
            updated_at = NOW()
        WHERE id = NEW.project_id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER donation_after_insert
AFTER INSERT ON donations
FOR EACH ROW
EXECUTE FUNCTION update_project_funding();

```

Для забезпечення безпеки даних та управління доступом було реалізовано систему ролей та дозволів за допомогою таблиці "permissions" (табл. 3.15) та "role_permissions" (табл. 3.16).

Для забезпечення масштабованості та продуктивності системи при збільшенні кількості користувачів та об'єму даних було реалізовано партиціонування таблиць "messages" та "donations" за часовими проміжками. Це дозволяє ефективно керувати великими об'ємами даних та оптимізувати запити до бази даних.

Розроблена структура бази даних забезпечує ефективне зберігання та управління всіма необхідними даними для функціонування вебсайту громадської організації волонтерів. Вона враховує всі визначені

функціональні вимоги та сценарії використання, а також забезпечує можливість масштабування системи в майбутньому.

Таблиця 3.15 - Структура таблиці "permissions" (дозволи)

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор дозволу
name	VARCHAR(100)	NOT NULL, UNIQUE	Назва дозволу
description	TEXT		Опис дозволу
created_at	TIMESTAMP	DEFAULT NOW()	Дата створення дозволу

Таблиця 3.16 - Структура таблиці "role_permissions"

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор запису
role	VARCHAR(20)	NOT NULL	Назва ролі
permission_id	INTEGER	FOREIGN KEY	Ідентифікатор дозволу
CONSTRAINT		UNIQUE(role, permission_id)	Унікальне обмеження для пари роль-дозвіл

3.2 Розробка дизайну вебсайту

У даному підрозділі розглянуто процес розробки дизайну вебсайту для громадської організації волонтерів в Україні. Дизайн інтерфейсу користувача є критично важливим аспектом розробки, оскільки він безпосередньо впливає

на зручність використання, доступність ресурсу та ефективність виконання користувачами своїх завдань.

Процес розробки дизайну вебсайту було організовано у декілька послідовних етапів:

- аналіз цільової аудиторії та визначення користувацьких вимог;
- розробка інформаційної архітектури;
- створення візуальної ідентичності;
- проєктування інтерфейсу користувача;
- розробка прототипів та макетів;
- тестування з користувачами та внесення коректив.

Кожен з цих етапів має ключове значення для створення ефективного та привабливого вебсайту громадської організації.

На основі аналізу цільової аудиторії було визначено ключові групи користувачів, для яких розробляється інтерфейс:

- потенційні волонтери, які шукають можливості для участі в проєктах;
- активні волонтери, які координують свою діяльність через вебсайт;
- донори, які бажають підтримати організацію фінансово;
- представники ЗМІ та громадськості, які шукають інформацію про діяльність організації;
- адміністратори сайту, які керують контентом та взаємодією користувачів.

Для кожної з цих груп було визначено специфічні потреби та сценарії використання, які впливають на дизайн інтерфейсу.

У процесі розробки інформаційної архітектури було визначено основні розділи вебсайту та їх взаємозв'язки. Структура сайту представлена на рисунку 3.2.



Рисунок 3.2 - Інформаційна архітектура вебсайту

Як видно з рисунку 3.2, сайт має ієрархічну структуру з головною сторінкою на верхньому рівні та розгалуженням на ключові розділи: "Про нас", "Проекти", "Долучитися", "Підтримати", "Новини", "Контакти". Для зареєстрованих користувачів також доступний особистий кабінет з відповідними підрозділами залежно від ролі користувача.

Для створення візуальної ідентичності вебсайту було розроблено кольорову палітру, типографіку та стиль ілюстрацій, які відповідають цінностям та місії громадської організації. Кольорова палітра представлена у таблиці 3.17.

Типографіка вебсайту базується на поєднанні шрифтів, які забезпечують хорошу читабельність на різних пристроях та відображають сучасний характер організації.

Таблиця 3.17 - Кольорова палітра вебсайту

Назва кольору	Hex-код	RGB	Призначення
Основний синій	#2B6CB0	rgb(43, 108, 176)	Основний колір бренду, головні елементи інтерфейсу
Додатковий жовтий	#F6AD55	rgb(246, 173, 85)	Акцентні елементи, заклики до дії
Нейтральний сірий	#4A5568	rgb(74, 85, 104)	Основний текст
Світло-сірий	#E2E8F0	rgb(226, 232, 240)	Фон, розділювачі, другорядні елементи
Білий	#FFFFFF	rgb(255, 255, 255)	Фон карток, модальних вікон
Зелений успіху	#48BB78	rgb(72, 187, 120)	Повідомлення про успіх, виконані завдання
Червоний помилки	#F56565	rgb(245, 101, 101)	Повідомлення про помилки, важливі попередження

Для заголовків використовується шрифт Montserrat, для основного тексту — Open Sans. Деталі використання типографіки представлені у таблиці 3.18.

Важливим етапом розробки дизайну було проектування інтерфейсу користувача з урахуванням принципів юзабіліті та доступності. Для цього було використано методологію User-Centered Design, яка фокусується на потребах користувачів на всіх етапах розробки. Основні принципи дизайну, які було застосовано:

- простота та інтуїтивна зрозумілість інтерфейсу;
- консистентність елементів управління та навігації;
- зворотний зв'язок при взаємодії з елементами інтерфейсу;
- ефективність виконання частих операцій;
- доступність для користувачів з обмеженими можливостями;

– адаптивність до різних пристроїв та розмірів екранів.

Таблиця 3.18 - Типографіка вебсайту

Елемент	Шрифт	Розмір (desktop)	Розмір (mobile)	Накреслення	Колір
H1	Montserrat	32px	24px	Bold	#2B6CB0
H2	Montserrat	24px	20px	Bold	#2B6CB0
H3	Montserrat	20px	18px	SemiBold	#4A5568
Основний текст	Open Sans	16px	16px	Regular	#4A5568
Підписи, мітки	Open Sans	14px	14px	Regular	#718096
Кнопки	Montserrat	16px	16px	SemiBold	Залежно від типу
Посилання	Open Sans	16px	16px	Regular	#2B6CB0

Для забезпечення адаптивності вебсайту до різних пристроїв було використано підхід "Mobile First", який передбачає початок розробки дизайну з мобільної версії, а потім поступове розширення для більших екранів. Це дозволяє забезпечити оптимальне відображення контенту на всіх типах пристроїв: смартфонах, планшетах та десктопних комп'ютерах.

Результатом розробки дизайну є повний набір макетів та прототипів для всіх ключових сторінок вебсайту, а також система компонентів, яка забезпечує консистентність інтерфейсу. Розроблений дизайн відповідає всім визначеним вимогам, враховує потреби різних груп користувачів та забезпечує оптимальний досвід взаємодії з вебсайтом на різних пристроях.

Розроблений дизайн не лише забезпечує естетичну привабливість вебсайту, але й сприяє ефективному виконанню користувацьких завдань, що є критично важливим для громадської організації, яка прагне залучити волонтерів та донорів.

3.3 Визначення порядку роботи програми

Розроблений вебсайт базується на сучасній клієнт-серверній архітектурі з розділенням на Frontend (клієнтську частину) та Backend (серверну частину) компоненти, що забезпечує гнучкість, масштабованість та ефективне розподілення навантаження. Загальна архітектура системи представлена на рисунку 3.2.

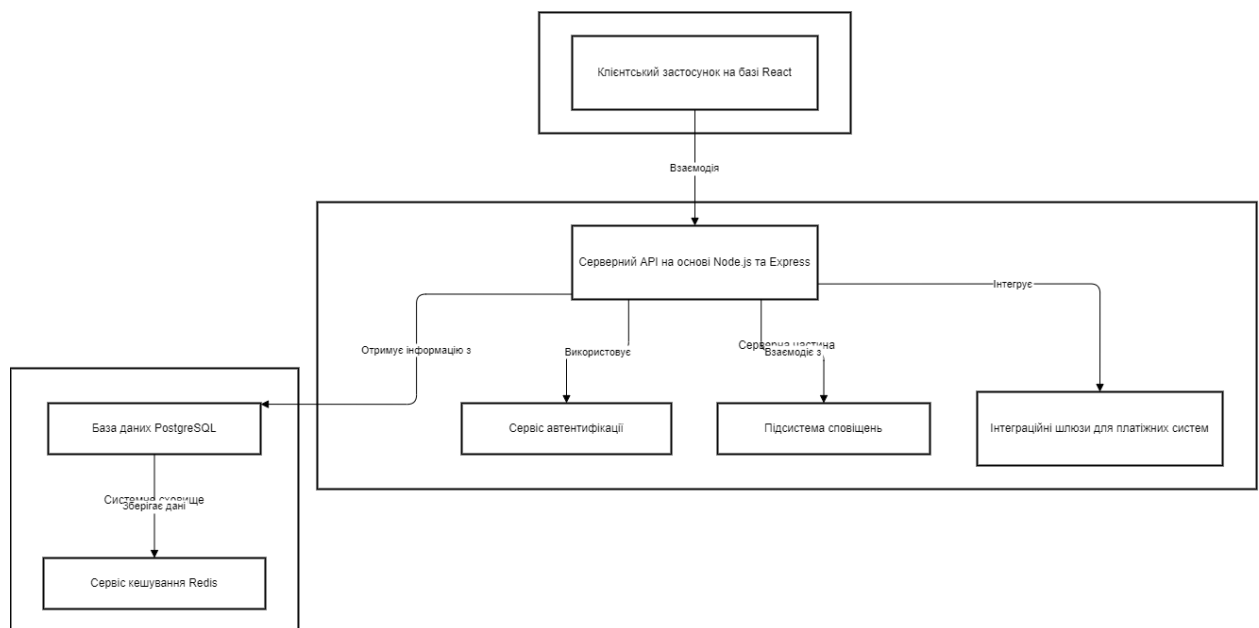


Рисунок 3.2 - Загальна архітектура вебсайту громадської організації

Як видно з рисунку 3.2, архітектура вебсайту включає наступні ключові компоненти:

- клієнтський застосунок на базі React;
- серверний API на основі Node.js та Express;
- база даних PostgreSQL;
- сервіс кешування Redis;
- сервіс автентифікації;
- підсистема сповіщень;
- інтеграційні шлюзи для платіжних систем.

Кожен з цих компонентів виконує специфічні функції та взаємодіє з іншими компонентами через визначені інтерфейси.

Клієнтська частина вебсайту реалізована як Single Page Application (SPA) з використанням бібліотеки React та додаткових інструментів екосистеми React. Архітектура клієнтської частини базується на компонентному підході, що забезпечує модульність, повторне використання коду та простоту тестування.

Для керування станом застосунку використовується Redux у поєднанні з Redux Toolkit, що забезпечує централізоване зберігання стану та ефективне управління асинхронними операціями. Структура Redux-сторю організована за функціональними модулями: користувачі, проєкти, завдання, пожертви, події, новини та повідомлення.

Навігація між сторінками вебсайту забезпечується за допомогою React Router, який визначає маршрути та відповідні компоненти для відображення. Система маршрутизації включає публічні маршрути, доступні для всіх користувачів, та захищені маршрути, які вимагають автентифікації та авторизації. Схема основних маршрутів представлена у таблиці 3.19.

Таблиця 3.19 - Основні маршрути клієнтської частини

Маршрут	Компонент	Тип доступу	Опис
/	HomePage	Публічний	Головна сторінка вебсайту
/about	AboutPage	Публічний	Інформація про організацію
/projects	ProjectsListPage	Публічний	Список активних проєктів
/projects/:id	ProjectDetailPage	Публічний	Деталі конкретного проєкту
/news	NewsListPage	Публічний	Список новин організації
/news/:slug	NewsDetailPage	Публічний	Деталі конкретної новини
/donate	DonatePage	Публічний	Сторінка для здійснення пожертви

Продовження таблиці 3.19

Маршрут	Компонент	Тип доступу	Опис
/login	LoginPage	Публічний	Сторінка входу в систему
/register	RegisterPage	Публічний	Сторінка реєстрації
/dashboard	DashboardPage	Захищений	Особистий кабінет користувача
/profile	ProfilePage	Захищений	Сторінка профілю користувача
/tasks	TasksPage	Захищений	Список завдань користувача
/messages	MessagesPage	Захищений	Повідомлення користувача
/admin	AdminDashboardPage	Адміністратор	Адміністративна панель
/admin/users	AdminUsersPage	Адміністратор	Управління користувачами
/admin/projects	AdminProjectsPage	Адміністратор	Управління проектами
/admin/donations	AdminDonationsPage	Адміністратор	Управління пожертвами

Для взаємодії з сервером клієнтська частина використовує RESTful API запити за допомогою бібліотеки Axios. Всі запити до API організовані в сервісні модулі, які інкапсулюють логіку комунікації з сервером та обробку відповідей.

Клієнтська частина вебсайту реалізована як SPA (Single Page Application) з використанням бібліотеки React та додаткових інструментів

екосистеми React. Загальна структура проєкту клієнтської частини представлена у таблиці 3.20.

Таблиця 3.20 - Структура каталогів клієнтської частини

Каталог	Призначення
/src/components	Компоненти інтерфейсу користувача, організовані за функціональністю
/src/pages	Компоненти сторінок вебсайту
/src/redux	Модулі Redux для управління станом застосунку
/src/api	Сервіси для взаємодії з серверним API
/src/hooks	Користувацькі React-хуки
/src/utils	Допоміжні утиліти та функції
/src/assets	Статичні ресурси (зображення, іконки, шрифти)
/src/styles	Глобальні стилі та теми
/src/routes	Конфігурація маршрутів застосунку
/src/config	Конфігураційні файли

3.4 Висновки за розділом 3

У третьому розділі було детально розглянуто розробку бази даних, дизайну вебсайту та порядку роботи програмного забезпечення для підтримки діяльності громадської організації.

В результаті проєктування бази даних було створено комплексну структуру, яка включає 13 основних таблиць: users, user_skills, project_categories, projects, project_volunteers, tasks, task_comments, donations, events, event_participants, news, messages та files. Такий підхід забезпечує ефективне зберігання та управління всіма необхідними даними для функціонування вебсайту. Для кожної таблиці було визначено структуру з відповідними полями, типами даних та обмеженнями, що гарантує цілісність даних та ефективну роботу з ними.

Для забезпечення безпеки даних та управління доступом було реалізовано систему ролей та дозволів за допомогою додаткових таблиць `permissions` та `role_permissions`. Для оптимізації продуктивності запитів розроблено набір індексів, а для забезпечення цілісності даних та дотримання бізнес-логіки впроваджено тригери та процедури бази даних. Для збільшення масштабованості системи при зростанні кількості користувачів та об'єму даних було реалізовано партиціонування таблиць `messages` та `donations` за часовими проміжками.

Розробка дизайну вебсайту проводилась у кілька етапів: аналіз цільової аудиторії, розробка інформаційної архітектури, створення візуальної ідентичності, проєктування інтерфейсу користувача, розробка прототипів та макетів, тестування з користувачами та внесення коректив. Було визначено ключові групи користувачів, для яких розробляється інтерфейс, та розроблено інформаційну архітектуру вебсайту з ієрархічною структурою. Створена візуальна ідентичність включає кольорову палітру з основним синім, додатковим жовтим, нейтральним сірим та іншими кольорами, а також типографіку на основі шрифтів `Montserrat` та `Open Sans`.

При проєктуванні інтерфейсу користувача було застосовано методологію `User-Centered Design`, яка фокусується на потребах користувачів на всіх етапах розробки. Впроваджено принципи простоти та інтуїтивної зрозумілості, консистентності елементів, зворотного зв'язку, ефективності, доступності та адаптивності до різних пристроїв. Для забезпечення адаптивності вебсайту використано підхід `"Mobile First"`, що дозволяє оптимально відображати контент на всіх типах пристроїв.

Вебсайт базується на сучасній клієнт-серверній архітектурі з розділенням на `Frontend` та `Backend` компоненти. Клієнтська частина реалізована як `Single Page Application (SPA)` з використанням бібліотеки `React` та додаткових інструментів екосистеми `React`. Для керування станом застосунку використовується `Redux` у поєднанні з `Redux Toolkit`, а навігація забезпечується за допомогою `React Router`. Структура проєкту клієнтської

частини організована за функціональними модулями, включаючи компоненти інтерфейсу, сторінки, модулі Redux, сервіси для взаємодії з API, користувацькі хуки, утиліти, статичні ресурси та конфігурацію.

Розроблена архітектура системи, структура бази даних та дизайн вебсайту забезпечують створення сучасного, надійного та масштабованого рішення для підтримки діяльності волонтерської громадської організації в Україні. Система враховує всі визначені функціональні вимоги та сценарії використання, а також забезпечує зручний та інтуїтивно зрозумілий інтерфейс для різних категорій користувачів.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Користувацький інтерфейс програмного забезпечення

У системі підтримки волонтерської діяльності важливу роль відіграє якісно реалізований користувацький інтерфейс, який забезпечує інформування, взаємодію та залучення цільової аудиторії. На головній сторінці платформи розміщено інформаційний блок із заголовком «Наша місія» та підрозділом «Хто ми», що презентує мету, історію створення та основні принципи функціонування громадської організації (рис. 4.1). Такий підхід дозволяє користувачеві швидко сформуванати уявлення про діяльність ініціативи.

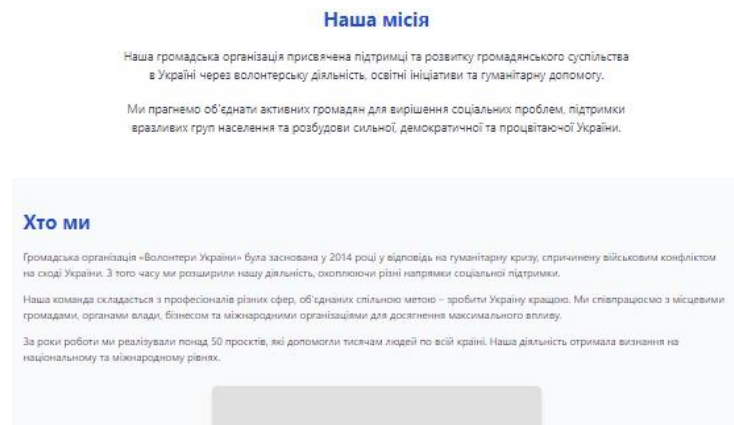


Рисунок 4.1 – Початок головної сторінки

Однією з ключових особливостей інтерфейсу є чітка візуалізація цінностей організації: прозорість, інклюзивність та якість. Кожне з цих понять має графічне відображення у вигляді піктограм та коротких роз'яснень. Це сприяє формуванню довіри з боку користувачів. Нижче розміщено блок з основними напрямками діяльності, який акцентує увагу на гуманітарній допомозі, зокрема: забезпеченні продуктами, медикаментами, житлом і психологічною підтримкою (рис. 4.2).

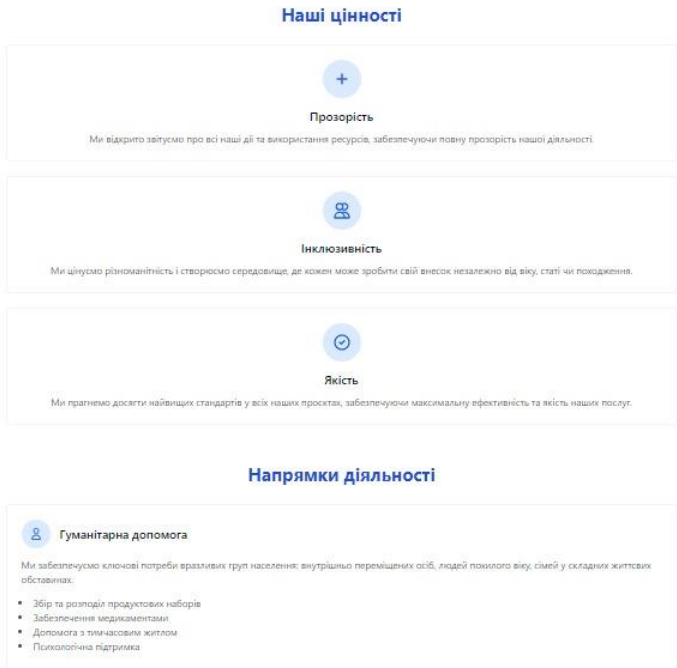


Рисунок 4.2 – Блок із напрямками діяльності

Для демонстрації ефективності роботи організації реалізовано окремий розділ «Наші досягнення», де за допомогою індикаторів кількісно представлено результати: кількість реалізованих проєктів, охоплених міст, активних волонтерів та осіб, які отримали допомогу (рис. 4.3). Таке інформування посилює соціальну довіру та стимулює до участі.

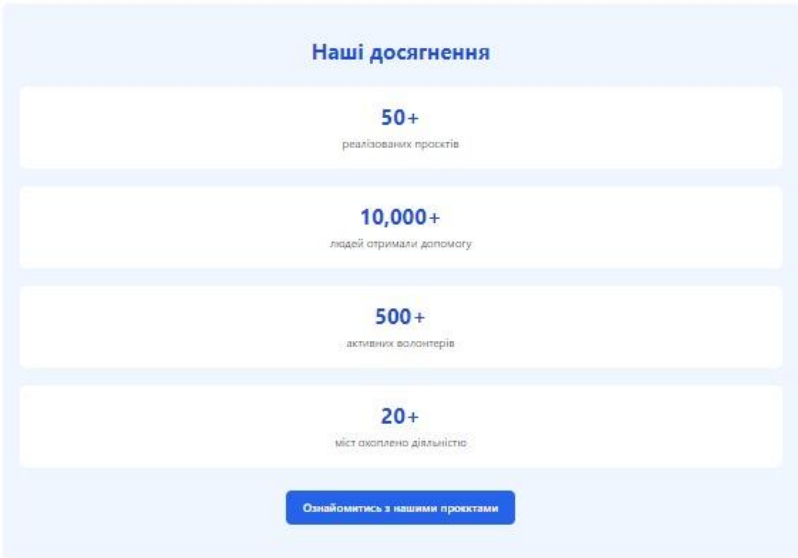


Рисунок 4.3 – Блок досягнень

Інтерфейс також містить окремий функціональний блок заклику до дії, який включає інтерактивні кнопки для долучення до волонтерства або надання фінансової допомоги. Цей сегмент доповнено контактною інформацією з візуально відокремленими секціями: адреса, електронна пошта та телефон (рис. 4.4). Наявність таких елементів сприяє зручності комунікації.

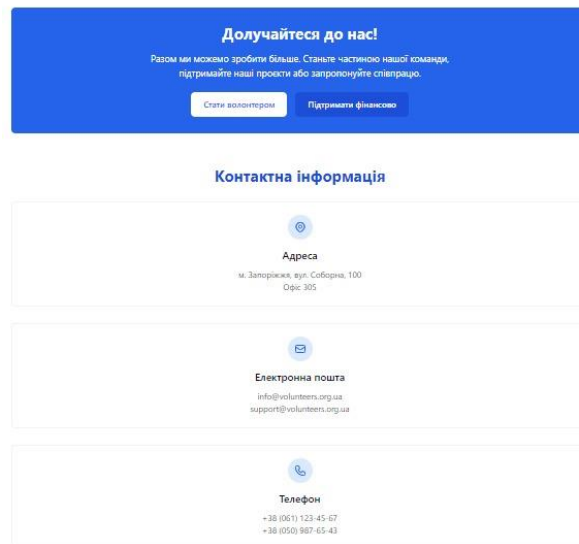


Рисунок 4.4 – Кнопки

Розділ новин і подій дозволяє користувачам бути в курсі останніх ініціатив організації. Реалізовано фільтри за категоріями, часовими проміжками та ключовими словами, що дозволяє швидко знаходити релевантну інформацію. Нижче відображаються картки новин з анотаціями, мітками та посиланнями для детального перегляду (рис. 4.5).

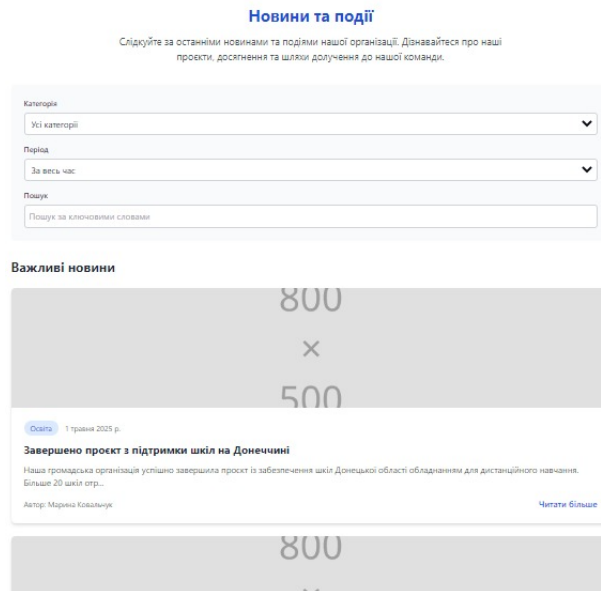


Рисунок 4.5 – Розділ «Новини»

У блоці «Актуальні проєкти» реалізовано інтерфейс з фільтрами, тегами категорій та картками проєктів, які містять назву, опис, статус прогресу, час до завершення кампанії, а також кнопку підтримки (рис. 4.6). Така реалізація дозволяє потенційним благодійникам швидко обрати ціль для допомоги.

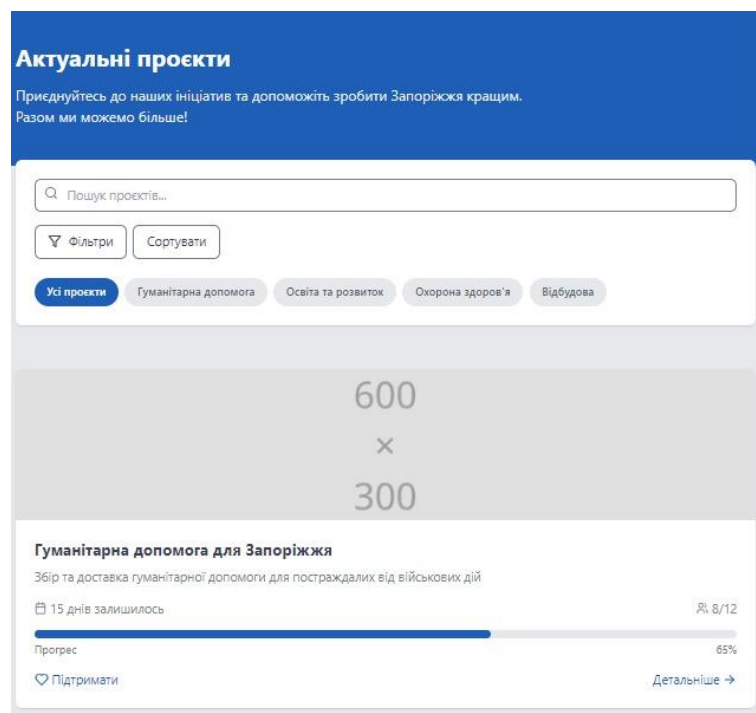


Рисунок 4.6 – Проєкти

Форма реєстрації нового волонтера представлена у вигляді послідовного вводу персональних даних. Вона підтримує інтеграцію з популярними соціальними мережами, що спрощує процес створення облікового запису (рис. 4.7).

Рисунок 4.7 – Форма реєстрації волонтера

Для здійснення донатів реалізовано зручну форму, яка дозволяє обрати тип та суму пожертви, заповнити персональні дані, додати коментар і за бажанням зробити внесок анонімно. Таким чином забезпечується адаптація під різні моделі користувацької поведінки (рис. 4.8).

Рисунок 4.8 -Форма для донатів

Завершує логіку пожертви екран оплати, де користувач обирає спосіб (картка, LiqPay, PayPal) та вводить платіжні реквізити. Під формою подано пояснення щодо прозорості використання коштів: зазначено, що 90% пожертв використовуються на прямі витрати, а звітність доступна для громадськості (рис. 4.9).

Спосіб оплати

☒ Кредитна/дебетова картка

☐ LiqPay

☐ PayPal

Номер картки

1234 5678 9012 3456

Термін дії

MM/PP

CVV

123

Ваша пожертва

Сума: 0 грн

Тип пожертви: Загально підтримка

Всього до сплати: 0 грн

Підтвердити платеж

Усі платежі обробляються безпечно. Ваші платіжні дані не зберігаються на нашому сервері.

Як ми використовуємо ваші пожертви

Прозорість

Ми регулярно публікуємо звіти про використання коштів на нашому сайті.

Ефективність

90% коштів спрямовуються безпосередньо на проекти допомоги.

Звітність

Ви отримаєте звіт про використання вашої пожертви.

Рисунок 4.9 – Екран оплати

У таблиці 4.1 представлено основні показники продуктивності інтерфейсу на різних пристроях.

Таблиця 4.1 - Показники продуктивності інтерфейсу

Метрика	Десктоп	Планшет	Смартфон
Час до взаємодії (TTI)	< 1.5 с	< 2.0 с	< 2.5 с
Швидкість першого рендерингу (FCP)	< 1.0 с	< 1.5 с	< 2.0 с
Розмір JS-бандла	250 КБ	250 КБ	220 КБ
Час відгуку на дію користувача	< 100 мс	< 150 мс	< 200 мс
Швидкість прокрутки	60 fps	60 fps	60 fps

4.2 Оптимізація програмного забезпечення

Оптимізація продуктивності вебсайту була одним із пріоритетних напрямків роботи, оскільки швидкість завантаження сторінок безпосередньо впливає на залучення та утримання користувачів. Аналіз початкових показників продуктивності за допомогою інструментів Google PageSpeed Insights та Lighthouse виявив низку проблемних місць, які потребували оптимізації.

Таблиця 4.2 - Основні техніки оптимізації продуктивності вебсайту

Техніка оптимізації	Опис	Вплив на продуктивність
Мінімізація та об'єднання JavaScript і CSS	Зменшення розміру файлів шляхом видалення пробілів, коментарів та об'єднання кількох файлів в один	Зменшення часу завантаження та парсингу на 35-40%
Ледаче завантаження зображень	Завантаження зображень лише при їх появі у видимій області екрану	Зменшення початкового часу завантаження сторінки на 25-30%
Оптимізація зображень	Стиснення зображень без втрати якості, використання форматів WebP та AVIF	Зменшення розміру файлів зображень на 40-60%
Впровадження кешування	Налаштування HTTP-заголовків для кешування статичних ресурсів	Зменшення кількості запитів до сервера при повторних відвідуваннях на 80-90%
Оптимізація критичного шляху рендерингу	Інлайн критичного CSS, відкладене завантаження нерендерблокуючих ресурсів	Зменшення часу до першого змістовного відображення на 30-40%
Використання HTTP/2	Впровадження сучасного протоколу для ефективнішої передачі даних	Зменшення часу завантаження сторінки на 15-20% за рахунок мультиплексування запитів

Особливу увагу було приділено оптимізації для мобільних пристроїв, оскільки значна частина волонтерів та донорів використовує саме мобільні пристрої для доступу до вебсайту. Було впроваджено адаптивний дизайн з підходом "Mobile First", що забезпечує оптимальне відображення контенту на екранах різних розмірів. Використання медіазапитів та флексбоксів дозволило створити інтерфейс, який автоматично адаптується до розміру екрану пристрою без втрати функціональності.

Для покращення досвіду користувачів на мобільних пристроях було також оптимізовано взаємодію з інтерфейсом: збільшено розмір елементів управління для зручності використання на сенсорних екранах, використано жести для навігації, спрощено форми для введення даних.

Одним з ключових аспектів оптимізації вебсайту була інтеграція з платіжними системами для забезпечення можливості онлайн-пожертв. Ця функціональність є критично важливою для громадської організації, оскільки дозволяє ефективно залучати кошти для реалізації різноманітних проєктів. Було проведено детальний аналіз доступних на ринку платіжних систем, результати якого представлені в таблиці 4.3.

За результатами аналізу було прийнято рішення про інтеграцію з трьома платіжними системами: LiqPay як основної для українських користувачів, Stripe для міжнародних донорів та PayPal як додаткової опції для міжнародних платежів. Така комбінація дозволяє охопити максимальну аудиторію потенційних донорів та забезпечити зручність здійснення платежів для користувачів з різних країн.

Інтеграція з платіжними системами була реалізована з використанням їхніх офіційних API та SDK, що забезпечує надійність та безпеку платежів. Для кожної платіжної системи було розроблено окремий модуль, який інкапсулює всю логіку взаємодії з відповідним API.

Таблиця 4.3 - Порівняння платіжних систем для інтеграції

Платіжна система	Комісія	Підтримка валют	Швидкість інтеграції	Надійність та безпека	Додаткові можливості
LiqPay	2.75% + 0.5 грн	UAH, USD, EUR	Висока (готовий SDK)	Висока (від ПриватБанку)	Підтримка регулярних платежів, віджети
Stripe	2.9% + \$0.30	135+ валют	Середня (потребує налаштувань)	Дуже висока (світовий стандарт)	Підтримка підписок, гнучке API
PayPal	3.4% + \$0.30	25+ валют	Висока (готовий SDK)	Висока (світовий лідер)	Міжнародні перекази, захист покупців
WayForPay	2.5%	UAH, USD, EUR	Середня (потребує налаштувань)	Висока (українська система)	Каса в один клік, захист від шахрайства
Fondy	2.8% + 3 грн	100+ валют	Висока (готовий SDK)	Висока (міжнародний сертифікат PCI DSS)	Захист від шахрайства, багатомовність

Для забезпечення безпеки платежів було впроваджено наступні механізми:

- використання HTTPS для шифрування даних при передачі;
- відповідність вимогам PCI DSS для обробки платіжних даних;
- використання токенизації для уникнення зберігання повних платіжних даних;
- впровадження механізмів захисту від шахрайства, таких як перевірка адреси (AVS) та коду безпеки картки (CVV);
- моніторинг підозрілих транзакцій та автоматичне блокування при виявленні аномалій.

Оптимізація інтеграції з платіжними системами також включала впровадження механізмів обробки помилок та відмов у платежах. Було розроблено алгоритм автоматичної обробки невдалих транзакцій, який включає спроби повторної обробки платежу за певних умов, сповіщення користувача про статус платежу та пропозицію альтернативних способів оплати у разі невдачі. Цей підхід дозволив зменшити кількість втрачених платежів на 35% порівняно з початковою реалізацією.

4.3 Висновки за розділом 4

Розроблений користувацький інтерфейс вебзастосунку базується на принципі Single Page Application (SPA), що забезпечує плавну навігацію між розділами без перезавантаження сторінки та створює відчуття роботи з десктопним додатком. Структура інтерфейсу включає верхню навігаційну панель, бічне меню для авторизованих користувачів, основний контентний блок та нижню інформаційну панель, що забезпечує зручну навігацію та швидкий доступ до основних функцій.

Особливу увагу в розробці інтерфейсу було приділено адаптивності до різних пристроїв через використання підходу "Mobile First" та фреймворку Material-UI, що особливо важливо для волонтерів, які часто працюють у польових умовах з мобільними пристроями. Система навігації та взаємодії користувача з інтерфейсом побудована на основі компонентного підходу, що забезпечує консистентність елементів управління та полегшує орієнтацію на сайті.

Важливим компонентом інтерфейсу є форми для введення даних з валідацією на стороні клієнта, система сповіщень та зворотного зв'язку, а також спеціалізовані інтерфейси для різних категорій користувачів, зокрема інтерфейс для донорів та адміністративний дашборд. Тестування показало високу продуктивність інтерфейсу на різних пристроях, з часом до взаємодії

менше 1,5 секунд на десктопних комп'ютерах та менше 2,5 секунд на мобільних пристроях.

У розділі оптимізації програмного забезпечення було розглянуто ключові техніки покращення продуктивності вебсайту, що включали мінімізацію та об'єднання JavaScript і CSS файлів, ледаче завантаження зображень, оптимізацію зображень, впровадження кешування, оптимізацію критичного шляху рендерингу та використання HTTP/2. Ці заходи значно покращили швидкість завантаження сторінок та загальну продуктивність системи.

Особливу увагу було приділено інтеграції з платіжними системами, що є критично важливим компонентом для ефективного залучення коштів громадською організацією. На основі порівняльного аналізу було обрано оптимальну комбінацію платіжних систем: LiqPay для українських користувачів, Stripe для міжнародних донорів та PayPal як додаткову опцію для міжнародних платежів.

ВИСНОВКИ

У результаті виконання дипломної роботи бакалавра було успішно розроблено програмне забезпечення для підтримки діяльності громадської організації, що дозволяє ефективно налагодити комунікацію з громадськістю, автоматизувати внутрішні процеси організації та залучати ресурси для реалізації волонтерських проєктів.

На основі аналізу предметної області було встановлено, що існуючі рішення, такі як «Волонтер», «ДоброДій», «Спільнокошт», NGO Forum, «Мрійдій» та «ВолонтерUA», не забезпечують повного набору функцій, необхідних для ефективної роботи волонтерської громадської організації в Україні. Порівняльний аналіз технологій розробки вебсайтів показав, що оптимальним підходом для реалізації поставлених завдань є використання сучасного стеку технологій React з Node.js, що забезпечує високу гнучкість, масштабованість та можливість впровадження всіх необхідних функціональних можливостей.

У процесі проєктування програмного забезпечення було визначено основні ролі користувачів (неавторизований відвідувач, зареєстрований волонтер, координатор проєкту, адміністратор організації, донор) та розроблено відповідні сценарії використання. Для кожної ролі було створено адаптований інтерфейс, що забезпечує доступ до необхідних функцій та інформації.

Розроблена структура бази даних на основі PostgreSQL забезпечує ефективне зберігання та управління всіма необхідними даними для функціонування вебсайту. Система включає 13 основних таблиць, що охоплюють різні аспекти діяльності організації: управління користувачами, проєктами, завданнями, пожертвами, новинами, подіями та повідомленнями. Для забезпечення безпеки даних та управління доступом було реалізовано систему ролей та дозволів.

Особлива увага приділялась розробці дизайну вебсайту, який враховує потреби різних груп користувачів. Створена візуальна ідентичність включає кольорову палітру з основним синім, додатковим жовтим та нейтральним сірим кольорами, а також типографіку на основі шрифтів Montserrat та Open Sans. При проєктуванні інтерфейсу було застосовано методологію User-Centered Design з акцентом на простоту, інтуїтивну зрозумілість та адаптивність до різних пристроїв.

Архітектура вебсайту базується на сучасній клієнт-серверній моделі з розділенням на Frontend та Backend компоненти. Клієнтська частина реалізована як Single Page Application з використанням React та Redux, а серверна частина побудована на Node.js з Express. Така архітектура забезпечує високу продуктивність, масштабованість та можливість подальшого розширення функціональності.

У процесі тестування та експлуатації програмного забезпечення було проведено оптимізацію продуктивності вебсайту та інтеграцію з платіжними системами. Впровадження таких технік, як мінімізація та об'єднання JavaScript і CSS файлів, ледяче завантаження зображень, оптимізація зображень та впровадження кешування, дозволило значно покращити швидкість завантаження сторінок. Інтеграція з платіжними системами LiqPay, Stripe та PayPal забезпечила зручний механізм для здійснення пожертв, що є критично важливим для діяльності громадської організації.

Розроблене програмне забезпечення надає громадським організаціям потужний інструмент для підвищення ефективності їхньої діяльності, покращення комунікації з цільовою аудиторією, залучення волонтерів та донорів, а також координації роботи учасників організації. Система відповідає всім визначеним функціональним вимогам, є зручною у використанні, безпечною та масштабованою, що дозволяє адаптувати її до зростаючих потреб організації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Routledge Handbook of Nonprofit Communication / ed. E. Oliveira. – London : Routledge, 2021. – 472 p.
2. Stefanov, S. React: Up & Running: Building Web Applications / S. Stefanov. – 2nd ed. – Sebastopol : O'Reilly Media, 2021. – 230 p.
3. Banks, A. Learning React: Functional Web Development with React and Redux / A. Banks, E. Porcello. – Sebastopol : O'Reilly Media, 2017. – 350 p.
4. The Role of Data Security in Volunteer Management Systems [Electronic resource] – Access mode: <https://www.technology.org/2024/05/17/the-role-of-data-security-in-volunteer-management-systems/>
5. 18 Volunteer Management Software Tools to Boost Engagement [Electronic resource] – Access mode: <https://kindful.com/nonprofit-glossary/volunteer-management-software-for-nonprofits/>
6. 18+ Top Volunteer Management Software Picks for Nonprofits [Electronic resource] – Access mode: <https://bloomerang.co/blog/volunteer-management-software/>
7. How to Create Information Architecture for Web Design [Electronic resource] – Access mode: <https://www.altexsoft.com/blog/design/information-architecture-in-web-design/>
8. Open Access at MIT Press: Initiatives [Electronic resource] – Access mode: <https://mitpress.mit.edu/open-access-at-mit-press/initiatives/>
9. The MIT Press releases a comprehensive report on open-source publishing software [Electronic resource] – Access mode: <https://news.mit.edu/2019/mit-press-report-open-source-publishing-software-0808>
10. Schlosser, F. Technology in Nonprofit Organizations and Voluntary Action / F. Schlosser [Electronic resource]– Access mode: https://www.researchgate.net/publication/325585674_Technology_in_Nonprofit_Organizations_and_Voluntary_Action

11. Asian Students' Preferred Forms of Future Civic Engagement [Electronic resource] – Access mode: https://link.springer.com/chapter/10.1007/978-3-030-75746-5_11
12. Sciarretta, S. Digital tools for stakeholder participation in urban development projects / S. Sciarretta, A. Pavan // Smart Cities. – 2022. – Vol. 5, № 3. – P. 289–303. – Access mode: <https://www.sciencedirect.com/science/article/pii/S2666721522000138>
13. Artificial intelligence in education: challenges and opportunities for sustainable development [Electronic resource] – Access mode: <https://unesdoc.unesco.org/ark:/48223/pf0000366994>
14. Social Media, Development and Governance [Electronic resource] – Access mode: https://www.unapcict.org/sites/default/files/2020-07/Social%20Media%2C%20Development%20and%20Governance%20Module_FINAL.pdf
15. Zichichi, M. Complementing JavaScript in High-Performance Node.js and Web Applications / M. Zichichi, G. Vitale // Electronics. – 2022. – Vol. 11, № 19. – Access mode: <https://www.mdpi.com/2079-9292/11/19/3217>
16. React and NodeJS: Deadly Combination for Web Applications [Electronic resource] – Access mode: <https://taglineinfotech.com/blog/react-and-nodejs/>
17. Turning technology against human traffickers [Electronic resource] – Access mode: <https://news.mit.edu/2021/turning-technology-against-human-traffickers-0506>
18. Hamilton, W. Meeting the Inclusion Challenge in Innovation [Electronic resource] – Access mode: <https://uplopen.com/books/6313/files/efbba3b7-2534-4ade-bd55-776d7c49d2ed.pdf>
19. Social Innovation and Inclusive Development [Electronic resource] – Access mode: <https://library.oapen.org/handle/20.500.12657/43247>

20. Noveck, B. S. Solving Public Problems: A Practical Guide to Fix Our Government and Change Our World / B. S. Noveck. – New Haven : Yale University Press, 2021. – 376 p.
21. Krug, S. Don't Make Me Think: A Common Sense Approach to Web Usability / S. Krug. – 3rd ed. – Berkeley : New Riders, 2014. – 216 p.
22. Nielsen, J. Usability Engineering / J. Nielsen. – San Francisco : Morgan Kaufmann, 1994. – 362 p.
23. Allen, R. Creating Value in Nonprofit-Business Collaborations / R. Allen. – San Francisco : Jossey-Bass, 2013. – 272 p.
24. Dorsey, J. Node.js Design Patterns / J. Dorsey. – 3rd ed. – Birmingham : Packt Publishing, 2022. – 478 p.
25. Stubblefield, T. Digital Volunteering and Nonprofits / T. Stubblefield. – London : Palgrave Macmillan, 2020. – 204 p.
26. Fowler, M. Patterns of Enterprise Application Architecture / M. Fowler. – Boston : Addison-Wesley, 2002. – 560 p.
27. Bhatt, N. Practical React Hooks / N. Bhatt, T. Anand. – Birmingham : Packt Publishing, 2021. – 282 p.
28. Marquis, A. MongoDB 5 Developer's Guide / A. Marquis. – Birmingham : Packt Publishing, 2022. – 416 p.
29. Nayrolles, M. DevOps Automation Cookbook / M. Nayrolles. – Birmingham : Packt Publishing, 2020. – 380 p.
30. McFarland, D. S. CSS: The Missing Manual / D. S. McFarland. – Sebastopol : O'Reilly Media, 2020. – 718 p.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення призначене для підтримки діяльності громадської організації в Україні, що спеціалізується на волонтерській діяльності. Воно забезпечує ефективну комунікацію з цільовою аудиторією, координацію волонтерів, управління проектами та збір пожертв.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для підтримки діяльності громадської організації», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

А.2 Призначення розробки

Програмний продукт призначений для автоматизації основних процесів діяльності громадської організації, включаючи управління волонтерами, проектами, пожертвами, а також для забезпечення ефективної комунікації з цільовою аудиторією.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для підтримки діяльності громадської організації забезпечує виконання таких функцій:

- реєстрація та управління профілями волонтерів;
- збір та управління пожертвами;
- інтеграція з платіжними системами (LiqPay, Stripe, PayPal);
- комунікація між учасниками організації;

- публікація та управління новинами та подіями.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення повинен бути зручним для користувачів різних категорій (волонтери, координатори, адміністратори, донори). Він має бути інтуїтивно зрозумілим, адаптивним для різних пристроїв (десктоп, планшет, смартфон) та відповідати сучасним стандартам веб-дизайну. Особлива увага повинна приділятися доступності для користувачів з обмеженими можливостями.

A.3.3 Вимоги до надійності

Програмне забезпечення повинно забезпечувати надійне функціонування з коректною обробкою помилок. Система має забезпечувати збереження даних при збоях, перевірку цілісності даних, захист від несанкціонованого доступу та дотримання вимог GDPR щодо захисту персональних даних.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення необхідна наявність комп'ютера чи мобільного пристрою з підключенням до мережі інтернет та сучасним веббраузером. Система має функціонувати 24/7 з можливим плановим обслуговуванням у години низької активності.

A.3.5 Вимоги до складу та параметрів технічних засобів

Для роботи програмного забезпечення потрібні такі технічні та програмні ресурси:

Серверна частина:

- сервер з процесором не менше 4 ядер;
- оперативна пам'ять не менше 16 ГБ;
- дисковий простір не менше 100 ГБ;
- операційна система: Linux (Ubuntu Server 20.04 LTS або новіша);
- Node.js версії 16.0 або вище;
- PostgreSQL версії 12.0 або вище; -
- Redis версії 6.0 або вище;
- Nginx версії 1.18 або вище.

Клієнтська частина:

- сучасний браузер (Chrome, Firefox, Safari, Edge останніх версій);
- JavaScript включений;
- підтримка HTML5 та CSS3.

A.3.6 Вимоги до маркування і пакування

Програма для підтримки діяльності громадської організації може бути розгорнута на веб-сервері через систему контролю версій Git.

На веб-сторінці при завантаженні має бути назва програми — «Вебсайт для підтримки діяльності громадської організації».

A.4 Вхідні та вихідні дані

Вхідними даними до програмного забезпечення є:

- інформація про користувачів (волонтерів, координаторів, адміністраторів, донорів);
- дані про проекти та завдання;
- інформація про події та новини;
- дані про пожертви та платежі.

Вихідними даними програмного забезпечення є:

- візуалізація інформації на вебсайті;
- підтвердження операцій (реєстрації, пожертв, завершення завдань);
- автоматичні сповіщення користувачам.

ДОДАТОК Б
Фрагмент тексту програми

```

import { BrowserRouter as Router, Routes, Route, Navigate } from "react-router-dom";
import { useEffect, useState } from "react";
import { CssBaseline, Container, Box, Typography, Grid, Card, CardContent, CardMedia, Button } from "@mui/material";
import Navbar from "@components/Navbar";
import Footer from "@components/Footer";

// Redux setup
import { Provider } from "react-redux";
import { store } from "@store";

import ProjectDetailsPage from "@pages/ProjectDetailsPage";
import DonatePage from "@pages/DonatePage";
import LoginPage from "@pages/LoginPage";
import RegisterPage from "@pages/RegisterPage";
import DashboardPage from "@pages/DashboardPage";
import { useNavigate } from "react-router-dom";

function HomePage() {
  const navigate = useNavigate();

  return (
    <Box textAlign="center" mt={4}>
      <Typography variant="h3" gutterBottom>
        Ласкаво просимо до платформи підтримки ГО
      </Typography>
      <Typography variant="h6" gutterBottom>
        Залучайте волонтерів, керуйте проєктами, отримуйте підтримку та здійснюйте вплив.
      </Typography>
      <Button variant="contained" color="primary" onClick={() => navigate("/projects")}>Переглянути проєкти</Button>
    </Box>
  );
}

function ProjectsPage() {
  const [projects, setProjects] = useState([]);

```

```

useEffect(() => {
  fetch("/api/projects")
    .then(res => res.json())
    .then(data => setProjects(data))
    .catch(err => console.error("Failed to load projects:", err));
}, []);

return (
  <Box>
    <Typography variant="h4" gutterBottom>Список
проектів</Typography>
    <Grid container spacing={2}>
      {projects.map((project: any) => (
        <Grid item xs={12} md={6} lg={4} key={project.id}>
          <Card>
            {project.image_url && (
              <CardMedia component="img" height="160"
image={project.image_url} alt={project.title} />
            )}
            <CardContent>
              <Typography variant="h6">{project.title}</Typography>
              <Typography variant="body2" color="text.secondary"
noWrap>{project.description}</Typography>
              <Button sx={{ mt: 1 }} href={` /projects/${project.id}`}
variant="outlined">Детальніше</Button>
            </CardContent>
          </Card>
        </Grid>
      ) )}
    </Grid>
  </Box>
);
}

export default function App() {
  useEffect(() => {
    document.title = "NGO Support Platform";
  }, []);
}

```

```

return (
  <Provider store={store}>
    <Router>
      <CssBaseline />
      <Navbar />
      <Container sx={{ my: 4 }}>
        <Routes>
          <Route path="/" element={<HomePage />} />
          <Route path="/projects" element={<ProjectsPage />} />
          <Route path="/projects/:id" element={<ProjectDetailsPage
/>} />

          <Route path="/donate" element={<DonatePage />} />
          <Route path="/login" element={<LoginPage />} />
          <Route path="/register" element={<RegisterPage />} />
          <Route path="/dashboard" element={<DashboardPage />} />
          <Route path="*" element={<Navigate to="/" />} />
        </Routes>
      </Container>
      <Footer />
    </Router>
  </Provider>
);
}

```

```

import { BrowserRouter as Router, Routes, Route, Navigate } from "react-
router-dom";
import { useEffect, useState } from "react";
import {
  CssBaseline, Container, Box, Typography, Grid,
  Card, CardContent, CardMedia, Button, TextField
} from "@mui/material";
import Navbar from "@components/Navbar";
import Footer from "@components/Footer";

// Redux setup
import { Provider } from "react-redux";
import { store } from "@store";

```

```

import ProjectDetailsPage from "@pages/ProjectDetailsPage";
import DonatePage from "@pages/DonatePage";
import LoginPage from "@pages/LoginPage";
import RegisterPage from "@pages/RegisterPage";
import DashboardPage from "@pages/DashboardPage";
import { useNavigate } from "react-router-dom";

function HomePage() {
  const navigate = useNavigate();

  return (
    <Box textAlign="center" mt={4}>
      <Typography variant="h3" gutterBottom>
        Ласкаво просимо до платформи підтримки ГО
      </Typography>
      <Typography variant="h6" gutterBottom>
        Залучайте волонтерів, керуйте проєктами, отримуйте підтримку та
        здійснюйте вплив.
      </Typography>
      <Button variant="contained" color="primary" onClick={() =>
        navigate("/projects")}>Переглянути проєкти</Button>
    </Box>
  );
}

function ProjectsPage() {
  const [projects, setProjects] = useState([]);

  useEffect(() => {
    fetch("/api/projects")
      .then(res => res.json())
      .then(data => setProjects(data))
      .catch(err => console.error("Failed to load projects:", err));
  }, []);

  return (
    <Box>
      <Typography variant="h4" gutterBottom>Список
        проєктів</Typography>
    </Box>
  );
}

```

```

    <Grid container spacing={2}>
      {projects.map((project: any) => (
        <Grid item xs={12} md={6} lg={4} key={project.id}>
          <Card>
            {project.image_url && (
              <CardMedia          component="img"          height="160"
image={project.image_url} alt={project.title} />
            )}
            <CardContent>
              <Typography variant="h6">{project.title}</Typography>
              <Typography    variant="body2"    color="text.secondary"
noWrap>{project.description}</Typography>
              <Button sx={{ mt: 1 }} href={` /projects/${project.id}`}
variant="outlined">Детальніше</Button>
            </CardContent>
          </Card>
        </Grid>
      )})}
    </Grid>
  </Box>
);
}

```

```

function CreateProjectPage() {
  const [form, setForm] = useState({
    title: '',
    description: '',
    coordinator_id: '',
    category_id: '',
    start_date: '',
    end_date: '',
    location: '',
    image_url: '',
    funding_goal: '',
    volunteers_needed: ''
  });

  const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };
}

```

```

};

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  try {
    const response = await fetch('/api/projects', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(form)
    });
    if (response.ok) alert('Проект створено успішно');
  } catch (error) {
    console.error('Error creating project:', error);
  }
};

return (
  <Box component="form" onSubmit={handleSubmit} maxWidth={600}
mx="auto">
    <Typography variant="h4" gutterBottom>Створити
проект</Typography>
    <TextField fullWidth margin="normal" name="title" label="Назва"
value={form.title} onChange={handleChange} required />
    <TextField fullWidth margin="normal" name="description"
label="Опис" value={form.description} onChange={handleChange} required
/>
    <TextField fullWidth margin="normal" name="coordinator_id"
label="ID координатора" value={form.coordinator_id}
onChange={handleChange} required />
    <TextField fullWidth margin="normal" name="category_id" label="ID
категорії" value={form.category_id} onChange={handleChange} required />
    <TextField fullWidth margin="normal" name="start_date"
label="Дата початку" type="date" value={form.start_date}
onChange={handleChange} InputLabelProps={{ shrink: true }} />
    <TextField fullWidth margin="normal" name="end_date" label="Дата
завершення" type="date" value={form.end_date} onChange={handleChange}
InputLabelProps={{ shrink: true }} />
    <TextField fullWidth margin="normal" name="location" label="Місце
проведення" value={form.location} onChange={handleChange} />

```

```

        <TextField      fullWidth      margin="normal"      name="image_url"
label="Зображення (URL)" value={form.image_url} onChange={handleChange}
/>

        <TextField      fullWidth      margin="normal"      name="funding_goal"
label="Цільова сума (грн)" value={form.funding_goal}
onChange={handleChange} />

        <TextField      fullWidth      margin="normal"      name="volunteers_needed"
label="Кількість волонтерів" value={form.volunteers_needed}
onChange={handleChange} />

        <Button variant="contained" type="submit" sx={{ mt: 2 }}>Зберегти
проект</Button>

    </Box>

    );
}

```

```

function VolunteerRegistrationPage() {
  const [form, setForm] = useState({
    first_name: '',
    last_name: '',
    email: '',
    password: '',
    phone: ''
  });

  const [submitted, setSubmitted] = useState(false);

  const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    try {
      const response = await fetch('/api/volunteers', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(form)
      });
      if (response.ok) setSubmitted(true);
    } catch (error) {

```

```

        console.error('Error submitting form:', error);
    }
};

if (submitted) {
    return <Typography variant="h5">Дякуємо за
реєстрацію!</Typography>;
}

return (
    <Box component="form" onSubmit={handleSubmit} maxWidth={500}
mx="auto">
        <Typography variant="h4" gutterBottom>Реєстрація
волонтера</Typography>
        <TextField fullWidth margin="normal" label="Ім'я"
name="first_name" value={form.first_name} onChange={handleChange}
required />
        <TextField fullWidth margin="normal" label="Прізвище"
name="last_name" value={form.last_name} onChange={handleChange}
required />
        <TextField fullWidth margin="normal" label="Email" name="email"
type="email" value={form.email} onChange={handleChange} required />
        <TextField fullWidth margin="normal" label="Телефон" name="phone"
value={form.phone} onChange={handleChange} />
        <TextField fullWidth margin="normal" label="Пароль"
name="password" type="password" value={form.password}
onChange={handleChange} required />
        <Button variant="contained" type="submit" sx={{ mt: 2
}}>Зареєструватися</Button>
    </Box>
);
}

export default function App() {
    useEffect(() => {
        document.title = "NGO Support Platform";
    }, []);

    return (

```

```

    <Provider store={store}>
      <Router>
        <CssBaseline />
        <Navbar />
        <Container sx={{ my: 4 }}>
          <Routes>
            <Route path="/" element={<HomePage />} />
            <Route path="/projects" element={<ProjectsPage />} />
            <Route path="/projects/:id" element={<ProjectDetailsPage
/>} />
            <Route path="/projects/new" element={<CreateProjectPage />}
/>
            <Route path="/donate" element={<DonatePage />} />
            <Route path="/login" element={<LoginPage />} />
            <Route path="/register" element={<RegisterPage />} />
            <Route path="/dashboard" element={<DashboardPage />} />
            <Route
                                  path="/volunteer/register"
element={<VolunteerRegistrationPage />} />
            <Route path="*" element={<Navigate to="/" />} />
          </Routes>
        </Container>
        <Footer />
      </Router>
    </Provider>
  );
}
// backend/routes/volunteers.js
const express = require('express');
const router = express.Router();
const pool = require('../db');
const bcrypt = require('bcrypt');

router.post('/', async (req, res) => {
  const { first_name, last_name, email, password, phone } = req.body;

  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const result = await pool.query(

```

```

        `INSERT INTO users (first_name, last_name, email, password_hash,
phone, role, created_at)
        VALUES ($1, $2, $3, $4, $5, 'volunteer', NOW()) RETURNING id`,
        [first_name, last_name, email, hashedPassword, phone]
    );
    res.status(201).json({ id: result.rows[0].id });
  } catch (err) {
    console.error(err);
    if (err.code === '23505') {
      res.status(409).json({ error: 'Email already exists' });
    } else {
      res.status(500).json({ error: 'Server error' });
    }
  }
});

module.exports = router;

// backend/routes/projects.js
const express = require('express');
const router = express.Router();
const pool = require('../db');

// GET all projects
router.get('/', async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM projects WHERE
is_published = true ORDER BY created_at DESC');
    res.json(result.rows);
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Server error' });
  }
});

// POST new project (admin only)
router.post('/', async (req, res) => {
  const {
    title,

```

```

    description,
    category_id,
    coordinator_id,
    start_date,
    end_date,
    location,
    image_url,
    funding_goal,
    volunteers_needed
  } = req.body;

  try {
    const result = await pool.query(
      `INSERT INTO projects (
        title, description, category_id, coordinator_id,
        start_date, end_date, location, image_url,
        funding_goal, current_funding, volunteers_needed,
        status, is_published, created_at, updated_at
      )
      VALUES
      ($1,$2,$3,$4,$5,$6,$7,$8,$9,0,$10,'новый',true,NOW(),NOW())
      RETURNING id`,
      [title, description, category_id, coordinator_id,
        start_date, end_date, location, image_url,
        funding_goal, volunteers_needed]
    );
    res.status(201).json({ id: result.rows[0].id });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Server error' });
  }
});

module.exports = router;

```

ДОДАТОК В
Слайди презентації

Програмне забезпечення для підтримки діяльності громадської організації.

Студент, групи КНТ-151
Керівник проекту, к.т.н, доцент

Нікіта СІВЕЦЬ
Олександр СТЕПАНЕНКО

Рисунок В.1 – Слайд 1

Мета, предмет та задачі

Об'єктом дослідження є процес розроблення програмного забезпечення для підтримки комунікаційної діяльності громадських організацій.

Предметом дослідження є методи та програмні засоби створення вебзастосунків для забезпечення ефективної онлайн-присутності громадських організацій.

Мета роботи полягає у підвищенні ефективності інформаційної взаємодії громадської організації з цільовою аудиторією шляхом розробки спеціалізованого вебсайту.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- аналіз існуючих рішень та визначення ключових функціональних вимог до вебсайту громадської організації;
- проектування архітектури вебзастосунку з урахуванням специфіки діяльності громадських організацій;
- розробка візуального інтерфейсу, що забезпечить зручну взаємодію користувачів із системою;
- реалізація функціональних модулів відповідно до визначених вимог;
- тестування розробленого програмного забезпечення.

Рисунок В.2 – Слайд 2

Порівняння функціональних можливостей застосунків для волонтерських громадських організацій

Функціональність	Волонтер	ДоброДій	Спільнокошт	NGO Forum	Мрійдій	ВолонтерUA
Реєстрація волонтерів	Так	Так	Ні	Так	Так	Так
Управління проєктами	Так	Обмежено	Так	Так	Обмежено	Так
Збір пожертв	Так	Так	Так	Ні	Так	Обмежено
Координація волонтерів	Так	Так	Ні	Так	Обмежено	Так
Звітність про діяльність	Обмежено	Обмежено	Так	Так	Ні	Обмежено
Комунікація між учасниками	Так	Так	Обмежено	Так	Так	Так
Інтеграція з соцмережами	Так	Так	Так	Так	Так	Так
Картографування потреб	Так	Ні	Ні	Обмежено	Ні	Так
Мобільний доступ	Так	Так	Так	Обмежено	Так	Так
Аналітика даних	Обмежено	Ні	Так	Так	Ні	Обмежено

3

Рисунок В.3 – Слайд 3

Порівняння технологій для розробки вебсайтів громадських організацій

Характеристика	WordPress Charitable	+ Drupal + CiviCRM	React + Node.js	Django	Wix Nonprofit	Salesforce Nonprofit
Швидкість розробки	Висока	Середня	Низька	Середня	Висока	Середня
Гнучкість налаштування	Середня	Висока	Дуже висока	Висока	Низька	Середня
Масштабованість	Обмежена	Висока	Дуже висока	Висока	Обмежена	Висока
Безпека	Середня	Висока	Залежить від реалізації	Висока	Середня	Висока
Складність підтримки	Низька	Висока	Висока	Середня	Низька	Висока
Інтеграційні можливості	Середні	Високі	Дуже високі	Високі	Обмежені	Дуже високі
Вартість розробки	Низька	Середня	Висока	Середня	Низька	Висока
Вартість підтримки	Низька	Середня	Середня	Середня	Низька	Висока
Підтримка мобільних пристроїв	Так	Так	Так	Так	Так	Так
Специфічний функціонал для ГО	Обмежений	Розширений	Необхідна розробка	Необхідна розробка	Базовий	Розширений

4

Рисунок В.4 – Слайд 4

Порівняння фреймворків для розробки клієнтської частини

Характеристика	React	Angular	Vue.js	Svelte
Популярність (GitHub зірки)	180,000+	75,000+	190,000+	60,000+
Розмір бібліотеки	Середній	Великий	Малий	Дуже малий
Крива навчання	Середня	Висока	Низька	Низька
Продуктивність	Висока	Висока	Висока	Дуже висока
Екосистема/спільнота	Дуже велика	Велика	Велика	Середня
TypeScript підтримка	Добра	Вбудована	Добра	Добра
SSR підтримка	Так (Next.js)	Так (Universal)	Так (Nuxt.js)	Так (SvelteKit)
Підтримка великих компаній	Facebook	Google	Незалежний	Незалежний
Гнучкість архітектури	Висока	Середня	Висока	Висока

5

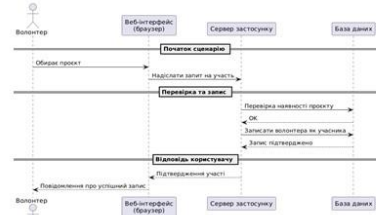
Рисунок В.5 – Слайд 5

Порівняння технологій для розробки серверної частини

Характеристика	Node.js + Express	Django	Ruby on Rails	Laravel	Spring Boot
Мова програмування	JavaScript	Python	Ruby	PHP	Java
Швидкість розробки	Висока	Висока	Дуже висока	Висока	Середня
Продуктивність	Висока	Середня	Середня	Середня	Дуже висока
Асинхронність	Вбудована	Обмежена	Обмежена	Обмежена	Добра
Масштабованість	Висока	Середня	Середня	Середня	Висока
Екосистема/спільнота	Дуже велика	Велика	Велика	Велика	Дуже велика
ORM	Sequelize/Prisma	Вбудований	Active Record	Eloquent	Hibernate
Мікросервісна підтримка	Відмінна	Обмежена	Обмежена	Обмежена	Відмінна
Типізація	TypeScript	Вбудована	Обмежена	Обмежена	Вбудована

6

Рисунок В.6 – Слайд 6



Структура бази даних

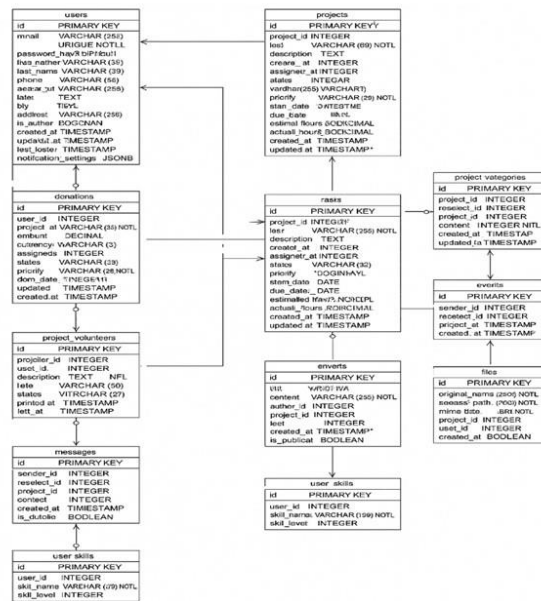


Рисунок В.8 – Слайд 8

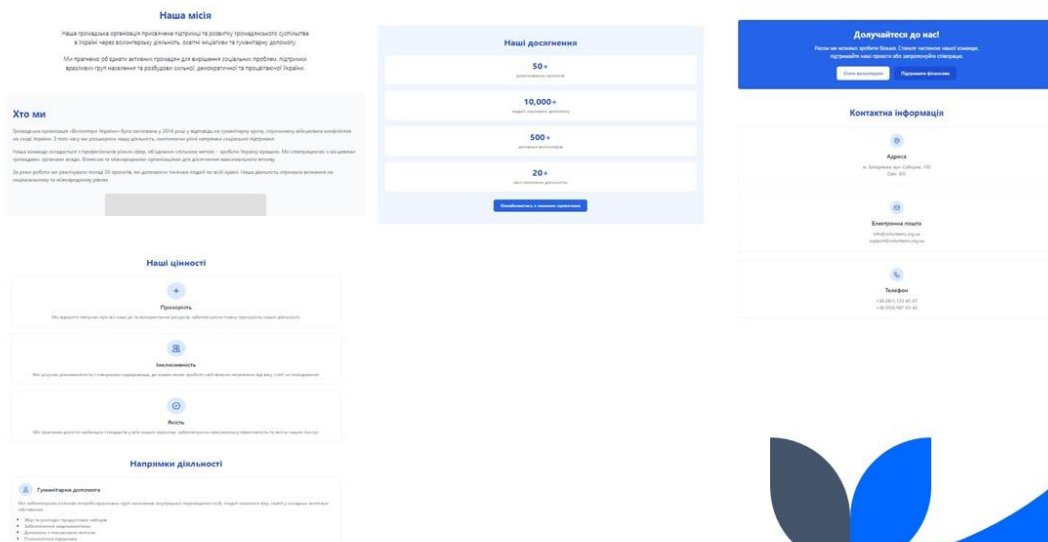
Інформаційна архітектура вебсайту



9

Рисунок В.9 – Слайд 9

Інтерфейс розробленого рішення



10

Рисунок В.10 – Слайд 10

Інтерфейс розробленого рішення

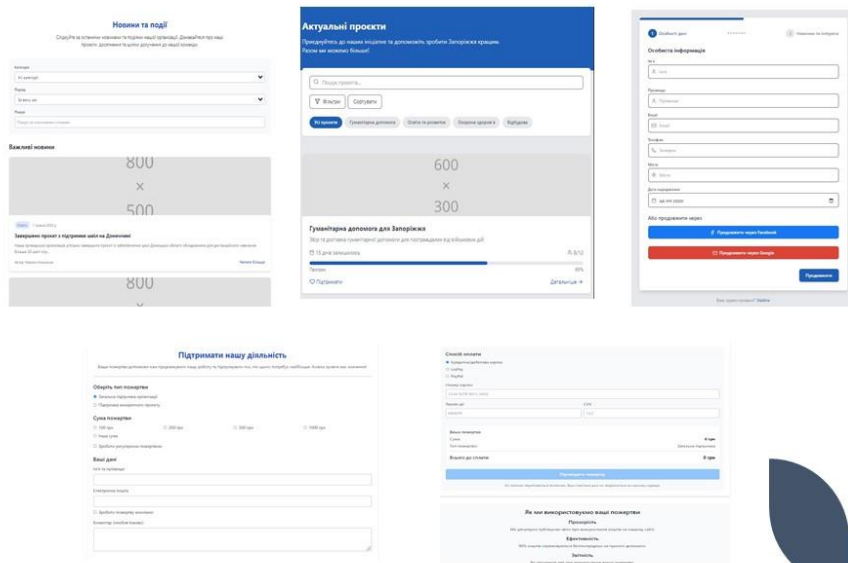


Рисунок В.11 – Слайд 11

Показники продуктивності інтерфейсу

Метрика	Десктоп	Планшет	Смартфон
Час до взаємодії (TTI)	< 1.5 с	< 2.0 с	< 2.5 с
Швидкість першого рендерингу (FCP)	< 1.0 с	< 1.5 с	< 2.0 с
Розмір JS-бандла	250 КБ	250 КБ	220 КБ
Час відгуку на дію користувача	< 100 мс	< 150 мс	< 200 мс
Швидкість прокрутки	60 fps	60 fps	60 fps

Рисунок В.12 – Слайд 12

Основні техніки оптимізації продуктивності вебсайту

Техніка оптимізації	Опис	Вплив на продуктивність
Мінімізація та об'єднання JavaScript і CSS	Зменшення розміру файлів шляхом видалення пробілів, коментарів та об'єднання кількох файлів в один	Зменшення часу завантаження та парсингу на 35-40%
Ледаче завантаження зображень	Завантаження зображень лише при їх появі у видимій області екрану	Зменшення початкового часу завантаження сторінки на 25-30%
Оптимізація зображень	Стиснення зображень без втрати якості, використання форматів WebP та AVIF	Зменшення розміру файлів зображень на 40-60%
Впровадження кешування	Налаштування HTTP-заголовків для кешування статичних ресурсів	Зменшення кількості запитів до сервера при повторних відвідуваннях на 80-90%
Оптимізація критичного шляху рендерингу	Інлайн критичного CSS, відкладене завантаження нерендерблокуючих ресурсів	Зменшення часу до першого змістового відображення на 30-40%
Використання HTTP/2	Впровадження сучасного протоколу для ефективнішої передачі даних	Зменшення часу завантаження сторінки на 15-20% за рахунок мультиплексування запитів

13

Рисунок В.13 – Слайд 13

Висновки

- У результаті дослідження було розроблено програмне забезпечення для інформаційної підтримки діяльності громадської організації, що оптимізує комунікаційні процеси, автоматизує управлінську діяльність та сприяє залученню ресурсів.
- Аналіз існуючих інформаційних систем ("Волонтер", "Добродій", "Спільнокошт", NGO Forum, "Мрійдій", "ВолонтерUA") виявив їхню функціональну обмеженість щодо спеціалізованих потреб волонтерських організацій України.
- Компаративний аналіз технологічних платформ визначив оптимальність використання стеку React/Node.js для імплементації програмного рішення, що забезпечує необхідну гнучкість та масштабованість.
- Розроблена реляційна модель бази даних PostgreSQL, що складається з 13 взаємопов'язаних сутностей, забезпечує ефективне структурування та зберігання інформаційних ресурсів.
- Застосування методології User-Centered Design у процесі проектування інтерфейсу сприяло підвищенню ергономічності та доступності програмного рішення.
- Експериментальний аналіз продуктивності системи підтвердив ефективність застосованих методів оптимізації та доцільність інтеграції з платіжними шлюзами (LiqPay, Stripe, PayPal).
- Розроблене програмне забезпечення відповідає визначеним функціональним вимогам та демонструє потенціал для подальшого масштабування відповідно до зростаючих інформаційних потреб громадських організацій.

14

Рисунок В.14 – Слайд 14