

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: "Розроблення інформаційної системи управління волонтерською діяльністю"

Виконав: студент групи 4147ст

 Зайцев О.В.
(підпис)

Керівник роботи:

(посада, науковий ступінь вчене звання)
 Беркунська Н. М.
(підпис)

м. Миколаїв — 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

" ____ " _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "бакалавр"

Студенту Зайцеву Олегу Валентиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Розроблення інформаційної системи управління волонтерською діяльністю

Керівник роботи: Беркунська Наталія Михайлівна

Затверджені наказом ректора: № 295-уч від "27" березня 2025 року.

2. Термін подання роботи: 20 червня 2025 р.

3. Вихідні дані до проекту (роботи): ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів): Вступ. 1. Розробка концепції інформаційної системи. 2. Розробка проекту інформаційної системи. 3. Реалізація проекту інформаційної системи. 4. Охорона праці.

Висновки.

5. Перелік презентаційних матеріалів: Слайди: Розроблення інформаційної системи управління волонтерською діяльністю; Мета роботи, Об'єкт дослідження, Предмет дослідження; Реалізація інформаційної системи, Результати впровадження.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Беркунська Н. М.	03.02.2025	22.02.2025
2	Беркунська Н. М.	22.02.2025	21.03.2025
3	Беркунська Н. М.	21.03.2025	25.04.2025
4	Беркунська Н. М.	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проекту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент


 (підпис)
Зайцев О. В.

(ПІБ)

Керівник роботи


 (підпис)
Беркунська Н. М.

(ПІБ)

АНОТАЦІЯ

Обсяг пояснювальної записки складає 82 сторінки. У роботі наведено 32 ілюстрації, 2 таблиці, 3 додатки та 35 джерел у списку літератури. Об'єктом проєкту є розробка інформаційної системи управління волонтерською діяльністю. Предметна область проєкту охоплює процеси організації, координації та підтримки волонтерських активностей. Метою проєкту є створення зручної платформи для взаємодії волонтерів та організацій, що сприятиме підвищенню ефективності управління заходами.

У рамках дослідження використовувалися методи аналізу вимог користувачів, проєктування архітектури системи, розробки програмного забезпечення та тестування. Результати дослідження включають функціональну інформаційну систему, що забезпечує ефективне управління активностями, а також підтримку відгуків та рекомендацій на основі вподобань користувачів.

Основні технологічні характеристики проєкту включають використання мови програмування Java з фреймворком Spring Boot для серверної частини та React з TypeScript для клієнтської. Для зберігання даних використовується база даних MySQL.

У результаті платформа може бути застосована у сфері волонтерської діяльності для полегшення процесів організації заходів та залучення нових учасників. Висновки та перспективи розвитку проєкту підтверджують його актуальність та значущість для підтримки волонтерського руху.

Ключові слова: волонтерська діяльність, інформаційна система, управління активностями, користувач, розробка, технології, база даних.

ABSTRACT

The explanatory note comprises 82 pages. The work includes 32 illustrations, 2 tables, 2 appendices, and 35 references.

The object of the project is the development of an information system for managing volunteer activities. The subject area of the project covers the processes of organizing, coordinating, and supporting volunteer activities. The goal of the project is to create a user-friendly platform for interaction between volunteers and organizations, which will enhance the efficiency of event management.

The study involved methods of user requirements analysis, system architecture design, software development, and testing. The results include a functional information system that provides effective management of activities, as well as support for feedback and recommendations based on user preferences.

The main technological features of the project include the use of the Java programming language with the Spring Boot framework for the server side and React with TypeScript for the client side. The MySQL database is used for data storage. As a result, the platform can be applied in the field of volunteer activities to simplify event organization processes and attract new participants. The conclusions and prospects for the development of the project confirm its relevance and significance in supporting the volunteer movement.

Keywords: volunteer activities, information system, activity management, user, development, technologies, database.

ЗМІСТ

РОЗДІЛ 1. Розробка концепції інформаційної системи управління волонтерською діяльністю	9
1.1. Опис об'єкта обстеження та аналіз предметної області	9
1.2. Формування вимог до інформаційної системи.....	14
1.3. Розробка концепції інформаційної системи	18
Висновок	23
РОЗДІЛ 2. Розробка технічного проекту інформаційної системи управління волонтерською діяльністю	24
2.1. Загальносистемні рішення	24
2.2. Рішення по інформаційному забезпеченню	31
2.3. Рішення з математичного забезпечення	46
2.4. Рішення з програмного забезпечення	48
2.5. Рішення з технічного забезпечення	50
Висновок	52
РОЗДІЛ 3. Реалізація проекту інформаційної системи управління волонтерською діяльністю	53
РОЗДІЛ 4. Охорона праці	58
ВИСНОВКИ.....	64
ЛІТЕРАТУРА	65
Додаток А – Інструкція користувачу	67
Додаток Б – Розгортання додатку.....	75
Додаток В – Текст програми	78

Перелік умовних позначень

ПЗ – Програмне забезпечення

ІС – Інформаційна система

ОС – Операційна система

СКБД, СУБД - Система керування (управління) базами даних.

API - Application Programming Interface.

HTTP - Hypertext Transfer Protocol.

REST - REpresentational State Transfer.

AWS - Amazon Web Services

JWT - JSON Web Token

CI/CD - Continuous Integration/Continuous Delivery

VPS - Virtual Private Server

SSD - Solid-State Drive

RAM - Random Access Memory

JDK - Java Development Kit

SQL - Structured Query Language

SPA – Single Page Application

ER - Entity-Relationship

ERD - Entity-Relationship Diagram

ВСТУП

У сучасних умовах активного розвитку волонтерських ініціатив зростає потреба у створенні ефективних інформаційних систем для підтримки співпраці між волонтерами та організаціями. Збільшення кількості проєктів та учасників вимагає нових підходів до управління даними, координації активностей та забезпечення зручного обміну інформацією. Традиційні методи організації роботи волонтерських спільнот часто є малоефективними та не дозволяють повноцінно задовольнити потреби сучасних користувачів.

Метою даної дипломної роботи є розробка інформаційної системи для підтримки взаємодії між волонтерами та волонтерськими організаціями. Система має забезпечувати зручний пошук активностей, можливість подачі заявок на участь у заходах, підписку на новини організацій, публікацію та перегляд відгуків. Окрім цього, передбачено налаштування особистих профілів користувачів та визначення їхніх уподобань для персоналізації взаємодії із системою.

Для досягнення поставленої мети проведено дослідження предметної області, сформульовано вимоги до функціоналу та архітектури системи, обґрунтовано вибір загальносистемних рішень, побудовано концептуальну модель даних і обрано найбільш придатну архітектурну модель для реалізації. У результаті виконано розробку інформаційної системи, яка дозволяє ефективно організовувати роботу волонтерських спільнот та сприяє розвитку громадських ініціатив.

РОЗДІЛ 1. Розробка концепції інформаційної системи управління волонтерською діяльністю

1.1. Опис об'єкта обстеження та аналіз предметної області

В умовах повномасштабної війни, яка триває в Україні, суспільство стикається з новими викликами, що вимагають активної участі громадян у волонтерській діяльності. Волонтерство відіграє ключову роль у підтримці військових, переселенців, постраждалих від бойових дій, а також у відбудові країни [1].

Волонтерський рух є по суті частиною кожної цивілізації та будь-якого суспільства. Волонтер – людина, яка добровільно здійснює соціально спрямовану некомерційну діяльність шляхом надання волонтерської допомоги.

Волонтерські організації є ключовими гравцями в економіці самі по собі в якості роботодавців і постачальників послуг, що збільшують загальну економічну віддачу країни і знижують навантаження на державні витрати. Цей сектор також відіграє ключову роль в створенні умов, в яких економіка може процвітати, вкладаючи кошти в людей засобом навчання, підвищення кваліфікації та поліпшення можливостей працевлаштування людей на ринку праці [2].

Волонтерство вносить значний вклад в економіку в усьому світі. Згідно з дослідженням Центру досліджень громадянського суспільства Джона Хопкінса, в 2017 році приблизно 140 мільйонів чоловік в 37 країнах беруть участь в добровольчій роботі в поточному році. Якби ці 140 мільйонів добровольців зібралися разом, це була б дев'ята за величиною країна в світі. Ці 140 мільйонів добровольців представляють еквівалент 20 мільйонів 80 тисяч робочих місць з повною зайнятістю. А діяльність волонтерів вирішує економічні проблеми [1].

Однак ці цифри про економічні значення не враховують переваги третіх сторін – співгромадян, які можуть насолоджуватися тим, що час присвячено цим

людям або причинам, які їх хвилюють. Ці цифри не враховують і менш відчутні вигоди, які випливають з волонтерства. Волонтерський сектор відіграє ключову роль у суспільному житті. Волонтерська діяльність об'єднує людей, які в іншому випадку не могли б контактувати один з одним [3].

Соціальну структуру можна зміцнити тільки завдяки практикам, які допомагають подолати наші соціально-економічні відмінності. Тут враховуються такі потенційні цінності, що лежать в основі соціального капіталу та поєднують організаційне життя спільноти, участь в громадських об'єднаннях, участь в громадських справах, неформальну комунікабельність і соціальну довіру.

Особливість волонтерського руху полягає в тому, що численною його групою є студентська молодь. Участь молодих людей у волонтерському русі дає їм можливість зробити особистий внесок у вирішення соціальних проблем, випробувати свої можливості, беручи участь в проектах і програмах соціальної спрямованості. Теоретична і практична підготовка молоді до волонтерської діяльності здійснюється в школах волонтерів [1].

Волонтерами можна вважати також неурядові (громадські) організації, що діють на громадських засадах. Серед них молодіжні та скаутські організації, молодіжні політичні, спортивні або екологічні організації, організації, які активно займаються розвитком населених пунктів і вирішенням конкретних проблем місцевого самоврядування [1].

У той же час перед волонтерами та волонтерськими організаціями виникають суттєві проблеми, які можна згрупувати за наступними напрямками:

1. Зниження обсягів допомоги з боку населення України, представників діаспори та інших благодійників;
2. Діяльність псевдоволонтерів і випадки зловживань з боку волонтерів;

3. Відсутність достатньої координації волонтерів між собою – недостатня обізнаність або відсутність координації між діями осіб і структур, що виконують схожі завдання;

4. Відсутність ефективного контролю за подальшою долею майна, переданого волонтерами;

5. Погана комунікація між волонтерами і суспільством;

6. Зниження мотивації, емоційне вигорання волонтерів. Не можна обійти увагою важливу проблему виснаження волонтерів (фізичного, емоційного, психологічного);

7. Безпека волонтерів. Окремою проблемою є безпека волонтерів, насамперед тих, хто самостійно доставляє допомогу в підрозділи, які знаходяться на лінії зіткнення;

8. Недостатнє законодавче регулювання волонтерської діяльності. Діюча нормативно-правова база багато в чому не враховує реалії, і час від часу перешкоджає вирішенню волонтерами актуальних проблем;

9. Консерватизм державних структур. Традиційний фактор ризику реформ – інерція системи і саботаж з боку чиновників.

Таким чином, не слід недооцінювати важливість таких процесів, як волонтерство і самоорганізація населення. Громадянська активність як тренд проявляється або в волонтерстві, або в участі в різних політичних і соціальних проектах, і, врешті-решт, сприяє залученню все більш широких верств населення до процесів самоорганізації на місцевому та загальнодержавному рівнях [1].

Отже, перед волонтерами та волонтерськими організаціями постає низка викликів, які ускладнюють їхню діяльність. Зменшення обсягів допомоги з боку громадян, діаспори та благодійників впливає на можливості забезпечення

необхідних ресурсів. Діяльність псевдоволонтерів та випадки зловживань підривають довіру до волонтерського руху, що ще більше ускладнює залучення підтримки. Відсутність ефективної координації між волонтерами призводить до дублювання зусиль та неефективного розподілу допомоги, а брак контролю за переданими ресурсами створює ризики їхнього нецільового використання [3].

Окрему проблему становить комунікація між волонтерами та суспільством, що впливає на швидкість та ефективність реагування на нагальні потреби. Додатково ускладнює ситуацію емоційне та фізичне вигорання волонтерів, адже тривала робота в умовах високого навантаження та стресу негативно позначається на їхній мотивації. Особливо важливою є безпека волонтерів, які здійснюють доставку допомоги в зони активних бойових дій [1].

На рівні державного регулювання також існує ряд проблем: законодавча база волонтерської діяльності часто не враховує реальних умов, що призводить до бюрократичних перепон, а консерватизм державних структур ускладнює реформування волонтерського руху та його ефективну інтеграцію в загальнодержавні процеси [2].

У зв'язку з цими викликами постає нагальна потреба у створенні централізованої системи обліку та координації волонтерської діяльності. Така система дозволить забезпечити ефективну взаємодію між волонтерами, уникнути дублювання ініціатив, посилити контроль за використанням ресурсів та підвищити прозорість і довіру до волонтерського руху. Крім того, централізована платформа сприятиме покращенню комунікації, забезпеченню безпеки волонтерів та зменшенню їхнього емоційного вигорання завдяки більш збалансованому розподілу навантаження. Впровадження подібної системи може стати важливим кроком у підвищенні ефективності волонтерської діяльності в умовах воєнного стану та майбутньої відбудови країни [1][3].

Тож створюється стартап системи управління волонтерською діяльністю. Структура стартапу для обліку волонтерської діяльності зображена на рисунку 1.1.

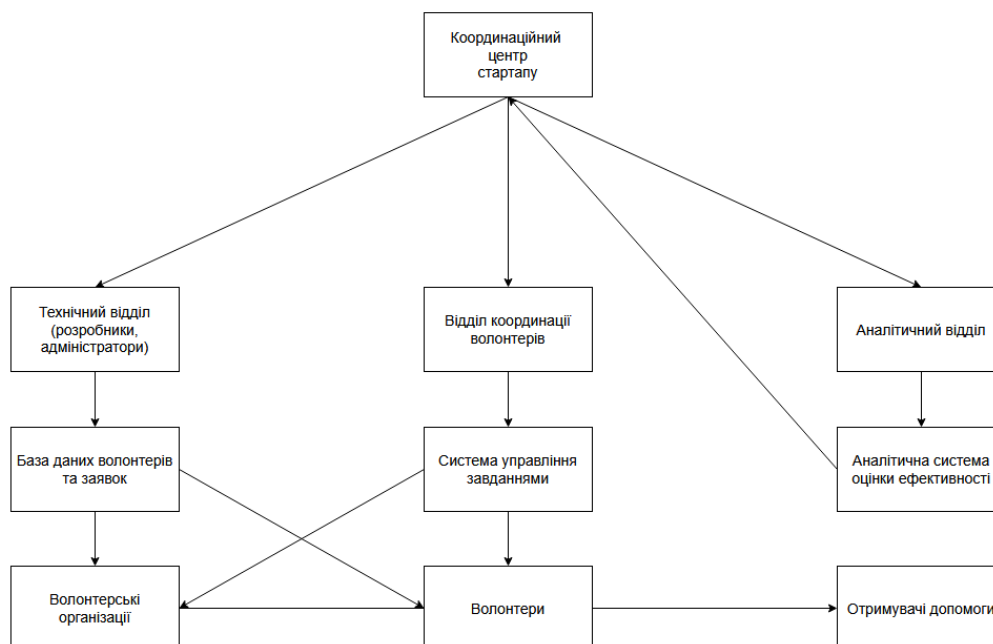


Рис. 1.1 – Структура стартапу обліку волонтерської діяльності

Структура стартапу, який забезпечує облік і координацію волонтерської діяльності організацій інституту, складається з кількох рівнів управління та функціональних модулів, що забезпечують ефективну взаємодію між усіма учасниками процесу.

На найвищому рівні знаходиться координаційний центр стартапу, який здійснює загальний контроль та нагляд за діяльністю волонтерської системи та функціонування сервісів.

Основними користувачами платформи є волонтерські організації, такі як ініціативні групи, а також окремі волонтери, які можуть долучатися до виконання завдань. Завдяки системі вони отримують доступ до координаційних інструментів, комунікаційних засобів та аналітичних даних, що сприяють більш

ефективній роботі. В кінцевому підсумку отримувачі допомоги, можуть швидко отримати необхідну підтримку від волонтерів.

Така структура дозволяє забезпечити організовану та прозору діяльність волонтерського руху, підвищує ефективність координації та сприяє максимальному охопленню потреб людей, які потребують допомоги.

На основі проведеного аналізу предметної області виникає потреба у визначенні вимог до інформаційної системи. Це дозволить забезпечити її відповідність практичним потребам користувачів та гарантувати належний рівень функціональності для ефективного управління волонтерською діяльністю.

1.2. Формування вимог до інформаційної системи

Для забезпечення ефективної та зручної роботи волонтерських організацій інформаційна система повинна відповідати певним вимогам. Вони охоплюють як технічні, так і користувацькі аспекти, що дозволяють створити надійний та продуктивний інструмент для координації волонтерської діяльності [10].

Вимоги користувачів

Одним із ключових факторів успішного впровадження системи є її зручність для кінцевих користувачів – волонтерів, адміністраторів та координаторів.

- Інтуїтивно зрозумілий інтерфейс – система повинна бути простою у використанні навіть для тих, хто не має спеціальної технічної підготовки. Навігація, доступ до основних функцій та візуальне оформлення повинні бути логічними і зрозумілими.

- Можливість доступу з будь-якого пристрою – у сучасних умовах важливо забезпечити гнучкий доступ до системи як із комп'ютерів, так і з

мобільних пристроїв. Це дозволить волонтерам швидко отримувати необхідну інформацію незалежно від місця перебування.

- Безпека даних та контроль доступу – система повинна гарантувати захист персональних даних користувачів, використовуючи сучасні механізми шифрування та багаторівневу систему доступу. Це дозволить уникнути витоку конфіденційної інформації та зловживань.
- Автоматизація процесів обліку та звітності – важливо мінімізувати ручну роботу, пов'язану з веденням звітності та обліком виконаних завдань. Автоматичне формування звітів, нагадування про завдання та інтеграція з іншими сервісами значно спростять адміністрування волонтерських ініціатив.

Системні вимоги

Інформаційна система повинна бути адаптивною та гнучкою, що дозволить їй ефективно працювати в різних умовах та легко масштабуватися відповідно до зростання кількості користувачів.

- Підтримка роботи у веб-браузерах – система має бути доступною через стандартні веб-браузери, що дозволить уникнути необхідності встановлення додаткового програмного забезпечення на пристрої користувачів.
- Масштабованість та можливість розширення функціоналу – система повинна підтримувати поступове розширення можливостей, наприклад, шляхом додавання нових модулів або інтеграції з іншими сервісами. Це дозволить у майбутньому розвивати функціонал без значних витрат на оновлення.

Функціональні вимоги

Щоб забезпечити ефективну взаємодію між волонтерами, організаціями та отримувачами допомоги, система повинна включати низку важливих функцій: Реєстрація та управління волонтерами – необхідно передбачити можливість створення облікових записів, перегляду інформації про волонтерів, їхні компетенції, історію виконаних завдань та рівень активності.

- Реєстрація та управління організаціями – волонтерські групи або окремі організації повинні мати змогу вести власний облік учасників, а також координувати діяльність у межах своєї структури.
- Призначення та контроль виконання завдань – система повинна дозволяти адміністраторам створювати завдання, розподіляти їх між волонтерами та контролювати статус виконання у реальному часі. Це забезпечить прозорість та ефективність волонтерської роботи.
- Аналітика та звітність про діяльність волонтерів – необхідно передбачити функції збору та аналізу статистичних даних щодо активності волонтерів, тривалості виконання завдань, рівня залученості та інших показників. Це допоможе організаціям оцінювати ефективність роботи та приймати обґрунтовані рішення.
- Інтеграція з месенджерами та сервісами комунікації – можливість сповіщення волонтерів про нові завдання, зміни у розкладі або важливі повідомлення через популярні месенджери (Telegram, Viber, WhatsApp) значно покращить рівень комунікації.

Вимоги до інформації

Оскільки система буде працювати з великим обсягом даних, необхідно забезпечити ефективне управління інформацією:

- Зберігання особистих даних волонтерів – передбачає надійний захист персональної інформації, з можливістю обмеження доступу до неї відповідно до ролей користувачів.
- Зберігання інформації про організації – кожна організація повинна мати власний профіль з детальним описом діяльності, контактною інформацією та списком учасників.
- Дані про наявні активності – у системі має бути передбачено ведення реєстру всіх заходів, ініціатив та подій, у яких беруть участь волонтери.

- База заявок на волонтерську допомогу – платформа повинна містити дані про запити на допомогу, що дозволить швидко розподіляти ресурси відповідно до потреб.

Вимоги до складу програмних засобів

Для повноцінного функціонування системи необхідно розділити її на серверну та клієнтську частини:

- Серверна частина – повинна містити REST API для взаємодії з клієнтськими додатками, а також базу даних для збереження всієї необхідної інформації.
- Клієнтська частина – включає веб-інтерфейс та мобільний додаток, які забезпечують доступ користувачів до основних функцій системи.

Вимоги до технічних засобів

Оскільки система має працювати у різних середовищах, важливо передбачити відповідні вимоги до обладнання:

- Хмарна або локальна серверна інфраструктура – можливість роботи як у хмарних середовищах (AWS, Azure, Google Cloud), так і на власних серверах організації.
- Підтримка мобільних пристроїв – система повинна коректно відображатися на мобільних пристроях та підтримувати адаптивний дизайн для зручності користувачів.

На основі проведеного аналізу було визначено основні вимоги до інформаційної системи управління волонтерською діяльністю. Її ключовими завданнями є створення бази даних волонтерської діяльності, автоматизація процесів координації та звітності, покращення комунікації між учасниками

волонтерського руху, а також впровадження аналітичних засобів для оцінки ефективності виконаної роботи.

Реалізація такої системи дозволить значно підвищити ефективність волонтерської діяльності, забезпечивши зручність, прозорість та оперативність взаємодії між усіма залученими сторонами.

На основі сформованих функціональних вимог необхідно розробити концепцію інформаційної системи що буде вирішувати проблеми визначені аналізом предметної області та відповідати сформованим вимогам.

1.3. Розробка концепції інформаційної системи

Концептуальна модель інформаційної системи управління волонтерською діяльністю зображена на рисунку 1.2.

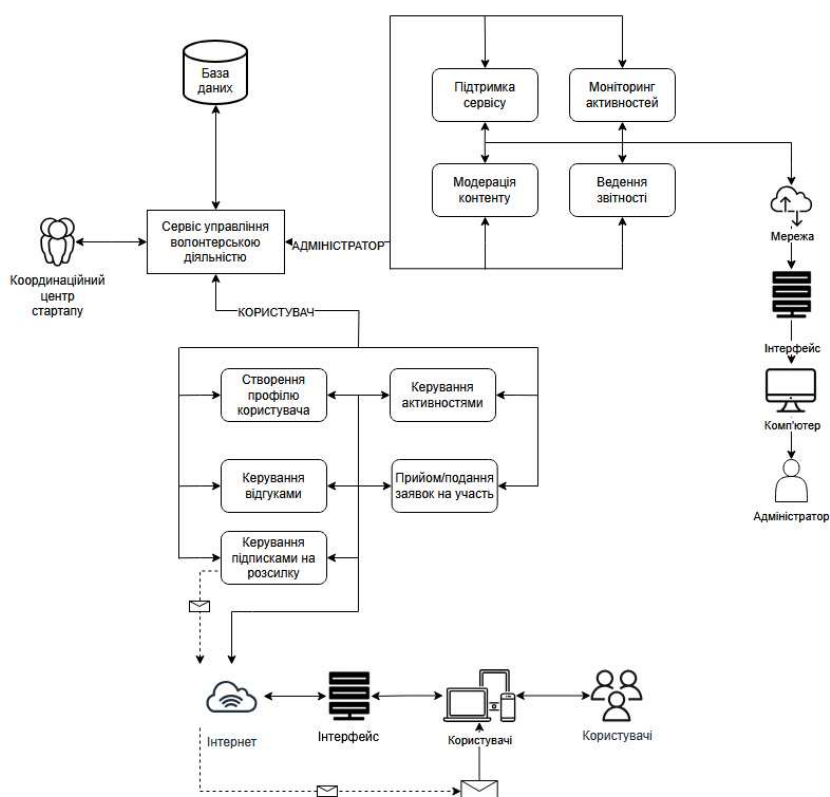


Рис. 1.2 – Концептуальна модель інформаційної системи

Розробка інформаційної системи управління волонтерською діяльністю передбачає впровадження різних рівнів доступу та функціональних можливостей для адміністраторів, волонтерських організацій та самих волонтерів. Це дозволить забезпечити гнучке управління, прозорість процесів та ефективність волонтерської роботи.

У системі передбачено три основні ролі користувачів: адміністратор, волонтерські організації та волонтери.

Передбачено роль адміністратора, який відіграватиме ключову роль у налаштуванні загальних параметрів системи та підтриманні її стабільності. Його функціонал включає: керування загальними налаштуваннями платформи, що дозволяє підтримувати її стабільну та безперебійну роботу; моніторинг активностей волонтерів та організацій з аналізом їх ефективності через вбудовані аналітичні модулі; контроль за модерацією контенту з метою запобігання порушенням правил платформи та забезпечення якісної взаємодії між користувачами.

Для волонтерських організацій передбачено широкий набір функцій для зручного управління своєю діяльністю. Функціонал організацій включає: створення активностей із можливістю публікації подій, ініціатив та потреб у волонтерах безпосередньо на платформі; прийом заявок від волонтерів із можливістю обробки запитів, перегляду профілів кандидатів та формування команд; керування активностями з моніторингом статусу завдань, оновленням деталей заходів та аналізом залучення волонтерів; налаштування власного профілю для ефективної презентації напрямків діяльності, контактної інформації та історії реалізованих проєктів.

Роль волонтера є ключовою для функціонування платформи, тому для волонтерів передбачено широкий спектр можливостей для зручної взаємодії із системою. Їх функціонал включає: подання заявок на участь у заходах та

виконання завдань; можливість підписки на сповіщення від організацій для оперативного інформування про нові події та зміни; перегляд усіх активностей із можливістю фільтрації за напрямками чи іншими критеріями; перегляд усіх зареєстрованих організацій та ознайомлення з їхньою діяльністю; публікацію, редагування та видалення відгуків про організації; налаштування власного профілю із зазначенням навичок та досвіду; встановлення користувацьких уподобань для отримання персоналізованих пропозицій.

При виборі технологій для розробки системи враховано низку факторів: продуктивність, безпека, зручність розгортання та підтримка сучасних стандартів веб-розробки, масштабованість [8].

Обрано базу даних, MySQL – реляційна база даних, яка відзначається високою швидкістю обробки запитів, гнучкістю та надійністю. Це дозволить зберігати великі обсяги інформації про волонтерів, організації, активності та звіти [19].

Для серверної частини обрано Java у поєднанні з Spring Framework. Це забезпечує високу продуктивність, гнучкість та безпеку сервісу. Використання Spring дозволить легко реалізувати REST API, що зробить систему модульною та масштабованою, для безпеки застосунку використовується Spring security + JWT [13][24].

Для розробки клієнтської частини обрано React Framework з TypeScript. Ця комбінація забезпечує високу продуктивність веб-застосунку, зручність у розробці та легкість у подальшій підтримці. TypeScript дозволить зменшити кількість помилок у коді та зробить систему більш стабільною, дозволить працювати з типізованими змінними що спростить розробку застосунку [28] [29].

Для забезпечення зручності розробників було передбачено можливість роботи в будь-якому середовищі, сумісному із засобами розробки – будь то

Windows, macOS або Linux. Це дозволяє кожному учаснику команди використовувати зручні інструменти для роботи.

Застосування програмного забезпечення з відкритим кодом (open-source) дозволяє уникнути значних витрат на комерційні ліцензії, що є особливо важливим для стартапів і неприбуткових організацій. Відкриті технології, такі як Linux, Docker, Kubernetes, MySQL, Redis та багато інших, надають потужні інструменти для розробки, розгортання та управління інформаційними системами без необхідності сплачувати за використання.

Крім того, активні спільноти розробників підтримують ці технології, що дозволяє швидко отримувати оновлення, виправлення вразливостей та нові можливості без додаткових витрат. Також використання відкритого ПЗ спрощує інтеграцію з іншими сервісами та дозволяє легко адаптувати систему під специфічні потреби, оскільки вихідний код доступний для модифікації. У довгостроковій перспективі це значно знижує експлуатаційні витрати та робить проєкт більш фінансово стійким.

Для забезпечення гнучкості, масштабованості та спрощеного розгортання інформаційної системи з управління волонтерською діяльністю прийнято рішення використовувати контейнеризацію. У рамках цього підходу кожен мікросервіс (наприклад, основний сервіс обробки даних, аналітичний модуль, рекомендаційна система тощо) буде розгортатися у вигляді окремого контейнера, ізольованого від решти середовища. Це дозволяє досягти високого рівня автономності, повторюваності середовища та спрощення процесів CI/CD (безперервної інтеграції та доставки).

Docker — найпопулярніший інструмент для створення, розгортання та управління контейнерами. Завдяки йому можливо «упакувати» застосунок разом з усіма залежностями (бібліотеками, системними утилітами, конфігураціями) в один контейнер, що гарантує ідентичність роботи у будь-якому середовищі [22].

Docker Compose — інструмент для конфігурації та запуску мультиконтейнерних середовищ. З його допомогою легко організувати одночасний запуск кількох сервісів (наприклад, бекенду, фронтенду, бази даних), задаючи їх взаємозв'язки в одному YAML-файлі [22].

Kubernetes — система оркестрації контейнерів, яка буде впроваджена на наступних етапах розвитку проєкту для автоматичного масштабування, балансування навантаження та управління станом сервісів у кластері [34].

Для розгортання інформаційної системи було обрано віддалений сервер, наданий компанією CloudHoster. Дана організація спеціалізується на наданні серверних ресурсів для розробників та малих бізнесів, пропонуючи високий рівень доступності, безпеки та можливості масштабування. Сервери компанії базуються на сучасному обладнанні з підтримкою контейнеризації та розподілених обчислень, що дозволяє ефективно використовувати ресурси для розміщення веб-застосунків.

CloudHoster надає можливість орендувати як віртуальні приватні сервери (VPS), так і фізичні виділені сервери, залежно від потреб проєкту. Завдяки підтримці різних операційних систем та зручним інструментам для адміністрування серверів, ця платформа є оптимальним вибором для розгортання інформаційної системи управління волонтерською діяльністю.

У якості операційної системи для розгортання веб-застосунку було обрано Linux (Ubuntu 24.0.2 LTS). Це рішення було прийнято з огляду на високу стабільність, безпеку та можливість безкоштовного використання без необхідності витрат на ліцензії [34].

Технічні вимоги до серверного обладнання

Для стабільної роботи веб-застосунку необхідний сервер з наступними характеристиками:

- Оперативна пам'ять (RAM): не менше 8 ГБ (рекомендовано 16 ГБ для підвищення продуктивності).
- Процесор: мінімум 4 ядра (рекомендовано 8 ядер).
- Місце на диску: від 100 ГБ SSD для зберігання бази даних та логів.
- Мережева пропускна здатність: не менше 1 Гбіт/с для стабільного доступу.

Програмні компоненти

- Операційна система: Linux (Ubuntu 24.0.2 LTS) — обрана через стабільність, безпеку та безкоштовне ліцензування.
- Версія JDK: OpenJDK 17, що забезпечує сумісність з Spring Boot 3.x.
- База даних: MySQL, оптимізована для роботи з великим обсягом даних та забезпечення цілісності транзакцій.
- Docker та Docker Compose: використовується для контейнеризації серверної частини та спрощення розгортання.
- Фронтенд: Node.js версії 18 для побудови та розгортання застосунку на базі React з TypeScript.
- Веб-сервер: Nginx — використовується як реверсний проксі та для обробки статичних ресурсів.

Висновок

Розроблена концепція дозволяє реалізувати інформаційну систему управління волонтерською діяльністю організацій в інституті відповідно до її цільового призначення. Обрані технології та функціональні можливості дозволяють створити потужну, зручну та масштабовану систему управління волонтерською діяльністю. Вона забезпечить комфортне використання для всіх категорій користувачів, автоматизує рутинні процеси, покращить комунікацію та дозволить ефективно оцінювати результати волонтерської роботи.

РОЗДІЛ 2. Розробка технічного проекту інформаційної системи управління волонтерською діяльністю

2.1. Загальносистемні рішення

На основі аналізу вимог було сформовано концептуальну модель даних, що відображає основні сутності та зв'язки між ними. Подальший вибір архітектурного підходу ґрунтувався на потребах у надійності, масштабованості та простоті підтримки системи.

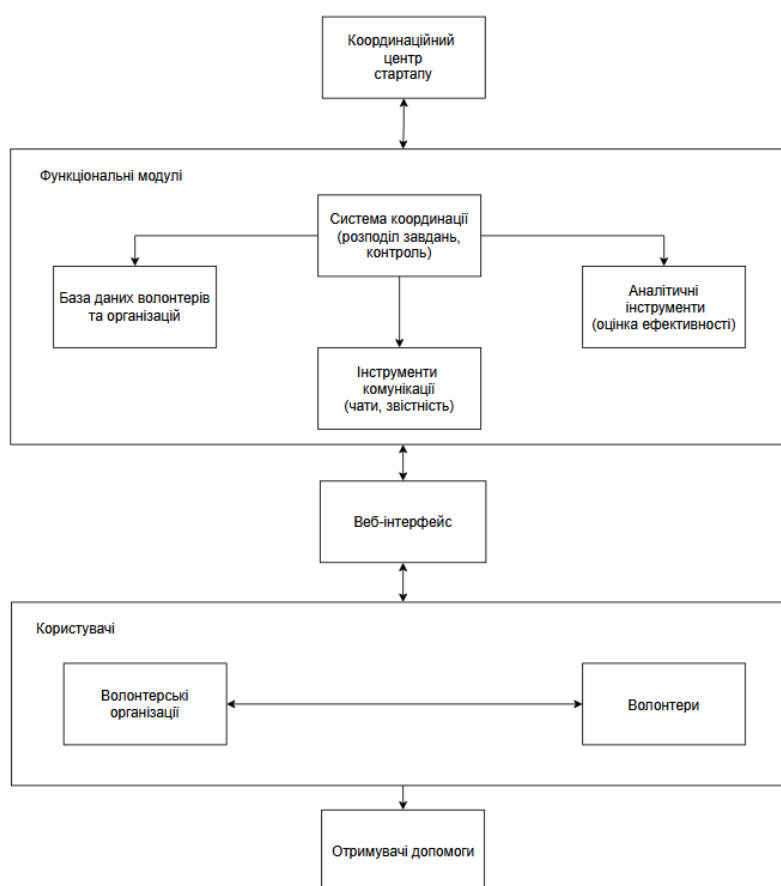


Рис. 2.1 – Концепція системи управління волонтерською діяльністю

На верхньому рівні розташовується координаційний центр стартапу, що виконує стратегічну функцію — визначення загального напрямку розвитку системи, функції управління та організації роботи внутрішніх модулів. Саме він контролює реалізацію технічної, координаційної та аналітичної частин платформи.

У центральній частині діаграми представлено основні функціональні компоненти системи, які забезпечують виконання ключових бізнес-процесів. Усі вони взаємодіють між собою та працюють як єдиний механізм:

Система координації відповідає за розподіл завдань, моніторинг їх виконання та контроль за активністю волонтерів та організацій. Це ядро функціональної архітектури, через яке проходить більшість дій користувачів.

База даних волонтерів та організацій забезпечує централізоване зберігання структурованої інформації про всіх зареєстрованих користувачів, включаючи волонтерські об'єднання, окремих волонтерів та їх участь у діяльності.

Інструменти комунікації дозволяють організовувати зв'язок між учасниками платформи. Передбачено використання внутрішніх чатів, посилань на соціальні мережі, а також відображення історії участі у подіях.

Аналітичні інструменти. Реалізовано модулі оцінки ефективності — окремі сервіси, які аналізують активність, рівень залученості користувачів, популярність заходів і виконання завдань. Дані використовуються адміністрацією та організаціями для прийняття управлінських рішень.

Усі функціональні модулі інтегруються у веб-інтерфейс, який є єдиним засобом взаємодії користувача з системою. Через нього здійснюється авторизація, заповнення профілів, вибір активностей, комунікація та перегляд аналітичних звітів.

Внизу діаграми відображено кінцевих користувачів, які взаємодіють із системою через веб-інтерфейс:

Волонтерські організації - це зареєстровані об'єднання, які розміщують заходи й управляють волонтерами у своїх проєктах;

Волонтери - фізичні особи, що приєднуються до діяльності, виконують завдання, залишають відгуки й формують особисту статистику активності;

Отримувачі допомоги - кінцеві бенефіціари, яким надається підтримка — безпосередньо не взаємодіють із платформою, проте є причиною залучення волонтерських ресурсів.

У такій формі структура системи дозволяє забезпечити логічне розділення обов'язків, централізоване управління, ефективну взаємодію користувачів і масштабованість з можливістю подальшого розвитку.

У межах дипломного проєкту прийнято рішення використовувати монолітну архітектуру для реалізації інформаційної системи управління волонтерської діяльності. Цей підхід є класичним варіантом побудови програмних систем, у якому вся бізнес-логіка, взаємодія з базами даних, обробка запитів, а також API інтерфейси розміщуються в одному програмному блоці та розгортаються як єдине цілісне рішення [4].

Обрання монолітної архітектури зумовлено кількома ключовими факторами:

1. Простота розробки: моноліт значно легше реалізувати на початкових етапах, не витрачаючи додатковий час на організацію взаємодії між сервісами, оркестрацію чи розподіл логіки;
2. Менша складність у розгортанні: достатньо єдиного серверного середовища або контейнера, що містить весь застосунок, без необхідності налаштування мережі взаємодії між сервісами;
3. Ефективне використання ресурсів на невеликих або середніх навантаженнях [4].

Зважаючи на широке застосування та адаптивність монолітного підходу, варто відзначити, що навіть у межах цієї архітектури існують різні підходи до її

реалізації. Кожен з них має свої особливості, переваги та недоліки, що дозволяє обрати найоптимальніший варіант залежно від цілей та масштабів проєкту [4]. Нижче розглянуто основні типи монолітної архітектури, які можуть бути застосовані на практиці:

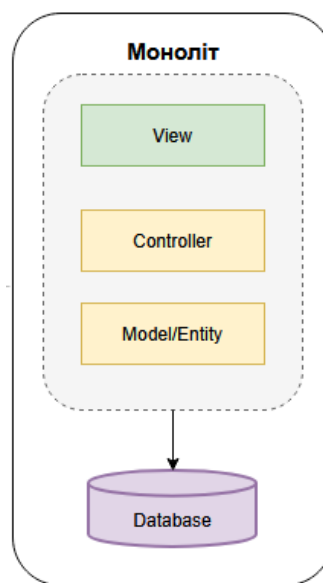


Рис. 2.2 – Монолітна архітектура

Монолітна архітектура передбачає об'єднання всіх складових програми — від інтерфейсу користувача та бізнес-логіки до доступу до даних — в одну єдину кодову базу та репозиторій. У ній часто використовуються шаблони, теми оформлення та патерн проектування Model-View-Controller (MVC). Такий підхід забезпечує високу швидкість розробки й впровадження, адже весь функціонал розташований у межах однієї структури. Це також сприяє повторному використанню коду — моделі та сутності можна застосовувати в різних частинах проєкту, що підвищує ефективність розробки. Крім того, єдина архітектура стимулює уніфікований стиль написання коду, що полегшує його читання й розуміння [5].

Втім, по мірі зростання проекту зростає і складність його підтримки. Обслуговування великого обсягу коду стає дедалі дорожчим і заплутанішим. Крім того, зміни в одній частині системи можуть викликати непередбачувані наслідки в інших, що створює ризики для стабільності. І нарешті, масштабування такої архітектури, зокрема горизонтальне, є складним, оскільки всі компоненти тісно пов'язані між собою [6].

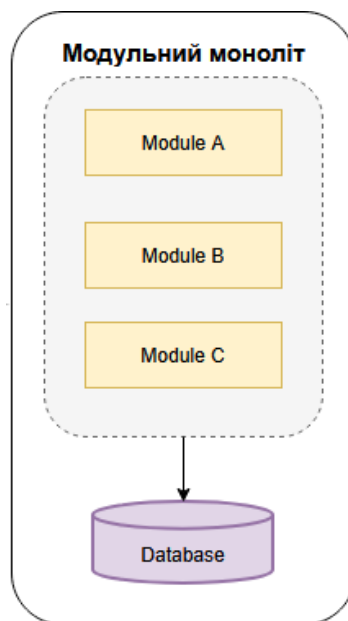


Рис. 2.3 – Модульний моноліт

Модульний моноліт відрізняється від класичного моноліту тим, що розділяє фронтенд і бекенд, які взаємодіють через API. Внутрішня логіка бекенду організовується у вигляді модулів, кожен з яких об'єднує пов'язані функції. Такий підхід забезпечує чіткий розподіл обов'язків, що, у свою чергу, покращує керованість кодовою базою, полегшує супровід і прискорює процес впровадження нових функціональних можливостей [6].

Попри більш структурований підхід, модульний моноліт все ще може страждати від побічних ефектів через спільну кодову базу. Крім того, коли всі бекенд-функції працюють одночасно, можуть виникати проблеми з продуктивністю. Взаємозалежність між модулями також ускладнює

горизонтальне масштабування, оскільки будь-яка зміна може вплинути на інші частини системи.

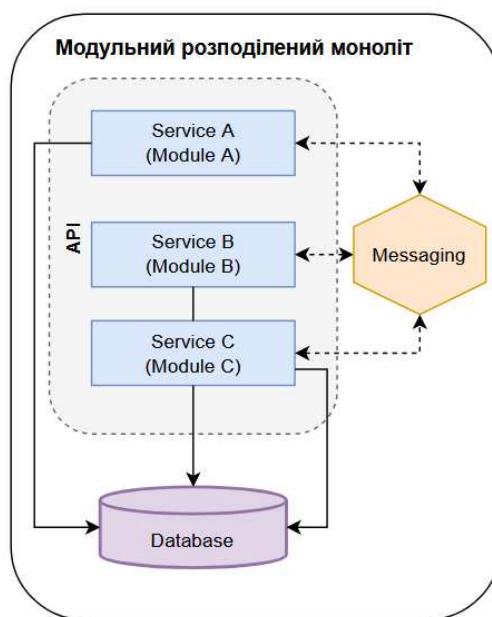


Рис. 2.4 – Модульний розподілений моноліт

Цей архітектурний підхід є розвитком ідеї модульного моноліту. Він передбачає розділення сервісів за функціональними ознаками: API, робітники (виробники/споживачі) тощо. Взаємодія між сервісами здійснюється за допомогою систем обміну повідомленнями, таких як RabbitMQ або Kafka, а обробка запитів і маршрутизація відповідей відбувається через API-шлюз [6].

Модульний розподілений моноліт поєднує переваги модульного підходу зі здатністю до більш гнучкого горизонтального масштабування. Компоненти можна використовувати повторно в різних частинах системи, що підвищує її ефективність. Така архітектура також є хорошою основою для подальшого переходу до мікросервісної моделі. Однак навіть попри поділ сервісів, спільна кодова база може призводити до небажаних побічних ефектів. До того ж незалежна розробка окремих частин системи може ускладнюватися через міжкомандні залежності [6].

З огляду на переваги та недоліки кожного з архітектурних підходів, для реалізації проєкту було обрано модульний моноліт як оптимальний баланс між структурністю, зрозумілою організацією коду та можливістю масштабування в майбутньому.

Моноліт реалізований із використанням Spring Boot як основного фреймворку для серверної частини. Spring Boot надає вбудовані засоби для роботи з REST API, ORM, безпекою, валідацією, обробкою запитів та конфігурацій, що дозволило сконцентрувати увагу на логіці взаємодії між користувачами та волонтерськими організаціями [17][24].

Для клієнтської частини обрано архітектурний підхід SPA (Single Page Application) у поєднанні з REST API.

SPA — це односторінковий веб-додаток, який динамічно оновлює вміст без перезавантаження сторінки. Такий підхід забезпечує більш плавну взаємодію користувача із системою, швидшу роботу інтерфейсу та зниження навантаження на серверну частину [15].

У проєкті SPA реалізовано за допомогою React — сучасного JavaScript-фреймворку для побудови інтерфейсів, яка дозволяє організовувати компонентну структуру, ефективну маршрутизацію (React Router), а також розділяти інтерфейс на окремі логічні частини.

REST (Representational State Transfer) — архітектурний стиль, який дозволяє будувати гнучкі веб-сервіси за допомогою HTTP-протоколу. У проєкті реалізовано набір REST-ендпоінтів для роботи з волонтерами, організаціями, подіями та аналітичними запитами. Усі відповіді повертаються у форматі JSON, що забезпечує простоту споживання даних з боку фронтенду [16].

Монолітна система поділяється на наступні функціональні блоки:

- Модуль користувача - управління реєстрацією, авторизацією, особистим кабінетом, інтересами волонтерів;
- Модуль організацій - створення, редагування, модерування організацій та їхніх подій;
- Модуль подій (активностей) - публікація заходів, реєстрація волонтерів, фільтрація за категоріями;
- Модуль аналітики - базова статистика по участі волонтерів, популярності заходів та внеску організацій;
- Модуль адміністратора - управління ролями, перевірка достовірності інформації, модерування.

Усі ці компоненти знаходяться в одному середовищі, що дає змогу ефективно управляти зв'язками між ними та скоротити кількість зовнішніх залежностей.

Хоча на поточному етапі обрана монолітна структура є оптимальною, у разі зростання навантаження або ускладнення бізнес-логіки система легко може бути рефакторингована в мікросервіси, оскільки логічні межі між модулями вже закладено у структурі коду. Наприклад, модулі аналітики чи рекомендаційної системи можуть бути винесені в окремі сервіси [15].

2.2. Рішення по інформаційному забезпеченню

У межах реалізації інформаційної системи для управління волонтерською діяльністю прийнято рішення використовувати систему керування базами даних MySQL. Цю СКБД було обрано з декількох причин:

Продуктивність - MySQL має високу швидкість виконання запитів, завдяки оптимізованому рушію та гнучкій структурі таблиць. Це дозволяє зменшити час відповіді бекенду при великій кількості запитів одночасно [19];

Стабільність і надійність - MySQL використовується багатьма великими компаніями, як-от Facebook, Google, Twitter, що свідчить про її перевірену часом стабільність та ефективність [20];

Транзакційність - Завдяки використанню рушія InnoDB, MySQL підтримує транзакції з дотриманням принципів ACID (атомарність, узгодженість, ізолюваність, довговічність), що критично важливо для забезпечення цілісності даних у багатокористувацькому середовищі [21];

Масштабованість - База може бути легко масштабована як вертикально (збільшення ресурсів сервера), так і горизонтально (через кластеризацію або реплікацію), що дозволяє розвивати систему без втрати продуктивності [19];

Підтримка сучасних інструментів - MySQL повністю інтегрується з фреймворком Spring Boot, який використовується в бекенді нашого проєкту. Також підтримуються ORM-інструменти, такі як Hibernate, що полегшує роботу з БД;

Доступність і поширеність - MySQL має широке ком'юніті, офіційну документацію та велику кількість освітніх матеріалів, що значно полегшує навчання та розв'язання проблем у процесі розробки [18][20].

MySQL дозволяє розпочати проєкт із мінімальними витратами завдяки відкритій ліцензії, а в майбутньому – масштабувати рішення під більшу кількість користувачів або у хмарну інфраструктуру.

2.2.1. Розробка концептуальної моделі даних

Концептуальна модель бази даних - це абстрактне представлення структури та взаємозв'язків між даними в базі даних без звернення до конкретних технічних деталей реалізації бази даних. Вона описує сутності (або таблиці), що представляють дані, та зв'язки між ними, але не визначає конкретні технічні аспекти, типи даних.

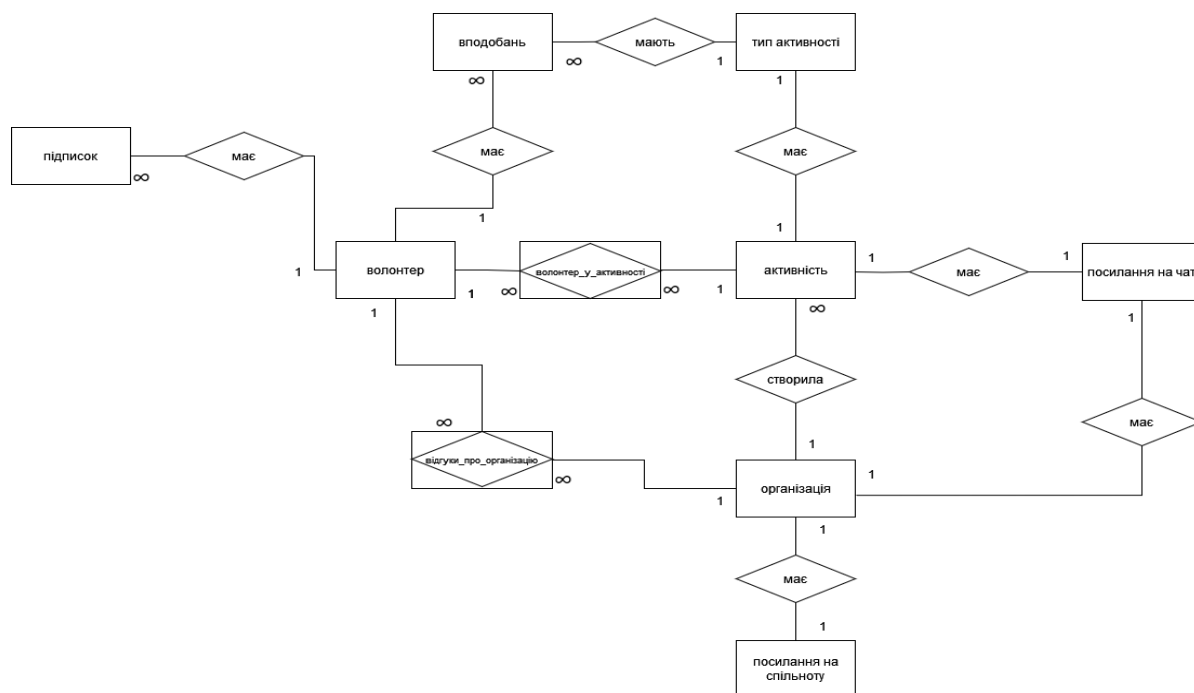


Рис. 2.5 – ER діаграма сутностей бази даних

У моделі даних інформаційної системи підтримки волонтерської діяльності центральною сутністю є волонтер, який бере участь у різноманітних зв'язках. Зокрема, він може мати кілька вподобань, що реалізується через проміжну таблицю «має», яка дозволяє створювати зв'язки типу багато-до-багатьох між волонтерами та вподобаннями. Подібним чином, волонтер може бути залученим до кількох активностей, що моделюється через зв'язок «бере участь у діяльності», також із типом багато-до-багатьох. Окрім цього, волонтери можуть підписуватися на організації, формуючи багатозв'язні відносини через сутність «підписки».

Сутність організація виступає джерелом створення активностей: одна організація може створити кілька заходів або подій. Також кожна організація може мати кілька посилань на спільноти, що дає змогу підтримувати присутність у соціальних мережах і покращувати комунікацію з волонтерами.

Активність у цій системі є окремим об'єктом, який пов'язується як із волонтерами (через їхню участь), так і з організацією, що її створила. Крім цього,

кожна активність має прив'язку до певного типу активності, що визначається через окрему допоміжну сутність. До того ж, активність може містити посилання на чат, що дозволяє учасникам швидко комунікувати між собою.

Допоміжні зв'язки, які мають додаткові атрибути, включають «підписку», що описує відношення між волонтером і організацією, «бере участь у діяльності», який також може містити, наприклад, дату приєднання до активності, та «має (вподобання)» і «має (тип активності)», які реалізують відповідні асоціації між об'єктами системи.

Сформована ER-модель забезпечує гнучку багаторівневу структуру даних, яка дозволяє повноцінно відстежувати участь волонтерів, враховувати їхні інтереси, керувати активностями, здійснювати ефективну комунікацію та реалізовувати аналітику для прийняття обґрунтованих рішень.

2.2.2. Розробка логічної моделі

Сутність "Організації" має зв'язки з наступними сутностями:

Діяльності : одна організація може мати багато діяльностей; зв'язок один до багатьох.

Відгуки : одна організація може мати багато відгуків; зв'язок один до багатьох.

Посилання групи організацій: одна організація може мати багато посилань на групи організацій; зв'язок один до багатьох.

Підписки волонтерів : одна організація може мати багато підписок волонтерів; зв'язок один до багатьох.

Сутність "Активності" має зв'язок з сутністю "Організації": багато активностей можуть бути пов'язані з однією організацією; зв'язок один до багатьох.

Сутність "Посилання чату активності" має зв'язок з сутністю "Організації" і "Активність" : кожне посилання чату активностей пов'язане з однією організацією та однією діяльністю; зв'язок один до одного.

Сутність "Відгуки" має зв'язок з сутностями "Організації" і "Волонтери": кожен відгук пов'язаний з однією організацією і одним волонтером; зв'язок один до одного.

Сутність "Волонтери" має зв'язки з наступними сутностями:

Відгуки : один волонтер може мати багато відгуків; зв'язок один до багатьох.

Волонтерські вподобання: один волонтер може мати багато вподобань; зв'язок один до багатьох.

Волонтери у діяльності: один волонтер може брати участь у багатьох діяльностях; зв'язок один до багатьох.

Підписки волонтерів: один волонтер може мати багато підписок на організації; зв'язок один до багатьох.

У результаті трансформації концептуальної моделі БД сутності будуть виглядати наступним чином (відображені тільки ті сутності що потребують додаткових зв'язків з ключами інших сутностей, та створені додаткові «проміжні» сутності) зображено у табл. 2.1:

Таблиця 2.1

Проміжні сутності БД

№	Назва сутності	Атрибути	Призначення
1	Активності	ІД активності, Місто, Країна, Заголовок, Опис,	Зберігає основну інформацію про волонтерські активності

		ID виду активності, Дата створення	
2	Посилання чату активності	ID посилання чату, ID організації, ID активності, Соціальна мережа, Посилання	Містить посилання на чати, пов'язані з активностями
3	Відгуки	ID відгуку, Оцінка, Опис, ID волонтера, ID організації	Зберігає оцінки та коментарі від волонтерів
4	Посилання на спільноту організацій	ID посилання групи, ID організації, Соціальна мережа, Посилання	Містить зовнішні посилання на соціальні сторінки організацій
5	Вподобання	ID вподобання, Назва	Категорії вподобань, що цікавлять волонтерів
6	Волонтерські вподобання	ID волонтера, ID вподобання	Проміжна сутність для реалізації зв'язку «багато-до-багатьох»
7	Волонтери у діяльності	ID волонтера, ID активності, Дата вступу	Проміжна сутність для фіксації участі волонтерів в активностях
8	Підписки волонтерів	ID волонтера, ID організації	Проміжна сутність для фіксації підписок волонтерів на організації

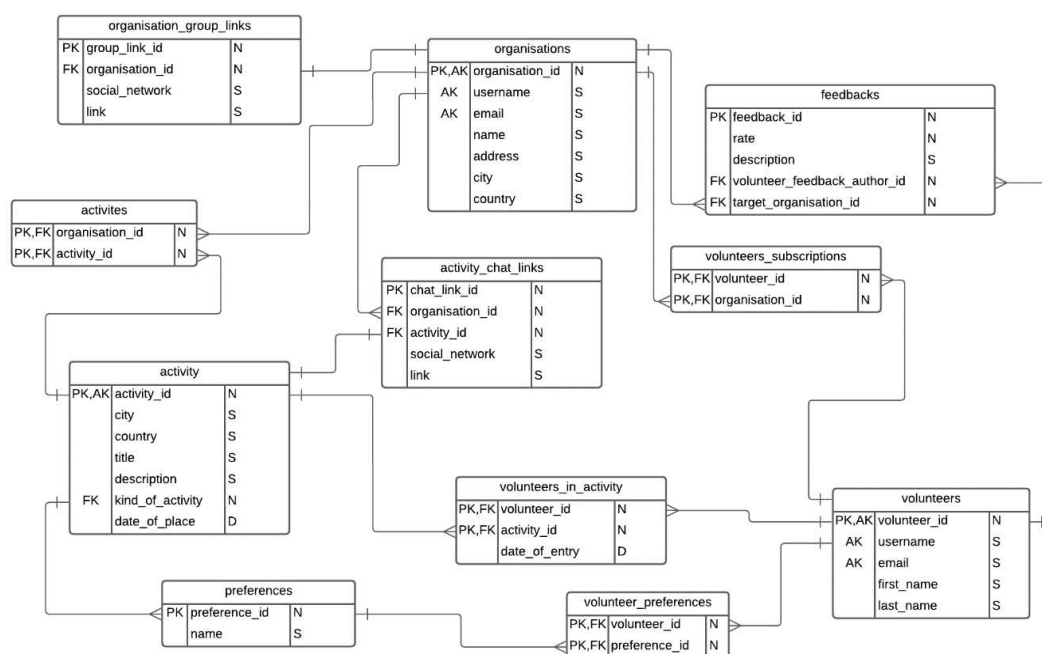


Рис. 2.6 – ERD діаграма зв'язків сутностей

У результаті розглядання ERD діаграми були виявлені зайві залежності такі як `activity_chat_links`, `organisation_group_links`, ці сутності, були перенесені в таблицю `activity` та таблицю `organisations` відповідно.

Також у результаті аналізу отриманих результатів було виявлено функціональну залежність в таблицях `organisations` та `activity`, у функціональній залежності поля `country` та `city`, для виправлення цієї ситуації буде створена таблиця словник, що буде містити в собі країну та місто, що вирішить не тільки проблему функціональної залежності (де місто задає країну), а й вирішить проблему пошуку чи додавання міст яких може не існувати в БД.

У таблиці `organizations` існує атрибут `Address` що є складним, тобто складається з декількох даних, тож було вирішено додати додаткову таблицю, що буде містити адресу і мати декілька атрибутів що будуть відповідати цій таблиці, що дозволить здійснювати пошук за адресами.

2.2.3. Розробка фізичної моделі даних

Етап розробки фізичної моделі бази даних є важливим етапом у процесі створення рішення і базується на вже розробленій концептуальній моделі. На цьому етапі відбувається перетворення концептуальної моделі в реальні таблиці та зв'язки між ними. На цьому етапі необхідно зробити нормалізацію відношень.

Нормалізація відношень - це процес структурування даних у таблицях з метою усунення аномалій та забезпечення ефективної організації даних. Для перевірки того, чи відповідає база даних третій нормальній формі (3НФ), необхідно перевірити, чи відповідають всі таблиці вимогам усіх нормальних форм, включаючи першу (1НФ) та другу (2НФ).

У першій нормальній формі (1НФ) кожна комірка таблиці має містити лише одне значення, а кожна колонка має унікальне ім'я.

Друга нормальна форма (2НФ) вимагає, щоб кожен неключовий атрибут у таблиці залежав від цілого первинного ключа.

Таблиця вважається в третій нормальній формі (3НФ), якщо вона відповідає вимогам другої нормальної форми і жоден неключовий атрибут не залежить від іншого неключового атрибута. Іншими словами, немає транзитивної залежності між неключовими атрибутами.

Для нормалізації відношень і об'єднання зв'язків у таблиці були створені наступні сутності:

`volunteer_preferences`: Ця таблиця дозволяє відображати зв'язок між волонтерами та їх вподобаннями. Вона вирішує проблему багато-до-багатьох відношень між волонтерами та їх вподобаннями, дозволяючи кожному волонтеру мати багато вподобань, а також кожному вподобанню належати до багатьох волонтерів.

`activity_chat_links`: Ця таблиця дозволяє відображати зв'язок між організаціями, активностями та посиланнями на чати. Вона вирішує проблему багато-до-багатьох відношень між організаціями та їх активностями, а також між активностями та посиланнями на чати.

`feedbacks`: Ця таблиця дозволяє відображати зв'язок між волонтерами, організаціями та зворотним зв'язком. Вона вирішує проблему багато-до-багатьох відношень між волонтерами та їх зворотніми зв'язками, а також між організаціями та зворотніми зв'язками.

Ці таблиці використовуються для створення відносин між сутностями, що допомагає уникнути аномалій бази даних та забезпечує більш ефективну організацію та структуру бази даних. Опис атрибутів сутностей представлений у табл. 2.2:

Таблиця 2.2

Опис атрибутів сутностей

Сутність активність				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_активності	activity_id	INT	NOT NULL, AUTO_INCREMENT	PRIMARY KEY
місто	city	VARCHAR (45)		
країна	country	VARCHAR (45)		
назва	title	VARCHAR (45)		
Опис_активності	description	VARCHAR (45)		
Вид_діяльності	kind_of_activity	VARCHAR (45)		
Сутність активності				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_організації	organisation_id	INT	NOT NULL	
ID_активності	activity_id	INT	NOT NULL, AUTO_INCREMENT	PRIMARY KEY
Сутність посилання на чат активності				

Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_посилання	chat_link_id	INT	NOT NULL, AUTO_INCREMENT	PRIMARY KEY
ID_організації	organisation_id	INT		
ID_активності	activity_id	INT		
Назва_соц_мережі	social_network	VARCHAR (45)	NOT NULL	
Посилання_на_чат	link	VARCHAR (225)	NOT NULL	
Сутність відгуки				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_відгуку	feedback_id	INT	NOT NULL, AUTO_INCREMENT	PRIMARY KEY
оцінка	rate	INT	NOT NULL	
Описання_відгуку	description	VARCHAR (225)	NOT NULL	
ID_автора_фідбеку	volunteer_feedback_author_id	INT		
Організація_для_якої_пишуть_відгук	target_organisation_id	INT		
Сутність посилання на спільноту організації				

Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_посилання	group_link_id	INT	NOT NULL, AUTO_INCREMENT	PRIMARY KEY
ID_організації	organisation_id	INT		
Назва_соц_мережі	social_network	VARCHAR (45)		
Посилання	link	VARCHAR (225)		
Сутність організації				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_Організації	organisation_id	INT	NOT NULL, AUTO_INCREMENT	PRIMARY KEY
Ім'я_користувача	username	VARCHAR (45)		
Пошта	email	VARCHAR (45)		
Назва_компанії	name	VARCHAR (45)		
Адреса	address	VARCHAR (45)		
Місто	city	VARCHAR (45)		
Країна	country	VARCHAR (45)		

Сутність вподобання				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_вподобання	preference_id	INT	NOT NULL, AUTO_INCREMENT	PRIMARY KEY
Назва_вподобання	name	VARCHAR (80)	NOT NULL	
Сутність волонтерські вподобання				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_волонтера	volunteer_id	INT	NOT NULL	
ID_вподобання	preference_id	INT	NOT NULL	
Сутність волонтери				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_волонтера	volunteer_id	INT	NOT NULL, AUTO_INCREMENT	PRIMARY KEY
Ім'я_користувача	username	VARCHAR (45)	NOT NULL	
Пошта	email	VARCHAR (45)		
Справжнє_ім'я	first_name	VARCHAR (45)		

Прізвище	last_name	VARCHAR (45)		
Сутність волонтери у активності				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_волонтера	volunteer_id	INT	NOT NULL	
ID_активності	activity_id	INT	NOT NULL	
Дата_вступу	date_of_entry	DATETIME		
Сутність підписки волонтерів				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
ID_волонтера	volunteer_id	INT	NOT NULL	
ID_організації	organisation_id	INT	NOT NULL	

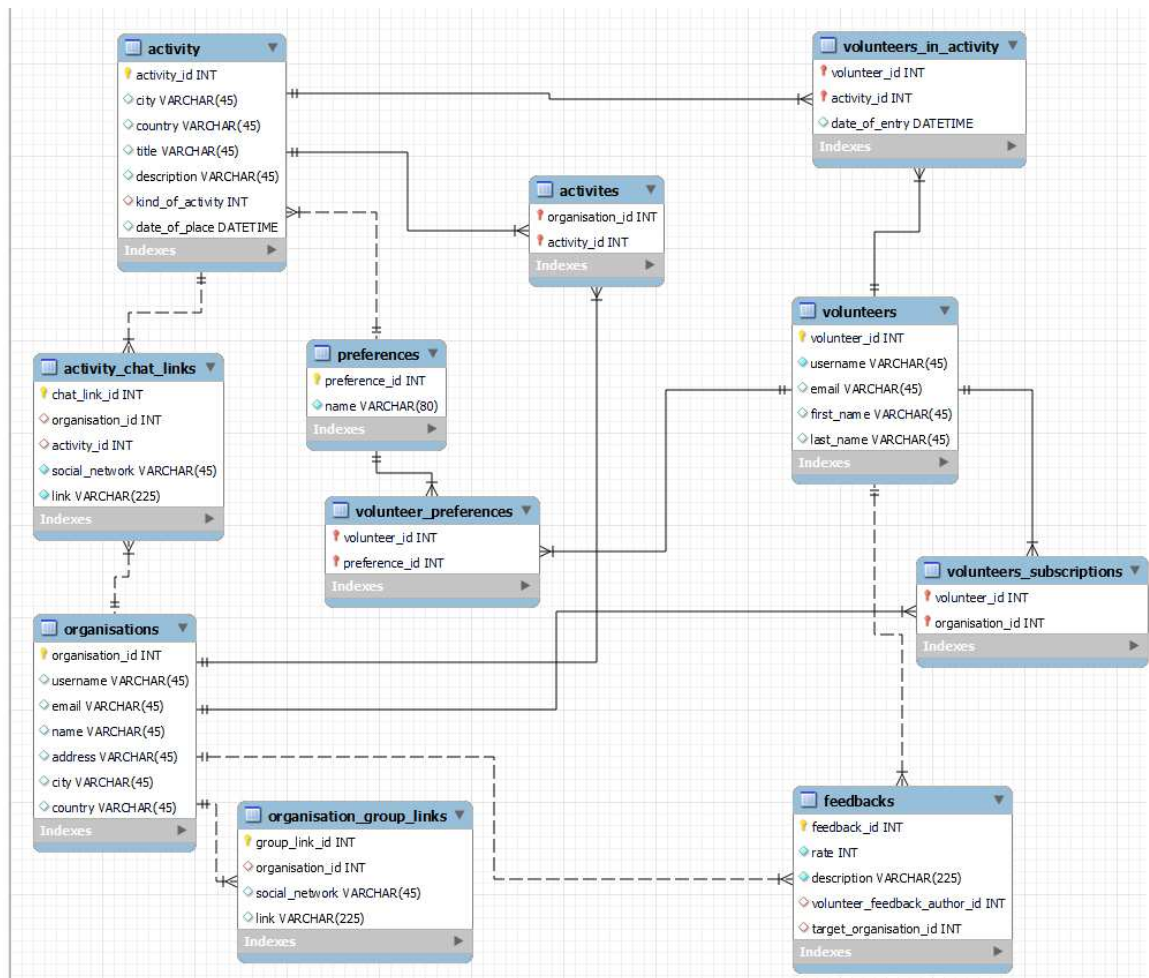


Рис. 2.7 – EER діаграма БД

На зображенні представлено EER-діаграму (Enhanced Entity-Relationship), яка ілюструє логічну структуру бази даних інформаційної системи управління волонтерської діяльності. Діаграма описує основні сутності системи, такі як волонтери, організації, активності, уподобання, а також зв'язки між ними. Вона слугує візуальним представленням структури даних і допомагає формалізувати логіку збереження та взаємодії об'єктів у базі даних.

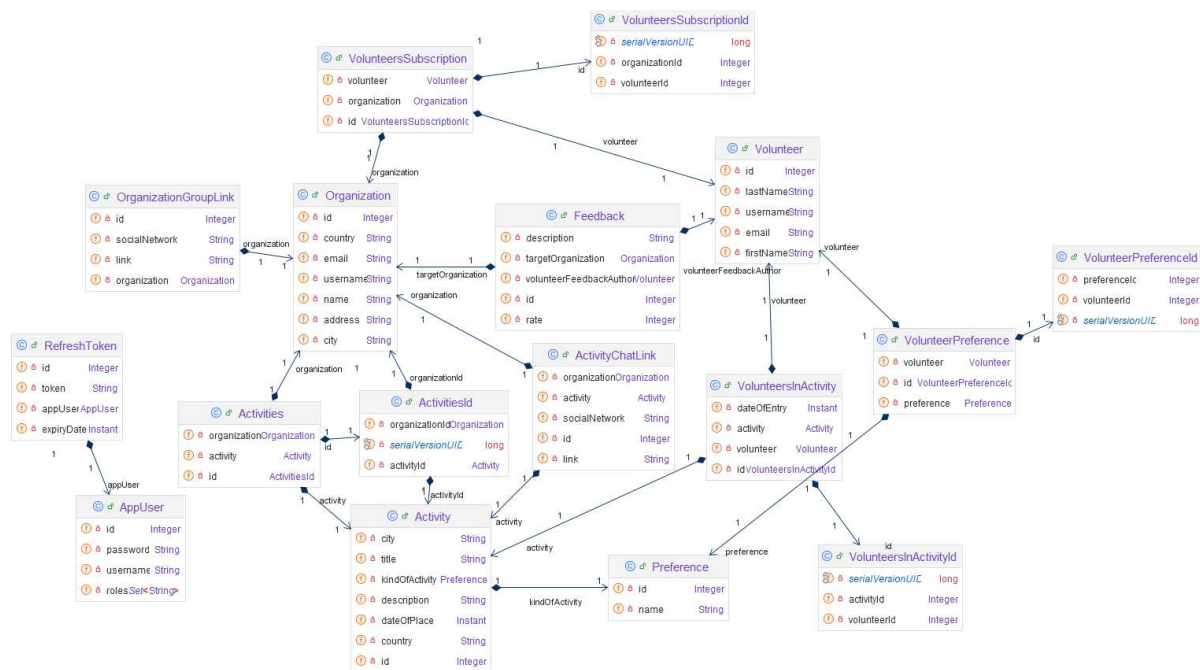


Рис. 2.8 – Діаграма класів-сутностей

Діаграма класів демонструє об'єктно-орієнтовану модель системи. Вона відображає основні класи програми, їхні властивості, методи, а також відношення між ними. Опис основних сутностей:

- «Volunteer» — інформаційна сутність про користувача - волонтера;
- «Organization» — інформаційна сутність про користувача - організацію.
- «Activity» — активність, організована організаціями для волонтерів.
- «VolunteerSubscription» — підписка волонтера на організацію.
- «VolunteerPreference» — вподобання волонтера, пов'язані з «Preference».
- «VolunteerInActivity» — участь волонтера в активності.
- «Feedback» — зворотний зв'язок між волонтерами та організаціями.
- «ActivityChatLink» — чат активності.
- «OrganizationGroupLink» — зв'язки між організаціями.
- «AppUser» / «RefreshToken» — користувачі додатку і токени для авторизації.

- «Activities» / «ActivitiesId» / «OrganizationOrganizationId» — допоміжні сутності для Many-to-Many зв'язків.

У результаті система добре нормалізована, передбачає багатокористувацьку взаємодію та підтримку гнучких зв'язків між об'єктами.

2.3. Рішення з математичного забезпечення

Блок-схему загальної роботи веб-застосунку зображено на рисунку 2.6.

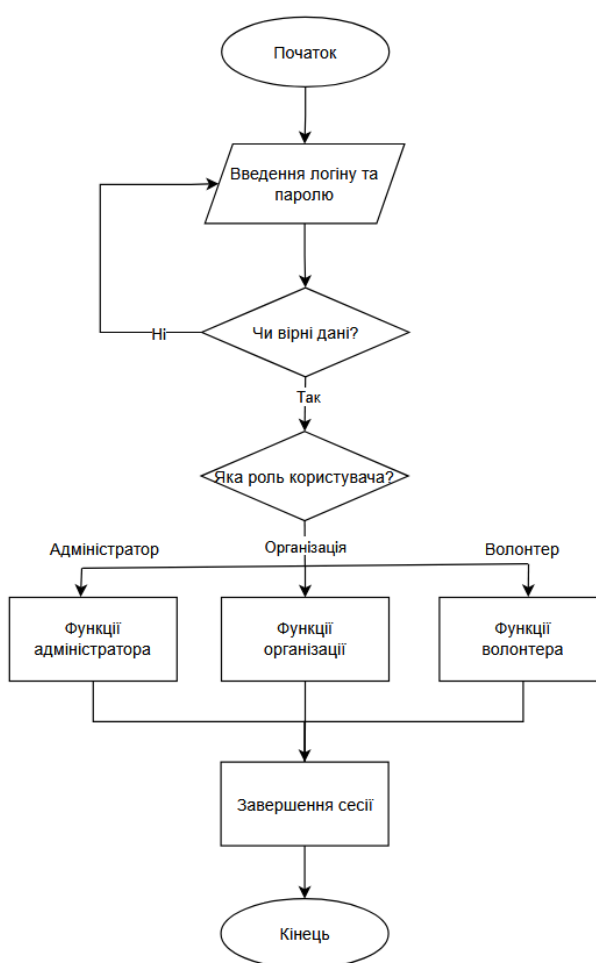


Рис. 2.9 – Блок-схема загальної роботи застосунку

Під час роботи з веб-додатком управління волонтерською діяльністю взаємодія користувача із системою починається з процесу автентифікації, тобто введення логіну та паролю. На цьому етапі система перевіряє правильність наданих даних. Якщо інформація введена некоректно, користувач повертається

до форми входу і має повторити спробу. У разі правильного введення облікових даних виконується авторизація — система визначає, до якої ролі належить користувач.

Кожна роль у системі — адміністратор, представник волонтерської організації або волонтер — має свій набір функцій, доступних після входу. Якщо система визначає, що користувач є адміністратором, йому надається доступ до функціоналу підтримки платформи, модерації контенту, ведення аналітики та моніторингу активностей усіх учасників. У випадку, коли користувач є представником організації, він отримує змогу створювати та редагувати волонтерські активності, обробляти заявки волонтерів, керувати інформацією про свою організацію, а також взаємодіяти з волонтерами через платформу. Якщо ж система авторизує користувача як волонтера, він отримує доступ до переліку подій, може подавати заявки на участь, залишати відгуки, керувати підписками на сповіщення та редагувати власний профіль.

Усі користувачі, незалежно від ролі, після виконання необхідних дій у системі мають змогу завершити сесію, що передбачає вихід із платформи. Таким чином, реалізовано чітку послідовність етапів: автентифікація, авторизація, доступ до ролезналежного функціоналу та завершення роботи з додатком.

Також у системі є функціонал що доступний і без авторизації, перегляд інформації щодо веб-застосунку, перегляд усіх наявних організацій та активностей і можливість створення аккаунту користувача.

У результаті розроблено математичне забезпечення, що включає алгоритмічні рішення для кожного функціонального модуля. Створені алгоритми забезпечують реалізацію основних обчислювальних процедур, обробку вхідних даних та генерування необхідних вихідних результатів. Це дозволяє ефективно виконувати поставлені завдання та забезпечує коректне функціонування інформаційної системи відповідно до сформульованих вимог.

2.4. Рішення з програмного забезпечення

Для реалізації інформаційної системи управління волонтерською діяльністю обрано сучасний, масштабований і гнучкий стек технологій, що забезпечує зручність використання, безпеку, адаптивність до змін і подальшу можливість масштабування.

Серверна частина (бекенд) побудована на мові програмування Java з використанням Spring Boot — фреймворку, який дозволяє швидко створювати надійні веб-сервіси з мінімальною конфігурацією. Spring Boot підтримує REST API, взаємодію з базами даних, модулі безпеки, логування та інші важливі функції для розробки повноцінних веб-застосунків. Його перевага — автоматична конфігурація, залежності за замовчуванням і зручна інтеграція з іншими бібліотеками з екосистеми Spring [24].

Java — це об'єктно-орієнтована мова програмування загального призначення, що вирізняється високою продуктивністю, портативністю, надійністю й активно використовується у створенні корпоративних, веб- та мобільних додатків. Java має велику екосистему бібліотек і фреймворків, що дозволяє ефективно реалізовувати складні програмні рішення [25].

Для збереження та обробки даних використовується MySQL — популярна система керування реляційними базами даних. Для спрощення взаємодії з базою даних у застосунку використовується ORM. Це підхід, який дозволяє розробнику працювати з базою даних через об'єкти мови програмування. У цьому проєкті застосовується бібліотека Spring Data JPA, яка дозволяє автоматично генерувати SQL-запити на основі об'єктів та інтерфейсів. Наприклад, замість написання запиту вручну можна просто викликати метод `findByEmail()` і фреймворк сам сформує потрібний SQL-запит. Це значно пришвидшує розробку, зменшує кількість помилок і забезпечує більшу читабельність коду [26, 33].

Для забезпечення аутентифікації та авторизації реалізовано Spring Security із використанням JWT. Цей механізм дозволяє уникнути збереження сесій на сервері: після успішного входу користувач отримує токен, який зберігається на клієнті у `localStorage` і надсилається у заголовках запитів. Сервер перевіряє токен, витягує з нього роль користувача (адміністратор, організація або волонтер) і надає доступ до відповідного функціоналу. Такий підхід є сучасним стандартом для SPA і забезпечує безпеку та масштабованість [27].

Фронтенд (інтерфейс користувача) реалізовано з використанням React — популярної бібліотеки JavaScript для створення інтерфейсів користувача. React дозволяє будувати складні, динамічні та ефективні SPA, де зміни на сторінці відображаються миттєво без перезавантаження браузера [28].

У React-проєкті використовується TypeScript — мова програмування, яка додає до JavaScript статичну типізацію. Це дозволяє покращити стабільність застосунку, оскільки помилки виявляються ще до запуску програми. TypeScript також покращує досвід розробки за рахунок автодоповнення, зручного рефакторингу і кращої підтримки структури коду — це особливо важливо при роботі з масштабними React-компонентами [29].

Для збірки клієнтської частини застосовано Vite — сучасний інструмент для розробки фронтенду, який забезпечує надзвичайно швидкий запуск, гаряче оновлення додатку HMR (Hot Module Replacement) та мінімальну конфігурацію. Завдяки Vite, розробка відбувається швидко та зручно навіть у великих проєктах [30].

Інтерфейс створено за допомогою Tailwind CSS — утилітарного CSS-фреймворку, що дозволяє створювати адаптивні стилі без написання складних CSS-файлів. Tailwind працює на основі класів, які додаються безпосередньо в HTML-розмітку компонентів, що значно спрощує розробку та підтримку дизайну. Додатково застосовується shadcn/ui — набір готових React-

компонентів, створених на основі Radix UI, які підтримують темну тему, доступність та мають сучасний вигляд. Вони чудово інтегруються з Tailwind і дозволяють будувати зручні інтерфейси без потреби створювати кожен елемент з нуля [31, 32].

Клієнтська частина взаємодіє з бекендом через REST API. Усі запити до серверної частини проходять через JWT-аутентифікацію, що дозволяє захищати дані на рівні маршруту. Наприклад, користувач із роллю волонтера не зможе отримати доступ до маршрутів, призначених для адміністратора чи організації.

Обрана архітектура дозволяє легко масштабувати додаток, інтегрувати нові сервіси, а також змінювати або розширювати логіку без серйозного втручання в існуючу систему. Крім того, система побудована за принципами розділення відповідальностей, де кожен компонент відповідає за конкретну частину логіки: бекенд — за обробку бізнес-процесів і безпеку, фронтенд — за інтерфейс користувача та взаємодію з API.

2.5. Рішення з технічного забезпечення

Для ефективної роботи інформаційної системи управління волонтерською діяльністю було обрано надійні апаратні та програмні засоби, що забезпечують стабільність функціонування, масштабованість, доступність і зручність експлуатації як для користувачів, так і для розробників.

Для розгортання інформаційної системи обліку волонтерської діяльності обрано віддалений сервер, наданий компанією CloudHoster. Дана організація пропонує високий рівень доступності, безпеки та масштабованості серверних ресурсів, що дозволяє ефективно розгортати веб-застосунки. Сервери компанії базуються на сучасному обладнанні, яке підтримує контейнеризацію та розподілені обчислення.

Для розгортання обрано віртуальний приватний сервер (VPS) з наступними технічними характеристиками:

- Оперативна пам'ять (RAM): 8 ГБ (з можливістю збільшення до 16 ГБ).
- Процесор: 4 ядра (з опцією розширення до 8 ядер).
- Місце на диску: 100 ГБ SSD для зберігання бази даних та логів.
- Мережева пропускна здатність: 1 Гбіт/с для стабільного доступу до платформи.
- Резервування даних: автоматичний бекап для забезпечення цілісності даних.

В якості операційної системи було обрано Linux (Ubuntu 24.0.2 LTS) через її стабільність, безпеку та безкоштовне використання. Дана ОС підтримує зручний пакетний менеджер APT, що спрощує встановлення та оновлення компонентів [34].

Такий вибір дозволяє ефективно масштабувати систему в разі зростання навантаження або потреби в додаткових ресурсах. Завдяки використанню VPS від CloudHoster, система залишається гнучкою та адаптованою до реальних потреб платформи, що забезпечує стабільну роботу веб-застосунку у різних умовах експлуатації.

Робочі станції користувачів Оскільки система є вебзастосунком, вона доступна з будь-якого сучасного браузера, що мінімізує апаратні вимоги до клієнтських пристроїв.

Основні вимоги:

- Пристрій: персональний комп'ютер, ноутбук, планшет або смартфон;
- Операційна система: Windows, macOS, Linux, Android або IOS;

- Браузер: Google Chrome, Mozilla Firefox, Safari або Microsoft Edge останньої версії;
- Інтернет-з'єднання: стабільне з пропускною здатністю не менше 1 Мбіт/с.

Інтерфейс застосунку є адаптивним і коректно відображається на екранах різної роздільної здатності, що забезпечує зручність користування на будь-яких пристроях.

Висновок

На основі розробленого технічного проєкту інформаційної системи управління волонтерською діяльністю постає задача реалізації цього проєкту, яка передбачає створення програмного забезпечення відповідно до визначених вимог, з урахуванням особливостей предметної області та забезпечення ефективного функціонування системи в реальних умовах.

РОЗДІЛ 3. Реалізація проекту інформаційної системи управління волонтерською діяльністю

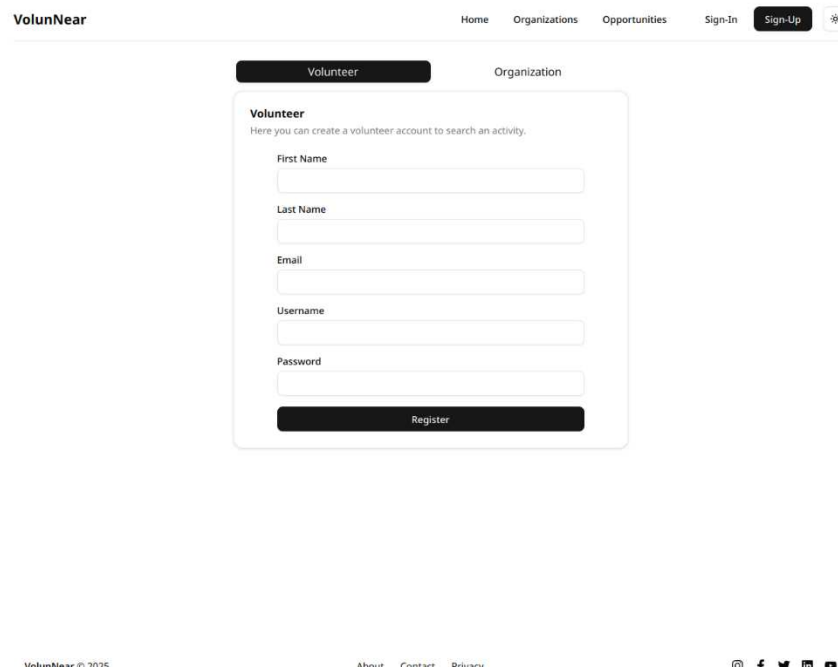
Основна увага приділена впровадженню створеної системи з урахуванням попередньо сформованих вимог та особливостей предметної області. У результаті реалізації проекту було створено веб-застосунок, що забезпечує користувачам функціонал для реєстрації, авторизації, управління активностями організацій, а також можливості залишення відгуків та отримання рекомендацій. Застосунок має широкий спектр можливостей і відповідає сучасним вимогам до функціональності та безпеки.

Для захисту даних реалізовано надійні механізми валідації введеної інформації, шифрування паролів та використання JWT-токенів для авторизації. Це гарантує конфіденційність та цілісність користувацьких даних. Реалізація функціоналу реєстрації та авторизації передбачає коректну перевірку даних та видачу токенів після успішного входу в систему.

Застосунок забезпечує зручне управління активностями, дозволяючи користувачам створювати, редагувати та видаляти події. Це сприяє ефективній організації заходів та залученню волонтерів. Також реалізовано можливість залишення відгуків про організації та отримання рекомендацій щодо заходів відповідно до вподобань користувача, що підвищує взаємодію в межах платформи.

Функціонал пошуку та фільтрації активностей дозволяє швидко знаходити потрібні події за різними параметрами, включаючи локацію, що значно спрощує користувачам процес планування своєї волонтерської діяльності. Крім того, система повідомлень з можливістю підписки на розсилку від обраних організацій забезпечує оперативне інформування про актуальні події та новини.

Для зручності створення профілю користувача було створено зручні форми с полями для реєстрації. На рисунку 3.1 зображено форму реєстрації волонтерського облікового запису.



The screenshot displays the VolunNear website interface. At the top, the logo 'VolunNear' is on the left, and navigation links 'Home', 'Organizations', 'Opportunities', 'Sign-In', and 'Sign-Up' are on the right. Below the navigation bar, there are two tabs: 'Volunteer' (selected) and 'Organization'. The 'Volunteer' tab contains a registration form with the following fields: 'First Name', 'Last Name', 'Email', 'Username', and 'Password'. A 'Register' button is located at the bottom of the form. The footer of the page includes 'VolunNear © 2020', 'About', 'Contact', 'Privacy', and social media icons for Facebook, Twitter, and Instagram.

Рис. 3.1 – Реєстраційна форма користувацького профілю волонтера

На рисунку 3.2 зображено форму реєстрації користувацького профілю організації.

The screenshot displays the VolunNear website's navigation bar with links for Home, Organizations, Opportunities, Sign-In, and Sign-Up. Below the navigation bar, there are two tabs: Volunteer and Organization, with the Organization tab selected. The main content area features a form titled "Organization" with the subtitle "Here you can create an organization account to publish activities and manage them." The form includes input fields for Organization Name, City, Address, Email, Username, and Password, followed by a Register button. The footer contains the text "VolunNear © 2020", links for About, Contact, and Privacy, and social media icons for Facebook, Twitter, and Instagram.

Рис. 3.2 – Реєстраційна форма користувацького профілю волонтера

На рисунку 3.3 зображено форму авторизації користувача

The screenshot displays the VolunNear website's navigation bar with links for Home, Organizations, Opportunities, Sign-In, and Sign-Up. Below the navigation bar, there is a large light gray rectangular area containing a form titled "Login" with the subtitle "Enter your username below to login to your account." The form includes input fields for Username and Password, followed by a Login button. At the bottom of the form, there is a link that says "Don't have an account? Sign up".

Рис. 3.3 – Форма авторизації користувача

На рисунку 3.4 зображена сторінка з усіма зареєстрованими на платформі волонтерськими організаціями.

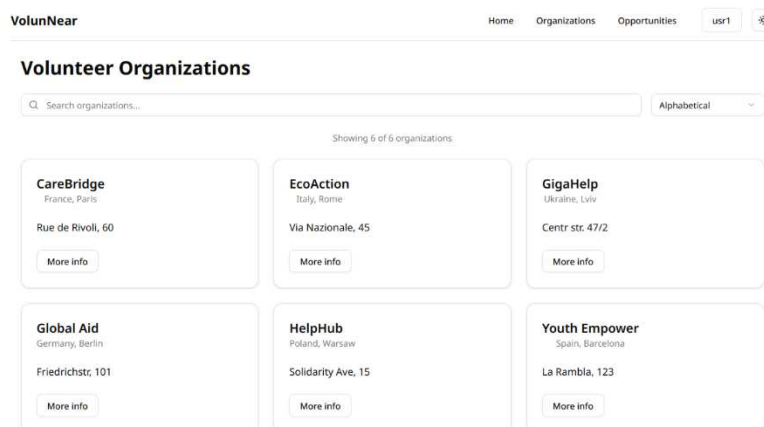


Рис. 3.4 – Сторінка з відображенням усіх волонтерських організацій

На рисунку 3.5 зображено сторінку зі списком доступних активностей з головною інформацією про них яка доступна по посиланню з навігаційного меню.

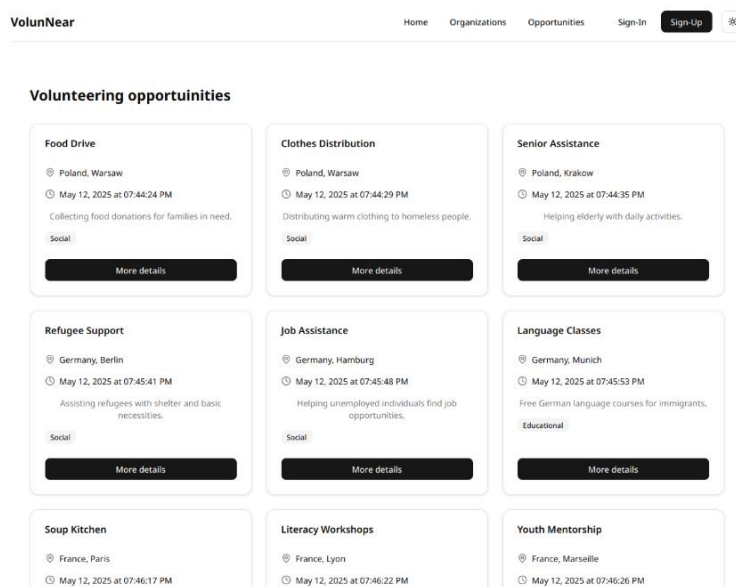


Рис. 3.5 – Сторінка з доступними активностями

У результаті тестування та експлуатації функціоналу застосунку можна зробити висновок, що розроблений застосунок відповідає початково сформульованим вимогам, надаючи користувачам зручний, функціональний та безпечний інструмент для участі в активностях волонтерських організацій.

Для зручного та ефективного використання системи була розроблена детальна документація, яка включає інструкцію для користувачів. У ній наведено покроковий опис основних функцій платформи, а також рекомендації щодо використання окремих модулів. Документація розміщена в додатку А.

РОЗДІЛ 4. Охорона праці

Вимоги безпеки та захисту здоров'я працівників під час роботи з екранними пристроями затверджені наказом Міністерства соціальної політики від 14.02.2018 № 207. Ці вимоги поширюються на всіх суб'єктів господарювання незалежно від форм власності, організаційно-правової форми та видів діяльності і встановлюють мінімальні вимоги безпеки та захисту здоров'я під час здійснення роботи, пов'язаної з використанням екранних пристроїв незалежно від їх типу та моделі.

Вимоги не поширюються на:

- робочі місця здобувачів освіти в комп'ютерних класах (кабінетах, аудиторіях тощо) закладів освіти;
- робочі місця пілотів, водіїв або операторів транспортних засобів, обладнані екранними пристроями в системах опрацювання даних на борту засобів сполучення, екранні пристрої у складі машин та устаткування, які переміщуються в процесі роботи;
- робочі місця працівників, які займаються обслуговуванням, ремонтом та налагодженням екранних пристроїв;
- портативні системи опрацювання даних, якщо вони не обладнані екранними пристроями; робочі місця працівників, що займаються обслуговуванням, ремонтом і наладкою екранних пристроїв; портативні системи опрацювання даних, якщо вони не

Роботодавець повинен проінформувати працівників під розписку про умови праці та наявність на їхніх робочих місцях небезпечних і шкідливих виробничих факторів (фізичних, хімічних, біологічних, психофізіологічних), що виникають під час роботи з екранними пристроями та ще не усунуті, а також про можливі наслідки їх впливу на здоров'я працівників.

Роботодавець повинен забезпечити навчання і перевірку знань працівників з питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними, а також у випадках модифікації та організації роботи обладнання.

Роботодавець повинен вжити відповідних заходів, щоб забезпечити відповідність робочого місця працівника цим вимогам.

Під час облаштування робочого місця працівника з екранними пристроями необхідно обирати таке обладнання, яке не створює зайвого шуму та не виділяє надлишкового тепла. Рівні шуму на робочих місцях осіб, які працюють з екранними пристроями, повинні відповідати вимогам Санітарних норм виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99, затверджених постановою Головного державного санітарного лікаря України від 01.12.1999 № 37.

Роботодавець повинен за рахунок тривалості робочої зміни організувати внутрішні регламентовані перерви для відпочинку відповідно до Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98, затверджених постановою Головного державного санітарного лікаря України від 10.12.1998 № 7.

Роботодавець повинен забезпечити за свій рахунок проведення медичних оглядів працівників відповідно до вимог Порядку проведення медичних оглядів працівників певних категорій, затвердженого наказом Міністерства охорони здоров'я України від 21.05.2007 № 246.

За результатами цих оглядів роботодавець за потреби повинен забезпечити виконання відповідних оздоровчих заходів.

Роботодавець зобов'язаний за потреби проводити лабораторні дослідження умов праці працівників з метою виявлення шкідливих і небезпечних факторів виробничого середовища, важкості та напруженості трудового процесу (зокрема щодо виявлення ризиків, пов'язаних із погіршенням зору, порушенням фізичного стану, стресом) та здійснювати заходи щодо усунення виявлених ризиків.

Робочі місця працівників з екранними пристроями мають бути спроектовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення та рухів.

Для забезпечення безпеки та захисту здоров'я працівників усе випромінювання від екранних пристроїв повинно бути зведене до гранично допустимого рівня (вплив на людину чинників середовища - шуму, вібрації, забруднювачів, температури тощо, який не спричиняє соматичних або психічних розладів, а також змін стану здоров'я, працездатності, поведінки, які виходять за межі пристосувальних реакцій) з погляду безпеки та охорони здоров'я працівників.

Організація робочого місця працівника з екранними пристроями повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним, антропологічним, психофізіологічним вимогам, а також характеру виконуваних робіт.

Освітлення робочого місця працівника з екранними пристроями повинно створювати відповідний контраст між екраном і навколишнім середовищем (з урахуванням виду роботи) і відповідати вимогам ДСанПіН 3.3.2.007-98.

Мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями повинен підтримуватися на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99, затверджених постановою Головного державного санітарного лікаря України від 0.12.1999 № 42 (далі - ДСН 3.3.6.042-99).

Робочий стіл або робоча поверхня мають бути достатнього розміру і мати поверхню з низькою відбивною здатністю, допускати гнучкість під час розміщення екрана, клавіатури, документів і відповідного обладнання.

Робоче крісло має бути стійким і давати змогу працівникові з екранними пристроями легко рухатися і займати зручне положення.

Сидіння має регулюватися за висотою, спинка сидіння - як за висотою, так і за нахилом. Слід передбачати підніжку для тих, кому це необхідно для зручності.

Мінімальні вимоги безпеки під час роботи з екранними пристроями:

- Щодня перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень;
- Після закінчення роботи екранні пристрої слід відключати від електричної мережі;
- У разі виникнення аварійної ситуації необхідно негайно відключити пристрій від електричної мережі.

Не допускається:

- виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника під час роботи з екранними пристроями;
- відключати захисні пристрої, самовільно проводити зміни в конструкції та складі екранних пристроїв або їх технічне налагодження;
- працювати з екранними пристроями, в яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності.

При виконанні робіт операторського типу, пов'язаних з нервово-емоційним напруженням, у приміщеннях під час роботи з екранними пристроями, на пультах і постах керування технологічними процесами та в інших приміщеннях мають дотримуватися оптимальні умови мікроклімату відповідно до вимог ДБН 3.3.6.042-99.

Мінімальні вимоги безпеки до екранних пристроїв:

- Екранні пристрої не повинні бути джерелом ризику для працівників;
- Усе випромінювання, за винятком видимої частини електромагнітного спектра, має бути зведене до незначного рівня з погляду безпеки та охорони здоров'я працівників;
- Символи на екранних пристроях мають бути чіткими, відповідного розміру. Між символами та рядками символів має бути належна відстань;
- Зображення на екрані має бути стабільним, без миготінь або інших видів нестабільності;
- Яскравість та/або контрастність символів повинні легко регулюватися працівником під час роботи з екранними пристроями, а також швидко адаптуватися до навколишніх умов;
- Обираючи екрани, слід надавати перевагу таким екранам, які легко та вільно обертаються та нахиляються відповідно до потреби працівника;
- За потреби може використовуватися окрема підставка або регульований стіл для розміщення екрана;
- Екран не повинен відблискувати або відбивати світло, щоб не викликати дискомфорту в працівника під час роботи з екранними пристроями;
- Обираючи клавіатуру, слід надавати перевагу такій клавіатурі, що відкидається і є автономною (відокремленою від екрана), щоб працівник міг обрати зручну робочу позу та уникнути втоми рук (кисті та верхньої частини руки);

- Поверхня клавіатури має бути матовою, щоб уникнути відбиття. Розташування клавіш і самі клавіші мають полегшувати роботу з клавіатурою. Позначення клавіш має бути достатньо контрастним і розбірливим;
- Обладнання, що входить до робочої станції, не повинно виділяти надлишкового тепла, що може заподіяти незручності працівникам під час роботи з екранними пристроями;

Відповідальність за порушення вимог безпеки

Відповідно до статті 43 Закону України «Про охорону праці» за порушення законодавства про охорону праці та невиконання приписів (розпоряджень) посадових осіб органів виконавчої влади з нагляду за охороною праці юридичні та фізичні особи, які відповідно до законодавства використовують найману працю, притягаються органами виконавчої влади з нагляду за охороною праці до сплати штрафу в порядку, встановленому законом.

Сплата штрафу не звільняє юридичну або фізичну особу, яка відповідно до законодавства використовує найману працю, від усунення виявлених порушень у визначені строки.

Максимальний розмір штрафу не може перевищувати п'яти відсотків середньомісячного фонду заробітної плати за попередній рік юридичної або фізичної особи, яка відповідно до законодавства використовує найману працю.

Несплата або неповна сплата юридичними або фізичними особами, які відповідно до законодавства використовують найману працю, штрафу тягне за собою нарахування пені на несплачену суму штрафу (його частини) з розрахунку 120 % річних облікової ставки Національного банку України, що діяла в період такої несплати, за кожний день прострочення [35].

ВИСНОВКИ

У рамках проєкту створено інформаційну систему управління волонтерською діяльністю, яка підвищує ефективність організації та координації активностей. Основною метою було створити зручний інструмент для взаємодії волонтерів з організаціями та оптимізувати процеси управління заходами.

Розробка системи почалася з аналізу вимог та створення технічної документації, що визначає структуру та функціонал платформи. У результаті реалізовано модульний моноліт, який забезпечує інтеграцію серверної та клієнтської частин. Для розробки використано Java з Spring Boot на бекенді та React з TypeScript на фронтенді.

Модульна архітектура забезпечила структурованість коду та спростила розширення функціоналу. Після завершення розробки проведено апробацію системи, що підтвердила стабільність та коректність роботи основних компонентів, зокрема реєстрації, авторизації, управління активностями та рекомендаційного сервісу.

Після завершення етапу розробки була проведена апробація системи, яка підтвердила стабільність та коректність роботи всіх ключових компонентів та модулів. Також створена документація з інструкцією для користувачів, що спрощує ознайомлення з функціями платформи.

Таким чином, розроблена інформаційна система повністю відповідає актуальним вимогам у сфері волонтерської діяльності, надаючи користувачам зручний та інтуїтивний інтерфейс для управління активностями. Впровадження цієї системи сприяє підвищенню ефективності організації волонтерських заходів, що в свою чергу позитивно впливає на розвиток волонтерського руху та залучення нових учасників.

ЛІТЕРАТУРА

1. Волонтерство в Україні – благодійний фонд UWCF - <https://uwcfoundation.com/ua/blagotvoritelnost-v-ukraine/volonterstvo-v-ukraini>
2. Фонд громади міста Херсон “Захист” - <https://zahyst.ks.ua/pro-fond-zahyst/>
3. Отримання статусу волонтерської організації: - <https://yustra.com.ua/uk/dop-uslugi/poluchenie-statusa-volonterskoj-organizacii/>
4. IBM – What is Monolithic Architecture - <https://www.ibm.com/think/topics/monolithic-architecture>
5. QALight – Монолітна архітектура ПЗ - <https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/>
6. Monolith Architecture - <https://tech.tamara.co/monolith-architecture-5f00270f384e>
7. Марченко А.В. Проектування інформаційних систем 2016 90с. - https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/content-20160217112601.pdf;
8. Вендров А. М. Проектування програмного забезпечення економічних інформаційних систем. М.: Фінанси та статистика, 2006. - 544 с.
9. Грекул В. І. Проектування інформаційних систем. Практикум. М: Національний відкритий університет «ІНТУІТ», 2012. -159 с.
10. Ременяк Л.В. Проектування інформаційних систем: конспект лекцій. Одеса, Одеський державний екологічний університет, 2016. 152.
11. Вікіпедія. Вільна енциклопедія - <https://uk.wikipedia.org/wiki/UML>
12. Джошуа Блох. Ефективна Java; [В оригіналі] 2016. —440 с
13. М. Фаулер UML. Основи, 3-видання; [В оригіналі]
14. Fowler, M. (2014) та Newman, S. (2019). *Building Microservices* (O'Reilly Media).
15. Introduction to single page – <https://www.geeksforgeeks.org/introduction-to-single-page-application/>

16. RESTfull API info – <https://restfulapi.net/>
17. MySQL – <https://www.mysql.com/>
18. MySQL Introduction from GeeksForGeeks – <https://www.geeksforgeeks.org/mysql-introduction/>
19. What is mysql – <https://www.redhat.com/en/topics/databases/what-is-mysql>
20. InnoDB introduction – <https://dev.mysql.com/doc/refman/8.0/en/innodb-introduction.html>
21. Docker Documentation – <https://docs.docker.com/>
22. RedHat. "What is containerization?" – <https://www.redhat.com/en/topics/containers/what-is-containerization>
23. Spring Boot Documentation – <https://spring.io/projects/spring-boot>
24. Vite Official Guide – <https://vitejs.dev/guide/>
25. Tailwind CSS – <https://tailwindcss.com/>
26. Shadcn UI – <https://ui.shadcn.com/>
27. React Documentation – <https://react.dev/>
28. TypeScript Documentation – <https://www.typescriptlang.org/docs/>
29. MySQL Documentation – <https://dev.mysql.com/doc/>
30. JWT.io — JSON Web Token Introduction – <https://jwt.io/introduction>
31. Spring Data JPA – <https://spring.io/projects/spring-data-jpa>
32. What is Java – <https://www.oracle.com/java/technologies/>
33. Ubuntu Server documentation - <https://documentation.ubuntu.com/server/>
34. Kubernetes Documentation - <https://kubernetes.io/docs/concepts/overview/#why-you-need-kubernetes-and-what-can-it-do>
35. Охорона праці - <https://hrliga.com/index.php?module=news&op=view&id=20212>

Додаток А – Інструкція користувачу

Для використання системи потрібно перейти на сторінку веб додатку. Після відкриття користувача зустріне головна сторінка проекту, на котрій буде навігаційне меню, контент головної сторінки, пошукове поле та інші навігаційні рішення, головна сторінка зображена на рисунку А1.

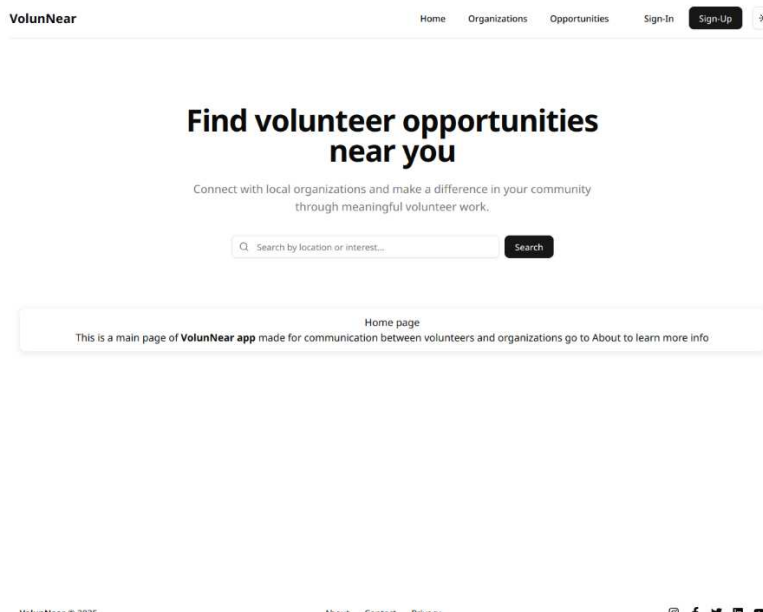


Рис. А1 – Головна сторінка веб-застосунку

Для перегляду усіх організацій потрібно в навігаційному меню натиснути «Organizations», сторінка з усіма організаціями зображена на рисунку А3.

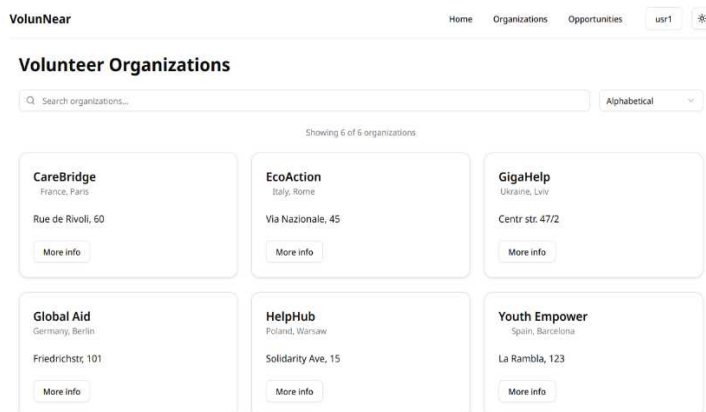


Рис. А2 – Сторінка з усіма зареєстрованими організаціями

Для перегляду усіх активностей потрібно в навігаційному меню натиснути «Opportunities», сторінка з усіма активностями зображена на рисунку А3.

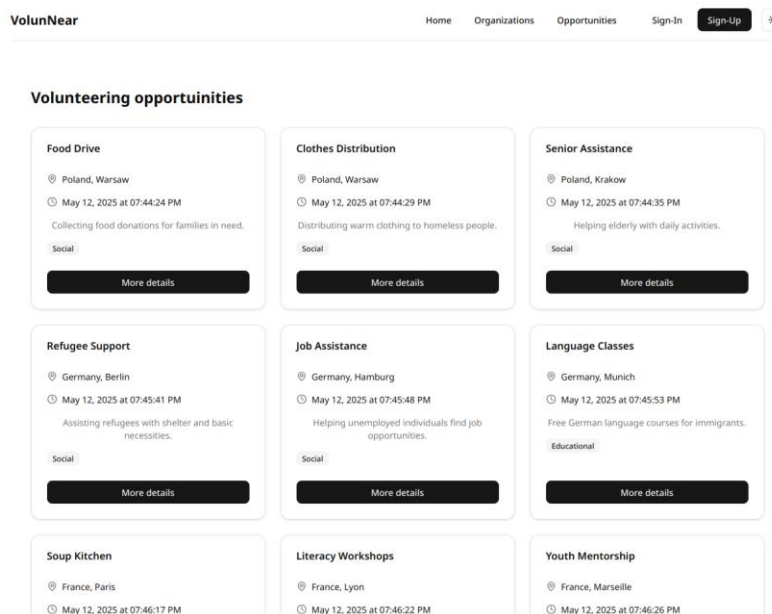


Рис. А3 - Сторінка з усіма доступними активностями

Для створення особистого кабінету, публікації та менеджменту активностей потрібно перейти на сторінку реєстрації натиснувши кнопку «Sign-up», й ввести всі необхідні дані для реєстрації і потім натиснути кнопку «Register» зображено на рисунку А4.

VolunNear Home Organizations Opportunities Sign-In **Sign-Up**

Volunteer **Organization**

Organization
Here you can create an organization account to publish activities and manage them.

Organization Name
HelpHub

Country
Poland

City
Warsaw

Address
Solidarity Ave, 15

Email
org3Email@gmail.com

Username
org3

Password
.....

Register

Рис. А4 – Сторінка створення користувацького профілю організації

Після створення користувацького профілю потрібно увійти в нього. На сторінці авторизації (або інакшими словами логіну). Для цього потрібно перейти на сторінку авторизації натисканням кнопки «Sign-in», та ввести дані котрі було вказано при реєстрації (а саме ім'я користувача та пароль) зображено на рисунку А6. В разі успішного входу буде переадресація на головну сторінку застосунку.

VolunNear Home Organizations Opportunities Sign-In **Sign-Up**

Login
Enter your username below to login to your account

Username
org3

Password
.....

Login

Don't have an account? [Sign up](#)

Рис. А6 – Сторінка авторизації

Після авторизації користувач – організація має можливість переглянути активності, додати нову, редагувати їх та видаляти. Для цього потрібно натиснути на ім'я користувача в навігаційному меню справа згори та натиснути в випадаючому меню «My activities» зображено на рисунку А7.

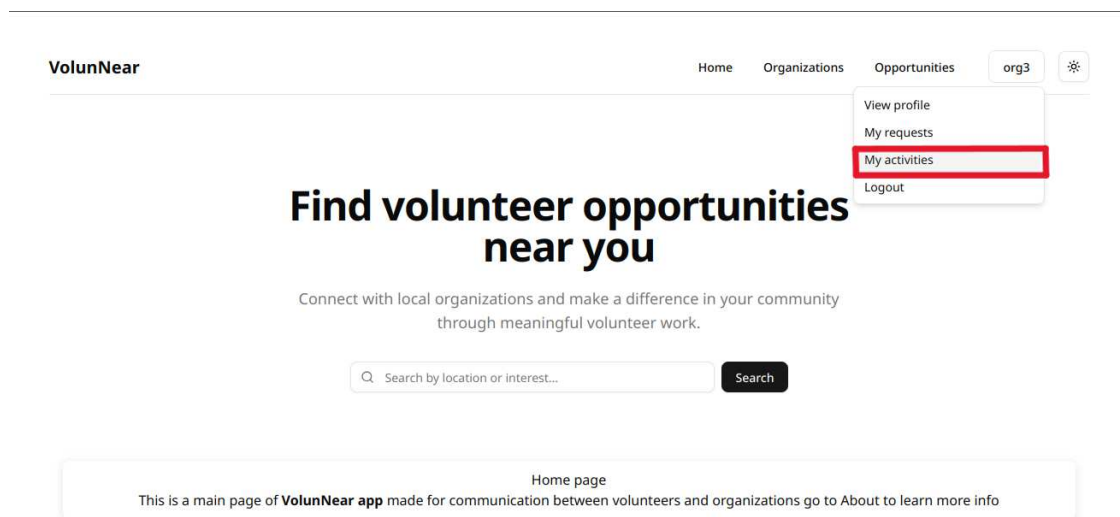


Рис. А7 – Випадаюче меню та кнопка «My activities»

Для створення нової активності потрібно натиснути кнопку «Add new activity», кнопка та сторінка менеджменту організації зображена на рисунку А8.

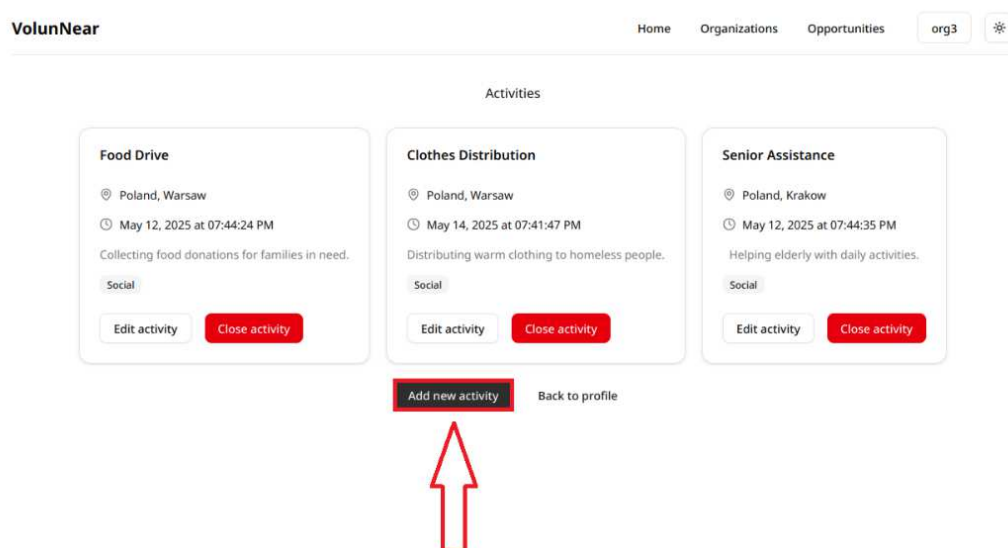
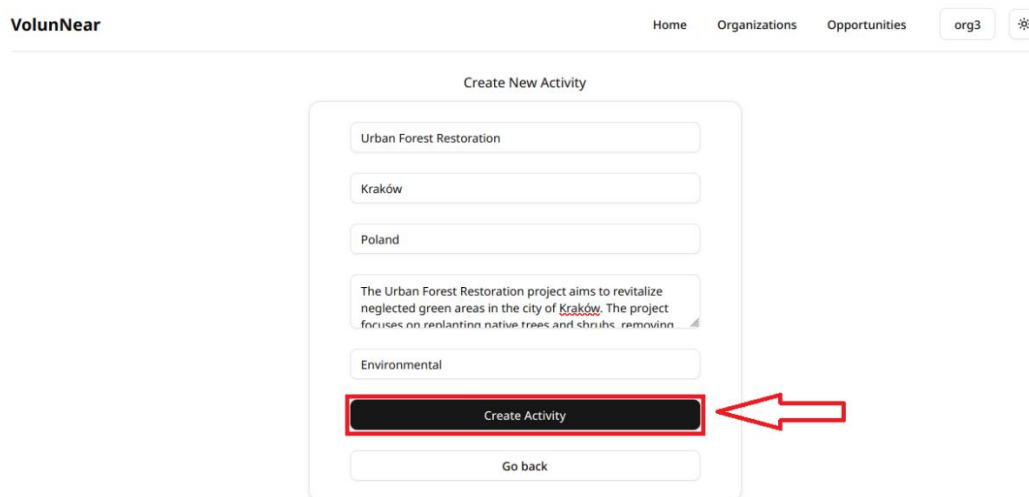


Рис. А8 – Сторінка менеджменту активностями та кнопка створення нової активності

Після натискання на кнопку створення активності відкриється сторінка з формою для надання інформації щодо активності, потрібно ввести всі дані котрі вказані в формі та натиснути кнопку «Create Activity», форма створення зображена на рисунку A9.



The screenshot shows the 'Create New Activity' form in the VolunNear application. The form is titled 'Create New Activity' and contains several input fields: 'Urban Forest Restoration', 'Kraków', 'Poland', a text area with a description, 'Environmental', and a 'Create Activity' button. The 'Create Activity' button is highlighted with a red box and a red arrow. Below it is a 'Go back' button. The top navigation bar includes 'VolunNear', 'Home', 'Organizations', 'Opportunities', 'org3', and a settings icon.

Рис. A9 – Форма створення активності

У результаті відобразиться повідомлення що активність була створена, повідомлення зображено на рисунку A10.

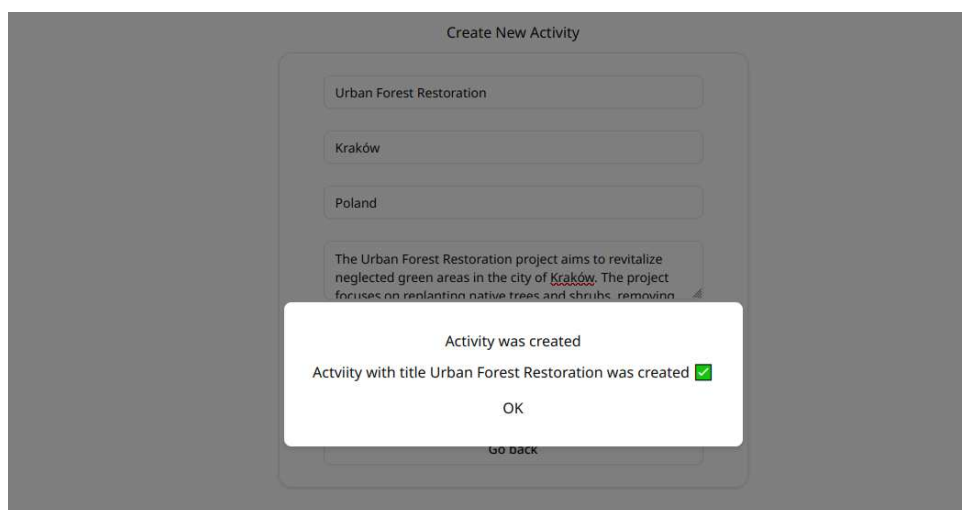


Рис. A10 – Повідомлення про статус створення активності

Результат додавання можна перевірити на сторінці менеджменту активностей користувача-організації чи на сторінці всіх доступних активностей.

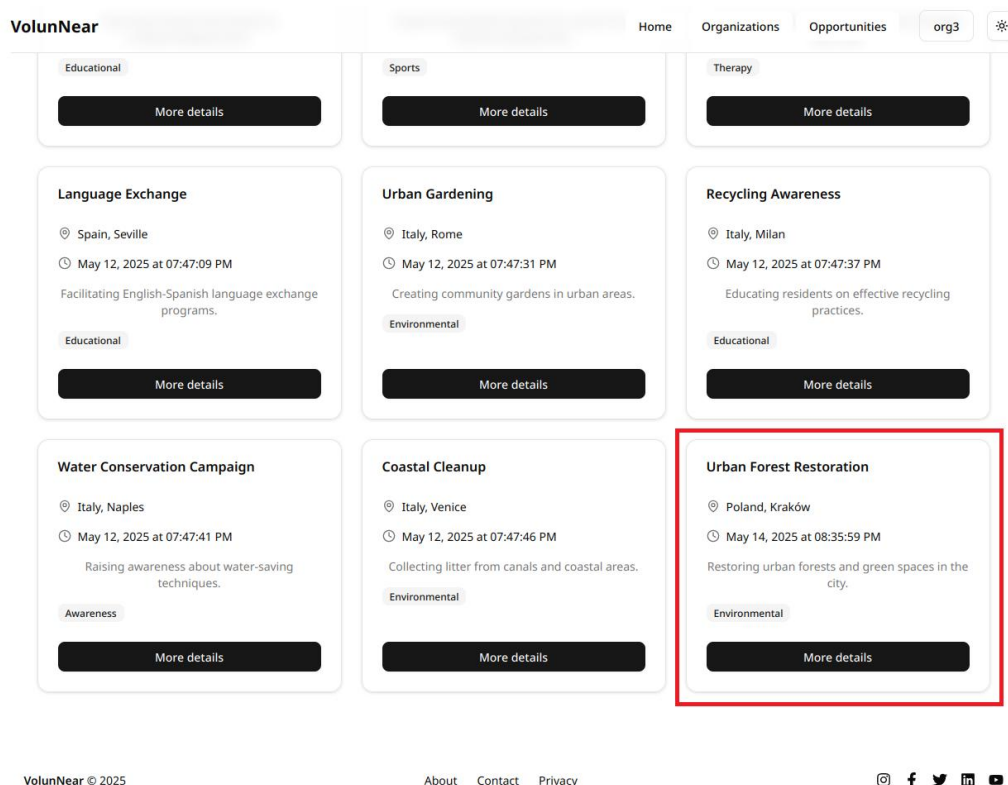


Рис. A11 – Створена активність на сторінці усіх активностей

Щоб волонтер приєднався до новоствореної активності потрібно. Користувачу волонтеру зайти в свій акаунт. Перейти на сторінку усіх активностей та натиснути кнопку «More Details» і натиснути кнопку «Join activity», що відправить запит до організації на приєднання до активності і тоді кнопка змінить свій вид на кнопку відміни запиту «Cancel my request», кнопка відправки запиту зображена на рисунку A12 та кнопки відміни запиту на A13 відповідно.

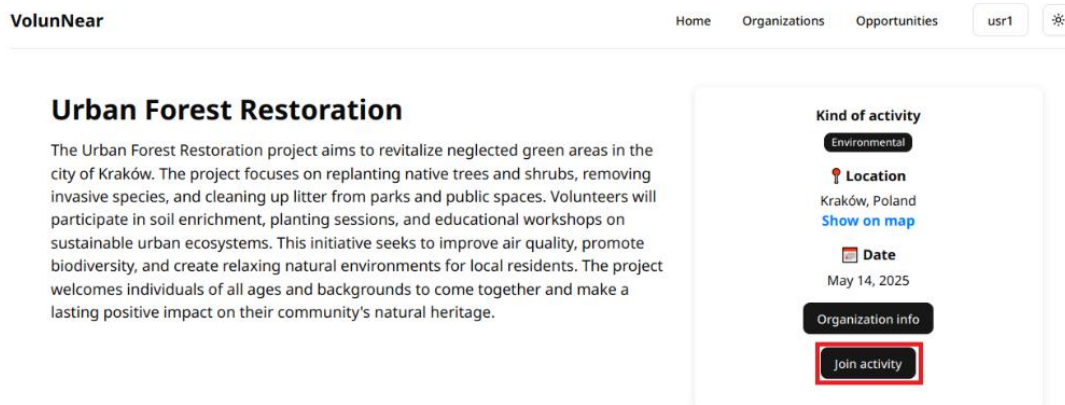


Рис. А12 – Кнопка подання заявки

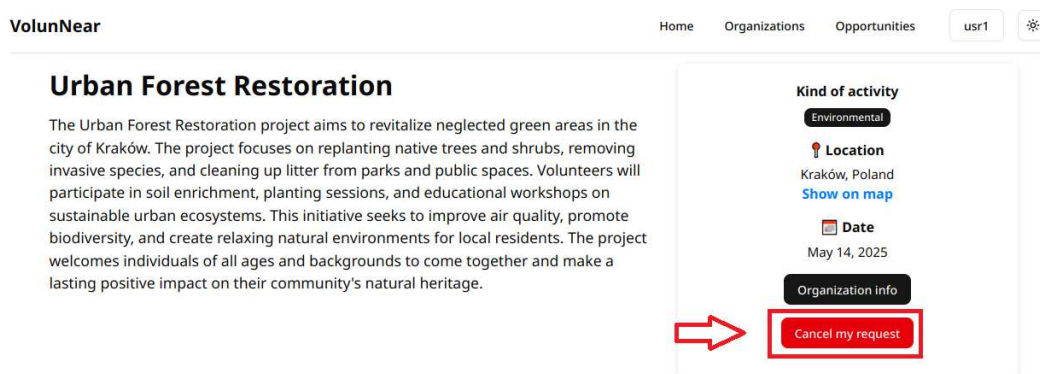


Рис. А13 – Кнопка відміни запиту

На цьому етапі потрібно щоб організація дозволила волонтеру доєднатись до активності, це робиться на сторінці запитів до якої можна перейти користувачу-організації натиснувши на ім'я користувача і потім натиснути у випадіючому меню «My requests» (користувач-волонтер у цьому меню побачить свої запити до активностей котрі він може відмінити, зображено на рисунку А14). А далі потрібно підтвердити участь конкретного користувача-волонтера в активності натиснувши «Accept request» після чого висвітиться повідомлення про успішне доєднання волонтера до активності, сторінка усіх запитів в профілі організації зображена на малюнку А15, повідомлення про успішне додання волонтера до активності на рисунку А16.

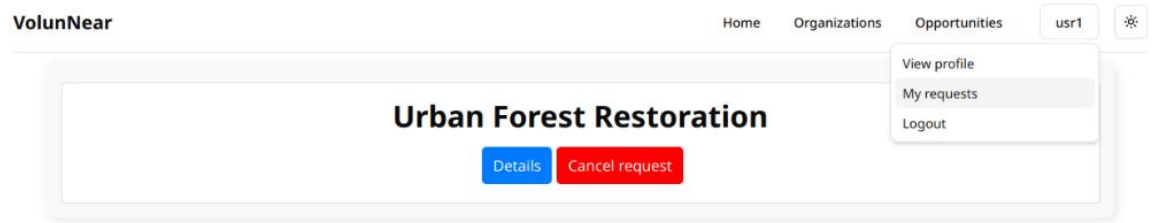


Рис. А14 – Сторінка запитів користувача-волонтера

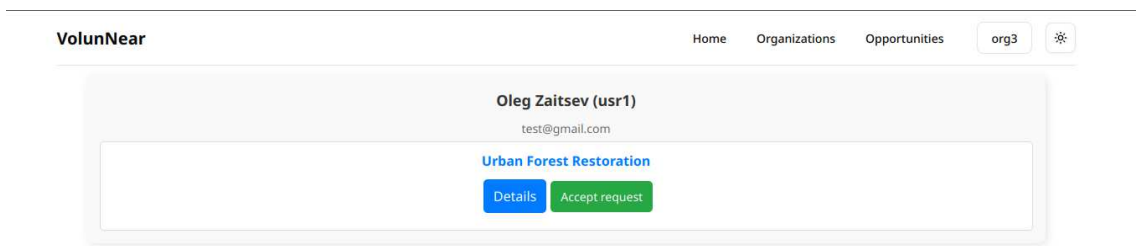


Рис. А15 – Сторінка запитів користувача-організації

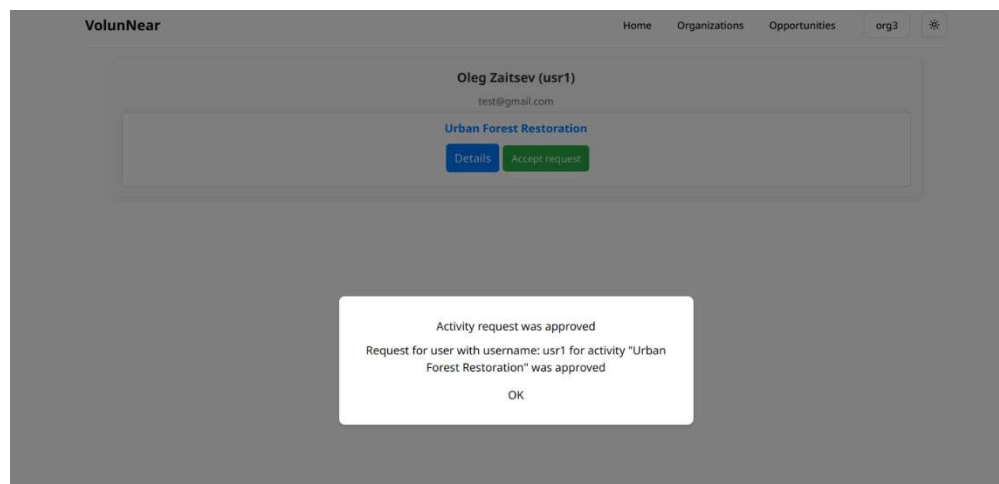


Рис. А16 – Повідомлення про успішне додання волонтера до активності

Додаток Б – Розгортання додатку

Проект складається з двох частин:

- Бекенд (Java Spring Boot) - [VolunNearApp Backend](#)
- Фронтенд (React + Vite) - [VolunNearApp Frontend](#)

1. Попередні вимоги

- Встановлений Docker та Docker Compose
- Встановлений Node.js (версія ≥ 18) та npm
- Встановлений Git

2. Клонування репозиторіїв

2.1 Клонування бекенду

```
git clone https://github.com/Essur/VolunNearApp.git
```

```
cd VolunNearApp
```

2.2 Клонування фронтенду

```
git clone https://github.com/Essur/volun-near-client-side.git
```

```
cd volun-near-client-side
```

3. Запуск бекенду за допомогою Docker

3.1 Перевірка Docker

Переконайтесь, що Docker працює:

```
docker --version
```

```
docker-compose --version
```

3.2 Запуск контейнерів (БД + Spring Boot)

Перейдіть в папку бекенду:

```
cd VolunNearApp
```

Запустіть Docker Compose:

```
docker-compose up --build
```

3.3 Що відбувається:

- MySQL запускається на порту 3600
- Spring Boot сервіс запускається на порту 8080

3.4 Перевірка роботи:

Перевірка стану контейнерів:

```
docker ps
```

Перегляд логів бекенду:

```
docker logs -f spring-boot-app
```

4. Запуск фронтенду з Vite:

4.1 Перехід до папки фронтенду:

```
cd volun-near-client-side
```

4.2 Встановлення залежностей:

```
npm install
```

4.3 Запуск фронтенду:

```
npm run dev
```

4.4 Перевірка роботи:

Перейдіть в браузер та відкрийте:

<http://localhost:5173>

5. Локальна перевірка роботи застосунку

1. Зайдіть на фронтенд за адресою:

<http://localhost:5173>

2. Створіть нового користувача або виконайте вхід.
3. Перевірте роботу основних функцій (додавання волонтерської активності, перегляд списку).

6. Зупинка застосунку

6.1 Зупинка бекенду:

У вкладці терміналу з Docker:

```
docker-compose down
```

6.2 Зупинка фронтенду:

Натисніть Ctrl + C у терміналі з Vite.

Додаток В – Текст програми

AuthController.java:

```
package com.volunnear.controllers;

import com.volunnear.Routes;
import com.volunnear.dtos.jwt.JwtRefreshRequest;
import com.volunnear.dtos.jwt.JwtRefreshTokenResponse;
import com.volunnear.dtos.jwt.JwtRequest;
import com.volunnear.dtos.jwt.JwtResponse;
import com.volunnear.entitiy.users.RefreshToken;
import com.volunnear.exception.TokenRefreshException;
import com.volunnear.services.security.AuthService;
import com.volunnear.services.security.RefreshTokenService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequiredArgsConstructor
public class AuthController {
    private final AuthService authService;
    private final RefreshTokenService refreshTokenService;

    @ResponseStatus(value = HttpStatus.OK)
    @PostMapping(value = Routes.LOGIN)
    public JwtResponse createAuthToken(@RequestBody JwtRequest
authRequest) {
        String token = authService.createAuthToken(authRequest);
        String role = authService.getAuthorities(token);
        RefreshToken refreshToken =
refreshTokenService.createRefreshToken(authService.getUsernameByToken(token));
        return new JwtResponse(token, role, refreshToken.getToken());
    }

    @ResponseStatus(value = HttpStatus.OK)
    @PostMapping(value = Routes.REFRESH_TOKEN)
    public JwtRefreshTokenResponse refreshAuthToken(@RequestBody
JwtRefreshRequest refreshTokenRequest) {
        String refreshToken = refreshTokenRequest.getRefreshToken();
        return refreshTokenService.findByToken(refreshToken)
            .map(refreshTokenService::verifyExpiration)
            .map(RefreshToken::getAppUser)
            .map(user -> {
                String token =
authService.recreateToken(user.getUsername());
                return new JwtRefreshTokenResponse(token,
refreshToken);
            })
            .orElseThrow(() -> new TokenRefreshException("Bad
credentials, try re-login"));
    }
}
```

AuthContext.tsx:

```

import React, { createContext, ReactNode, useContext, useEffect, useState
} from "react";
import { REFRESH_TOKEN } from "../config/ApiRoutes";

interface AuthContextType {
  isLoggedIn: boolean;
  username: string | null;
  role: string | null;
  login: (username: string, token: string, refreshToken: string, role:
string, expiresIn: number) => void;
  logout: () => void;
  refreshToken: () => Promise<void>;
}

export const AuthContext = createContext<AuthContextType |
undefined>(undefined);

export const AuthProvider: React.FC<{ children: ReactNode }> = ({ children
}) => {
  const [isLoggedIn, setIsLoggedIn] =
useState<boolean>(!!localStorage.getItem("jwtToken"));
  const [username, setUsername] = useState<string |
null>(localStorage.getItem("username"));
  const [role, setRole] = useState<string |
null>(localStorage.getItem("role"));

  const login = async (username: string, token: string, refreshToken:
string, role: string, expiresIn: number) => {
    const expiryTime = Date.now() + expiresIn * 1000;

    localStorage.setItem("jwtToken", token);
    localStorage.setItem("refreshToken", refreshToken);
    localStorage.setItem("username", username);
    localStorage.setItem("role", role);
    localStorage.setItem("tokenExpiry", expiryTime.toString());

    setIsLoggedIn(true);
    setUsername(username);
    setRole(role);
  };

  const logout = () => {
    localStorage.removeItem("jwtToken");
    localStorage.removeItem("refreshToken");
    localStorage.removeItem("username");
    localStorage.removeItem("role");
    localStorage.removeItem("tokenExpiry");
    setIsLoggedIn(false);
    setUsername(null);
    setRole(null);
  };

  const refreshToken = async () => {
    const storedRefreshToken = localStorage.getItem("refreshToken");
    if (!storedRefreshToken) {
      logout();
      return;
    }

    try {

```

```

        const response = await fetch(REFRESH_TOKEN, {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
                Authorization: `Bearer
${localStorage.getItem("jwtToken")}`,
            },
            body: JSON.stringify({ refreshToken: storedRefreshToken
        })),
    });

    if (!response.ok) throw new Error("Failed to refresh token");

    const data = await response.json();
    const expiryTime = Date.now() + data.expiresIn * 1000;

    localStorage.setItem("jwtToken", data.token);
    localStorage.setItem("tokenExpiry", expiryTime.toString());

    setIsLoggedIn(true);
    setUsername(localStorage.getItem("username"));
    setRole(localStorage.getItem("role"));
} catch (error) {
    console.error("Token refresh failed:", error);
    logout();
}
};

useEffect(() => {
    const checkTokenExpiry = () => {
        const expiryTime = localStorage.getItem("tokenExpiry");
        if (!expiryTime) return;

        const timeLeft = parseInt(expiryTime, 10) - Date.now();
        if (timeLeft < 60000) { // Refresh 1 minute before expiry
            refreshToken();
        }
    };

    const interval = setInterval(checkTokenExpiry, 20000); // Check
every 30 sec
    return () => clearInterval(interval);
}, []);

return (
    <AuthContext.Provider value={{ isLoggedIn, username, role, login,
logout, refreshToken }}>
        {children}
    </AuthContext.Provider>
);
};

export const useAuth = () => {
    const context = useContext(AuthContext);
    if (context === undefined) {
        throw new Error("useAuth must be used within an AuthProvider");
    }
    return context;
};

```

JwtTokenProvider.java:

```
package com.volunnear.security.jwt;

import com.volunnear.services.users.UserService;
import io.jsonwebtoken.*;
import jakarta.servlet.http.HttpServletRequest;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

import java.util.Date;
import java.util.List;

@Component
@RequiredArgsConstructor
public class JwtTokenProvider {
    @Value("${jwt.token.secret}")
    private String secret;

    @Value("${jwt.token.lifetime}")
    private Long jwtLifetime;
    private final UserService userService;

    public String createToken(UserDetails userDetails) {
        Claims claims =
        Jwts.claims().setSubject(userDetails.getUsername());
        claims.put("roles",
        userDetails.getAuthorities().stream().map(GrantedAuthority::getAuthority).
        toList());

        Date now = new Date();
        Date expireTime = new Date(now.getTime() + jwtLifetime);

        return Jwts.builder()
            .setClaims(claims)
            .setExpiration(expireTime)
            .signWith(SignatureAlgorithm.HS256, secret)
            .compact();
    }

    public String resolveToken(HttpServletRequest request) {
        String bearerToken = request.getHeader("Authorization");
        if (bearerToken != null && bearerToken.startsWith("Bearer ")) {
            return bearerToken.substring(7);
        }
        return null;
    }

    public boolean validateToken(String token) {
        try {
            Jws<Claims> claims =
            Jwts.parser().setSigningKey(secret).parseClaimsJws(token);
            return !claims.getBody().getExpiration().before(new Date());
        } catch (JwtException | IllegalArgumentException e) {
            return false;
        }
    }
}
```

```

    }
}

public Authentication getAuthentication(String token) {
    UserDetails userDetails =
userService.loadUserByUsername(getUsernameFromToken(token));
    return new UsernamePasswordAuthenticationToken(userDetails, "",
userDetails.getAuthorities());
}

public String getUsernameFromToken(String token) {
    return getClaimsFromToken(token).getSubject();
}

public List<String> getRolesFromToken(String token) {
    return getClaimsFromToken(token).get("roles", List.class);
}

private Claims getClaimsFromToken(String token) {
    return Jwts.parser()
        .setSigningKey(secret)
        .parseClaimsJws(token)
        .getBody();
}
}

```

Повний код проекту можна переглянути за двома посиланнями:

Back-end: <https://github.com/Essur/VolunNearApp>

Front-end: <https://github.com/Essur/volun-near-client-side>