НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Вебзастосунок для допомоги військовим в запасі з пошуком роботи, житла та місця навчання, і допомоги організації волонтерських заходів»

Виконав:
студент IV курсу, групи КП-13
Попович Юрій Петрович _____

Керівник:
доцент кафедри ПЗКС, к.е.н., доцент,
Ткаченко Костянтин Олександрович _____

Консультант з нормоконтролю:
доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
доцент кафедри СПСКС, к.т.н, с.н.с.,
Боярінова Юлія Євгенівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2024 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Поповичу Юрію Петровичу

1. Тема проєкту «Вебзастосунок для допомоги військовим в запасі та їхнім сім'ям з пошуком роботи, житла та місця навчання, і допомоги організації волонтерських заходів», керівник проєкту Ткаченко Костянтин Олександрович, доцент кафедри ПЗКС, к.т.н., доцент, затверджені наказом по університету №1808-С від «29» травня 2025 р.

2. Термін подання студентом проєкту «13» червня 2025 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз предметної області та існуючих рішень;
   - обґрунтування вибору засобів реалізації вебзастосунку;
   - опис структурно-алгоритмічної організації системи;
   - аналіз розробленого вебзастосунку.

5. Перелік обов'язкового графічного матеріалу:
   - алгоритм користування вебзастосунком військовим (креслення);
   - схема бази даних (креслення);
   - архітектура вебзастосунку (плакат);
   - дерево проблем (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2024	
2.	Розроблення та узгодження технічного завдання	22.11.2024	
3.	Розроблення структури вебзастосунку	15.12.2024	
4.	Підготовка матеріалів першого розділу дипломного проєкту	05.02.2025	
5.	Розроблення дизайну сторінок та графічних елементів	10.02.2025	
6.	Підготовка матеріалів другого розділу дипломного проєкту	02.03.2025	
7.	Програмна реалізація вебзастосунку	11.03.2025	
8.	Тестування вебзастосунку	24.03.2025	
9.	Підготовка матеріалів третього розділу дипломного проєкту	25.03.2025	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	28.04.2025	
11.	Підготовка графічної частини дипломного проєкту	15.05.2025	
12.	Оформлення документації дипломного проєкту	27.05.2025	

Студент Юрій ПОПОВИЧ

Керівник проєкту Костянтин ТКАЧЕНКО

АНОТАЦІЯ

Даний дипломний проєкт присвячений розробці вебзастосунку RefugeUA, основною метою якого є надання допомоги демобілізованим військовим, ветеранам, військовозобов’язаним в запасі та їхнім родинам у пошуку житла, працевлаштування, освіти, психологічної підтримки, а також сприяння волонтерській діяльності шляхом організації заходів та зборів.

Вебзастосунок реалізовано як інтерактивну онлайн-платформу, що дозволяє користувачам переглядати та розміщувати оголошення про доступне житло, вакансії, освітні програми та волонтерські ініціативи. Особливістю системи є можливість створення волонтерськими організаціями власних груп, додавання детальної інформації про заходи (час, місце, програма, організатори), а також участь користувачів у цих заходах за наявності вільних місць. Події відображаються у календарному інтерфейсі, що дозволяє зручно орієнтуватися в часі проведення заходів.

Застосунок також може використовуватись місцевими адміністраціями як інструмент зворотного зв’язку та моніторингу потреб громади. У проєкті реалізовано архітектуру клієнтської та серверної частин, інтерфейси для різних ролей користувачів, механізми авторизації та реєстрації, базу даних, а також систему обліку участі у заходах.

ABSTRACT

This diploma project is dedicated to the development of the *RefugeUA* web application, whose primary goal is to support demobilized soldiers, veterans, reservists, and their families in finding housing, employment, educational opportunities, and psychological assistance, as well as to promote volunteer activity through the organization of events and gatherings.

The web application is implemented as an interactive online platform that allows users to browse and publish listings for available housing, job vacancies, educational programs, and volunteer initiatives. A key feature of the system is the ability for volunteer organizations to create their own groups, add detailed information about events (time, location, schedule, organizers), and allow users to participate if spots are available. Events are presented in a calendar interface, making it easy to navigate and plan participation.

The platform can also be used by local authorities as a feedback and monitoring tool to assess the needs of the community. The project implements both client-side and server-side architecture, user interfaces for different roles, authentication and registration mechanisms, a database system, and a participation tracking module for events.

ДП.045440-01-90 Вебзастосунок для допомоги військовим в запасі та їхнім сім’ям з пошуком роботи, житла та місця навчання, і допомоги організації волонтерських заходів. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Вебзастосунок для допомоги	5	
	військовим в запасі та їхнім		
	сім’ям з пошуком роботи, житла		
	та місця навчання, і допомоги		
	організації волонтерських заходів.		
	Технічне завдання		
ДП.045440-03-81	Вебзастосунок для допомоги	88	
	військовим в запасі та їхнім		
	сім’ям з пошуком роботи, житла		
	та місця навчання, і допомоги		
	організації волонтерських заходів.		
	Пояснювальна записка		
ДП.045440-04-51	Вебзастосунок для допомоги	4	
	військовим в запасі та їхнім		
	сім’ям з пошуком роботи, житла		
	та місця навчання, і допомоги		
	організації волонтерських заходів.		
	Програма та методика тестування		
ДП.045440-05-34	Вебзастосунок для допомоги	37	
	військовим в запасі та їхнім		
	сім’ям з пошуком роботи, житла		
	та місця навчання, і допомоги		
	організації волонтерських заходів.		
	Керівництво користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Вебзастосунок для допомоги	2	
	військовим в запасі та їхнім		
	сім’ям з пошуком роботи, житла		
	та місця навчання, і допомоги		
	організації волонтерських заходів.		
	Алгоритм користування		
	застосунком. Схема алгоритму		
ДП.045440-07-99	Вебзастосунок для допомоги	1	
	військовим в запасі та їхнім		
	сім’ям з пошуком роботи, житла		
	та місця навчання, і допомоги		
	організації волонтерських заходів.		
	Структура бази даних.		
	ERD-діаграма		
ДП.045440-08-98	Вебзастосунок для допомоги	1	
	військовим в запасі та їхнім		
	сім’ям з пошуком роботи, житла		
	та місця навчання, і допомоги		
	організації волонтерських заходів.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ДОПОМОГИ ВІЙСЬКОВИМ В ЗАПАСІ ТА ЇХНІМ СІМ'ЯМ З ПОШУКОМ РОБОТИ, ЖИТЛА ТА МІСЦЯ НАВЧАННЯ, І ДОПОМОГИ ОРГАНІЗАЦІЇ ВОЛОНТЕРСЬКИХ ЗАХОДІВ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Костянтин ТКАЧЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Юрій ПОПОВИЧ

2024

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: вебзастосунок для допомоги військовим в запасі та їхнім сім’ям з пошуком роботи, житла та місця навчання, і допомоги організації волонтерських заходів

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання як вебзастосунок, що забезпечує військових у запасі та їхні родини доступом до актуальних оголошень щодо житла, роботи та навчання, а також підтримує громади та волонтерські організації в інтеграції військових у місцеве середовище й організації волонтерських заходів.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Серверна частина вебзастосунку повинна бути розроблена на фреймворку ASP.NET Core, клієнтська частина – на фреймворку Angular, а база даних – за допомогою СУБД Microsoft SQL Server.

Вебзастосунок повинен забезпечувати такі основні функції:

1. Реєстрація та вхід користувача в систему.

2. Створення, перегляд, редагування та видалення оголошень відповідно до прав доступу.
3. Можливість взяти участь у волонтерських заходах.
4. Можливість перегляду волонтерських заходів
5. Можливість відгукнутися на оголошення та перегляду власних відгуків.
6. Можливість перегляду власного профілю.
7. Можливість перегляду статей з психологічної підтримки.
8. Можливість перегляду контактів спеціалістів з психологічної допомоги.
9. Можливість перегляду волонтерських груп.
10. Можливість слідкувати за волонтерськими групами.

Додаткові вимоги вебзастосунку:

1. Зберігання паролей в захешованому вигляді.
2. Реєстрація в системі має відбуватись з підтвердженням пошти та дозволом адміністратора.
3. Вебзастосунок повинен мати функціональні можливості для модерації та перевірки оголошень.
4. Дизайн сторінок виконувати у відтінках синього кольору.
5. Наявність навігаційної панелі.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Алгоритм користування застосунком. Схема алгоритму»;

– «Структура бази даних. ERD-діаграма».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи 13.11.2024
Розроблення та узгодження технічного завдання 22.11.2024
Розроблення структури вебзастосунку 15.12.2024
Розроблення дизайну сторінок та графічних елементів 10.02.2025
Програмна реалізація вебзастосунку 11.03.2025
Тестування вебзастосунку 25.04.2025
Підготовка матеріалів текстової частини проєкту 30.04.2025
Підготовка матеріалів графічної частини проєкту 15.05.2025
Оформлення технічної документації проєкту 27.05.2025

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ДОПОМОГИ ВІЙСЬКОВИМ В ЗАПАСІ ТА ЇХНІМ СІМ'ЯМ З ПОШУКОМ РОБОТИ, ЖИТЛА ТА МІСЦЯ НАВЧАННЯ, І ДОПОМОГИ ОРГАНІЗАЦІЇ ВОЛОНТЕРСЬКИХ ЗАХОДІВ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Костянтин ТКАЧЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Юрій ПОПОВИЧ

2025

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

HR-фахівець – фахівець з управління людськими ресурсами, який займається підбором кадрів, адаптацією працівників і створенням сприятливого робочого середовища.

SQL (Structured Query Language) – мова структурованих запитів, що використовується для взаємодії з реляційними базами даних.

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту, що використовується для обміну інформацією між веббраузером та вебсервером.

SQL (Structured Query Language) – мова структурованих запитів, що використовується для взаємодії з реляційними базами даних.

SQL ін'єкції – тип вразливості вебзастосунків, що дозволяє зловмисникам виконувати SQL-запити на сервері.

SPA (Single Page Application, односторінковий застосунок) – це вебзастосунок, який завантажує тільки одну сторінку і динамічно змінює її вміст без необхідності перезавантаження всієї сторінки.

JWT (JSON Web Token) – це стандарт токена доступу на основі JSON, стандартизованого в RFC 7519. Як правило, використовується для передачі даних для аутентифікації в клієнт-серверних програмах.

XSS (Cross-Site Scripting) – вразливість вебзастосунків, яка дозволяє виконувати шкідливі скрипти на стороні користувача.

CSRF (Cross-Site Request Forgery) – атака, при якій зловмисник змушує жертву виконати небажану дію через її авторизований сеанс.

ORM (Object-Relational Mapping) – технологія, що дозволяє зберігати об'єкти програмування в реляційних базах даних.

DRY (Don't Repeat Yourself) – принцип програмування, який полягає в уникненні дублювання коду.

JVM (Java Virtual Machine) – віртуальна машина для виконання Java-програм.

IoC (Inversion of Control) – принцип програмування, коли контроль над об’єктами та їх залежностями передається фреймворку.

REST (Representational State Transfer) – архітектурний стиль для проєктування вебсервісів, що базується на HTTP протоколі.

Web API – набір інтерфейсів для обміну даними між різними програмами через HTTP.

API (Application Programming Interface) – інтерфейс програмування застосунків, набір функцій та процедур, які дозволяють взаємодіяти з програмним забезпеченням.

DOM (Document Object Model) – об’єктна модель документа, структура, яка дозволяє доступ до елементів HTML або XML документа.

ACID – це набір властивостей, які гарантують надійність транзакцій у системах керування базами даних.

JSON (JavaScript Object Notation) – це легкий текстовий формат обміну даними, який використовується для представлення структурованої інформації.

BSON (Binary JSON) – це бінарний формат, що базується на JSON. Він розроблений для ефективнішого зберігання і передачі даних, підтримуючи додаткові типи даних (наприклад, дату, двійкові дані).

HTML (HyperText Markup Language) – це мова розмітки, яка використовується для створення вебсторінок. Вона визначає структуру документа за допомогою елементів (тегів), які вказують браузеру, як відображати контент.

UI (User Interface, користувацький інтерфейс) – це візуальна частина програмного забезпечення або вебзастосунку, з якою взаємодіє користувач. UI включає всі елементи, що дозволяють користувачу вводити дані та отримувати результати.

Endpoint (кінцева точка) – це URL-адреса, яка є точкою доступу до функцій або даних API.

ВСТУП

На теперішній час в умовах війни питання соціальної адаптації та підтримки військових, які повертаються до цивільного життя, є вкрай актуальним. Демобілізовані військовослужбовці, військовозобов'язані в запасі, ветерани та їхні сім'ї стикаються з багатьма проблемами, зокрема з пошуком житла, роботи, можливостей для перекваліфікації, отриманням якісної психологічної допомоги та інтеграції в місцеві громади. Водночас волонтерські організації потребують ефективного інструменту для організації заходів та координації допомоги. Відсутність єдиної централізованої платформи, що об'єднувала б ці напрями, ускладнює надання необхідної підтримки.

Зважаючи на зазначені проблеми, у межах даного дипломного проєкту передбачається розробка вебзастосунку RefugeUA. Основна мета цього застосунку – створення платформи, яка сприятиме соціальній адаптації військових та їхніх родин, забезпечуючи зручні інструменти для пошуку житла, роботи, можливостей навчання та перекваліфікації, а також доступ до психологічної допомоги. Крім того, вебзастосунок дозволить волонтерським організаціям ефективно координувати свою діяльність, розміщувати оголошення про волонтерські заходи, а також залучати учасників.

Розроблюваний продукт матиме широкий спектр функціональних можливостей, серед яких пошук доступного житла, інструменти для розміщення вакансій та пошуку роботи, інформація про навчальні заклади та програми перекваліфікації, а також ресурси для психологічної підтримки. Волонтерські організації отримають змогу створювати власні групи, розміщувати оголошення заходів та збирати необхідну кількість учасників для їхньої реалізації.

Запропонований вебзастосунок стане ефективним рішенням для вирішення низки соціальних проблем, сприятиме інтеграції ветеранів у

цивільне життя та підвищить ефективність волонтерської діяльності. Використання сучасних інформаційних технологій дозволить зробити процес підтримки більш організованим, прозорим та доступним для всіх зацікавлених сторін.

1. ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1. Огляд проблеми, яка вирішується ПЗ

Інтеграція військових у запасі та їхніх сімей у нові громади ускладнюється відсутністю єдиного інформаційного ресурсу, через який місцеві жителі могли б розміщувати оголошення про доступне житло та безпосередньо комунікувати з ветеранами. Це створює додаткові труднощі при пошуку житла для тих, хто повертається до цивільного життя, і не дозволяє формувати міцні зв'язки між військовими та місцевою спільнотою.

Крім того, відсутність спеціалізованої платформи для пошуку вакансій разом зі стереотипами роботодавців значно обмежує можливості працевлаштування для військових у запасі. Це не тільки негативно впливає на фінансове становище їхніх родин, але й знижує конкурентоспроможність ветеранів на ринку праці, що може призвести до тривалого безробіття.

Додатковою проблемою є недостатня інформаційна підтримка щодо можливостей освіти та перекваліфікації. Без доступу до актуальних даних про безкоштовні курси та освітні програми військові і їхні сім'ї втрачають шанс швидко адаптуватися до нових умов життя.

Ще однією важливою проблемою є відсутність ефективної організації волонтерських заходів. Нестача системної координації волонтерських ініціатив ускладнює реєстрацію на заходи та участь у них, що позбавляє громаду можливості активної підтримки військових. Це спричиняє недостатню взаємодію між волонтерами та ветеранами, що поглиблює соціальну ізоляцію та ускладнює процес інтеграції у місцеве середовище.

Низький рівень соціальної інтеграції та відсутність системної підтримки спричиняють ізоляцію військових у громадах, що ускладнює їх участь у соціальному житті і може призвести до зростання соціальних конфліктів.

Для досягнення поставленої мети було сформовано такі задачі:

1) аналіз ринку ПЗ для підтримки та інтеграції військових;

2) порівняння конкурентів з виявленням їхніх сильних та слабких сторін;
3) визначення функціональних можливостей системи;
4) обґрунтування вибору стеку технологій для розробки ПЗ;
5) розробка вебзастосунку для допомоги військовим у запасі та їхнім сім’ям з пошуком роботи, житла та місця навчання, а також організації волонтерських заходів.

1.2. Український фонд ветеранів

Український фонд ветеранів – це вебсайт, який має на меті всебічну підтримку ветеранів війни та членів їхніх родин. Його основна місія полягає в наданні достовірної та актуальної інформації, а також у забезпеченні доступу до різноманітних видів допомоги, необхідної для соціальної адаптації та покращення якості життя. Зокрема, ресурс охоплює такі ключові сфери, як соціальні послуги, медична допомога, реабілітація та інші корисні ресурси. Це включає інформування про пільги, державні програми підтримки, допомогу в оформленні документів, доступ до лікування, спеціалізовану медичну допомогу, а також фізичну і психологічну реабілітацію.

Крім того, сайт покликаний об’єднувати самих ветеранів, громадські організації та ініціативи, які працюють задля покращення їхнього добробуту. Він сприяє створенню спільноти підтримки та взаємодії, де кожен ветеран може знайти відповіді на важливі для нього питання [1].

Основні функціональні можливості сайту Українського фонду ветеранів включають:

- надання інформації про фінансову, соціальну та медичну підтримку ветеранів;
- заповнення та подачу онлайн-заявок на отримання допомоги;
- участь у конкурсних програмах для отримання грантів і фінансування;

- перегляд запланованих заходів, конференцій та зустрічей;
- ознайомлення з актуальними новинами та оновленнями щодо діяльності фонду;
- отримання психологічної підтримки, консультацій та контактів спеціалістів;
- запис на медичні та соціальні послуги через інтерактивні форми.

Одна з важливих особливостей сайту – це наявність інтерактивних форм (рис. 1.1), через які ветерани можуть подати заявки на різні види допомоги. Користувач може заповнити форму для отримання фінансової або медичної підтримки, що значно спрощує процес звернення за допомогою.

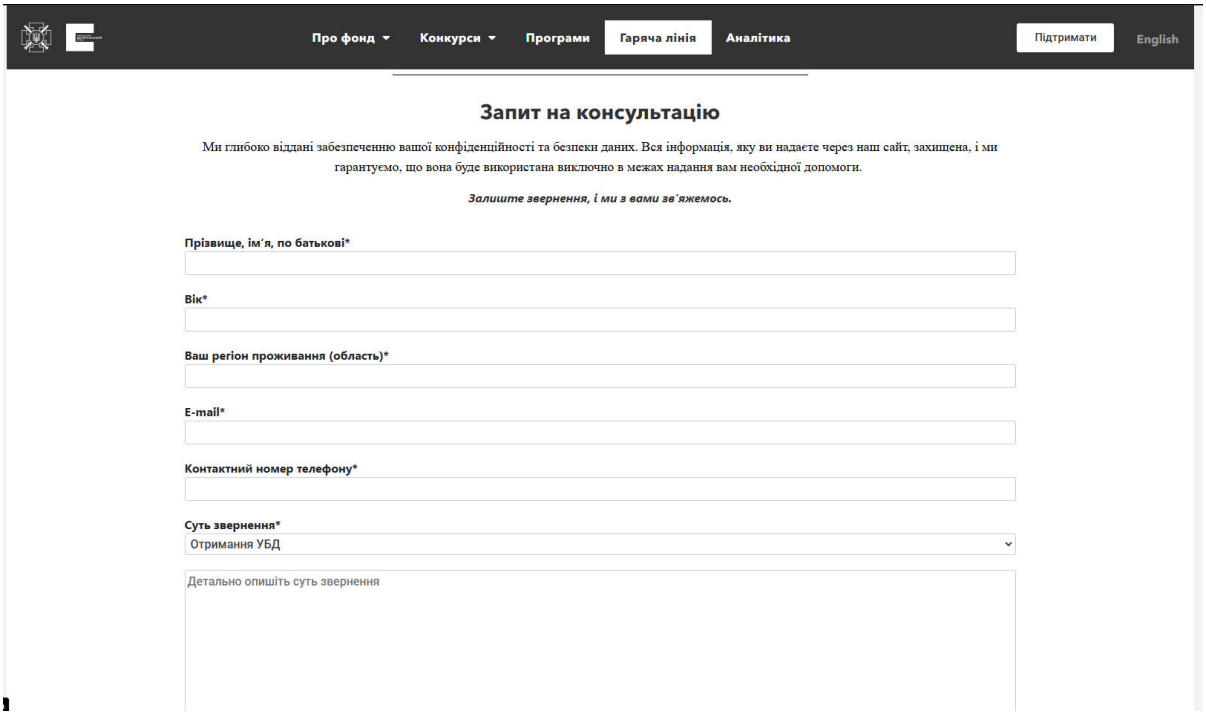


Рис. 1.1. Форма запиту для консультації

Крім того, на сайті є спеціальний розділ для організації заходів, де користувачі можуть переглядати заплановані події, конференції та зустрічі для ветеранів. Всі заходи доступні на сторінці «Програми» (рис. 1.2), а для перегляду актуальної інформації про найближчі події та оновлення є розділ «Новини» (рис. 1.3).

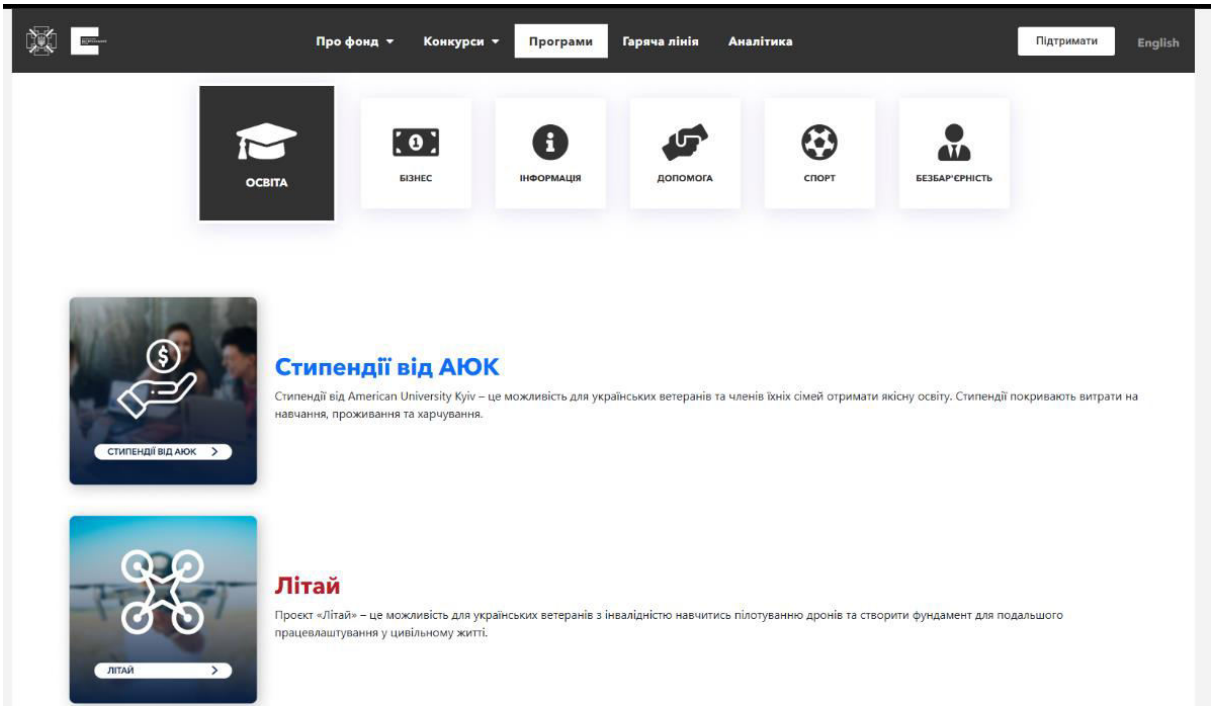


Рис. 1.2. Розділ вебсайту з програмами

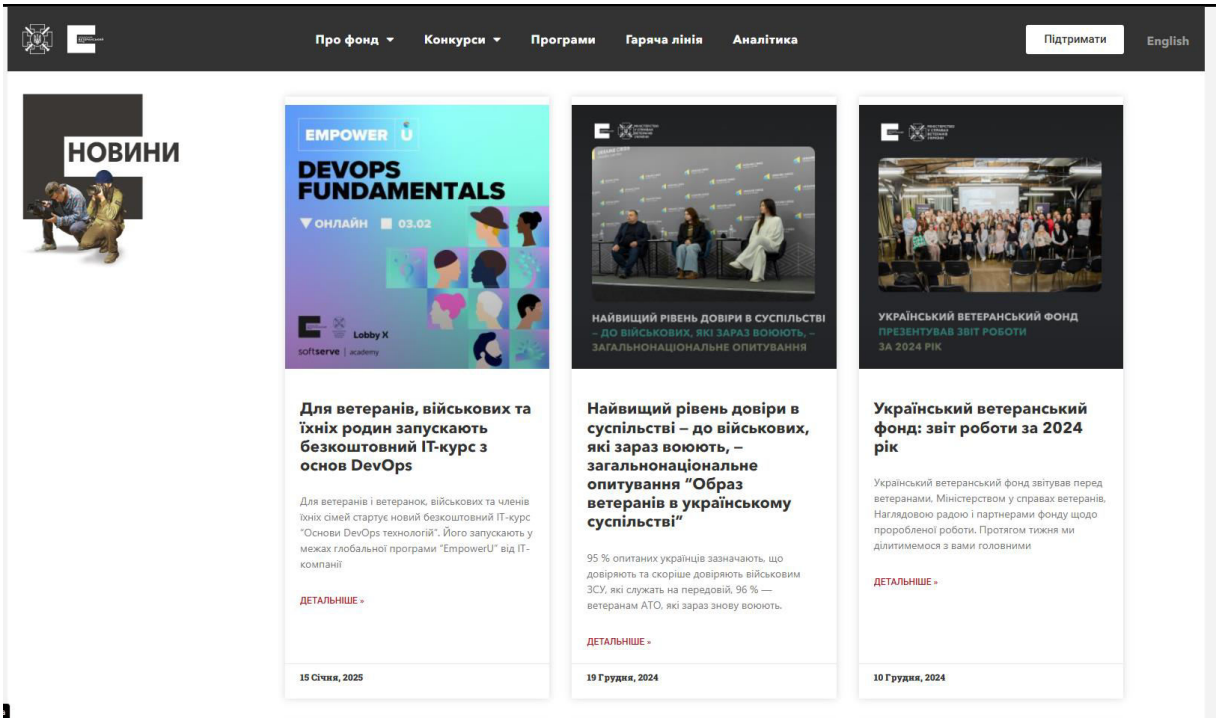


Рис. 1.3. Розділ вебсайту з новинами

Окремо представлено розділ «Конкурси» (рис. 1.4), де розміщуються можливості для отримання фінансування, грантів та підтримки ветеранських ініціатив. У ньому можна знайти інформацію про поточні

конкурси, умови участі, терміни подачі заявок і критерії відбору переможців (рис 1.5).

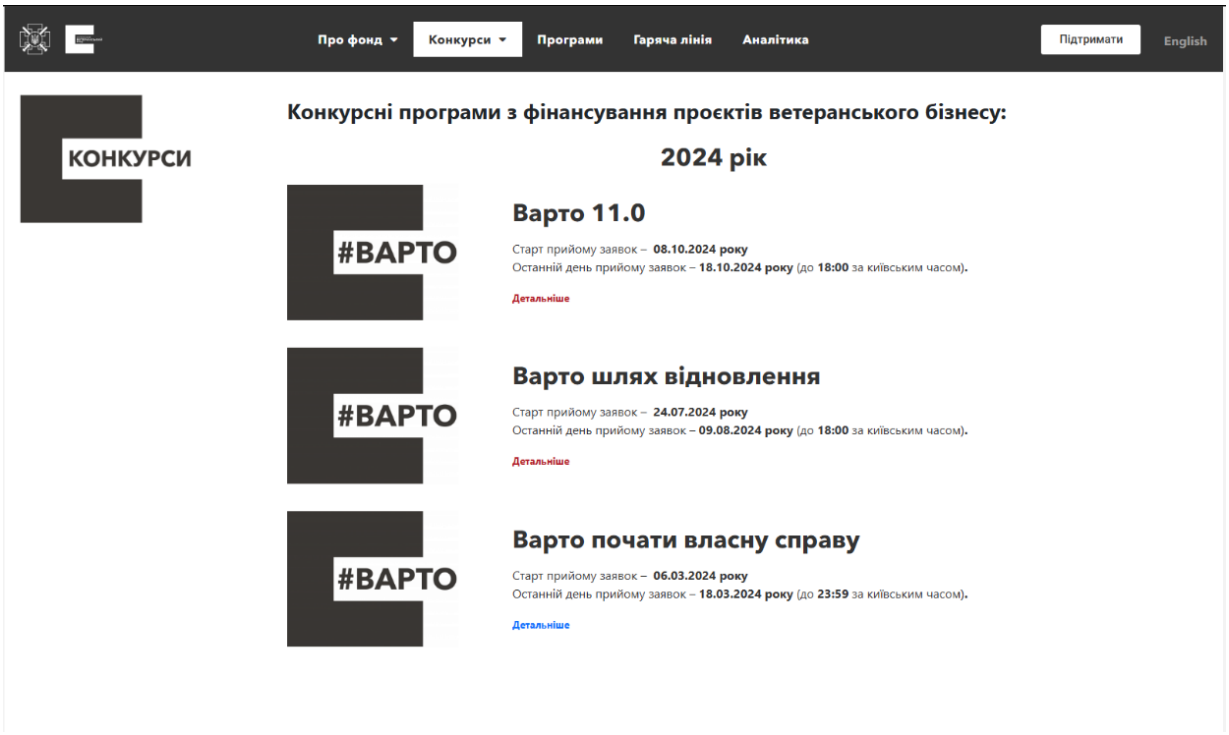


Рис. 1.4. Розділ вебсайту з конкурсними програмами

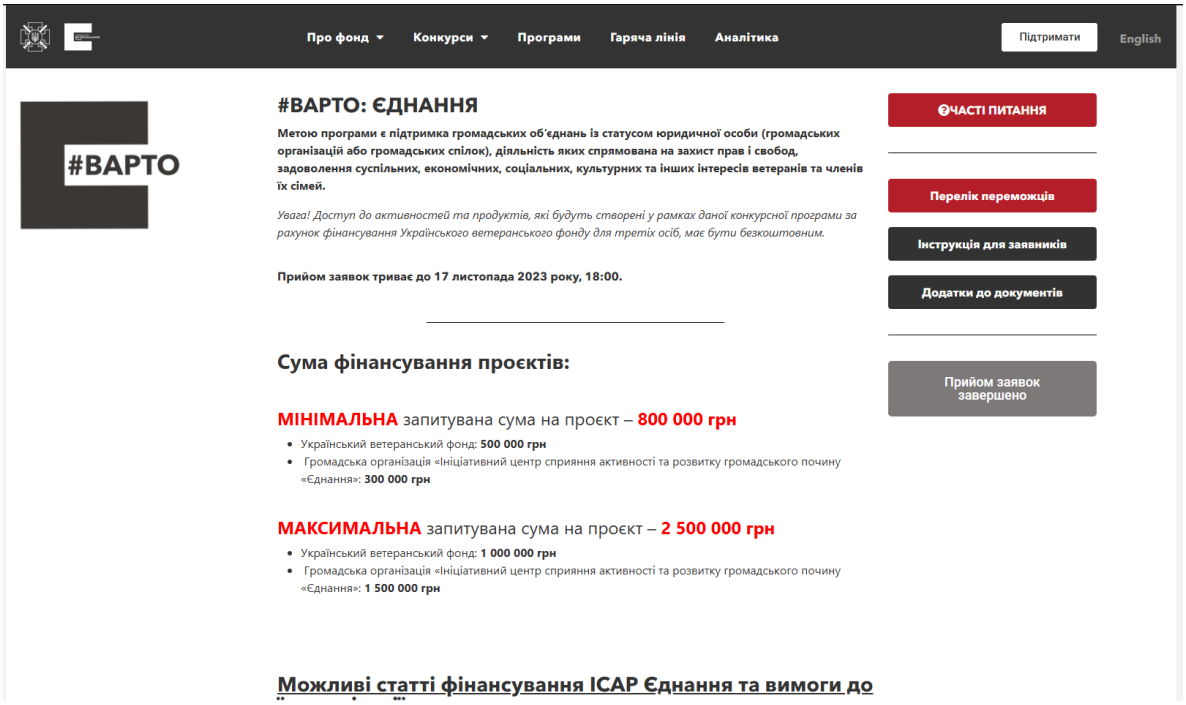


Рис. 1.5. Сторінка з інформацією про конкурс

Ресурс також включає розділ для отримання психологічної підтримки (рис. 1.6-1.7), де представлені контакти фахівців, а також поради щодо того, як знайти допомогу при різних психологічних проблемах, що виникають у ветеранів після повернення з війни.

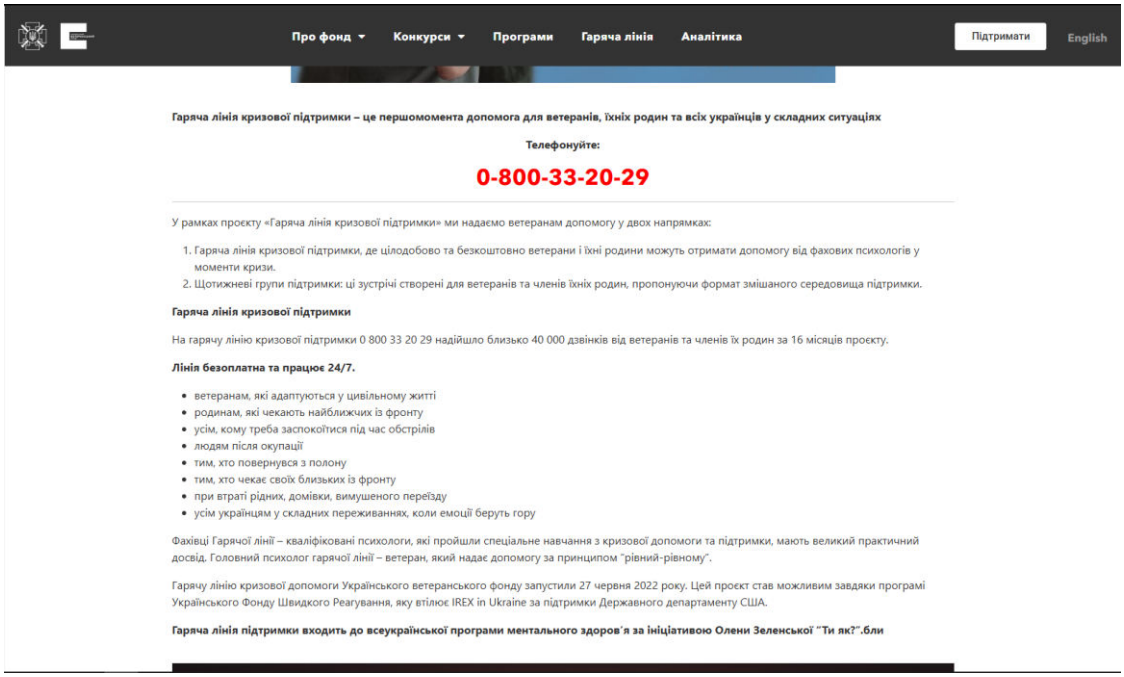


Рис. 1.6. Сторінка вебсайту з психологічної допомоги

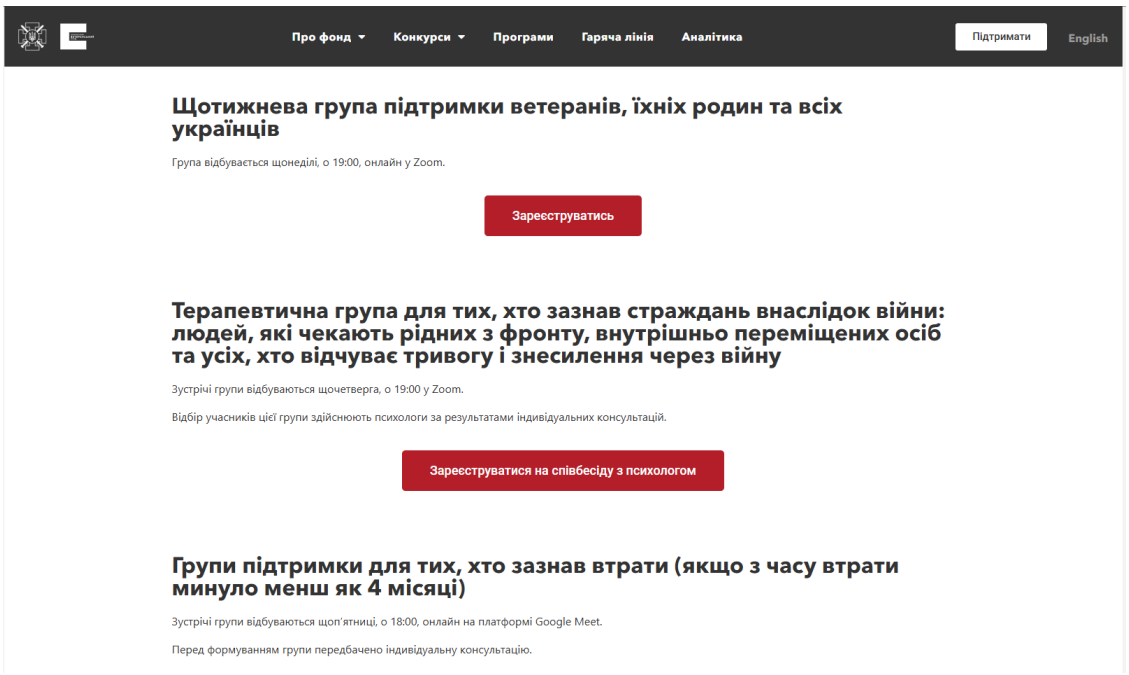


Рис. 1.7. Посилання на конференції та співбесіди з психологічної допомоги

Переваги сайту Українського фонду ветеранів:

- простий і доступний інтерфейс для ветеранів усіх вікових категорій;
- наявність інтерактивних функцій для подачі заявок та запису на медичні та соціальні послуги;
- наявність психологічної підтримки та консультацій для ветеранів;
- зручний доступ до інформації про державні та приватні програми підтримки;
- прозорість вебсайту, яка забезпечується відкритим доступом до фінансових та діяльнісних звітів, що дозволяє користувачам відстежувати використання коштів і результати реалізованих програм.

Проте є і деякі недоліки:

- відсутність персоналізованих рекомендацій щодо програм допомоги;
- необхідність удосконалення пошукової системи сайту для швидкого доступу до інформації;
- потреба в розширенні інтеграції з державними, громадськими та волонтерськими організаціями;
- відсутність повноцінного календаря подій для зручного планування;
- відсутність сервісів з працевлаштування для ветеранів;
- брак співпраці з місцевими громадами для залучення підтримки ветеранів на місцевому рівні.

1.3. Veteran Hub

Veteran Hub – це вебзастосунок, призначений для підтримки ветеранів, військовослужбовців та їхніх родин під час адаптації до цивільного життя. Основні напрями діяльності включають персональний супровід, юридичну та психологічну допомогу, надання можливостей працевлаштування, а також дослідження потреб ветеранів і пропонування

рішень для їхньої успішної інтеграції. Сайт орієнтований не лише на ветеранів, а й на їхні сім'ї та близьких, надаючи комплексну підтримку та інформацію про важливі програми та ініціативи. Крім-того сайт надає посібники та інформацію про те, як взаємодіяти, спілкуватися та працювати з ветеранами для цивільних та бізнесів [2].

Функціональні можливості сайту:

- можливість отримання юридичних, психологічних та соціальних консультацій онлайн, особисто або через мобільний офіс;
- можливість вибору ролі користувача (ветеран, подружжя, фахівець з психічного здоров'я, HR-фахівець) для персоналізованого перегляду контенту на вебсайті;
- перегляд вакансій для працевлаштування в проєкті Veteran Hub;
- функціональна можливість персонального супроводу для допомоги у вирішенні соціальних та професійних питань;
- база інформаційних матеріалів, статей та практичних посібників щодо адаптації, працевлаштування та психологічної підтримки;
- інтеграція з платформами донорів, волонтерських і бізнес-організацій для отримання фінансової допомоги;
- можливість перегляду досліджень та аналітичних матеріалів щодо потреб ветеранів;
- система навігації для переходу на допоміжні сайти та ресурси відповідно до конкретних програм підтримки.

Вебсайт містить розділ з послугами (рис. 1.8), що надаються фахівцями у сфері права, психології та соціальної роботи. Вони пропонують консультації для підтримки користувачів у різних життєвих ситуаціях. Доступні наступні види консультацій: юридичні консультації, психологічна підтримка, персональний супровід, допомога у пошуку роботи та освітніх можливостей (рис. 1.9).

Отримати консультацію можна одним із трьох способів: онлайн, особисто або через мобільний офіс, який виїжджає у різні регіони для

надання підтримки. Щоб записатися на консультацію, користувачі можуть звернутися на лінію підтримки за вказаним номером телефону та узгодити зручний формат і час зустрічі.

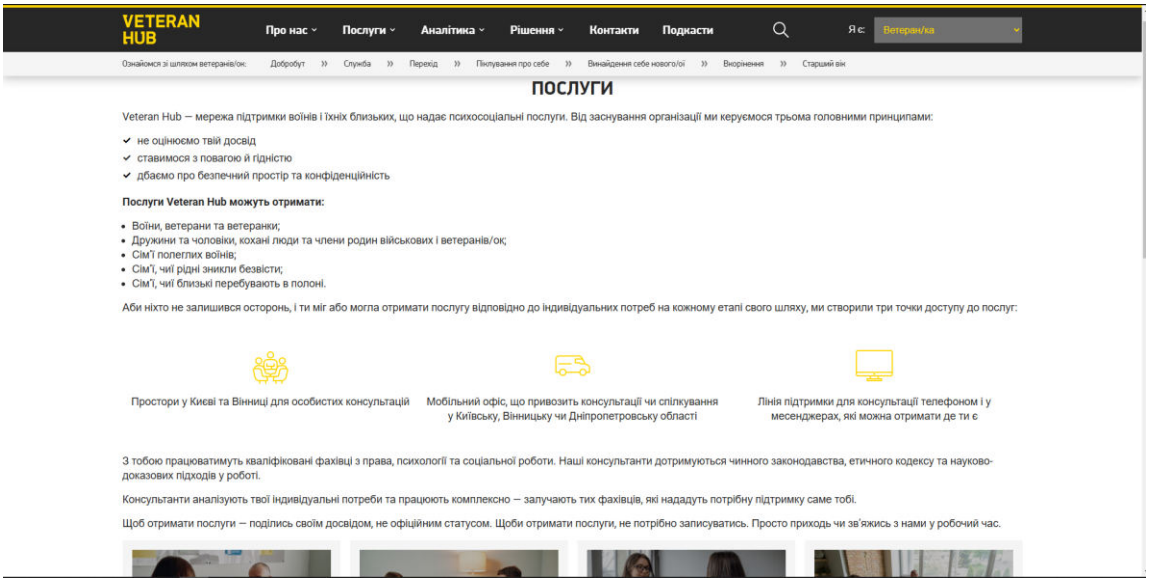


Рис. 1.8. Сторінка «Послуги» з послугами консультації

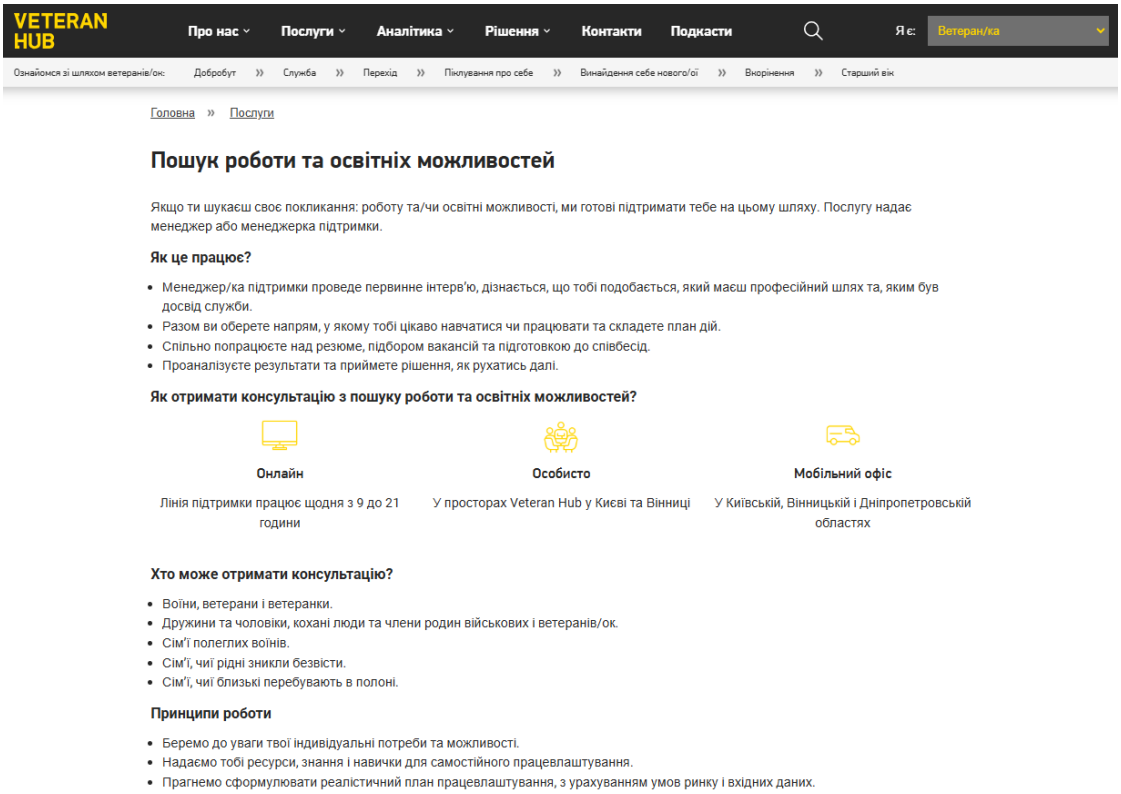


Рис. 1.9. Сторінка послуги консультації з пошуком роботи та освітніх можливостей

Вебзастосунок дозволяє користувачам обирати свою роль у меню навігації. Залежно від вибраної ролі змінюється головна сторінка, відображаючи релевантні ресурси. Наприклад, для ветеранів доступні статті та дослідження (рис. 1.10), для їхніх близьких – посібники та рекомендації з підтримки (рис. 1.11), для фахівців із психічного здоров'я – протоколи й методичні матеріали (рис. 1.12), а для HR-спеціалістів – інструкції щодо створення сприятливого робочого середовища (рис. 1.13).

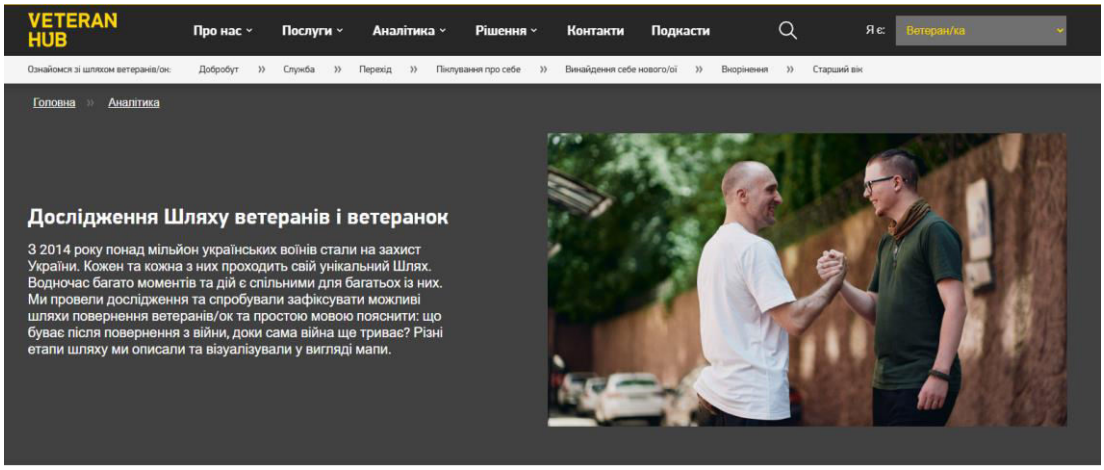


Рис. 1.10. Головна сторінка сайту з вибраної роллю «Ветеран/ка»

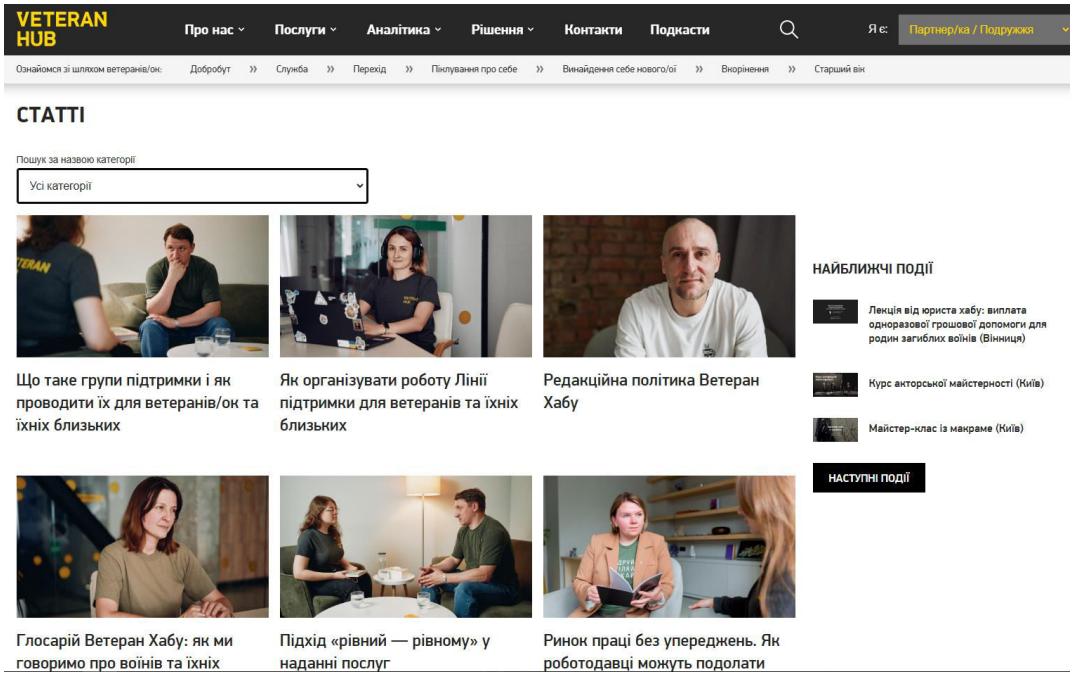


Рис. 1.11. Головна сторінка сайту з вибраної роллю «Партнер/ка / Подружжя»

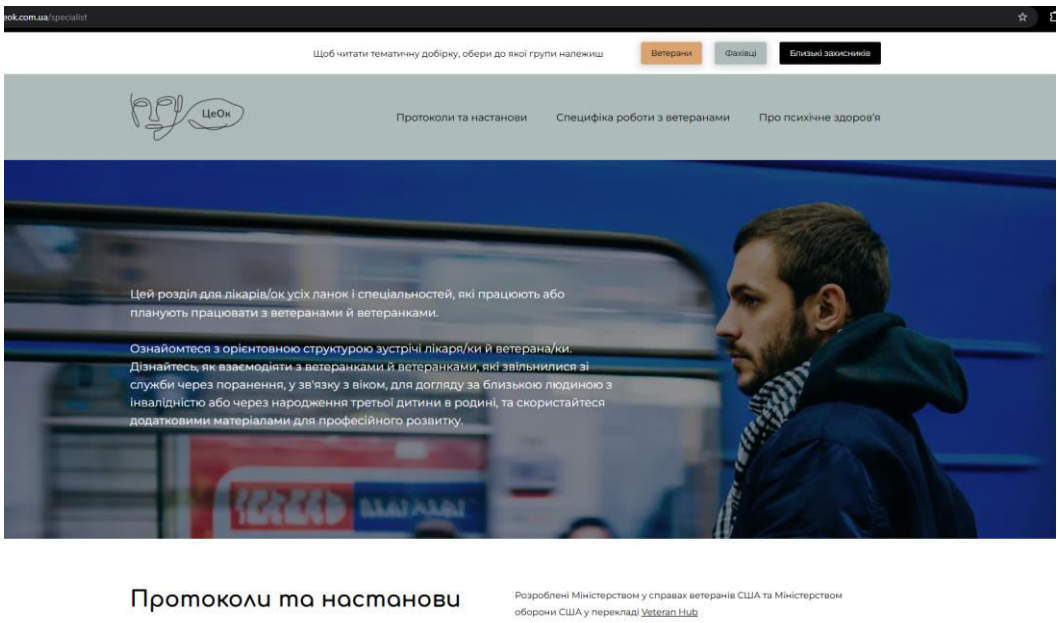


Рис. 1.12. Перехід на окремий сайт з вибраною роллю «Фахівці з психічного здоров'я»

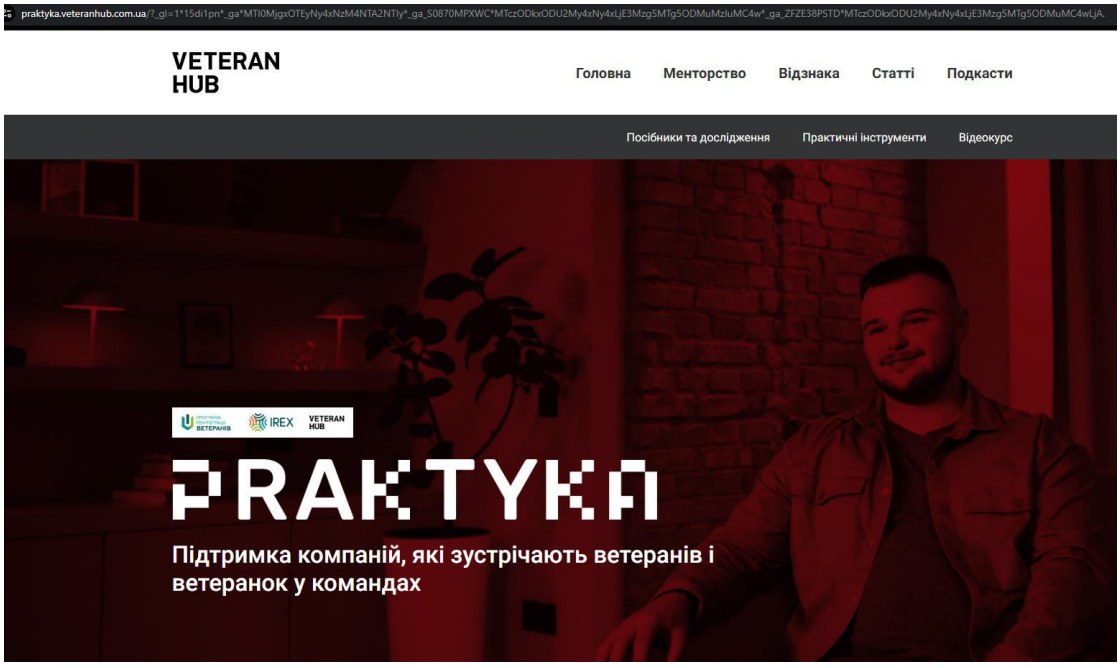


Рис. 1.13. Перехід на окремий сайт з вибраною роллю «HR фахівці»

На сайті також передбачена окрема сторінка з вакансіями (рис. 1.14), що відкриті в проєкті Veteran Hub. Робочі місця призначені не лише для ветеранів, а для всіх, хто має відповідний досвід та експертизу.

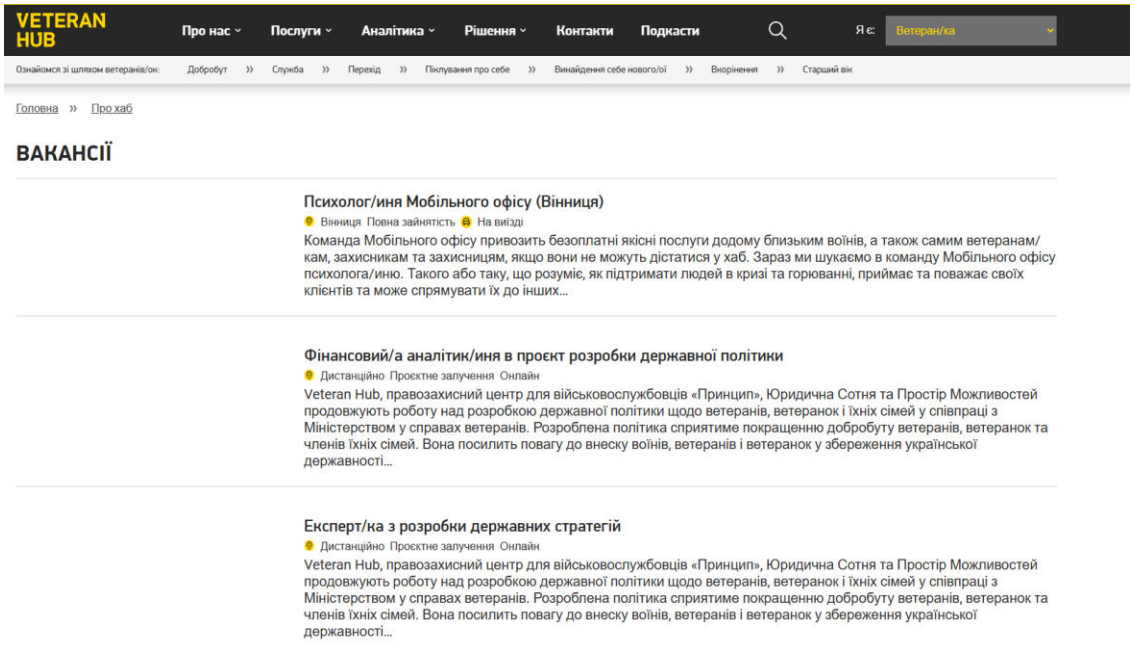


Рис. 1.14. Сторінка з доступними вакансіями на проєкті Veteran Hub

Однією з ключових особливостей вебсайту є база знань (рис. 1.15), яка містить усі статті, дослідження та аналітичні матеріали, підготовлені командою Veteran Hub. Цей розділ допомагає користувачам отримати доступ до актуальної інформації, рекомендацій та досліджень, що можуть бути корисними для ветеранів, їхніх родин, роботодавців та фахівців.

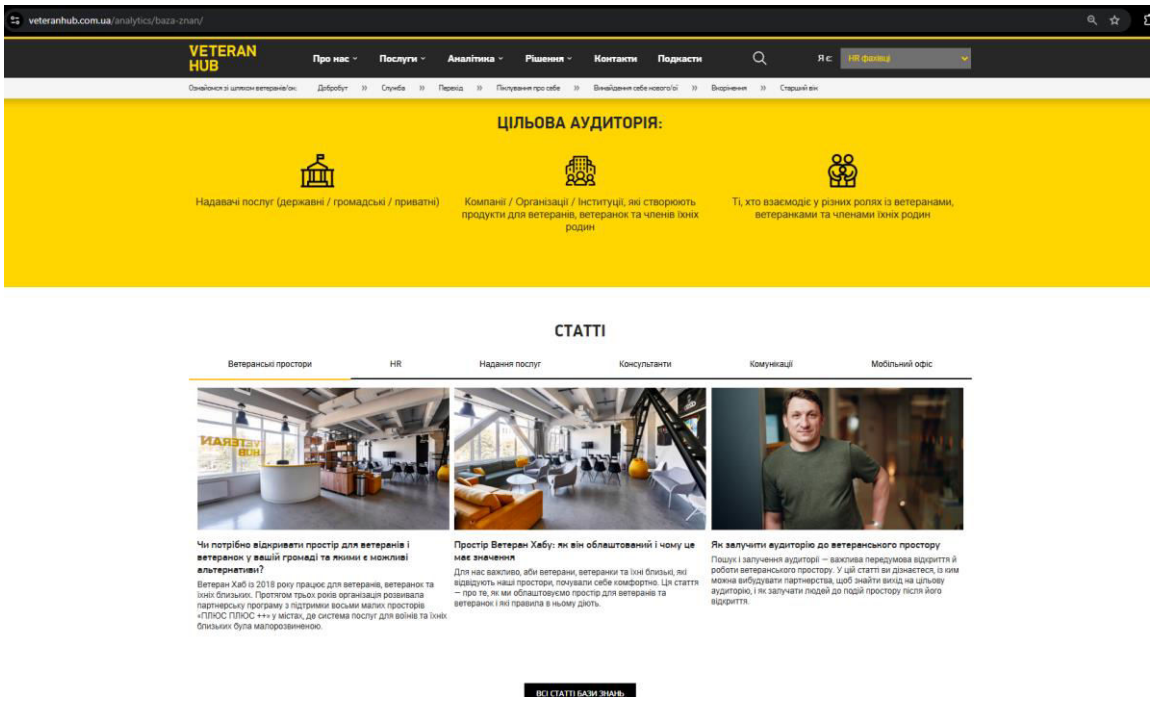


Рис. 1.15. Розділ вебсайту «База знань» з дослідженнями та статтями

Переваги вебсайту Veteran Hub:

- комплексний підхід до підтримки ветеранів, що охоплює юридичну, психологічну та соціальну допомогу і відповідає основним потребам військових та їхніх сімей;
- персоналізований контент з можливістю вибору ролі користувача, що підвищує релевантність інформації для ветеранів, їхніх родин, HR-фахівців та фахівців з психічного здоров'я;
- база знань та інформаційних ресурсів з доступом до досліджень, аналітики та методичних матеріалів для різних груп користувачів;
- гнучкі формати консультацій, включаючи онлайн-зустрічі, особисті прийоми та мобільний офіс, що забезпечує зручність для користувачів;
- інтеграція з бізнесами та волонтерськими організаціями, що дає можливість отримання фінансової допомоги та взаємодії з працедавцями;
- розділ вакансій з пропозиціями роботи, що підтримує працевлаштування ветеранів та сприяє їхній професійній реінтеграції;
- навігація по додаткових ресурсах, яка допомагає користувачам швидко знаходити спеціалізовані сервіси.

Недоліки вебсайту Veteran Hub:

- відсутність функціональної можливості для пошуку житла, що є критично важливим для ветеранів та їхніх сімей;
- обмежена взаємодія волонтерських організацій, оскільки вони не мають можливості координувати заходи та формувати спільноти;
- відсутність інструментів для навчання та перекваліфікації, які допомагають ветеранам адаптуватися до нового життя;
- немає інтерактивного календаря заходів, що ускладнює координацію подій для волонтерів та учасників спільноти;

- обмежений підхід до громадської інтеграції, оскільки фокус спрямований переважно на підтримку ветеранів без активного залучення місцевих громад;
- обмежена кількість вакансій для працевлаштування.

1.4. Порівняння конкурентів

Під час аналізу двох вебзастосунків конкурентів було виявлено, що обидва сервіси надають схожі базові функціональні можливості, орієнтовані на підтримку ветеранів та внутрішньо переміщених осіб, однак вони зосереджені на різних напрямах інтеграції. Один із сервісів більше акцентує увагу на працевлаштуванні та кар'єрному розвитку, тоді як інший надає перевагу соціальній адаптації та психологічній допомозі. У табл. 1.1 представлено детальний результат порівняння функціональних можливостей обох вебсайтів, що дозволяє зробити обґрунтовані висновки щодо їхніх переваг та недоліків. Проведене порівняння дає змогу оцінити найкращі практики, які можна застосувати у власному рішенні, а також виявити потенційні напрямки для вдосконалення функціональності майбутнього вебзастосунку.

Таблиця 1.1

Порівняння функціональних можливостей конкурентів

Вебзастосунок / Функціональна вимога	Український фонд ветеранів	Veteran Hub
Можливість перегляду інформації про консультаційні заходи та контакти психологів для звернення	Присутнє	Присутнє
Публікація оголошень про доступне житло для ветеранів	Відсутнє	Відсутнє

Продовження табл. 1.1

Можливість розміщення вакансій і подання відгуків на них	Частково присутнє	Відсутнє
Каталог навчальних закладів та програм перекваліфікації	Частково присутнє (перекваліфікація)	Відсутнє
Створення груп волонтерів та управління їх діяльністю	Відсутнє	Відсутнє
Публікація оголошень про волонтерські заходи	Частково присутнє	Відсутнє
Можливість публікації та перегляду оголошень	Відсутнє	Відсутнє
Створення оголошень про заходи від громади для підтримки ветеранів	Відсутнє	Відсутнє

1.5. Висновки

Під час аналізу двох вебзастосунків конкурентів було виявлено, що обидва сервіси мають схожу функціональність, проте зосереджені на різних аспектах інтеграції ветеранів. Жоден із них не пропонує достатньо можливостей для взаємодії з місцевими громадами та волонтерськими організаціями. Основний акцент цих платформ спрямований на психологічну підтримку ветеранів для їхньої соціальної адаптації, тоді як питання пошуку житла, роботи, навчальних закладів та підтримки волонтерських ініціатив залишаються недостатньо охопленими.

Veteran Hub надає індивідуальні та групові консультації з психологами, проте не має функціональності для допомоги ветеранам у пошуку роботи, житла чи освітніх можливостей. Український фонд ветеранів також приділяє увагу психологічній підтримці, але частково охоплює питання перекваліфікації через освітні програми. Водночас обидва сервіси не містять можливостей для розміщення вакансій, створення

каталогів навчальних закладів, інтеграції ветеранів у місцеві громади чи залучення їх до волонтерської діяльності.

Крім того, в жодному з аналізованих застосунків немає інструментів для організації волонтерських ініціатив, створення волонтерських груп чи публікації оголошень про відповідні заходи. Це створює прогалину у можливості залучення ветеранів до активної участі в житті громад. Також відсутні функціональні можливості для публікації інформації про доступне житло, що є критично важливим для тих, хто повертається до мирного життя.

Таким чином, обидва сервіси зосереджені переважно на соціальній та психологічній адаптації ветеранів, проте не надають комплексного рішення для їхньої інтеграції у суспільство через пошук житла, роботи, навчання та участь у громадському житті.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

На даному етапі потрібно сформувати основні вимоги до засобів реалізації програмного забезпечення та провести аналіз стеку технологій, які будуть використовуватись для розробки проєкту, та виконати їхнє порівняння.

Архітектура вебзастосунку для підтримки демобілізованих військових, ветеранів, їхніх сімей та волонтерських організацій складається із 3-ох основних частин: база даних, серверна та клієнтська частина. Даний підхід забезпечить ефективну роботу застосунку та зменшить «зчеплення» компонентів системи та дозволить в подальшому швидко та легко змінювати кожну з частин. Також це підвищить швидкість роботи застосунку, адже зменшиться навантаження на клієнтську частину, та впровадить додатковий захист, за який відповідає сервер.

База даних гарантує цілісність, безпеку та доступність інформації, необхідної для функціонування системи. Вона забезпечує ефективну організацію та зміну даних, здійснюючи їх обробку через запити.

Серверна частина виконує бізнес-логіку, обробляє клієнтські запити, взаємодіє з базою даних і клієнтською частиною. Вона відповідає за автентифікацію та авторизацію користувачів, обробку транзакцій, збереження даних у відповідних форматах та управління фоновими процесами. Для захисту даних та запровадження додаткової безпеки використовуються обмеження доступу, шифрування інформації, перевірка підозрілих дій та механізми захисту від поширених атак.

Клієнтська частина взаємодіє з сервером через HTTP-запити, обробляючи введені користувачем дані та відправляючи їх на сервер для подальшої обробки. Вона отримує відповіді сервера, відображає отриману інформацію в інтерфейсі та керує станом застосунку відповідно до змін у даних. Основні функції клієнтської частини включають аутентифікацію користувачів, отримання та відправлення даних, управління елементами

інтерфейсу відповідно до прав доступу та підтримку інтерактивних операцій, таких як оновлення списків, відображення повідомлень про статус запитів та обробку дій користувачів.

2.1. Вибір технологій для розроблення серверної частини

Для реалізації серверної частини вебзастосунку мова програмування та відповідний до неї фреймворк мають задовольняти наступні критерії.

1) підтримка високопродуктивного асинхронного виконання;
2) модульність та гнучкість розширення;
3) широкий набір доступних бібліотек та інструментів;
4) зручність роботи з залежностями та пакетами;
5) інструменти для безпечної обробки даних;
6) масштабованість системи та підтримка високих навантажень;
7) підтримка інтеграцій та комунікації з іншими сервісами;
8) документація та доступність навчальних матеріалів.

2.1.1. Python Django

Python – високорівнева інтерпретована об’єктно-орієнтована мова програмування, яка здобула популярність завдяки простоті синтаксису, динамічній типізації та розвиненій структурі даних. Завдяки підтримці модулів та пакетів, розробники можуть легко реорганізувати код і ізолювати помилки на рівні окремих компонентів. Відсутність компіляції перед виконанням сприяє швидкому циклу розробки та виявленню помилок, хоча цей підхід може негативно впливати на продуктивність при виконанні коду рядок за рядком. Крім того, наявність великої кількості вбудованих функцій значно спрощує розробку, але інколи призводить до більшого споживання пам’яті порівняно з іншими мовами [3].

Серед основних переваг Python можна виділити [4]:

1) підтримку об’єктно-орієнтованої парадигми, що сприяє структурованій розробці;

2) модульність, яка дозволяє повторне використання коду;

3) динамічну типізація;

4) кросплатформеність, яка гарантує сумісність застосунків із різними операційними системами;

5) зрозумілий і легкий для читання синтаксис;

6) швидше писати в порівнянні з іншими мовами;

7) високий вибір бібліотек та модулів.

Водночас існують і деякі недоліки [4]:

1) нижча продуктивність та швидкість через інтерпретовану природу мови;

2) обмежена підтримка багатопоточності;

3) високе споживання пам’яті.

Django – це потужний вебфреймворк на мові Python, який легко справляється з взаємодією з базами даних. Його представлення (views) приймають HTTP-запити та повертають відповіді, що дозволяє будувати логіку застосунку. Шаблони в Python і Django дають змогу відокремити код відображення від бізнес-логіки, а маршрутизатор URL-адрес направляє запити до відповідних представлень. Система обробки форм і функції автентифікації спрощують роботу з введенням користувачів і безпекою. Нарешті, адміністративний інтерфейс автоматизує CRUD-операції з моделями даних. [5].

Django має наступні переваги:

1. Швидка розробка: завдяки наявності ORM, вбудованої системи автентифікації, адміністративного інтерфейсу та інших компонентів, розробка стає ефективнішою, що дозволяє зосередитись на бізнес-логіці.
2. Безпека: фреймворк автоматично впроваджує захист від типових вебзагроз (SQL-ін’єкції, XSS, CSRF тощо). Це робить його надійним для комерційних застосунків.

3. Масштабованість: Django розроблений для обробки сайтів з високим трафіком. Тому він чудово підходить для створення масштабованих застосунків.
4. Вбудовані інструменти: фреймворк постачається з великою кількістю вбудованих компонентів: аутентифікація, керування сесіями, обробка форм, взаємодія з базами даних тощо. Це скорочує час розробки та зменшує потребу у сторонніх бібліотеках.
5. Велика документація: якісна та обширна документація дозволяє швидко знаходити рішення для складних завдань.

Однак, Django має і певні недоліки:

1. Стрімка крива навчання: попри хорошу документацію, Django має досить стрімку криву навчання, особливо для новачків. Через складну архітектуру та велику кількість можливостей, потрібно час, щоб глибоко його опанувати.
2. Монолітна структура: Django має жорстко визначену архітектуру з багатьма вбудованими компонентами, що спрощує розробку, але ускладнює адаптацію або інтеграцію зі сторонніми системами.
3. Обмежена гнучкість: через підхід "конвенція понад конфігурацію" Django може обмежувати свободу розробника в прийнятті нестандартних архітектурних рішень.
4. Навантаження на продуктивність: високорівневі абстракції та ORM можуть створювати додаткове навантаження, особливо при складних запитах до бази даних або інтенсивній обробці даних.
5. Неідеальний для SPA: Django орієнтований на серверний рендеринг, тому не завжди зручний для створення односторінкових застосунків із великим обсягом логіки на клієнті.

1.1.3. Java Spring

Java – це об’єктно-орієнтована, кросплатформна мова програмування, яка забезпечує високу продуктивність, безпеку та стабільність у великих

програмних системах. Однією з головних переваг Java є принцип «Write Once, Run Anywhere», що дозволяє виконувати скомпільований код на будь-якій платформі, де встановлена Java Virtual Machine (JVM). Завдяки цьому Java активно використовується у корпоративному секторі, фінансових системах, мобільних застосунках (Android), хмарних рішеннях та великих вебзастосунках.

Java підтримує статичну типізацію та має суворий механізм управління пам’яттю за допомогою Garbage Collector. Це допомагає зменшити кількість помилок та підвищує надійність коду. Також Java активно використовується у високонавантажених системах завдяки можливостям багатопотокового програмування та ефективного керування потоками. Велика кількість бібліотек дозволяє прискорити процес розробки та адаптувати Java для різних сценаріїв використання [6, 7].

Переваги Java:

1) кросплатформеність – можливість виконання коду на будь-якій системі з JVM;
2) висока безпека – сувора типізація та контроль виконання коду;
3) масштабованість – ефективне використання багатопотоковості та паралельних обчислень;
4) розвинена екосистема – підтримка великої кількості бібліотек та інструментів;
5) підтримка ООП – використання класів, об’єктів, наслідування та поліморфізму;
6) широка спільнота – велика кількість документації, форумів та активних розробників;
7) модульність – легкість розділення коду на незалежні частини;
8) управління пам’яттю – автоматичне прибирання непотрібних об’єктів за допомогою Garbage Collector.

Недоліки Java:

1) високе споживання ресурсів – JVM може використовувати багато пам’яті та процесорного часу;
2) повільний запуск – час запуску застосунківів довший, ніж у мовах із рідною компіляцією;
3) необхідність компіляції – будь-які зміни в коді потребують повторної компіляції перед виконанням;
4) багатослівність – код у Java довший у порівнянні з іншими мовами (наприклад, Python);
5) залежність від JVM – програма не може працювати без встановленої віртуальної машини;
6) не гнучкий синтаксис – може вимагати написання більшої кількості шаблонного коду та ускладнювати розробку.

Spring – це потужний фреймворк для Java, який використовується для створення вебзастосунків, мікросервісів та корпоративних систем. Головною особливістю Spring є інверсія управління (IoC), яка дозволяє легко керувати залежностями між компонентами застосунку, що значно покращує його гнучкість і масштабованість. Завдяки механізму аспектно-орієнтованого програмування (AOP), розробники можуть ефективно реалізовувати логіку, що повторюється, наприклад, логування, безпеку або управління транзакціями.

Одним із найпопулярніших модулів фреймворку є Spring Boot, який значно спрощує розробку та розгортання застосунків, автоматично налаштовуючи базові параметри та інтегруючи готові компоненти. Spring також підтримує роботу з REST API, інтеграцію з базами даних (Spring Data), безпеку (Spring Security) та мікросервісну архітектуру (Spring Cloud) [8].

Переваги Spring [9, 10]:

1) гнучкість та масштабованість – підходить як для монолітних, так і для мікросервісних архітектур;

2) модуль Spring Boot – спрощує розробку та автоматично налаштовує середовище;
3) висока продуктивність – оптимізоване управління ресурсами та потоками;
4) широка підтримка спільноти – велика кількість документації та прикладів використання;
5) контейнер керування залежностями – автоматичне створення об’єктів та ін’єкція залежностей;
6) підтримка REST API – вбудовані механізми для створення RESTful-сервісів;
7) зручна конфігурація – підтримка анотацій та YAML-файлів для налаштувань;
8) безпека – Spring Security дозволяє легко реалізувати авторизацію та автентифікацію.

Недоліки [10]:

1) високий поріг входу – складність у вивченні через велику кількість модулів та концепцій;
2) споживання ресурсів – Spring-застосунки можуть вимагати більше оперативної пам’яті;
3) складність конфігурації – для кастомних налаштувань може знадобитися багато часу;
4) залежність від зовнішніх бібліотек – для повноцінної роботи потрібно багато додаткових бібліотек;

2.1.3. C# ASP.NET

C# – це одна з основних мов програмування, що використовується для створення програмного забезпечення в екосистемі .NET. Завдяки своїй інтеграції з платформою .NET, C# є надзвичайно популярним інструментом серед розробників для створення як стаціонарних, так і мобільних застосунків, а також для веброзробки. Мова має потужний набір

інструментів, які підтримують різноманітні архітектурні підходи, включаючи об’єктно-орієнтоване програмування, мікросервіси, а також асинхронне та багатопотокове виконання. Важливими перевагами C# є чітка типізація, яка знижує ймовірність помилок під час компіляції, а також вбудовані функції для роботи з пам’яттю через механізм автоматичного збору сміття (Garbage Collection), що забезпечує ефективне управління ресурсами.

Крім того, C# має сильну підтримку багатьох сучасних стандартів програмування, таких як LINQ (Language Integrated Query), що дозволяє зручно працювати з колекціями даних, і async/await, що полегшує написання асинхронного коду, зокрема для високонавантажених серверів та клієнтських застосунків. Платформа .NET дає змогу розробляти продуктивні й масштабовані програми завдяки оптимізаціям на рівні Common Language Runtime (CLR), що дозволяє компілювати код в рідний код і забезпечує високу швидкість виконання програм [11, 12].

Переваги C#:

1. Об’єктно-орієнтоване програмування (ООП) – C# підтримує всі основні принципи ООП, що дозволяє створювати модульні, масштабовані та підтримувані застосунки.
2. Інтеграція з платформою .NET – .NET надає доступ до великої кількості бібліотек та інструментів для розробки, включаючи ASP.NET для вебзастосунків, Entity Framework для роботи з базами даних, та багато інших.
3. Сувора типізація – завдяки статичній типізації, помилки можуть бути виявлені на етапі компіляції, що зменшує ймовірність багів під час виконання програми.
4. Робота з багатопоточністю та асинхронним виконанням – C# пропонує потужні механізми для роботи з паралельним виконанням задач.

5. Висока продуктивність – завдяки компіляції в рідний код і оптимізаціям CLR, C# забезпечує високу швидкість виконання.
6. Підтримка сучасних стандартів програмування – включає інноваційні функції, такі як LINQ для запитів до даних та async/await для асинхронного програмування, що спрощує розробку складних систем.
7. Кросплатформеність (через .NET Core) – за допомогою .NET Core, C# програми можуть працювати на різних операційних системах, таких як Windows, Linux та macOS.

Недоліки C#:

1. Залежність від .NET – мова C# тісно пов'язана з .NET Framework або .NET Core, що обмежує вибір середовищ розробки й ускладнює використання поза екосистемою Microsoft.
2. Повільний цикл розробки – зміни в коді вимагають повторної компіляції програми, що може сповільнювати процес розробки, особливо при роботі з великими проєктами.
3. Обмеження платформою Microsoft – незважаючи на наявність .NET Core, який підтримує кросплатформенність, C# все ще частіше використовується в межах Windows.
4. Обмежена підтримка для функціонального програмування – хоча екосистема .NET досить велика, кількість бібліотек і інструментів, доступних для C#, може бути меншою, ніж у популярних мовах, таких як Python або JavaScript.

ASP.NET – це потужний вебфреймворк від Microsoft, який використовується для створення динамічних вебзастосунків, API та сервісів. Він є частиною платформи .NET і підтримує різні архітектурні стилі, включаючи MVC (Model-View-Controller) та Web API. Завдяки інтеграції з .NET ASP.NET забезпечує високу продуктивність, безпеку та гнучкість у розробці. Фреймворк підтримує асинхронне програмування, що дозволяє ефективно обробляти велику кількість запитів без значного

навантаження на сервер. Також він має розширені можливості для роботи з базами даних, аутентифікації та авторизації користувачів [13].

Переваги ASP.NET:

1. Висока продуктивність – завдяки компіляції в проміжний код та оптимізації на рівні CLR, ASP.NET забезпечує швидке виконання вебзастосунків. Використання технологій на кшталт Kestrel дає можливість обробляти тисячі запитів одночасно.
2. Інтеграція з екосистемою .NET – ASP.NET безшовно працює з іншими бібліотеками .NET, що спрощує розробку серверної логіки, роботи з базами даних через Entity Framework та інтеграцію з іншими сервісами Microsoft.
3. Гнучкість у розробці – підтримка різних шаблонів проєктування, таких як MVC, Web API, Razor Pages, SignalR, дозволяє вибрати оптимальну архітектуру для конкретного випадку.
4. Безпека – ASP.NET містить вбудовані механізми безпеки, включаючи захист від XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery) та SQL-ін’єкцій. Також є інтеграція з Microsoft Identity для автентифікації та авторизації.
5. Кросплатформеність – починаючи з .NET Core, ASP.NET можна використовувати не лише на Windows, а й на Linux та macOS, що значно розширює можливості розробки та розгортання.

Недоліки ASP.NET:

1. Залежність від .NET – хоча ASP.NET працює незалежно від Windows, він все ж потребує середовища .NET, що може ускладнити використання його у проєктах, де потрібна повна незалежність від платформи.
2. Обмежена кількість сторонніх бібліотек у порівнянні з іншими фреймворками – попри активний розвиток .NET, деякі специфічні бібліотеки для веброзробки, доступні у JavaScript або Python, можуть бути недоступні або мати менш розвинені аналоги.

3. Складність налаштування та конфігурації – деякі аспекти конфігурації ASP.NET, особливо при роботі з контейнерами або мікросервісами, можуть вимагати глибокого розуміння інструментів DevOps.

3.1.3. Результат проведеного аналізу

Були встановлені основні вимоги до технологій, що використовуватимуться для розробки серверної частини застосунку. Під час проведення аналізу було детально розглянуто кожну з технологій та визначено їх переваги і недоліки (Python, Java, C#). Нижче представлено порівняння (табл. 2.1) за обраними критеріями.

Таблиця 2.1

Порівняння технологій для реалізації серверної частини

Технологія / Критерій порівняння	Python Django	Java Spring	C# ASP.NET
Високопродуктивне асинхронне виконання	+/	+	+
Модульність та гнучкість розширення	+/–	+	+
Широкий набір доступних бібліотек та інструментів	+	+	+
Робота з залежностями та пакетами	+/–	+	+
Інструменти для безпечної обробки даних	+	+	+
Масштабованість та підтримка високих навантажень	+/–	+	+
Інтеграції та комунікація з іншими сервісами	+	+	+
Документація та навчальні матеріали	+	+	+

Результати порівняння технологій для розробки серверної частини застосунку показують, що Java Spring та C# ASP.NET задовольняють усі поставлені вимоги. Обидва фреймворки є потужними інструментами для розробки масштабованих вебзастосунків, підтримують високу продуктивність, безпеку та інтеграцію з іншими сервісами.

Python Django є зручним фреймворком, що пропонує швидку розробку, широкий набір бібліотек та зручну систему безпеки. Проте, у порівнянні з Java Spring та ASP.NET, Django має певні обмеження у продуктивності та масштабованості, що може створювати труднощі при розробці великих та високонавантажених систем. Це робить Django чудовим вибором для стартапів або невеликих вебзастосунків, але менш придатним для проєктів, що вимагають складних мікросервісних архітектур та розподілених систем.

Java Spring демонструє відмінну масштабованість, підтримку мікросервісної архітектури та широкий набір інструментів для інтеграції з іншими сервісами. Його перевагою є велика кількість бібліотек та готових рішень для корпоративного програмного забезпечення. Проте, Java має досить складний синтаксис та високу складність налаштувань, що може ускладнювати роботу розробників, особливо якщо вони не мають досвіду з цією технологією. Також управління залежностями в Java Spring може бути менш зручним у порівнянні з конкурентами.

C# ASP.NET є найкращим вибором для розробки серверної частини застосунку, оскільки поєднує високу продуктивність, простоту налаштувань, ефективну роботу з залежностями та безпеку. Його інтеграція з екосистемою Microsoft, включаючи підтримку Azure, надає додаткові переваги для хмарних рішень. Крім того, C# має більш приємний і зрозумілий синтаксис у порівнянні з Java, що робить його доступнішим для розробників та спрощує написання чистого та підтримуваного коду. ASP.NET також має потужну підтримку асинхронного програмування, що

дозволяє створювати високонавантажені системи з мінімальними затримками.

Таким чином, C# ASP.NET є оптимальним вибором для розробки серверної частини вебзастосунку, оскільки забезпечує найкраще поєднання продуктивності, зручності розробки, інтеграції з хмарними технологіями та легкого для читання коду.

2.2. Вибір технологій для реалізації клієнтської частини

Для реалізації клієнтської частини вебзастосунку мова програмування та відповідний до неї фреймворк мають задовольняти наступні критерії:

1) продуктивність та ефективність рендерингу;
2) гнучкість та масштабованість;
3) зручність роботи з API та інтеграціями;
4) кросплатформеність та адаптивність;
5) механізми керування станом застосунку;
6) безпека та захист від вразливостей;
7) підтримка розширених можливостей UI/UX;
8) документація та наявність навчальних матеріалів.

2.2.1. Angular

Angular – це open-source фреймворк, який підтримує Google і має велику спільноту розробників. Першою версією був AngularJS, який базувався на архітектурі MVC (Model-View-Controller) і дозволяв створювати односторінкові застосунки (SPA) із двонапрямленим зв’язком даних. Наступні версії Angular (2 і вище) перейшли на компонентний підхід і використовують TypeScript, що зробило розробку більш структурованою та масштабованою.

AngularJS забезпечує двонапрямлений зв’язок даних, що автоматично оновлює інтерфейс при зміні моделі, зменшуючи навантаження на розробника. Велика кількість навчальних матеріалів, плагінів і форумів

робить роботу з ним зручною та швидкою. Концепція ін’єкції залежностей допомагає розділяти код на незалежні компоненти, підвищуючи його гнучкість і повторне використання. Директиви розширюють HTML, даючи змогу створювати інтерактивний і насичений контент.

Водночас продуктивність великих SPA на AngularJS іноді залишає бажати кращого, через що застосунки можуть працювати з затримками. Ще одним недоліком є необхідність підтримки JavaScript у браузері, адже без нього застосунок не працюватиме.

Сучасні версії Angular базуються на компонентній архітектурі, що підвищує повторне використання коду та робить його більш структурованим. Кожен елемент інтерфейсу – окремий компонент, що спрощує розробку, тестування і масштабування застосунків. Водночас фреймворк продовжує підтримувати принципи MVC, розділяючи логіку моделі, подання і контролера, що допомагає підтримувати чисту архітектуру.

Angular демонструє високу продуктивність завдяки ієрархічній ін’єкції залежностей, диференціальному завантаженню, серверному рендерингу через Angular Universal та компілятору Ivy, який пришвидшує збірку та оптимізацію коду. Вбудовані елементи і директиви дозволяють розширювати HTML і використовувати компоненти в різних контекстах [14].

Однією з ключових особливостей сучасного Angular є Ahead of Time (AoT) компіляція, яка дозволяє перевіряти і трансформувати шаблони під час збірки, а не під час виконання програми. Це знижує ризики, пов’язані з ін’єкціями шаблонів і іншими вразливостями, оскільки більшість помилок і небезпечного коду виявляються ще на етапі компіляції. Для боротьби з атаками типу Cross-Site Scripting (XSS) Angular за замовчуванням використовує контекстно-залежне кодування виводу та очищення шкідливого коду. Крім того, у нових версіях поліпшено іменування методів

і API, що дозволяє розробникам свідомо застосовувати безпечні практики [15].

2.2.2. React

React – це сучасна JavaScript бібліотека, розроблена для створення інтерфейсів користувача. Вона дозволяє будувати складні вебзастосунки, розбиваючи їх на окремі компоненти – невеликі, автономні частини UI, які можна повторно використовувати. Кожен компонент у React має власний стан і властивості, що дає змогу гнучко керувати тим, як виглядає і поводиться користувацький інтерфейс.

Особливістю React є те, що він працює на основі віртуального DOM – внутрішньої копії структури сторінки, яка дозволяє ефективно відстежувати зміни та оновлювати тільки ті елементи, які дійсно потребують перерендерингу. Це значно підвищує продуктивність та плавність роботи інтерфейсу. React підтримує клієнтську маршрутизацію, що дає змогу створювати односторінкові застосунки (SPA) без повного перезавантаження сторінок. При цьому логіка маршрутизації виконується безпосередньо у браузері, що дозволяє швидко змінювати вміст і зберігати стан програми.

Ще один важливий аспект – це інтеграція React з іншими вебтехнологіями і сервісами. React часто використовують разом з REST API чи GraphQL для отримання даних з серверів, а також для реалізації сучасних механізмів авторизації, зокрема через кукі чи JWT. Однак робота з зовнішніми сервісами потребує врахування обмежень браузера, таких як політика CORS.

React активно розвивається та має велику спільноту, що створює безліч додаткових бібліотек і інструментів – від UI-компонентів до засобів тестування. Це робить React дуже гнучким та адаптованим для будь-яких задач – від простих вебсайтів до складних корпоративних систем [16].

2.2.3. Vue.js

Vue.js – це бібліотека для створення інтерактивних вебінтерфейсів. Її мета – надати переваги реактивного зв'язування даних і композиційних компонентів з максимально простою API.

Vue.js зосереджена лише на шарі відображення, тому її легко вивчити і інтегрувати з іншими бібліотеками чи існуючими проєктами. Водночас, у поєднанні з відповідними інструментами та бібліотеками, Vue.js може стати основою складних односторінкових застосунків.

В основі Vue.js лежить реактивна система зв'язування даних, яка дуже спрощує синхронізацію даних і DOM. На відміну від маніпуляцій DOM вручну через jQuery, тут використовується декларативне прив'язування, що означає, що DOM автоматично оновлюється при зміні даних. Це робить код простішим для написання, розуміння і підтримки.

Архітектура Vue.js базується на шаблоні MVVM (Model-View-ViewModel), де ViewModel (екземпляр Vue) зв'язує модель і представлення. Наприклад, при зміні властивості моделі, відповідні частини DOM оновлюються автоматично без необхідності писати код для прямої роботи з DOM.

Vue.js також підтримує спеціальні директиви – атрибути, що починаються з v-, які надають реактивну поведінку для елементів DOM. Наприклад, v-if керує видимістю елемента залежно від стану, а v-for дозволяє циклічно відображати списки. Завдяки цим директивам можна не лише зв'язувати текст, а й динамічно керувати структурою DOM.

Ще одним ключовим поняттям є система компонентів, яка дозволяє будувати великі застосунки з маленьких автономних, часто повторно використовуваних частин. Інтерфейс будь-якого застосунку можна представити у вигляді дерева компонентів, де кожен компонент відповідає за свою частину UI.

Загалом, система компонентів є основою для побудови великих застосунків у Vue.js, а екосистема пропонує додаткові інструменти і бібліотеки, що дозволяють створювати повноцінні фреймворки на базі Vue [17].

2.2.4. Результат проведеного аналізу

Було проведено аналіз технологій для розробки клієнтської частини застосунку, зокрема Angular, React і Vue.js. Тепер складемо порівняння (табл. 2.2), у якому кожен із фреймворків буде оцінений відповідно до визначених вимог.

Таблиця 2.2

Порівняння технологій для реалізації клієнтської частини

Технологія / Критерій порівняння	Angular	React	Vue.js
Продуктивність та ефективність рендерингу	+/–	+	+
Гнучкість та масштабованість	+	+	+/–
Зручність роботи з API та інтеграціями	+	+	+
Кросплатформеність та адаптивність	+	+	+
Механізми керування станом застосунку	+	+	+
Безпека та захист від вразливостей	+	+/–	+/–
Підтримка розширених можливостей UI/UX	+	+	+
Документація та наявність навчальних матеріалів	+	+	+

Результати порівняння технологій для розробки клієнтської частини застосунку показують, що Angular, React і Vue.js мають свої сильні та слабкі сторони. Angular забезпечує високий рівень безпеки, потужні механізми роботи з API та вбудовані інструменти для маршрутизації та керування станом, що робить його оптимальним вибором для масштабованих і корпоративних рішень. Проте його складна структура може впливати на швидкість рендерингу, поступаючись у цьому аспекті React та Vue.js.

Vue.js вирізняється гнучкістю у створенні UI/UX та має простий API для роботи з сервером. Його легкість у використанні робить його чудовим вибором для швидкої розробки, однак для забезпечення високого рівня безпеки може знадобитися додаткове налаштування. React, своєю чергою, добре інтегрується з API та підтримує популярні менеджери стану, що робить його зручним для динамічних застосунків. Водночас питання безпеки в React вирішуються переважно за допомогою сторонніх бібліотек, що може ускладнювати підтримку та інтеграцію.

Таким чином, серед розглянутих технологій Angular є найкращим вибором для побудови клієнтської частини вебзастосунку RefugeUA, оскільки вже містить усі необхідні інструменти для безпечної та ефективної роботи без потреби у встановленні додаткових залежностей.

2.3. Обґрунтування вибору системи керування базами даних

Для вибору системи керування базами даних, щоб побудувати базу даних для вебзастоснуку, було сформовано такі критерії:

1) ефективну обробку великої кількості даних і запитів;
2) захист збереженої інформації від несанкціонованого доступу та видалення;
3) підтримку механізмів реплікації для підвищення доступності даних;
4) можливість відновлення даних у разі збоїв;
5) використання індексів для прискорення пошуку інформації;

6) відповідність принципам ACID для забезпечення цілісності транзакцій;
7) масштабованість для підтримки зростаючого навантаження;
8) інтеграція з хмарними сервісами Azure.

2.3.1. MongoDB

MongoDB – це документно-орієнтована база даних NoSQL, яка зберігає дані у форматі BSON (бінарна форма JSON). Вона відома своєю здатністю ефективно працювати з неструктурованими та напівструктурованими даними. MongoDB підтримує такі можливості, як горизонтальне масштабування, динамічне проєктування схеми та аналітику в режимі реального часу [18].

Переваги MongoDB:

1. Динамічне проєктування схеми – гнучка структура дозволяє додавати нові поля до документів без необхідності змінювати існуючу схему.
2. Масштабованість – дозволяє горизонтально масштабувати базу даних через шардинг, розподіляючи дані між декількома серверами.
3. Висока продуктивність – оптимізовані операції запису та зчитування завдяки індексування та використання відображеного в пам’яті сховища для швидкого доступу до даних.
4. Реплікація та розподіл навантаження – завдяки розподіленню та копіюванню даних між багатьма серверами, MongoDB забезпечує швидкий та надійний доступ до даних.
5. Підтримка складних структур даних: MongoDB дозволяє зберігати вкладені та ієрархічні дані без використання join’ів.
6. Обробка даних в реальному часі – агрегаційний фреймворк дозволяє виконувати обробку великих обсягів даних, що є дуже зручно для виконання аналітики та звітування.

7. Кросплатформеність – MongoDB є сумісним з багатьма основними мовами програмування, такі як Python, JavaScript, Java та C#.
8. Простота підтримки – відсутність строгої структури дозволяє легко підтримувати базу даних MongoDB в порівнянні з традиційними SQL базами даних.

Недоліки MongoDB:

1. Обмеженість ACID транзакцій – нестача підтримки повноцінних ACID транзакцій між всіма документами та колекціями.
2. Високе використання пам’яті – зберігання назв полів разом з кожним документом призводить до більшого споживання пам’яті в порівняння з реляційними базами даних.
3. Відсутність підтримки join’ів – в MongoDB немає вбудованої підтримки join-операцій, а їхні альтернативи є затратними.
4. Обмеження на розмір документа: максимальний розмір становить 16 MB, що створює труднощі при обробці великих обсягів даних.
5. Складність налаштування шардингу – конфігурація та управління шардингу можуть бути складними.
6. Високе споживання ресурсів – архітектура MongoDB може вимагати більше серверних ресурсів порівняно з реляційними базами даних.

2.3.2. Microsoft SQL Server

MS SQL Server – це реляційна система управління базами даних (RDBMS), розроблена Microsoft для ефективного зберігання та обробки структурованих даних. Вона використовує мову T-SQL (Transact-SQL) для виконання транзакцій та складних запитів. Спочатку SQL Server працював лише у середовищі Windows, але з 2016 року став доступним і для Linux. Основні компоненти системи включають двигун бази даних (відповідає за зберігання, індексацію та виконання запитів), реляційний двигун (керує

обробкою запитів і пам’яттю) та двигун зберігання (забезпечує фізичне розміщення даних на носіях).

MS SQL Server широко використовується у бізнесі та корпоративних середовищах. Він добре інтегрується з іншими продуктами Microsoft, підтримує складні транзакції та дозволяє працювати із великими обсягами даних [19].

Переваги MS SQL Server [20, 21]:

1. Наявність інструментів – Microsoft SQL Server пропонує потужні та корисні інструменти, що сприяють покращенню проєктування, розробки баз даних і забезпечують зручне обслуговування.
2. Висока продуктивність – високою продуктивністю при обробці великих обсягів даних та здатність виконувати швидкі та ефективні запити.
3. Міцна безпека – наявність кілька рівнів безпеки, включаючи шифрування даних, автентифікацію користувачів та моніторинг доступу.
4. Масштабованість – дозволяє легко масштабувати систему, включаючи збільшення місткості зберігання або поліпшення продуктивності системи.
5. Інтеграція з продуктами Microsoft – підтримка інтеграції з такими інструментами, як Azure, Power BI, Excel та .NET.

Недоліки MS SQL Server [20, 21]:

1. Вартість – ліцензії можуть бути досить дорогими, особливо для більш розширених версій. Це може стати перепоною для малих бізнесів або стартапів з обмеженим бюджетом.
2. Складність налаштування – налаштування Microsoft SQL Server може бути складним і вимагати глибоких технічних знань.
3. Апаратні вимоги – Microsoft SQL Server може бути вимогливим до апаратного забезпечення. Для його оптимальної роботи потрібні потужні сервери та добре організоване IT-середовище.

4. Обмежена сумісність – можуть виникати проблеми сумісності з програмним забезпеченням сторонніх виробників, що може обмежити можливості інтеграції та вимагати розробки власних рішень.

2.3.3. PostgreSQL

PostgreSQL – це одна з найстаріших та найпередовіших систем керування базами даних з відкритим кодом. Вона була розроблена в рамках проєкту POSTGRES в Університеті Каліфорнії в Берклі в 1986 році і на сьогодні має понад 35 років активного розвитку. PostgreSQL підтримується великою та активною спільнотою, яка регулярно надає підтримку та допомогу в розв'язанні проблем, що виникають під час роботи з цією системою.

PostgreSQL є потужною об'єктно-реляційною СУБД, яка розширює SQL та має безліч функцій для надійного зберігання та масштабування складних даних. Вона працює на основних операційних системах, таких як Windows, Linux та Mac, і дозволяє розробникам, менеджерам проєктів та адміністраторам систем будувати продукти, вебсайти та інструменти. PostgreSQL потребує мінімального обслуговування завдяки своїй стабільності, а також дозволяє визначати власні типи даних і індексів, а також розробляти власні плагіни для задоволення специфічних вимог.

Система підтримує важливі функції, як-от повну відповідність стандарту ACID та мультиверсійну конкуренцію, що дозволяє забезпечити надійні транзакції та підтримку при високих навантаженнях. PostgreSQL також підтримує такі стандартні програми, як MySQL, ANSI SQL, MongoDB, Oracle і багато інших. Вона є високоефективною, розширюваною та підтримує типи індексів GIN і GIST, а також бази даних NoSQL.

Завдяки своїй архітектурі, надійності та широкому набору функцій PostgreSQL здобула визнання серед розробників та організацій по всьому

світу. Вона активно підтримується спільнотою з відкритим кодом, що забезпечує постійний розвиток та інновації [22].

Переваги PostgreSQL [23]:

1. Транзакції – PostgreSQL підтримує транзакції на рівні DDL, що дозволяє здійснювати зміни в базі даних у рамках єдиного транзакційного процесу, зокрема при зміні схеми бази даних.
2. Розширюваність – PostgreSQL має високу розширюваність, дозволяючи додавати нові функції, типи даних, мови програмування через розширення.
3. Безпека – PostgreSQL має надійні механізми безпеки, зокрема управління доступом на основі привілеїв і конфігурації на рівні операційної системи для захисту даних. Також підтримується параметрична безпека і контроль доступу на рівні застосунків.
4. Сумісність ACID – PostgreSQL є повністю ACID-сумісним, що гарантує надійність і стабільність при роботі з транзакціями.
5. Масштабованість – PostgreSQL може масштабуватись як вертикально, так і горизонтально, підтримуючи реплікацію та кластеризацію для роботи з великими обсягами даних.
6. Сумісність з різними типами даних – PostgreSQL підтримує широкий спектр типів даних, зокрема текстові, числові, часові, а також складні типи, як JSON, JSONB, XML, які дозволяють зберігати і обробляти різноманітні дані.

Недоліки PostgreSQL [24]:

1. Структура бази даних – PostgreSQL є реляційною базою даних, що вимагає чіткої схеми для таблиць і суворого визначення полів. Це може бути обмеженням при роботі з неструктурованими даними або даними, що постійно змінюються, порівняно з NoSQL базами даних.
2. Відкрите джерело – як система з відкритим кодом, PostgreSQL не має одного комерційного постачальника, що може створити

труднощі в плані підтримки, гарантій, ліцензування та сумісності з іншими продуктами. Також відсутність централізованого контролю може призводити до нестабільності чи змін у майбутніх версіях.

3. Сумісність з комерційними рішеннями – в порівнянні з комерційними СУБД, такими як Oracle або DB2, PostgreSQL може мати меншу підтримку специфічних функцій або інтерфейсів, з якими користувачі можуть бути знайомі.
4. Продуктивність – при великих обсягах даних або складних запитах PostgreSQL може демонструвати знижену продуктивність, оскільки має обмеження при обробці великих таблиць і запитів з великою кількістю полів.
5. Складність налаштування та адміністрування – PostgreSQL має багато налаштувань і параметрів, що може вимагати від адміністратора значних зусиль для оптимізації продуктивності та безпеки системи.
6. Обмеження на високу навантаженість – при обробці дуже великих обсягів одночасних запитів PostgreSQL може бути менш ефективним, порівняно з іншими базами даних, такими як Cassandra чи MongoDB, що спроєктовані спеціально для високих навантажень і масштабування.
7. Немає вбудованого користувацького інтерфейсу – PostgreSQL зазвичай не має стандартного інтерфейсу користувача, і його налаштування або адміністрування може вимагати використання сторонніх інструментів або командного рядка.

2.3.4. Результат проведеного аналізу

Були визначені основні вимоги до технологій для створення та використання баз даних, а також проведено аналіз таких технологій, як MongoDB, MS SQL Server та PostgreSQL. Для кожної із технологій було визначено основні переваги та недоліки.

Складемо порівняння (табл. 2.3), в якому кожна з баз даних буде оцінена відповідно до визначених вимог.

Таблиця 2.3

Порівняння технологій для реалізації бази даних

Технологія / Критерій порівняння	MongoDB	MS SQL Server	PostgreSQL
Ефективна обробка великої кількості даних і запитів	+	+	+
Захист збереженої інформації від несанкціонованого доступу та видалення	+/–	+	+
Підтримка механізмів реплікації для підвищення доступності даних	+	+	+
Можливість відновлення даних у разі збоїв	+/–	+	+
Використання індексів для прискорення пошуку інформації	+	+	+
Відповідність принципам ACID для забезпечення цілісності транзакцій	–	+	+
Масштабованість для підтримки зростаючого навантаження	+	+/–	+
Інтеграція з хмарними сервісами Azure	+	+	+/–

Результати порівняння технологій баз даних для розробки програмного забезпечення показують, що MongoDB, MS SQL Server та PostgreSQL мають різні переваги та недоліки в контексті вимог проєкту. MongoDB є гарним вибором для обробки великих обсягів даних і підтримки масштабованості, однак її відсутність повної відповідності принципам

ACID може бути важливим фактором для забезпечення цілісності транзакцій. Водночас, вона має хорошу інтеграцію з хмарними сервісами Azure, що робить її зручним варіантом для розгортання в хмарі.

MS SQL Server, завдяки повній підтримці принципів ACID, відновленню даних та високому рівню безпеки, є найбільш надійним вибором для проєктів, де важлива цілісність транзакцій і захист інформації. Однак її масштабованість може бути менш гнучкою в порівнянні з MongoDB, що може стати обмеженням при зростанні навантаження.

PostgreSQL поєднує в собі високий рівень підтримки ACID, захисту та масштабованості, що робить її хорошим вибором для проєктів, де важлива надійність і доступність даних. Проте інтеграція з Azure може вимагати додаткових налаштувань, що ускладнює процес розгортання.

Таким чином, з огляду на прості бізнес-вимоги і необхідність розгортання в хмарі Azure, MS SQL Server є оптимальним вибором для цього проєкту. Вона забезпечує високу надійність, підтримку принципів ACID і хорошу інтеграцію з Azure, що відповідає вимогам програмного забезпечення, без необхідності в додаткових налаштуваннях та інтеграціях.

2.4. Висновки

У процесі роботи над другим розділом пояснювальної записки були виконані наступні завдання:

1. Визначено перелік вимог та критеріїв для вибору технологій реалізації програмного продукту, зокрема для серверної частини, клієнтської частини та системи керування базою даних.
2. Обрано найпопулярніші та безкоштовні технології.
3. Проведено детальний аналіз кожної з технологій, розглянувши їхні переваги та недоліки.
4. Створено порівняльні таблиці, що оцінюють відповідність технологій сформованим вимогам.

5. На основі порівняння сформульовано висновки та обрано найбільш підходящі технології для реалізації серверної частини, клієнтської частини та системи керування базою даних.

В результаті було обрано C# ASP.NET для реалізації серверної частини, Angular для клієнтської частини та MS SQL Server для системи керування базою даних.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ СИСТЕМИ

У цьому розділу дипломного проєкту буде розглянуто:

1) аналіз проблематики та цілей вебзастосунку;

2) попереднє проєктування вебзастосунку;

3) загальна характеристика архітектури програмного забезпечення;

4) опис вимог програмного забезпечення;

5) аналіз структури моделі даних програмного забезпечення.

3.1. Аналіз проблематики та цілей вебзастосунку

Для того, щоб зрозуміти яку проблему вирішує програмне забезпечення «RefugeUA», було побудовано дерево проблем (рис. 3.1), яке визначає головну проблему, причини та наслідки. Головну проблему на діаграмі дерева проблем (рис. 3.1) позначено зеленим кольором, причини – червоним, наслідки – блакитним.

Головною проблемою є важка інтеграція військових в запасі та їхніх сімей в нову громади. Основними причинами проблеми є:

1. Відсутність централізованого ресурсу для військових, на якому місцеві жителі могли б розміщувати оголошення про доступне житло та комунікувати з військовими.
2. Обмежений доступ до вакансій через відсутність онлайн-платформи, спеціально орієнтованої на військових, а також через стереотипи роботодавців щодо військових у запасі.
3. Недостатня інформація про можливості для освіти, зокрема про безкоштовні курси перекваліфікації.
4. Низький рівень соціальної інтеграції, що зумовлений відсутністю спільних заходів для інтеграції та недостатньою взаємодією з волонтерами в громаді.

5. Нестача організованих волонтерських ініціатив через відсутність системи координації волонтерів та онлайн-платформи для реєстрації на заходи.

Усі ці фактори та проблеми призводять до таких наслідків:

1. Обмежене спілкування з місцевими жителями та військовими в громаді.
2. Відсутність зв’язку між військовими в запасі та місцевою громадою.
3. Погіршення фінансового становища та примусова еміграція військових і їхніх родин.
4. Низька конкурентоспроможність на ринку праці та тривале безробіття серед військових та їхніх сімей.
5. Проживання в тимчасових умовах (гуртожитки, прихистки), що негативно впливає на якість життя.
6. Зростання конфліктів у громаді та низька участь у громадському житті.

Отже, метою програмного забезпечення є сприяння соціальній адаптації військовозобов’язаних в запасі, ветеранів та їхніх сімей шляхом надання доступу до оголошень про доступне житло, роботу, освітні заклади, можливості перекваліфікації та психологічну допомогу, а також створення платформи для організації волонтерських заходів і інтеграції військових у громаду.

Вирішення цих проблем досягається шляхом створення багатофункціонального вебзастосунку, який забезпечує централізований доступ до релевантної інформації та сервісів. Через інтерфейс застосунку користувачі зможуть переглядати структуровані оголошення за категоріями (житло, робота, навчання), залишати відгуки, звертатися за психологічною допомогою, читати корисні матеріали та контактувати з фахівцями. Також реалізовано модулі для реєстрації на волонтерські події, взаємодії з організаторами та пошуку можливостей залучення до громади.

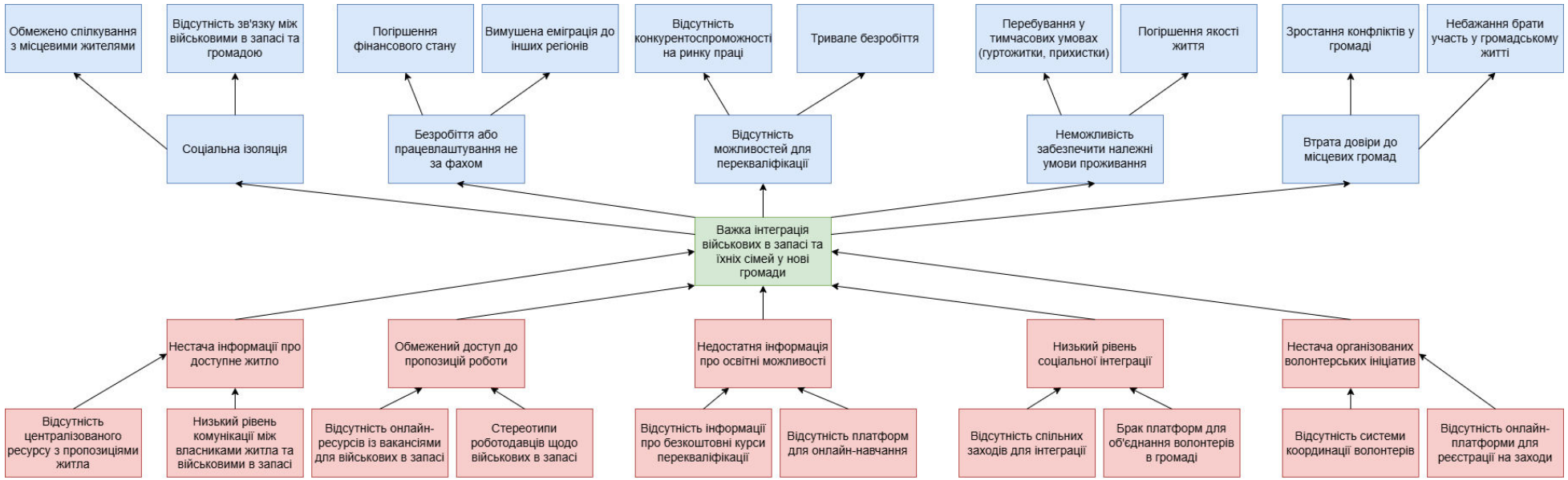


Рис. 3.1. Діаграма дерева проблем, які вирішує програмне забезпечення «RefugeUA»

3.2. Попереднє проєктування вебзастосунку

Основне завдання вебзастосунку – забезпечити користувачам доступ до оголошень про роботу, місця для навчання та перекваліфікацій, місця проживання та волонтерські заходи. Користувачам з особливими правами надається можливість створювати оголошення про волонтерські заходи, житло, роботу та навчання. Також передбачена можливість для представників громади редагувати та групувати оголошення, що стосуються жителів їхньої громади.

Для кожної із ролей існує відповідна панель навігації, за якою користувач може переглядати відповідні сторінки та виконувати певні операції.

- Панель адміністратора та представника громади:
  - Перегляд та редагування особистого профілю.
  - Перегляд користувачів та управління їх реєстрацією.
  - Перегляд власних груп.
  - Модерація оголошень.
  - Перегляд власних оголошень.
  - Створення та керування волонтерськими подіями.
  - Перегляд подій, у яких користувач бере участь.
  - Управління оголошеннями.
- Панель волонтера:
  - Головна сторінка волонтера.
  - Перегляд доступних волонтерських заходів.
  - Створення оголошень про волонтерські заходи.
  - Подання заявок на участь у заходах.
  - Перегляд своїх заявок та статусів.
- Панель місцевого жителя:
  - Перегляд та редагування особистого профілю.
  - Перегляд і керування власними групами.
  - Перегляд власних створених оголошень.

- Створення та керування власними волонтерськими подіями.
- Перегляд волонтерських подій, у яких бере участь.
- Керування власними волонтерськими подіями.

- Панель військового або члена сім’ї військового:
  - Перегляд та редагування особистого профілю.
  - Перегляд і керування власними групами.
  - Перегляд відгуків, залишених на оголошення.
  - Перегляд волонтерських подій, у яких бере участь.

Для зображення реалізованого функціональних можливостей розроблюваного програмного забезпечення було побудовано діаграму прецедентів (рис. 3.1). Вона показує різні використання та різні типи користувачів програмного забезпечення. Діаграма прецедентів використання програмного забезпечення для вебзастосунку, який має на меті допомогти військовим в запасі та їхнім сім’ям, включає в собі шість акторів: неавторизований користувач, місцевий житель, адміністратор, представник громади, військовий або член сім’ї військового, волонтер.

Неавторизований користувач може зареєструватися або увійти в систему. Після входу авторизовані користувачі отримують доступ до перегляду оголошень (житло, робота, освіта), участі у волонтерських заходах, читання статей із психологічної підтримки та редагування власного профілю.

Військові та їхні родини додатково можуть переглядати контакти фахівців з психологічної допомоги та надсилати відповіді на оголошення.

Волонтери мають змогу створювати заходи, формувати розклад заходу та організовувати волонтерські групи.

Представники громади можуть редагувати оголошення, створені місцевими жителями, забезпечуючи їх актуальність. Адміністратор керує користувачами та оголошеннями, підтримуючи порядок і безпеку системи.

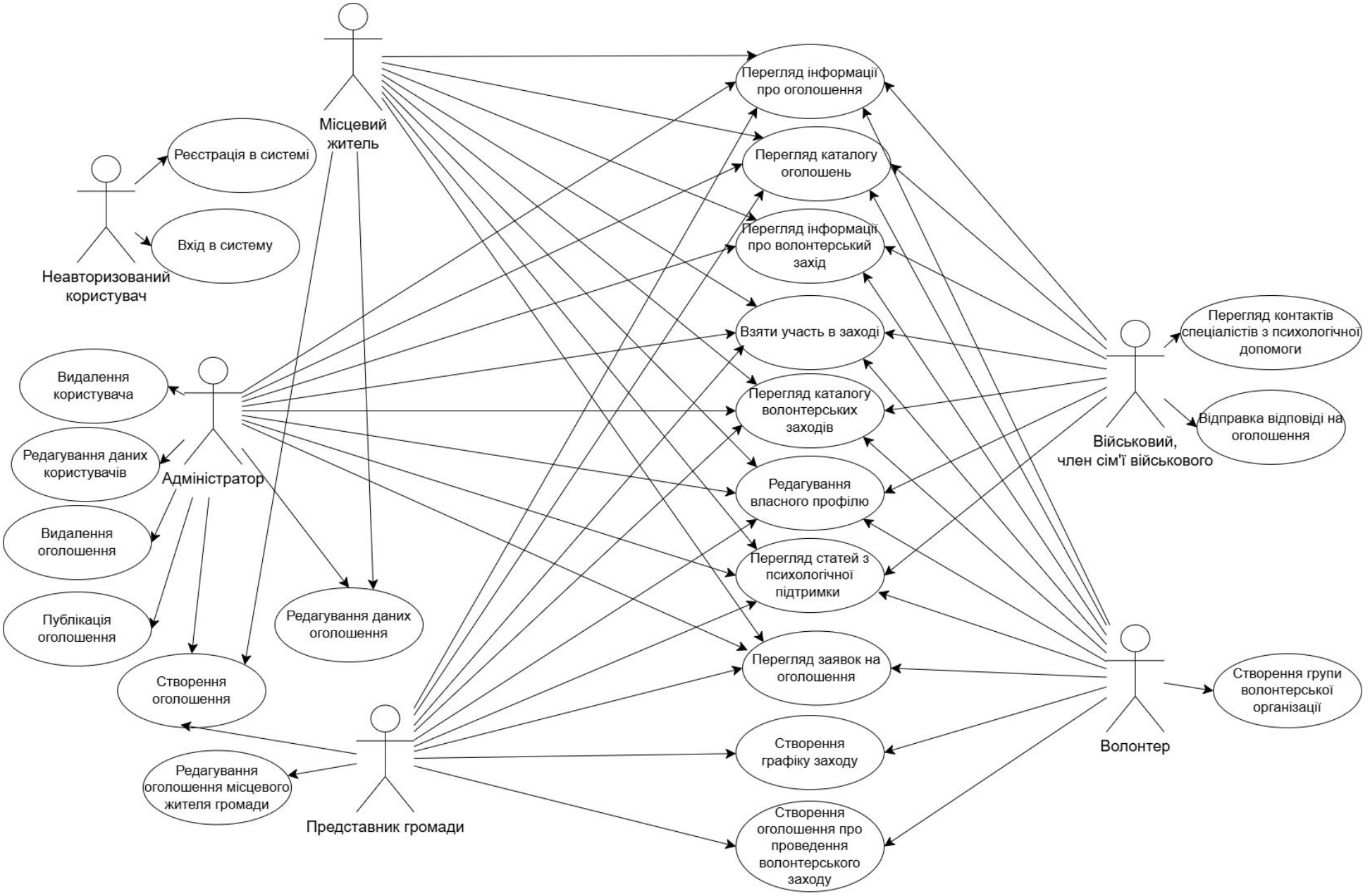


Рис. 3.2. Діаграма прецедентів вебзастосунку

Для зображення прикладу користування вебзастосунком користувачем, було побудовано алгоритм, який покроково демонструє типову взаємодію між користувачем та системою (рис. 3.3-3.4).

Користувач починає з перегляду головної сторінки вебсайту. На цьому етапі він ознайомлюється з доступними функціональними можливостями, коротким описом розділів та загальним призначенням ресурсу. Далі він обирає один із трьох основних напрямів: волонтерські заходи, оголошення або психологічна підтримка.

Якщо користувач переходить до розділу волонтерських заходів, система відображає перелік майбутніх подій, організованих місцевими громадами або волонтерськими групами. Користувач може переглядати список, фільтрувати заходи за датою, натискати на конкретний захід для перегляду його опису, організаторів. Якщо захід є актуальним і користувача зацікавила участь – він може пройти просту реєстрацію. Після цього система підтверджує участь і пропонує переглядати інші події.

Якщо обрано розділ оголошень, користувач має можливість обрати одну з підкатегорій: житло, робота або навчання і перекваліфікація. У кожній з цих категорій можна застосовувати фільтри для зручного пошуку. Після перегляду списку оголошень користувач може перейти до детальної сторінки конкретного оголошення, де розміщено всю необхідну інформацію: контактні дані, умови та опис. Користувач також має можливість залишити відгук. Після ознайомлення він може повернутися до списку, змінити категорію або завершити роботу.

У разі переходу до розділу психологічної підтримки користувач обирає один із підрозділів: статті з допомогою, або контакти спеціалістів – перелік фахівців-психологів, які можуть надати кваліфіковану підтримку. Користувач може ознайомитися з профілями спеціалістів, доступними каналами зв’язку та умовами консультацій. Після перегляду цієї інформації користувач може повернутися на головну сторінку, продовжити перегляд інших розділів або завершити роботу з вебзастосунком.

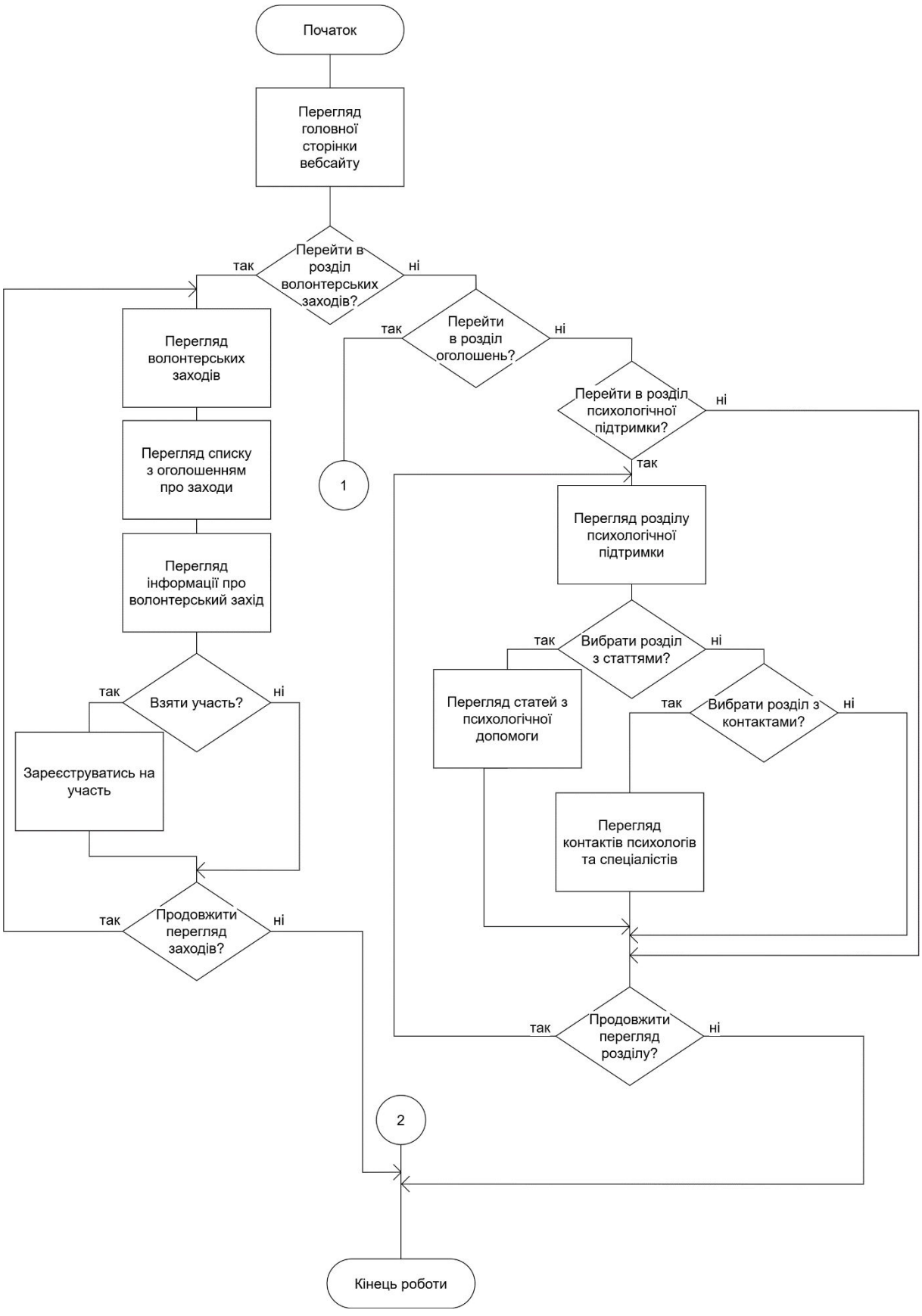


Рис. 3.3. UML діаграма користування вебзастосунком користувачем з роллю «військовий, член сім’ї військового», частина 1

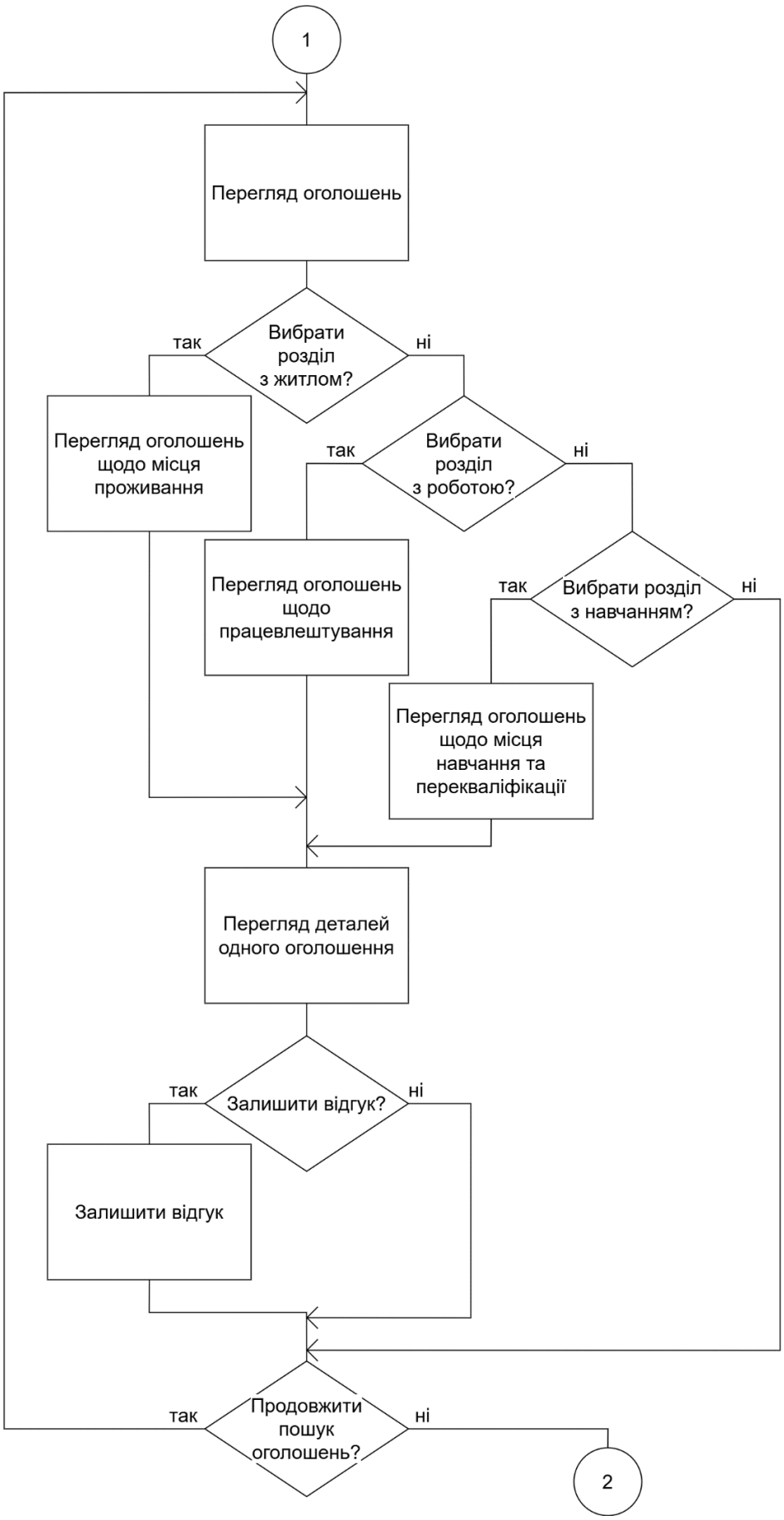


Рис. 3.4. UML діаграма користування вебзастосунком користувачем з роллю «військовий, член сім’ї військового», частина 2

3.3. Загальна характеристика архітектури програмного забезпечення

Архітектура програмного забезпечення визначає основні компоненти системи, їхню організацію та взаємодію, що впливає на продуктивність, гнучкість і підтримку застосунку. Добре спроєктована архітектура забезпечує модульність і масштабованість, дозволяючи легко замінювати або розширювати окремі компоненти без значних змін у кодовій базі. Це полегшує супровід та оновлення системи, зменшуючи витрати на її розвиток. Крім того, правильно обрана архітектура сприяє підвищенню безпеки, розподілу навантаження та забезпеченню високої доступності застосунку.

Вебзастосунок для підтримки та інтеграції військових у громаду побудований на основі клієнт-серверної архітектури. Основними компонентами є клієнтська та серверна частини, які взаємодіють через API, обмінюючись даними у захищеному форматі. Клієнтська частина відповідає за інтерфейс користувача та відображення інформації, тоді як серверна обробляє запити, керує базами даних, автентифікацією та бізнес-логікою застосунку.

Взаємодія між клієнтською та серверною частинами вебзастосунку здійснюється через протокол передачі даних HTTP, що забезпечує стандартизований обмін інформацією. Для опису та документування API використовується специфікація REST API, яка визначає чітку структуру endpoint’ів, параметрів запитів і форматів відповідей. Усі дані передаються у форматі JSON. REST API підтримує основні методи HTTP, такі як GET (отримання даних), POST (створення нового ресурсу), PUT (оновлення ресурсу) і DELETE (видалення ресурсу). Запити до API захищені механізмами аутентифікації та авторизації, що обмежує доступ до певних функцій залежно від ролі користувача [24].

Загальна схема архітектури вебзастосунку (рис. 3.5).

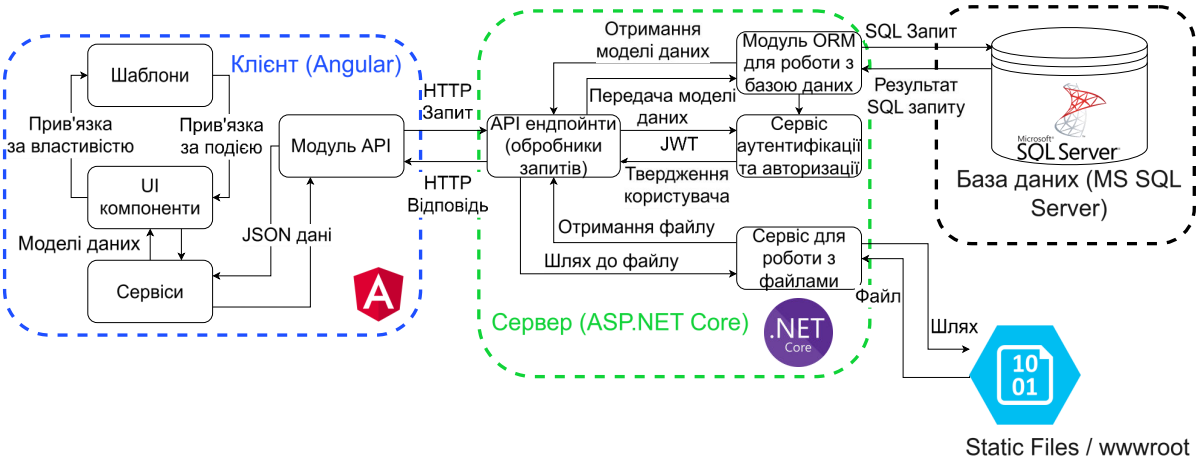


Рис. 3.5. Діаграма архітектури застосунку

Клієнт (Angular) складається з шаблонів, які визначають HTML-структуру компонентів. Взаємодія між шаблонами та логікою відбувається через прив’язку до властивостей і подій. UI-компоненти відповідають за відображення інформації та взаємодію з користувачем [25]. Сервіси забезпечують бізнес-логіку на клієнтській стороні та комунікацію з сервером через HTTP-запити.

Сервер (ASP.NET Core) містить API-ендпоінти, які обробляють HTTP-запити від клієнта. Для організації ендпоінтів використовується підхід архітектури Vertical Slices Architecture.

Vertical Slices Architecture – це підхід до організації коду, який фокусується на незалежних функціональних сценаріях, а не на традиційних горизонтальних шарах (UI, бізнес-логіка, дані). Замість того щоб структурувати застосунок за рівнями (наприклад, шар контролерів, шар сервісів, шар репозиторіїв), код організовується навколо окремих функцій, таких як "Створити оголошення", "Отримати список користувачів" чи "Прийняти оголошення". У межах кожного такого сценарію знаходяться всі необхідні компоненти: контролер, бізнес-логіка, доступ до бази даних та інші залежності [26].

ORM-модуль (Entity Framework Core) забезпечує взаємодію з базою даних, сервіс аутентифікації та авторизації керує доступом користувачів, а сервіс для роботи з файлами виконує завантаження файлів за шляхом.

База даних (MS SQL Server) отримує SQL-запити від ORM-модуля та повертає результати. Це забезпечує збереження даних та їхню обробку відповідно до запитів від сервера.

Статичні файли зберігаються безпосередньо на сервері у визначеній директорії wwwroot. Сервер формує шлях до збереженого файлу та передає його клієнту для подальшого доступу.

3.4. Опис вимог програмного забезпечення

Було розглянуто основну структуру вебзастосунку, а саме зміст вебсторінок та панелі навігації користувачів. Кожен користувач, залежно від своєї ролі, має відповідний рівень доступу до функціональних можливостей вебзастосунку. Тепер визначимо функціональні вимоги програмного забезпечення, що включають вимоги для кожної із ролей користувачів (табл. 3.1-3.7).

Таблиця 3.1

Функціональні вимоги для ролі адміністратора

Код вимоги	Зміст вимоги	Атрибути	
		Пріоритет	Складність
F-1	Можливість реєстрації в системі з вибором ролі користувача	Високий	Середня
F-2	Можливість авторизації користувачів за допомогою JWT-токенів	Високий	Середня

Продовження табл. 3.1

F-3	Можливість перегляду, редагування, видалення та отримання детальної інформації про оголошення	Високий	Середня
F-4	Можливість створення оголошення для волонтерів, адміністраторів та представників громади	Високий	Середня
F-5	Можливість адміністрування користувачів, оголошень, волонтерських заходів та відгуків	Високий	Висока
F-6	Можливість перегляду, редагування, видалення та отримання детальної інформації про статті з психологічної підтримки та контактів спеціалістів-психологів	Високий	Середня

Таблиця 3.2

Функціональні вимоги для ролі волонтера

Код вимоги	Зміст вимоги	Атрибути	
		Пріоритет	Складність
F-7	Можливість створення оголошень про волонтерські заходи	Високий	Середня
F-8	Можливість формування та перегляду розкладу волонтерського заходу	Середній	Середня

Продовження табл. 3.2

F-9	Можливість перегляду календаря із запланованими волонтерськими заходами	Середній	Низька
F-10	Можливість реєстрації на волонтерський захід через оголошення	Високий	Середня
F-11	Можливість створення груп волонтерських організацій	Середній	Середня
F-12	Можливість додавання та видалення користувачів у групах	Середній	Середня

Таблиця 3.3

Функціональні вимоги для ролі представника громади

Код вимоги	Зміст вимоги	Атрибути	
		Пріоритет	Складність
F-13	Можливість перегляду, редагування, видалення оголошень місцевих жителів	Високий	Висока
F-14	Можливість об’єднання оголошень в окремі групи	Середній	Середній

В таблиці 3.4 наведено детальний опис основних функціональних можливостей, які стають доступними для всіх користувачів після успішного входу в систему. Ці вимоги забезпечують базовий рівень взаємодії з вебзастосунком, дозволяючи користувачам виконувати ключові операції та ефективно користуватися основними функціональними можливостями.

Таблиця 3.4

Функціональні вимоги для ролі базового користувача

Код вимоги	Зміст вимоги	Атрибути	
		Пріоритет	Складність
F-15	Можливість пошуку оголошення за назвою, громадою	Високий	Низька
F-16	Можливість перегляду детальної інформації про оголошення стосовно житла, навчання та роботи	Високий	Низька
F-17	Можливість пошуку волонтерського заходу	Високий	Низька
F-18	Можливість перегляду інформації про волонтерський захід, включаючи розклад, кількість зареєстрованих учасників та опис події	Високий	Середня
F-19	Можливість залишення відгуку на оголошення	Середній	Середня
F-20	Можливість реєстрації на участь у волонтерському заході	Високий	Середня
F-21	Можливість редагування профілю, перегляду заявок у особистому кабінеті	Високий	Висока
F-22	Можливість перегляду контактів психологів спеціалістів та статей з психологічної підтримки	Середній	Низька

Продовження табл. 3.4

F-23	Панель базових користувачів: можливість перегляду оголошень, реєстрації на заходи, перегляду календаря	Високий	Середня

Таблиця 3.5

Додаткові функціональні вимоги для ролі базового користувача, місцевого жителя

Код вимоги	Зміст вимоги	Атрибути	
		Пріоритет	Складність
F-24	Можливість створення оголошення стосовно житла, роботи та навчання	Високий	Низька

Наступним етапом є визначення нефункціональних вимог, які охоплюють аспекти безпеки та якості. Вони визначають, як саме повинно працювати програмне забезпечення, забезпечуючи його надійність, ефективність і зручність для користувачів, а також захищеність даних. Нижче наведенні нефункціональні вимоги вебзастосунку.

Таблиця 3.6

Нефункціональні вимоги до безпеки

Код вимоги	Зміст вимоги
SEC-1	Всі користувачі повинні мати різні рівні доступу, визначені за допомогою ролей (наприклад, користувачі, волонтери, адміністратори).

Продовження табл. 3.6

SEC-2	Паролі користувачів повинні зберігатися в зашифрованому вигляді за допомогою сучасних алгоритмів хешування.
SEC-3	Вебзастосунок повинен підтримувати безпечну аутентифікацію користувачів, з використанням багатофакторної аутентифікації (MFA) для адміністративних акаунтів.
SEC-4	Програмне забезпечення повине бути захищена від атак типу SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), а також від атак типу Denial of Service (DoS).

Таблиця 3.7

Нефункціональні вимоги до якості

Код вимоги	Зміст вимоги
Q-1	Вебзастосунок має забезпечувати швидкий доступ до оголошень і подій, з оптимізованим завантаженням сторінок (не більше 3 секунд на завантаження головної сторінки).
Q-2	Програмне забезпечення має бути здатна розширюватися для обробки збільшення кількості користувачів і даних без значного погіршення продуктивності.
Q-3	Вебзастосунок повенен бути кросплафторменим та працювати на операційних системах Windows, Linux та macOS
Q-4	Клієнтська частина програмного забезпечення повинна працювати в сучасних веббраузерах, таких як Google Chrome, Mozilla Firefox, Microsoft Edge та Safari.

3.5. Аналіз структури моделі даних програмного забезпечення

Фізичне проєктування – це ключовий етап розробки інформаційної системи, який дозволяє виділити всі бізнес-сутності, необхідні для реалізації логіки застосунку. На цьому етапі проєктується структура бази даних, що включає в себе конкретні таблиці, зв’язки між ними, типи даних

для кожного поля, індекси, обмеження цілісності (constraints), а також можливі тригери та збережені процедури. Основна мета фізичного проєктування – забезпечити ефективне зберігання, доступ і обробку даних з урахуванням особливостей обраної системи керування базами даних (СКБД).

Під час цього етапу здійснюється створення таблиць, які відповідають виділеним бізнес-сутностям. Для кожної сутності визначаються поля, які зберігатимуть атрибути, первинні ключі (Primary Keys) для унікальної ідентифікації записів, зовнішні ключі (Foreign Keys) для забезпечення зв'язків між таблицями, а також застосовуються додаткові обмеження типу NOT NULL, UNIQUE, CHECK, що гарантують цілісність і якість даних.

Важливою частиною фізичного проєктування є оптимізація структури: вона досягається шляхом створення індексів, які пришвидшують пошук, або денормалізації, якщо це виправдано з точки зору продуктивності. Крім того, враховується унікальність записів, щоб уникнути дублювання, та забезпечується збереження інформації в атомарному вигляді – тобто поділ складних даних на найменші логічні одиниці, що сприяє узгодженості та простоті обробки.

Фізичне проєктування також передбачає впровадження обмежень транзакційності та цілісності відповідно до ACID-властивостей, що гарантує правильне виконання операцій оновлення, вставки та видалення. На завершальному етапі базу наповнюють тестовими даними та перевіряють її працездатність – виконуючи контрольні запити, тестуючи цілісність даних і швидкість доступу [27].

Наведемо опис сутностей у базі даних (рис. 3.6), які реалізують ключові функції системи. Кожна сутність відображає окрему функціональну частину системи та пов'язана з іншими через відповідні відношення – один до одного, один до багатьох або багато до багатьох.

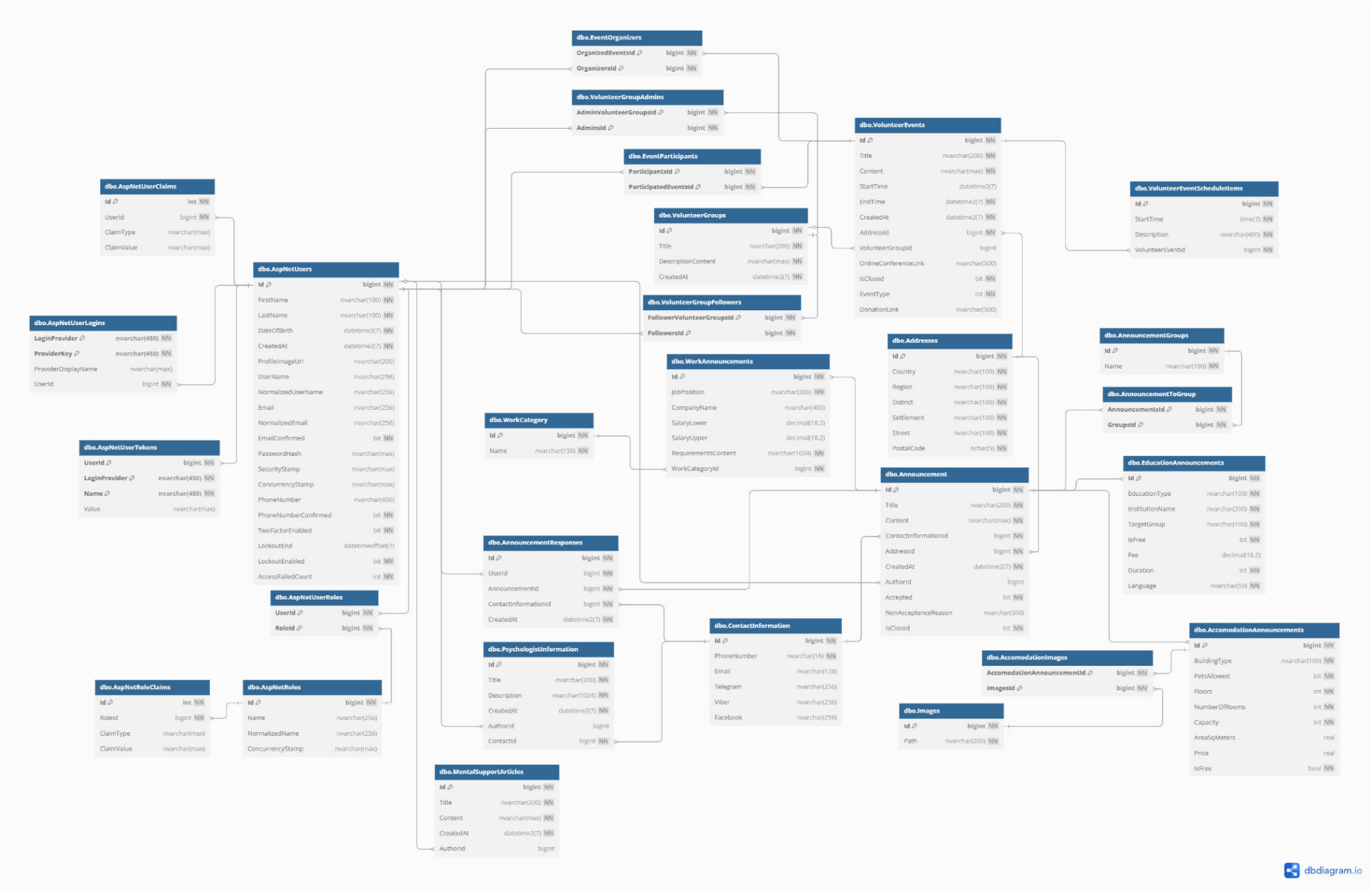


Рис. 3.6. Діаграма взаємозв’язків між сутностями

Тепер наведемо таблиці даних сутностей, що зберігаються в базі даних вебзастосунку.

1. Оголошення про житло (AccomodationAnnouncements).
2. Зображення житла (AccomodationImages).
3. Адреси (Addresses).
4. Універсальні оголошення (Announcement).
5. Групи оголошень (AnnouncementGroups).
6. Відгуки на оголошення (AnnouncementResponses).
7. Зв’язок між оголошеннями та групами (AnnouncementToGroup).
8. Контактна інформація (ContactInformation).
9. Освітні оголошення (EducationAnnouncements).
10. Організатори подій (EventOrganizers).
11. Учасники подій (EventParticipants).
12. Зображення (Images).
13. Статті з психологічної підтримки (MentalSupportArticles).
14. Інформація про психологів (PsychologistInformation).
15. Волонтерські події (VolunteerEvents).
16. Розклад волонтерських подій (VolunteerEventScheduleItems).
17. Волонтерські групи (VolunteerGroups).
18. Адміністратори волонтерських груп (VolunteerGroupAdmins).
19. Підписники волонтерських груп (VolunteerGroupFollowers).
20. Оголошення про роботу (WorkAnnouncements).
21. Категорія роботи (WorkCategory).
22. Користувачі (AspNetUsers).
23. Таблиці автентифікації та авторизації.
    - 23.1. AspNetRoles – зберігає ролі (наприклад, "Admin", "User", "Moderator").
    - 23.2. AspNetUserRoles – таблиця зв’язку між користувачами і ролями.

23.3. AspNetUserClaims – клейми (атрибути) користувачів.

23.4. AspNetRoleClaims – клейми, пов’язані з ролями.

23.5. AspNetUserLogins – зовнішні способи входу (Google, Facebook тощо).

23.6. AspNetUserTokens – зберігає токени для відновлення доступу, MFA тощо.

4. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ВЕБЗАСТОСУНКУ

Метою даного розділу є опис реалізації проєкту, а саме її серверної та клієнтської частини, користувацького інтерфейсу, опису проведеного тестування та подальші кроки для покращення вебзастосунку для допомоги військовим та волонтерським організаціям.

4.1. Особливості реалізації

Вебзастосунок для військових в запасі та волонтерських організацій розроблено з урахуванням зручності подальшого супроводу та розширення. Його структура є модульною, що спрощує внесення змін і додавання нових функцій. Завдяки зрозумілому та впорядкованому коду розробники можуть швидко орієнтуватися в системі, ефективно виявляти й усувати помилки та забезпечувати стабільну роботу застосунку в довгостроковій перспективі. Крім того, реалізовано чіткий розподіл між клієнтською та серверною частинами, що забезпечує кращу масштабованість, полегшує тестування окремих компонентів та підвищує загальну продуктивність і безпеку системи.

4.1.1. Серверна частина вебзастосунку

Вебзастосунок для розміщення оголошень для військових в запасі та волонтерських організацій було реалізовано з чітким поділом на клієнтську та серверну частини. Серверна частина розроблена з використанням ASP.NET і ORM-бібліотеки Entity Framework для роботи з базою даних.

Архітектура побудована за підходом Vertical Slices, що забезпечує логічну ізоляцію функціональних сценаріїв та спрощує масштабування, тестування й підтримку окремих частин системи. Згідно з цим принципом, кожен endpoint реалізує окрему бізнес-операцію та використовує лише ті залежності, які безпосередньо потрібні для її виконання.

Побудована серверна частина містить наступні модулі (рис. 4.1).

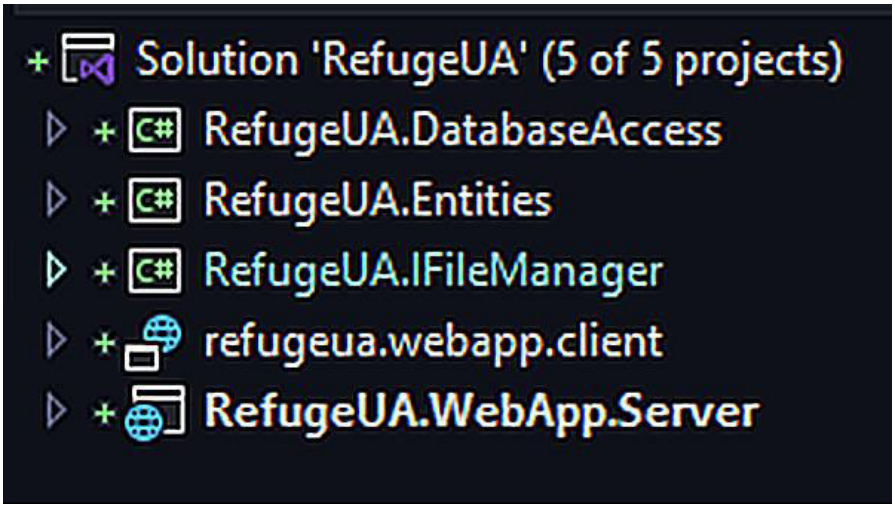


Рис. 4.1. Загальна архітектура серверної частини

Розділи серверної частини проєкту:

- RefugeUA.DatabaseAccess – модуль для роботи з базою даних.
- RefugeUA.Entities – модуль, який містить всі бізнес-сутності застосунку.
- RefugeUA.IFileManager – модуль, який містить інтерфейс для роботи з файлами.
- RefugeUA.WebApp.Server – проєкт вебзастосунку у вигляді web api, який містить всю серверну бізнес-логіку та зберігає файли.

Модуль вебсервера (рис. 4.2) побудовано відповідно до визначеної архітектури Vertical Slices, де кожна операція – це ізольований та окремий endpoint. Кожен endpoint містить всю необхідну функціональність для своєї роботи, а саме код бізнес-логіки, валідатори для перевірки даних, обробку запитів і відповідей, а також об’єкти введення (DTO). Для валідації застосовується бібліотека FluentValidation.

Крім endpoint’ів, сервер містить також усі інші необхідні для роботи ресурси, зокрема сервіси для авторизації, автентифікації, надсилання повідомлень на електронну пошту користувачів, обробки файлів та

зображень, а також конфігураційні файли для налаштування середовища, підключення до бази даних і зовнішніх сервісів.

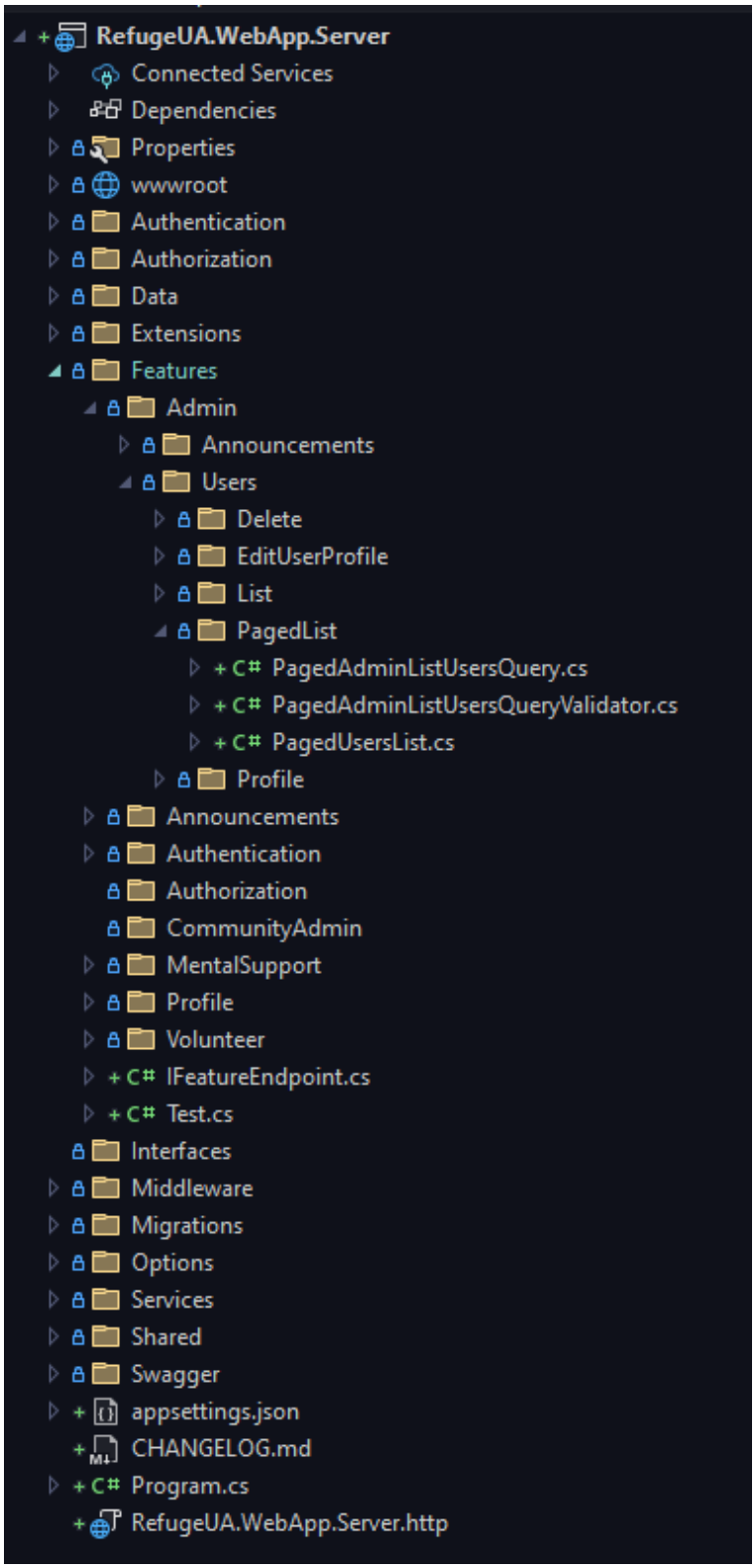


Рис. 4.2. Структура головного модулю вебзастосунку

4.1.2. Клієнтська частина вебзастосунку

Клієнтська частина вебзастосунку було реалізована за допомогою фреймворку Angular. Інтерфейс побудовано на основі компонентів, що відповідають за відображення інформації, та шаблонів, які визначають структуру HTML з використанням прив'язки до властивостей і подій.

Для обміну інформацією з серверною частиною впроваджено сервіси (рис. 4.4), які виконують HTTP-запити до API та обробляють отримані відповіді. Усі запити до серверу вже винесені в окремі сервіси, що дозволяє легко масштабувати систему та спрощує супровід коду. Реалізовано валідацію форм на клієнтському рівні для зниження навантаження на сервер та дозволяючи перевіряти дані без перезавантаження вебсторінки.

Загальний вигляд структури клієнтської частини (рис. 4.3).

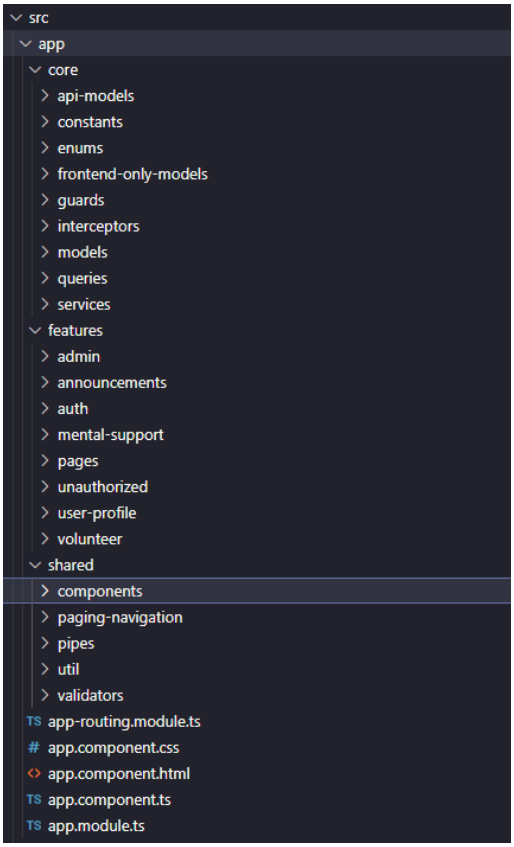


Рис. 4.3. Загальна структура клієнтської частини

Загальна структура клієнтського проєкту була поділена на наступні розділи:

- «core», що містить в собі сутності, моделі запитів та сервіси.
- «features», що містить основні сторінки та UI-компоненти.
- «shared», що містить в собі всі спільні функціональні можливості та компоненти користувацького інтерфейсу, що використовується всіма сторінками.

Кожен UI-компонент в клієнтській частині містить в собі три файли розширень .css, .ts та .html (рис. 4.4), який відповідає за зовнішній вигляд, поведінку та структуру компоненти відповідно.

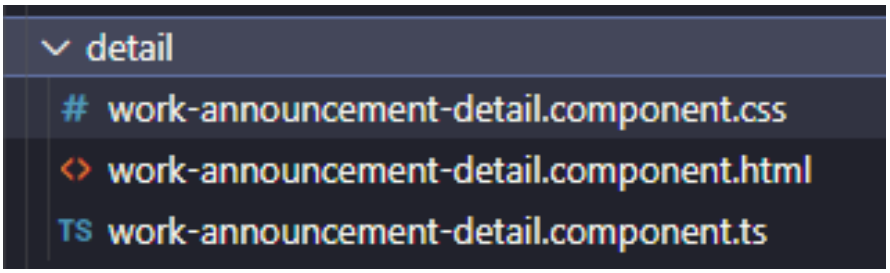


Рис. 4.4. Приклад UI-компоненти

Для взаємодіїї з сервером використовується вбудований сервіс «HttpClient», який дозволяє надсилати та отримувати дані через протокол HTTP. Кожен із таких запитів було інкапсульовано в окремі сервіси, які згодом використовуються у відповідних компонентах.

Клієнтську частину було налаштовано на комунікацію з серверною частиною за допомогою проксі. Для цього використовується конфігураційний файл, у якому запити до шляху /api перенаправляються на сервер ASP.NET Core.

4.2. Опис реалізованого користувацького інтерфейсу

Користувацький інтерфейс вебзастосунку для розміщення оголошень для військових в запасі та волонтерських організацій реалізовано у вигляді

односторінкового вебзастосунку. За допомогою інструментів фреймворку Angular, повторюванні елементи дизайну було згруповано в окремі компоненти.

Вебзастосунок містить наступні найголовніші сторінки:

1) сторінка реєстрації та входу;
2) головна сторінка;
3) сторінка з оголошеннями;
4) сторінка оголошення;
5) сторінка створення оголошення;
6) сторінка відгуків на оголошення;
7) сторінка волонтерських подій;
8) сторінка волонтерської події;
9) сторінка створення волонтерської події;
10) сторінка волонтерських груп;
11) сторінка волонтерської групи;
12) сторінка створення волонтерської групи;
13) сторінка статей з психологічної підтримки;
14) сторінка статті з психологічної підтримки;
15) сторінка створення статті з психологічної підтримки;
16) сторінка контактів психологів;
17) сторінка контакту психолога;
18) сторінка створення контакту психолога;
19) сторінка профілю користувача;
20) сторінка з своїми оголошеннями;
21) сторінка з своїми відгуками;
22) сторінка модерації оголошень;
23) сторінка модерації користувачів та заявок на реєстрацію;
24) сторінка груп оголошення.

Для прикладу дизайну компонентів користувацького інтерфейсу наведемо декілька зображень.

Сторінка вебзастосунку складається із двох основних частин: панелі навігації та головної панелі відображення сторінки. Панель навігації містить посилання на розділи вебзастосунку: оголошення, волонтерські події, психологічна підтримка. Головна панель відповідає за відображення відповідного розділу залежно від того, на який саме перейшов користувач.

Оскільки вебзастосунок є односторінковим (SPA – Single Page Application), перехід між розділами відбувається без повного перезавантаження сторінки.

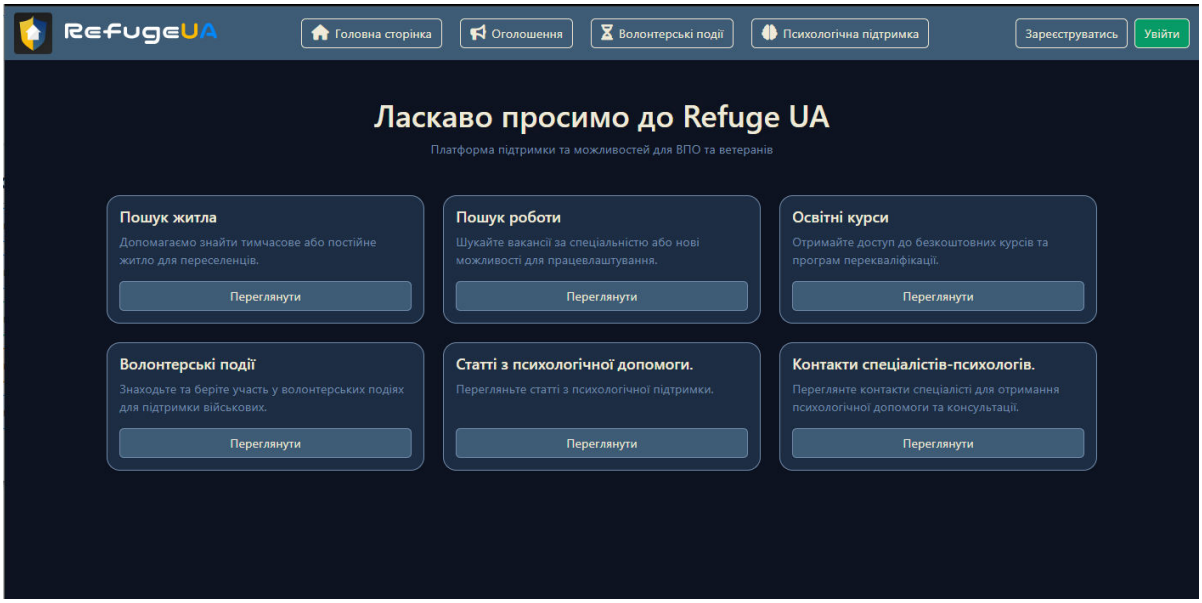


Рис. 4.5. Головна сторінка вебзастосунку

Для створення об’єктів використовуються форми, які мають спільний дизайн (рис. 4.6). У формах передбачено як обов’язкові, так і необов’язкові поля для введення. Усі форми є динамічними – їхній вміст адаптується залежно від обраного типу об’єкта або інших введених даних. Було реалізовано клієнтську валідацію, яка перевіряє введені у форму дані на відповідність визначеним форматам для кожного поля. Після правильного заповнення всіх обов’язкових полів і успішного проходження перевірки, кнопка надсилання форми стає активною, що дає змогу відправити дані на обробку серверною частиною вебзастосунку.

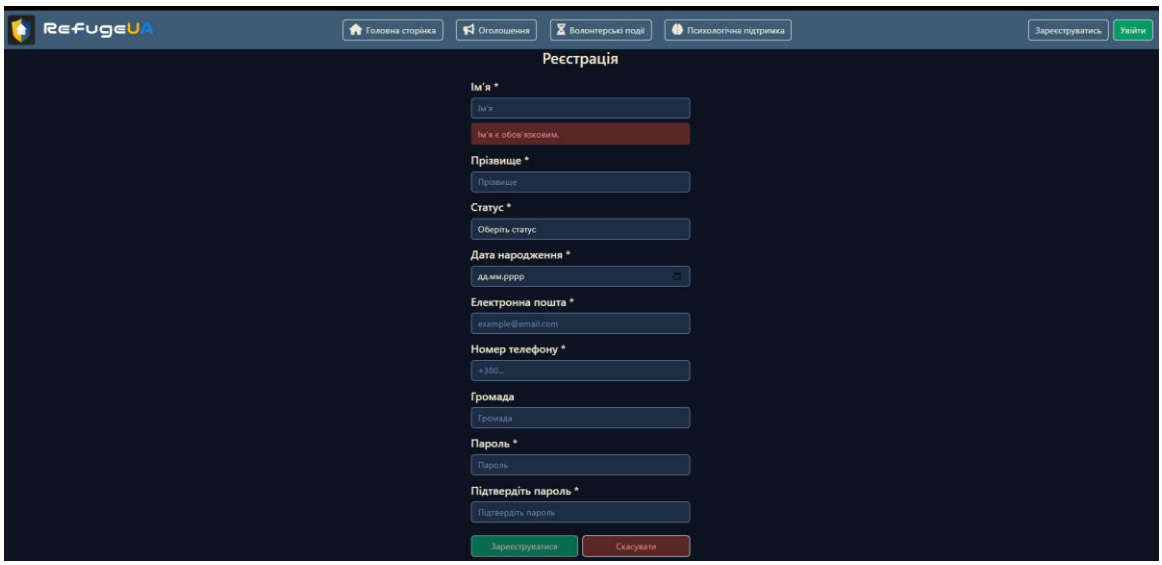


Рис. 4.6. Приклад форми

Для пошуку у відповідних розділах вебзастосунку передбачено пошуковий рядок та панель фільтрів, які дозволяють знаходити необхідну інформацію за ключовими словами або за заданими критеріями. Для прикладу наведемо зображення розділу з пошуком роботи (рис. 4.7).

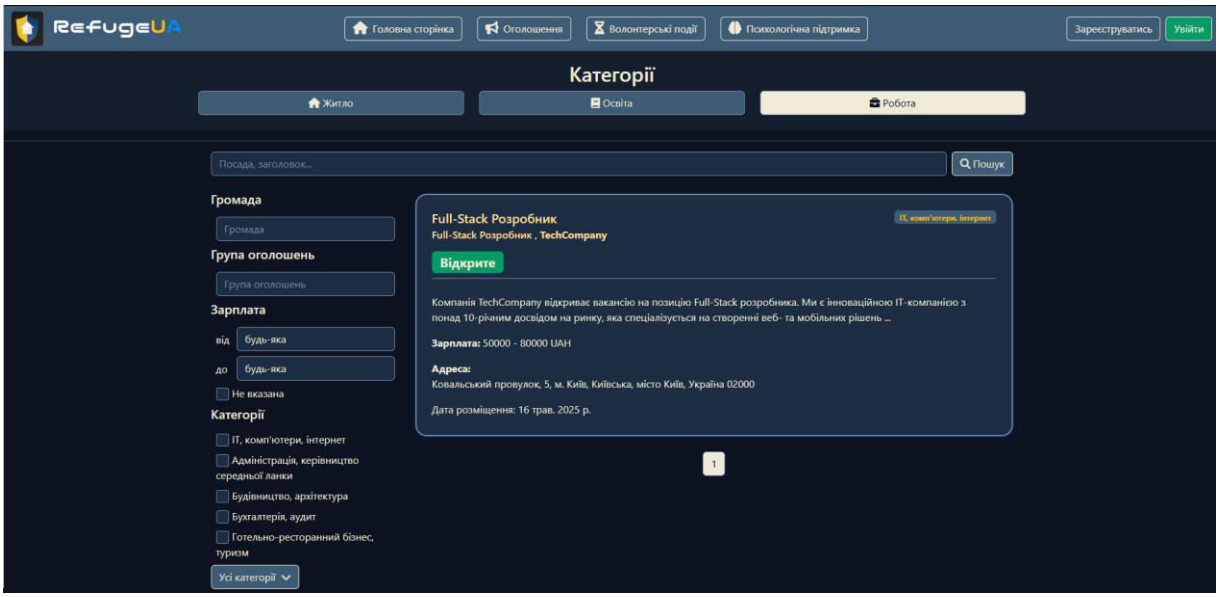


Рис. 4.7. Пошук оголошень стосовно роботи

Результатом пошуку є список з короткою інформацією про кожен об’єкт. Натиснувши на елемент, користувач переходить до детального

перегляду, де залежно від ролі та авторства може редагувати, видаляти або переглядати додаткові дані (рис. 4.8).

Рис. 4.8. Перегляд деталей оголошення про роботу адміністратором

4.4. Особливості проведеного тестування

Тестування системи проводиться згідно з тестовим планом, який відображає бізнес-вимоги застосунку та включає набір тестових випадків. Для перевірки вебзастосунку застосовується метод димового тестування, що передбачає перевірку усіх основних сценаріїв його використання.

Основними сценаріями використання є:

1. Можливість реєстрації у системі.
2. Можливість входу в систему.
3. Перегляд та фільтрація оголошень щодо житла, роботи, навчання.
4. Перегляд та фільтрація волонтерських подій.
5. Перегляд та фільтрація волонтерських груп.
6. Перегляд та фільтрація статей з психологічної підтримки.
7. Перегляд та фільтрація контактів спеціалістів.
8. Перегляд детальної інформації про оголошення.
9. Відгук на оголошення.
10. Перегляд власних відгуків.

11. Створення оголошення (житло, робота, навчання).
12. Редагування оголошення.
13. Видалення оголошення.
14. Перегляд відгуків на своє оголошення.
15. Створення волонтерської події.
16. Редагування волонтерської події.
17. Видалення волонтерської події.
18. Участь у волонтерській події.
19. Перегляд волонтерських подій, у яких бере участь користувач.
20. Перегляд заявок на реєстрацію адміністратором.
21. Дозвіл або скасування доступу до системи.
22. Перегляду сторінки модерації оголошень про роботу, навчання та житло.
23. Прийняти або відхилити оголошення про роботу, навчання та житло.
24. Створення статті з психологічної допомоги.
25. Редагування статті з психологічної допомоги.
26. Видалення статті з психологічної допомоги.
27. Створення контактів спеціаліста з психологічної допомоги.
28. Редагування контактів спеціаліста з психологічної допомоги.
29. Видалення контактів спеціаліста з психологічної допомоги.
30. Створення волонтерської групи.
31. Редагування опису волонтерської групи.
32. Видалення волонтерської групи.
33. Приєднання користувачем до волонтерської групи.

Для тестування серверної частини застосунку було використано вбудований інструмент Swagger (рис. 4.9), який автоматично генерує інтерактивну документацію для всіх доступних API- endpoint’ів. Після запуску застосунку через браузер відкрився інтерфейс Swagger, де було здійснено перевірку роботи всіх ключових маршрутів, включно зі

створенням, редагуванням, переглядом і видаленням даних. Для кожного endpoint'а були введені відповідні дані у форматі JSON, після чого відправлено HTTP-запити й проаналізовано відповіді від сервера. Таким чином, за допомогою Swagger було успішно протестовано функціональність API, що підтвердило готовність серверної частини до подальшої інтеграції з клієнтською частиною.

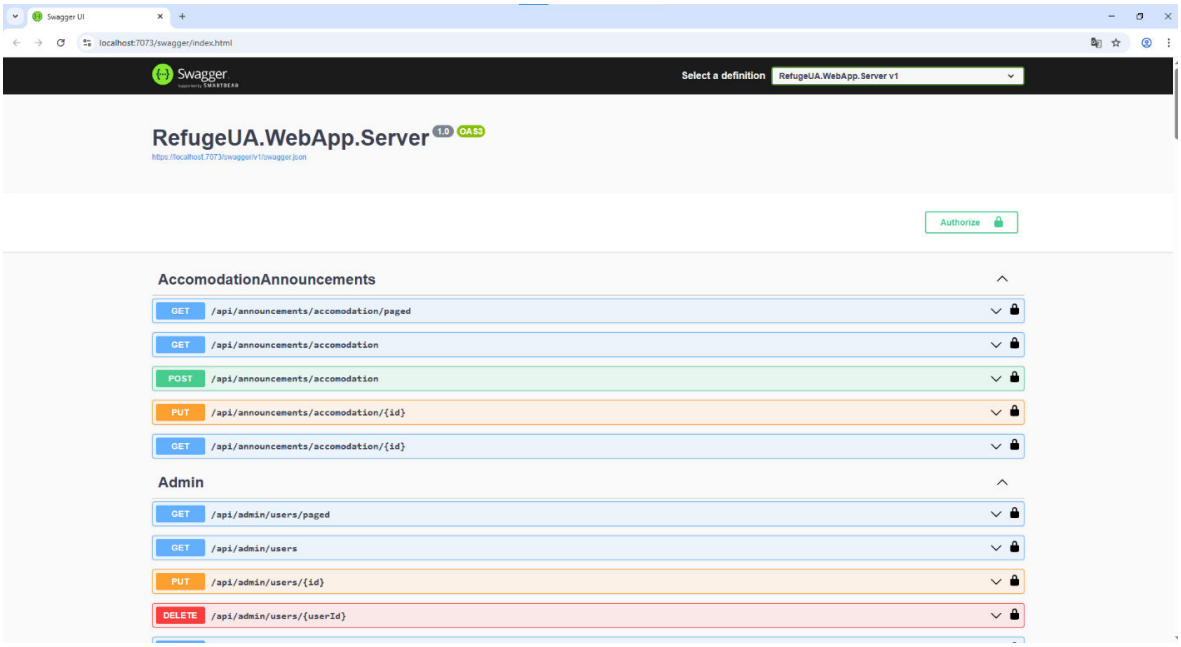


Рис. 4.9. Інтерфейс Swagger, початок списку endpoint'ів

4.4. Рекомендації щодо подальшого вдосконалення

На основі проведеного тестування вебзастосунку RefugeUA можна виокремити кілька напрямків для його подальшого вдосконалення.

Корисним покращенням буде створення календаря подій, в якому користувач може швидко всі події, які проходять в задані періоди часу.

Також можна розробити механізм нагадувань про майбутні події (наприклад, електронною поштою або через мобільні сповіщення), щоб підвищити відвідуваність заходів і покращити організацію.

Ще одним кроком до вдосконалення є розробка стратегії поширення вебзастосунку за межами України, зокрема серед української діаспори,

міжнародних волонтерських фондів та благодійних організацій. На такому етапі треба розробити мультимовний інтерфейс.

З метою підвищення довіри й активності користувачів, варто впровадити механізми зворотного зв’язку, рейтингу оголошень і волонтерських груп.

4.5. Висновки

У рамках даного розділу було розглянуто особливості реалізації вебзастосунку RefugeUA, призначеного для підтримки демобілізованих військових, ветеранів, їхніх родин та волонтерських організацій. Детально описано реалізацію клієнтської та серверної частин системи, а також наведено приклади інтерфейсів.

Крім того, представлено результати тестування основних сценаріїв використання, які підтвердили стабільність роботи основних функцій. Також сформульовано рекомендації щодо подальшого вдосконалення, зокрема інтеграцію з зовнішніми сервісами, покращення пошукової системи, реалізацію нагадувань та підтримку для мультимовного інтерфейсу.

ВИСНОВКИ

Метою цього дипломного проєкту є створення рішення, яке допоможе демобілізованим військовим, військовозобов'язаним у запасі, ветеранам та їхнім сім'ям у пошуку житла, роботи, можливостей для навчання та перекваліфікації, а також отриманні психологічної допомоги. Крім того, проєкт спрямований на підтримку волонтерських організацій у розміщенні оголошень про волонтерські заходи.

Результатом роботи став вебзастосунок із користувацьким інтерфейсом, доступний для використання через мережу Інтернет.

У процесі реалізації проєкту було проведено аналіз предметної області, пов'язаної з інтеграцією та підтримкою військових. Визначено основні проблеми, з якими стикаються військові у запасі, та проаналізовано наявні програмні рішення, їхні переваги та недоліки. Зібрано функціональні та нефункціональні вимоги до системи, обрано найбільш відповідні технології для її реалізації.

Порівняльний аналіз показав, що жодне з існуючих рішень не задовольняє повною мірою потреби цільової аудиторії. Тому розроблений вебзастосунок є актуальним і затребуваним.

У результаті аналізу технологій для розробки програмного забезпечення було обрано ASP.NET Core, Angular і MS SQL. Архітектура застосунку побудована за принципом клієнт-серверної моделі, де для серверної частини використано шаблон Vertical Slice, а для клієнтської – підхід Model-View-Controller.

У дипломному проєкті реалізовано користувацький інтерфейс, бізнес-логіку на сервері та спроєктовано базу даних. Вебзастосунок повністю відповідає поставленим вимогам. Його тестування було проведено згідно з затвердженою програмою та методикою.

Розроблення виконано у повному обсязі, воно відповідає висунутим до нього вимогам, що описані у даному документі та технічному завданні,

тестування продукту здійснено у відповідності до затвердженої програми та методики тестування.

Використання цього вебзастосунку сприятиме працевлаштуванню військових у запасі та членів їхніх сімей, пошуку ними житла й можливостей для навчання, а також полегшить волонтерським організаціям розміщення інформації про заходи. Крім того, застосунок стане інструментом для органів місцевого самоврядування щодо інтеграції військових у громаду.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Вебсайт для підтримки ветеранів Український фонд ветеранів [Електронний ресурс]. – Режим доступу: https://veteranfund.com.ua/. – Дата доступу: 05.02.2025 р.
2. Вебсайт для підтримки ветеранів Veteran Hub [Електронний ресурс]. – Режим доступу: https://veteranhub.com.ua/. – Дата доступу: 07.02.2025 р.
3. What is Python? Executive Summary [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/essays/blurb/. – Дата доступу: 02.03.2025 р.
4. Python Pros and Cons in 2024 [Електронний ресурс]. – Режим доступу: https://www.netguru.com/blog/python-pros-and-cons. – Дата доступу: 02.03.2025 р.
5. The Django Framework’s Pros and Cons: A Coder’s Guide [Електронний ресурс]. – Режим доступу: https://medium.com/@spectricssolutions/the-django-frameworks-pros-and-cons-a-coder-s-guide-ddf27db8bf08. – Дата доступу: 02.03.2025 р.
6. Що таке Java і де вона використовується [Електронний ресурс]. – Режим доступу: https://goit.global/ua/articles/shcho-take-java-i-de-vona-vykorystovuietsia/. – Дата доступу: 04.03.2025 р.
7. Advantages and Disadvantages of Java [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-java/. – Дата доступу: 04.03.2025 р.
8. Spring Framework Overview [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-framework/reference/overview.html. – Дата доступу: 04.03.2025 р.
9. What is Spring Framework and its Advantages [Електронний ресурс]. – Режим доступу: https://www.simplilearn.com/tutorials/spring-boot-tutorial/what-is-spring-framework-and-its-advantages. – Дата доступу: 04.03.2025 р.

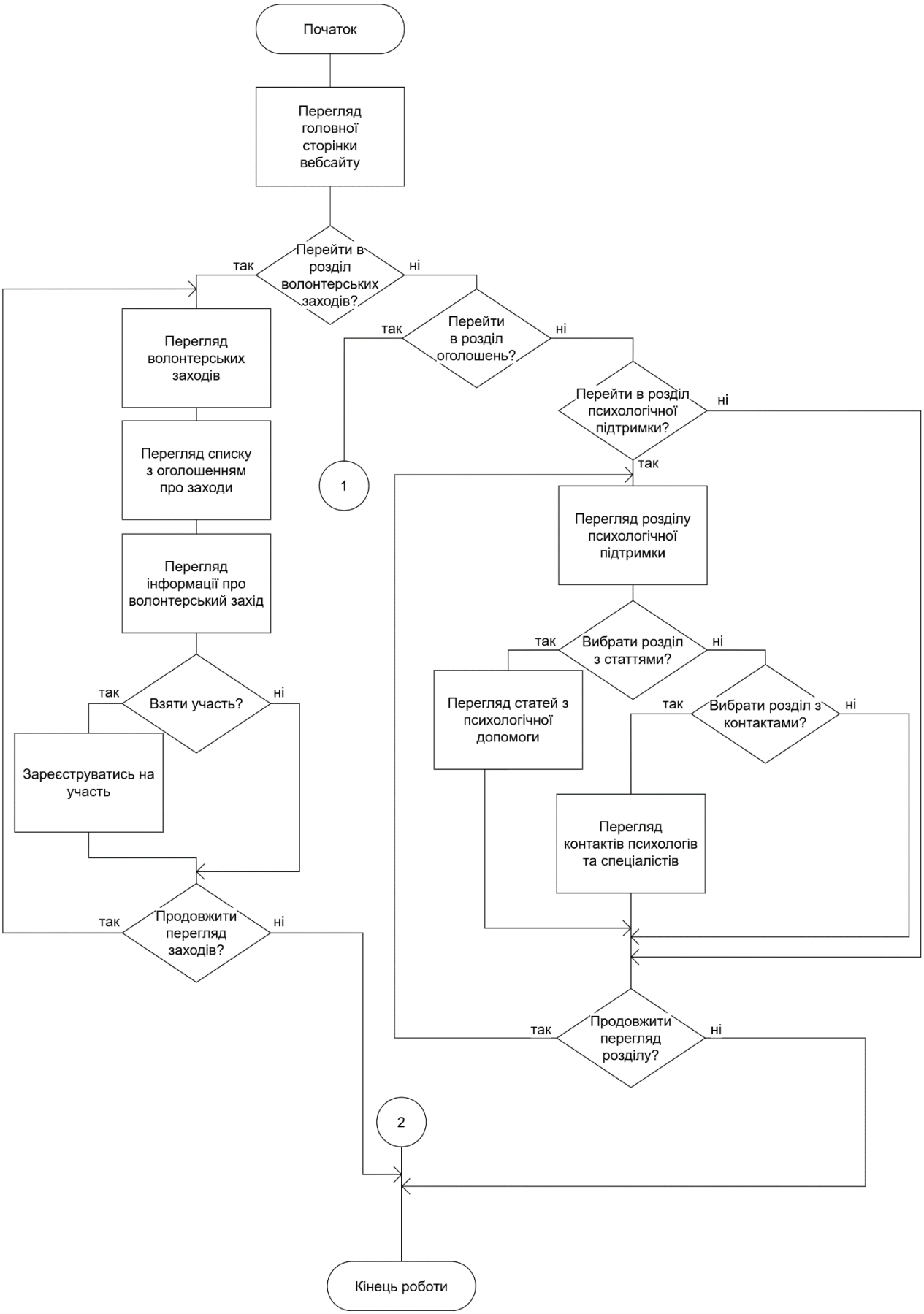
10. Getting Started with Spring Boot: Advantages, Disadvantages, and Use Cases [Електронний ресурс]. – Режим доступу: https://medium.com/@jayeshwarke011/getting-started-with-spring-boot-advantages-disadvantages-and-use-cases-497b0f04fb86. – Дата доступу: 04.03.2025 р.
11. Introduction to C# [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/introduction-to-c-sharp/. – Дата доступу: 04.03.2025 р.
12. C# documentation [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/uk-ua/dotnet/csharp/tour-of-csharp/. – Дата доступу: 04.03.2025 р.
13. ASP.NET Core Documentation [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-9.0. – Дата доступу: 04.03.2025 р.
14. The Pros and Cons of Angular Development: All You Need to Know [Електронний ресурс]. – Режим доступу: https://medium.com/@jobs8918/the-pros-and-cons-of-angular-development-all-you-need-to-know-90d9dc636ea3. – Дата доступу: 04.03.2025 р.
15. Comparing React and Angular secure coding practices [Електронний ресурс]. – Режим доступу: https://lirantal.medium.com/comparing-react-and-angular-secure-coding-practices-snyk-96315e3faf7d. – Дата доступу: 04.03.2025 р.
16. Advantages and Disadvantages of React JS [Електронний ресурс]. – Режим доступу: https://medium.com/@reactmasters.in/advantages-and-disadvantages-of-react-js-e6c80b25763b. – Дата доступу: 04.03.2025 р.
17. The Progressive JavaScript Framework (Vue.js) [Електронний ресурс]. – Режим доступу: https://vuejs.org/. – Дата доступу: 04.03.2025 р.
18. MongoDB Advantages & Disadvantages [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/. – Дата доступу: 24.03.2025 р.

19. Introduction of MS SQL Server [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/introduction-of-ms-sql-server/. – Дата доступу: 25.03.2025 р.
20. Microsoft SQL Server Pros and Cons [Електронний ресурс]. – Режим доступу: https://learnsql.com/blog/microsoft-sql-server-pros-and-cons/. – Дата доступу: 25.03.2025 р.
21. Microsoft SQL Server: Advantages and Disadvantages for your Business [Електронний ресурс]. – Режим доступу: https://licendi.com/en/blog/microsoft-sql-server-advantages-and-disadvantages%20/. – Дата доступу: 25.03.2025 р.
22. What is PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/about/.– Дата доступу: 25.03.2025 р.
23. PostgreSQL Advantages and Disadvantages [Електронний ресурс]. – Режим доступу: https://www.aalpha.net/blog/pros-and-cons-of-using-postgresql-for-application-development/ – (25.03.2025).
24. What Is a REST API? Examples, Uses, and Challenges [Електронний ресурс]. – Режим доступу: https://blog.postman.com/rest-api-examples/. – Дата доступу: 28.03.2025 р.
25. Architecture Overview [Електронний ресурс]. – Режим доступу: https://v2.angular.io/docs/ts/latest/guide/architecture.html. – Дата доступу: 28.03.2025 р.
26. Vertical Slice Architecture [Електронний ресурс]. – Режим доступу: https://www.milanjovanovic.tech/blog/vertical-slice-architecture. – Дата доступу: 28.03.2025 р.
27. Database Design in DBMS [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/database-design-in-dbms/. – Дата доступу: 28.03.2025 р.

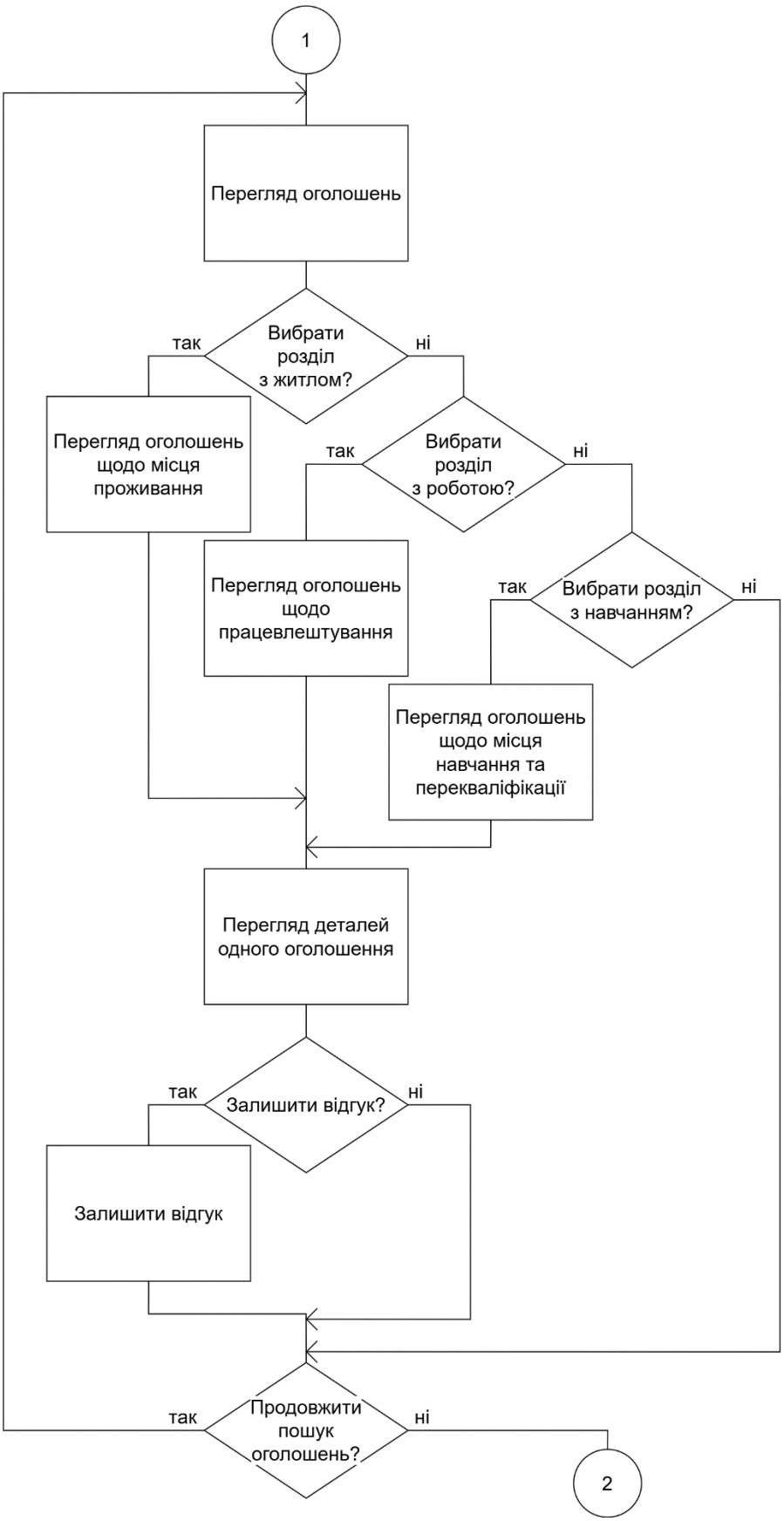
ДОДАТКИ

Додаток 1

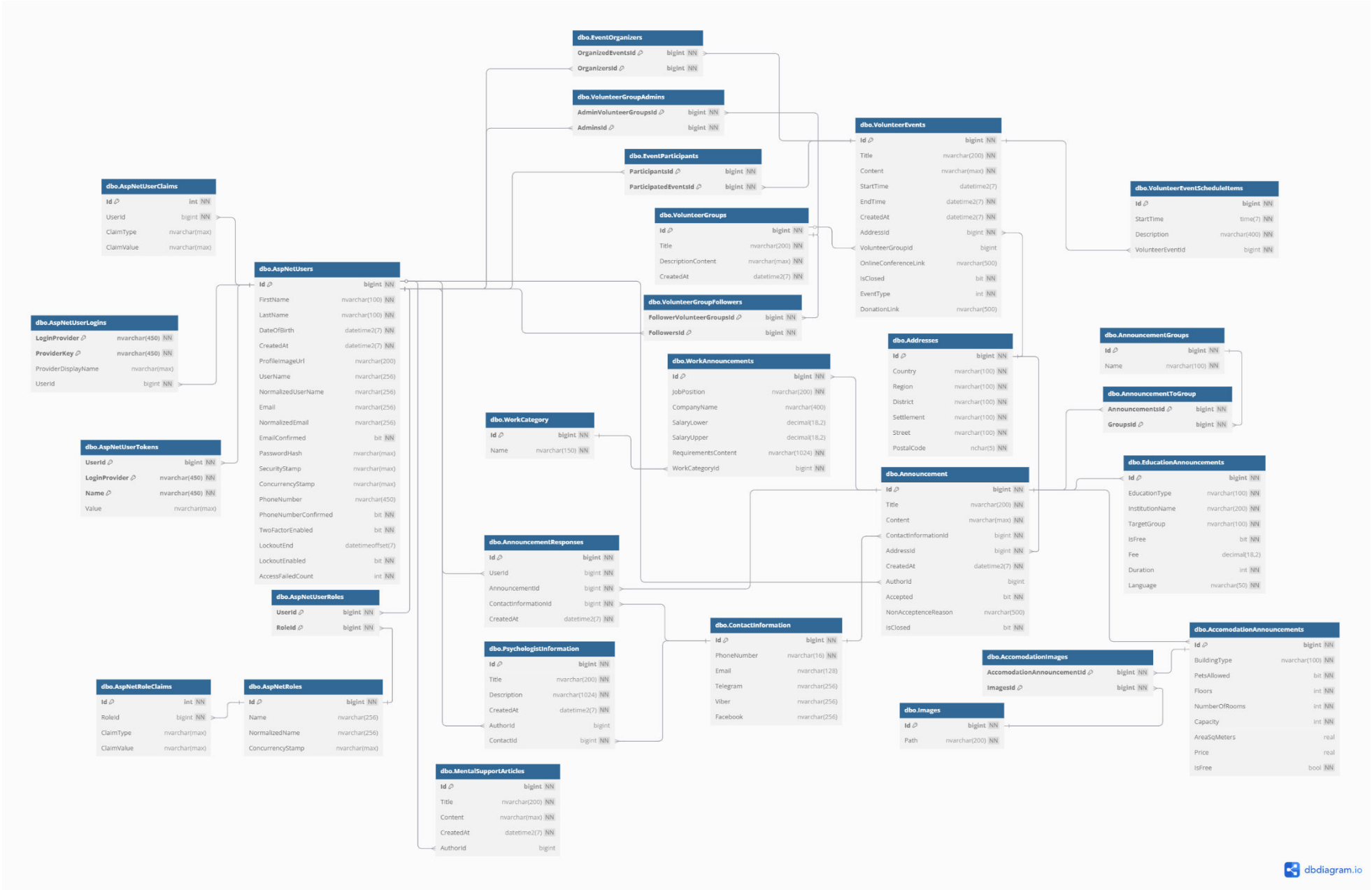
Копії графічних матеріалів



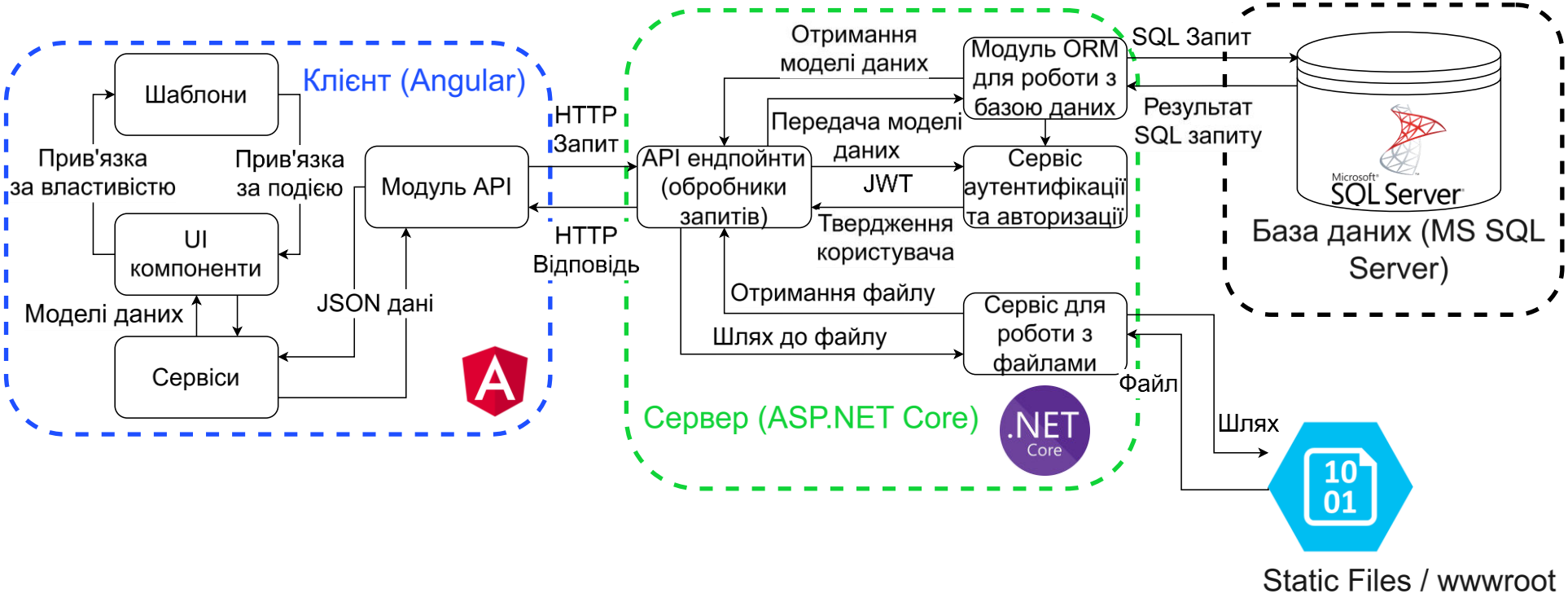
ДП.045440-06-99. Вебзастосунок для допомоги військовим. Алгоритм користування застосунком, частина 1. Схема алгоритму



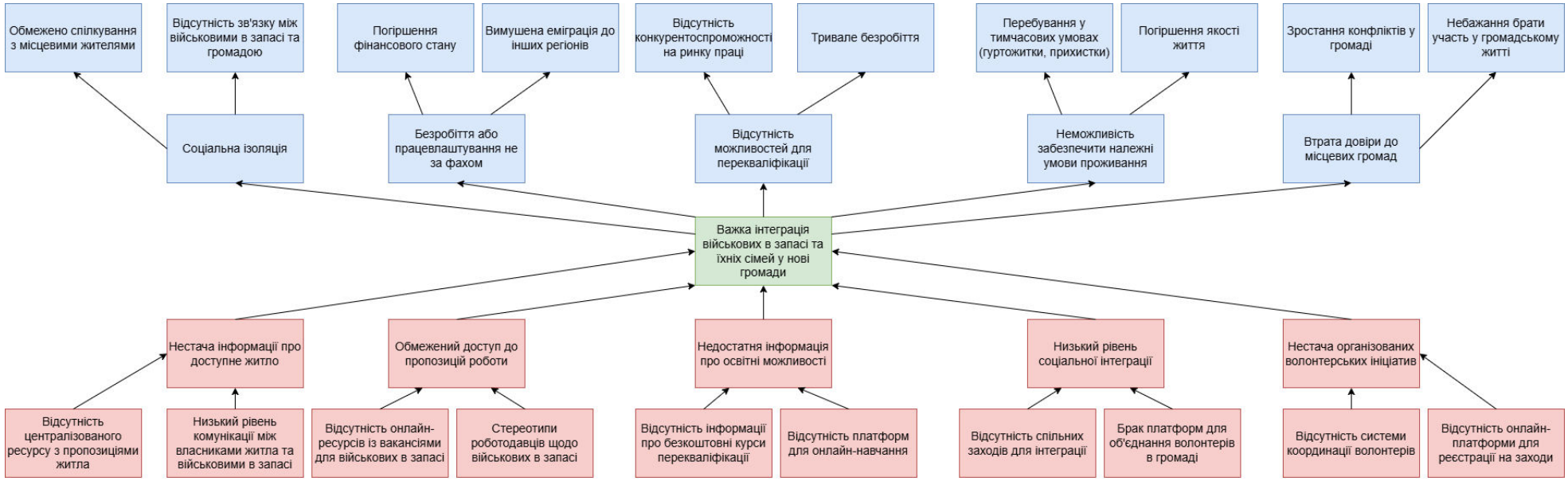
ДП.045440-06-99. Вебзастосунок для допомоги військовим. Алгоритм користування застосунком, частина 2. Схема алгоритму



ДП.045440-07-99. Вебзастосунок для допомоги військовим. Структура бази даних. ERD-діаграма



Архітектура вебзастосунку
Попович Ю.П., група КП-13



Дерево проблем користувачів системи
Попович Ю.П., група КП-13

Додаток 2

Фрагменти коду програми

```
using FluentValidation;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using RefugeUA.DatabaseAccess;
using RefugeUA.Entities;
using RefugeUA.WebApp.Server.Authorization.Constants;
using RefugeUA.WebApp.Server.Extensions.Authentication;
using RefugeUA.WebApp.Server.Extensions.Mapping;
using RefugeUA.WebApp.Server.Features.Announcements.Work.Common;

namespace RefugeUA.WebApp.Server.Features.Announcements.Work.Create
{
    public class CreateWorkAnnouncement : IFeatureEndpoint
    {
        [Authorize(Roles = "Admin,Volunteer,LocalCitizen,CommunityAdmin")]
        public static async Task<IResult> CreateWorkAnnouncementAsync(
            [FromBody] EditOrCreateWorkAnnouncementCommand command,
            [FromServices] RefugeUADbContext dbContext,
            [FromServices] IValidator<EditOrCreateWorkAnnouncementCommand>
validator,
            [FromServices] IHttpContextAccessor httpContextAccessor)
        {
            var validationResult = await validator.ValidateAsync(command);
            if(!validationResult.IsValid)
            {
                return
Results.ValidationProblem(validationResult.ToDictionary());
            }

            long userId = httpContextAccessor!.HttpContext!.User.GetId() ??
0;

            var workAnnouncement = new WorkAnnouncement
            {
                Title = command.Title,
                Content = command.Content,
                JobPosition = command.JobPosition,
                CompanyName = command.CompanyName,
                SalaryLower = command.SalaryLower,
                SalaryUpper = command.SalaryUpper,
                RequirementsContent = command.RequirementsContent,
                WorkCategoryId = command.WorkCategoryId,
                AuthorId = userId,
                Address = command.Address.MapToEntity(),
                ContactInformation =
command.ContactInformation.MapToEntity()
            };

            dbContext.WorkAnnouncements.Add(workAnnouncement);
            await dbContext.SaveChangesAsync();

            return Results.Created();
        }

        public void AddEndpoint(IEndpointRouteBuilder app)
        {
            app.MapPost("api/announcements/work",
CreateWorkAnnouncementAsync)
                .Produces(StatusCodes.Status201Created)
                .Produces(StatusCodes.Status400BadRequest)
                .Produces(StatusCodes.Status403Forbidden)
                .Produces(StatusCodes.Status401Unauthorized)
                .WithName("CreateWorkAnnouncement")
                .WithTags("WorkAnnouncements")
```

```
                .RequireAuthorization();
        }
    }
}

using RefugeUA.WebApp.Server.Features.Announcements.Common;

namespace RefugeUA.WebApp.Server.Features.Announcements.Work.Common
{
    public class EditOrCreateWorkAnnouncementCommand :
BaseEditOrCreateAnnouncementCommand
    {
        public string JobPosition { get; set; } = default!;

        public string CompanyName { get; set; } = default!;

        public decimal? SalaryLower { get; set; }

        public decimal? SalaryUpper { get; set; }

        public string RequirementsContent { get; set; } = default!;

        public long WorkCategoryId { get; set; }
    }
}

using RefugeUA.Entities.Interfaces;
using RefugeUA.Entities;
using RefugeUA.WebApp.Server.Shared.Dto.Address;
using RefugeUA.WebApp.Server.Shared.Dto.ContactInformation;

namespace RefugeUA.WebApp.Server.Features.Announcements.Common
{
    public abstract class BaseEditOrCreateAnnouncementCommand
    {
        public string Title { get; set; } = default!;

        public string Content { get; set; } = default!;

        public ContactInformationDto ContactInformation { get; set; } =
default!;

        public AddressDto Address { get; set; } = default!;
    }
}

using FluentValidation;
using RefugeUA.DatabaseAccess;
using RefugeUA.WebApp.Server.Features.Announcements.Common;

namespace RefugeUA.WebApp.Server.Features.Announcements.Work.Common
{
    public class EditOrCreateWorkAnnouncementCommandValidator :
AbstractValidator<EditOrCreateWorkAnnouncementCommand>
    {
        public EditOrCreateWorkAnnouncementCommandValidator(
            RefugeUADbContext refugeUADbContext,
            IValidator<BaseEditOrCreateAnnouncementCommand>
validatorBaseAnnouncementCommand)
        {
            Include(validatorBaseAnnouncementCommand);

            RuleFor(x => x.WorkCategoryId).MustAsync(
                async (id, cancellation) =>
```

```
                {
                    var workCategory = await
refugeUADbContext.WorkCategories.FindAsync(id);
                    return workCategory != null;
                })
                .WithMessage("Work category does not exist.");

            RuleFor(x => x.JobPosition)
                .NotEmpty()
                .WithMessage("Job position is required.")
                .MaximumLength(200)
                .WithMessage("Job position must not exceed 100
characters.");

            RuleFor(x => x.CompanyName)
                .NotEmpty()
                .WithMessage("Company name is required.")
                .MaximumLength(400)
                .WithMessage("Company name must not exceed 100
characters.");

            RuleFor(x => x.SalaryLower)
                .GreaterThan(0).When(x => x.SalaryLower.HasValue)
                .WithMessage("Salary lower must be greater than 0.")
                .LessThan(x => x.SalaryUpper)
                    .When(x => x.SalaryUpper.HasValue)
                    .WithMessage("Salary lower must be less than salary
upper.");

            RuleFor(x => x.SalaryUpper)
                .GreaterThan(0).When(x => x.SalaryUpper.HasValue)
                .WithMessage("Salary upper must be greater than 0.")
                .GreaterThan(x => x.SalaryLower)
                    .When(x => x.SalaryLower.HasValue)
                    .WithMessage("Salary upper must be greater than salary
lower.");

            RuleFor(x => x.RequirementsContent)
                .NotEmpty().WithMessage("Requirements content is
required.")
                .MaximumLength(1024).WithMessage("Requirements content must
not exceed 1024 characters.");
        }
    }
}

using FluentValidation;
using RefugeUA.DatabaseAccess;
using RefugeUA.WebApp.Server.Shared.Dto.Address;
using RefugeUA.WebApp.Server.Shared.Dto.ContactInformation;

namespace RefugeUA.WebApp.Server.Features.Announcements.Common
{
    public class BaseEditOrCreateAnnouncementCommandValidator :
AbstractValidator<BaseEditOrCreateAnnouncementCommand>
    {
        public
BaseEditOrCreateAnnouncementCommandValidator(RefugeUADbContext
refugeUADbContext,
            IValidator<ContactInformationDto> validatorContactInfoDto,
            IValidator<AddressDto> validatorAddressDto)
        {
            RuleFor(x => x.Title)
                .NotEmpty()
```

```
                .WithMessage("Title is required.")
                .MaximumLength(200)
                .WithMessage("Title must not exceed 200 characters.");

            RuleFor(x => x.Content)
                .NotEmpty()
                .WithMessage("Content is required.")
                .MaximumLength(4096)
                .WithMessage("Content must not exceed 4096 characters.");

            RuleFor(x => x.ContactInformation)
                .Cascade(CascadeMode.Stop)
                .NotNull()
                .WithMessage("Contact information is required.")
                .SetValidator(validatorContactInfoDto);

            RuleFor(x => x.Address).Cascade(CascadeMode.Stop)
                .NotNull()
                .WithMessage("Address is required.")
                .SetValidator(validatorAddressDto);
        }
    }
}
```

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, FormArray } from '@angular/forms';
import { ActivatedRoute, ActivatedRouteSnapshot, Params, Router } from
'@angular/router';
import { AnnouncementWorkService } from
'../../../core/services/announcements/work/announcement-work-service';
import { WorkCategory } from '../../../core/models/work-category';
import { IsValueNullNanOrUndefined } from '../../../shared/util/value-not-
null-undefined-or-nan-checker';
import { WorkAnnouncement } from '../../../core/models';
import { WorkAnnouncementQuery } from '../../../core/queries/work-
announcement-query';
import { BaseAnnouncementComponent } from '../base-announcement/base-
announcement.component';
import { filter } from 'rxjs';
import { SalaryOption, salaryOptionsLower, salaryOptionsUpper } from
'../../../core/constants/salary-options';
import { AuthService } from '../../../core/services/auth/auth-service';

@Component({
  selector: 'app-announcement-work',
  standalone: false,
  templateUrl: './announcement-work.component.html',
  styleUrl: './announcement-work.component.css'
})
export class AnnouncementWorkComponent extends BaseAnnouncementComponent
implements OnInit {
  categories: WorkCategory[] = [];
  displayedCategories: WorkCategory[] = [];
  currentWorkAnnouncements: WorkAnnouncement[] = [];

  currentlyAvailableSalaryOptionsLower: SalaryOption[] =
[...salaryOptionsLower];
  currentlyAvailableSalaryOptionsUpper: SalaryOption[] =
[...salaryOptionsUpper];

  announcementWorkSearchForm: FormGroup = new FormGroup({});
```

```
  constructor(router: Router, route: ActivatedRoute, authService:
AuthService, private announceWorkService: AnnouncementWorkService) {
    super(router, route, authService);
  }

  ngOnInit(): void {
    this.announceWorkService.getCategories().subscribe({
      next: (data) => {
        this.categories = data;
        this.updateDisplayedCategories();
      },
      error: (error) => console.error('Failed to load work categories.',
error),
    });

    this.initializeForm();

    this.initializeDataFromQuery(this.route.snapshot.queryParams);

    this.announcementWorkSearchForm.valueChanges.subscribe(() => {
      this.onFormsDataChanged();
      this.updateDisplayedCategories();
    });

    this.route.queryParams.subscribe(params => {
      window.scrollTo(0, 0);
      if (this.announcementWorkSearchForm) {
        this.initializeDataFromQuery(params);
      }

      this.loadAnnouncementsWithQuery(params);
    });
  }

  protected override get activatedRoute(): ActivatedRoute{
    return this.route;
  }

  get selectedCategories(): FormArray<FormControl<number>> {
    return this.announcementWorkSearchForm.get('jobCategories') as
FormArray<FormControl<number>>;
  }

  initializeForm() {
    this.announcementWorkSearchForm = new FormGroup({
      salaryLower: new FormControl<number | null>(null),
      salaryUpper: new FormControl<number | null>(null),
      salaryNotSet: new FormControl<boolean>(false),
      jobCategories: new FormArray<FormControl<number>>([]),
      district: new FormControl<string | null>(null),
      announcementGroup: new FormControl<string | null>(null),
    });
  }

  initializeDataFromQuery(params: Params) {
    if (!this.announcementWorkSearchForm) {
      console.error('Form is not initialized!');
      return;
    }

    if (params['salaryLower'] && !Number.isNaN(+params['salaryLower'])) {

this.announcementWorkSearchForm.get('salaryLower')?.setValue(+params['salar
yLower'], { emitEvent: false });
```

```
    }
    else {
      this.announcementWorkSearchForm.get('salaryLower')?.setValue(null, {
emitEvent: false });
    }

    if (params['salaryUpper'] && !Number.isNaN(+params['salaryUpper'])) {

this.announcementWorkSearchForm.get('salaryUpper')?.setValue(+params['salar
yUpper'], { emitEvent: false });
    }
    else {
      this.announcementWorkSearchForm.get('salaryUpper')?.setValue(null, {
emitEvent: false });
    }

    if (params['salaryNotSet']) {

this.announcementWorkSearchForm.get('salaryNotSet')?.setValue(params['salar
yNotSet'] === 'true', { emitEvent: false });
    }

    if (params['page']) {
      this.currentPage = +params['page'];
    }

    if (params['district']) {

this.announcementWorkSearchForm.get('district')?.setValue(params['district'
], { emitEvent: false });
    }

    if (params['announcementGroup']) {

this.announcementWorkSearchForm.get('announcementGroup')?.setValue(params['
announcementGroup'], { emitEvent: false });
    }

    if(params['prompt'])
    {
      this.prompt = params['prompt'];
    }

    let ids = [] as number[];
    if (params['jobCategories']) {
      ids = Array.from(new Set((params['jobCategories'] as
string).split('+').map(s => +s)));
      const controls = ids.map(id => new FormControl<number>(id, {
nonNullable: true }));
      const newArray = new FormArray(controls, { updateOn: 'change' });

      this.announcementWorkSearchForm.setControl('selectedCategories',
newArray, { emitEvent: false });
    }

    this.updateDisplayedCategories();
    this.updateAvailableSalaryOptions();
    this.announcementWorkSearchForm.markAsDirty();
  }

  onFormsDataChanged(): void {

    let formValue = this.announcementWorkSearchForm.value;
    let queryParams: any = {};
```

```
    if (!IsValueNullNanOrUndefined(formValue.salaryLower)) {
      queryParams.salaryLower = formValue.salaryLower;
    }

    if (!IsValueNullNanOrUndefined(formValue.salaryUpper)) {
      queryParams.salaryUpper = formValue.salaryUpper;
    }

    if (!IsValueNullNanOrUndefined(formValue.salaryNotSet)) {
      queryParams.salaryNotSet = formValue.salaryNotSet;
    }

    let jobCategories = Array.from(new
Set(formValue.jobCategories)).join('+');

    if (!IsValueNullNanOrUndefined(formValue.jobCategories) &&
(jobCategories?.length ?? 0) > 0) {
      queryParams.jobCategories = jobCategories;
    }

    if (!IsValueNullNanOrUndefined(formValue.district) &&
formValue.district.trim() != "") {
      queryParams.district = formValue.district.trim();
    }

    if (!IsValueNullNanOrUndefined(formValue.announcementGroup) &&
formValue.announcementGroup.trim() != "") {
      queryParams.announcementGroup = formValue.announcementGroup.trim();
    }

    queryParams.page = 1;

    if(this.prompt)
    {
      queryParams.prompt = this.prompt;
    }

    this.isDisplayedFull = false;
    this.router.navigate([], {
      relativeTo: this.route,
      queryParams: queryParams,
      queryParamsHandling: ''
    });
  }

  loadAnnouncementsWithQuery(params: Params) {
    this.announcementsLoaded = false;
    let workAnnouncementQuery: WorkAnnouncementQuery = { pageLength: 7,
page: 1, isClosed: false };

    if (params['salaryLower'] && !Number.isNaN(+params['salaryLower'])) {
      workAnnouncementQuery.salaryLower = +params['salaryLower'];
    }

    if (params['salaryUpper'] && !Number.isNaN(+params['salaryUpper'])) {
      workAnnouncementQuery.salaryUpper = +params['salaryUpper'];
    }

    if (params['salaryNotSet']) {
      workAnnouncementQuery.salaryNotSet = params['salaryNotSet'] ===
'true';
    }
```

```
    if (params['page']) {
      workAnnouncementQuery.page = +params['page'];
    }

    if (params['district']) {
      workAnnouncementQuery.district = params['district'];
    }

    if (params['announcementGroup']) {
      workAnnouncementQuery.announcementGroup =
params['announcementGroup'];
    }

    if(params['prompt'])
    {
      workAnnouncementQuery.prompt = params['prompt'];
    }

    let ids = [] as number[];
    if (params['jobCategories']) {
      ids = Array.from(new Set((params['jobCategories'] as
string).split('+').map(s => +s)));
    }

    if (ids.length > 0) {
      workAnnouncementQuery.jobCategories = ids;
    }


this.announceWorkService.getWorkAnnouncements(workAnnouncementQuery).subscr
ibe({
      next: (result) => {
        console.log(result);
        this.currentWorkAnnouncements = result.items;
        this.pages = result.pagesCount;
        this.announcementsLoaded = true;
        this.announcementsFound = true;
      },
      error: (err) => {
        this.announcementsFound = false;
        this.announcementsLoaded = true;
        console.log(err);
      }
    });
  }

  onAllCategoriesButtonClick() {
    this.isDisplayedFull = true;
  }

  onCheckboxChange(event: Event) {
    const input = event.target as HTMLInputElement;
    const value = +input.value;

    if (input.checked) {
      this.selectedCategories.push(new FormControl<number>(value, {
nonNullable: true }));
    } else {
      const index = this.selectedCategories.controls.findIndex(c => c.value
=== value);
      if (index !== -1) this.selectedCategories.removeAt(index);
    }
  }
```

```
  isCategorySelected(id: number): boolean {
    return this.selectedCategories.value.includes(id);
  }

  updateDisplayedCategories() {
    this.displayedCategories = [];
    let ids = this.selectedCategories.value;
    this.displayedCategories.push(...this.categories.filter(c =>
ids.includes(c.id!)));
    this.displayedCategories.push(...this.categories.filter(c =>
!ids.includes(c.id!)));
  }

  updateAvailableSalaryOptions() {
    var formValue = this.announcementWorkSearchForm.value;
    if (formValue.salaryUpper == null) {
      this.currentlyAvailableSalaryOptionsLower = [...salaryOptionsLower];
    }
    else {
      this.currentlyAvailableSalaryOptionsLower =
salaryOptionsLower.filter(o => o.value <= formValue.salaryUpper);
    }

    if (formValue.salaryLower == null) {
      this.currentlyAvailableSalaryOptionsUpper = [...salaryOptionsUpper];
    }
    else {
      this.currentlyAvailableSalaryOptionsUpper =
salaryOptionsUpper.filter(o => o.value >= formValue.salaryLower);
    }
  }
}


<div class="row m-2">
    <app-search-bar class="mb-4" placeholder="Посада, заголовок..."
[onSearchBarSearch]="this.getOnSearchBarCallback()"></app-search-bar>
    @if(this.authService.authSubject.value &&
this.authService.userClaims.role != Roles.MilitaryOrFamily)
    {
    <div class="mb-4">
        <a routerLink="create" class="btn btn-custom-
green"><strong>Створити оголошення про роботу</strong></a>
    </div>
    }
    <div class="col-3">
        <form [formGroup]="announcementWorkSearchForm" class="d-flex flex-
column w-100">
            <div class="form-group text-start">
                <label class="h5">Громада</label>
                <ul class="list-unstyled my-0 mx-2">
                    <li class="no-style d-flex flex-align-center mb-0 mt-2
gap-2">
                        <input type="text" class="form-control"
placeholder="Громада" formControlName="district"/>
                    </li>
                </ul>
            </div>
            <div class="form-group mt-2 text-start">
                <label class="h5">Група оголошень</label>
                <ul class="list-unstyled my-0 mx-2">
                    <li class="no-style d-flex flex-align-center mb-0 mt-2
gap-2">
```

```
                        <input type="text" class="form-control"
placeholder="Група оголошень" formControlName="announcementGroup"/>
                    </li>
                </ul>
            </div>
            <div class="form-group mt-2 text-start">
                <label class="h5">Зарплата</label>
                <ul class="list-unstyled my-0 mx-2">
                    <li class="no-style d-flex flex-align-center mb-0 mt-2
gap-2">
                        <span class="mr-sm mt-2 option-nav-
label">від</span>
                        <select class="form-control"
formControlName="salaryLower">
                            <option [ngValue]="null">будь-яка</option>
                            @for (item of
this.currentlyAvailableSalaryOptionsLower; track $index)
                            {
                            <option
[ngValue]="item.value">{{item.label}}</option>
                            }
                        </select>
                    </li>
                    <li class="no-style d-flex flex-align-center mb-0 mt-2
gap-2">
                        <span class="mr-sm mt-2 option-nav-label">до</span>
                        <select class="form-control"
formControlName="salaryUpper">
                            <option [ngValue]="null">будь-яка</option>
                            @for (item of
this.currentlyAvailableSalaryOptionsUpper; track $index)
                            {
                            <option
[ngValue]="item.value">{{item.label}}</option>
                            }
                        </select>
                    </li>
                    <li class="no-style d-flex flex-align-center mb-0 mt-2
gap-2">
                        <label class="custom-checkbox-container">
                            <input readonly style="width: 1.25rem; height:
1.25rem;" type="checkbox"
                                formControlName="salaryNotSet"
[checked]="announcementWorkSearchForm.value.salaryNotSet"
                                class="mt-2" />
                            <span class="custom-checkbox"></span>
                            <span class="mr-sm mt-2 checkbox-label">Не
вказана</span>
                        </label>
                    </li>
                </ul>
            </div>
            <div class="form-group mt-2 text-start">
                <label class="h5">Категорії</label>
                <ul class="list-unstyled my-0 mx-2">
                    @if(this.displayedCategories.length > 0)
                    {
                    @for(item of ((this.isDisplayedFull) ?
this.displayedCategories : this.displayedCategories.slice(0,
                    Math.max(this.displayedCountMin,
this.selectedCategories.value.length))); track item.id)
                    {
                    <li class="no-style d-flex flex-align-center mb-0 mt-
2">
```

```
                        <label class="custom-checkbox-container">
                            <input style="width: 1.25rem; height: 1.25rem;"
type="checkbox" [value]="item.id"

[checked]="this.isCategorySelected(item.id!)"
(change)="this.onCheckboxChange($event)"
                                class="mt-2" />
                            <span class="custom-checkbox"></span>
                            <span class="mr-sm mt-2 checkbox-
label">{{item.name}}</span>
                        </label>
                    </li>
                    }
                    }
                    @else {
                    <img src="assets/gif/loading.gif" width="50em"
style="margin-left: auto; margin-right: auto;">
                    }
                </ul>
                @if(!this.isDisplayedFull && this.selectedCategories.length
!= this.displayedCategories.length)
                {
                <button (click)="onAllCategoriesButtonClick()" class="btn
btn-custom mt-2">
                    Усі категорії
                    <i class="fa-solid fa-chevron-down ms-1"></i>
                </button>
                }
            </div>
        </form>
    </div>
    <div class="col-9 d-flex flex-column gap-4 mt-1">
        @if(this.announcementsLoaded)
        {
        @if(!this.announcementsFound)
        {
        <span class="error-display">Не було знайдено оголошення!</span>
        }
        @else {
        @for (item of this.currentWorkAnnouncements; track $index) {
        <app-announcement-work-preview [workAnnouncement]="item"></app-
announcement-work-preview>
        }
        <app-paging-navigation [pagesCount]="this.pages"
[onNavigationCallback]="getNavigationCallback()"
            [currentPage]="this.currentPage"></app-paging-navigation>
        }
        }
        @else {
        <img src="assets/gif/loading.gif" width="75em" style="margin-left:
auto; margin-right: auto;">
        }
    </div>
</div>


import { HttpClient } from "@angular/common/http";
import { CookieService } from "ngx-cookie-service";
import { catchError, map, Observable, of } from "rxjs";
import { Injectable } from "@angular/core";
import { BehaviorSubject } from "rxjs";
import { RegisterCommand } from "../../api-models/register-command";
import { throwError } from "rxjs";
```

```
import { SendEmailConfirmation } from "../../api-models/send-email-
confirmation";
import { Token } from "../../api-models/token";
import { decodeToken } from "../../../shared/util/token-decoder/token-
decode";
import { Router } from "@angular/router";
import { SessionStorageService } from "../util/session-storage-service";
import { UserClaims } from "../../api-models/user-claims";

@Injectable({
    providedIn: 'root'
})
export class AuthService {
    authSubject = new BehaviorSubject<boolean>(false);
    isAdminSubject = new BehaviorSubject<boolean>(false);
    isCommunityAdminSubject = new BehaviorSubject<boolean>(false);
    isVolunteerSubject = new BehaviorSubject<boolean>(false);
    isLocalCitizenSubject = new BehaviorSubject<boolean>(false);
    isMilitaryOrFamilySubject = new BehaviorSubject<boolean>(false);

    public userClaims!: UserClaims;

    isAuthenticated$ = this.authSubject.asObservable();
    isAdmin$ = this.isAdminSubject.asObservable();
    isCommunityAdmin$ = this.isCommunityAdminSubject.asObservable();
    isVolunteer$ = this.isVolunteerSubject.asObservable();
    isLocalCitizen$ = this.isLocalCitizenSubject.asObservable();
    isMilitaryOrFamily$ = this.isMilitaryOrFamilySubject.asObservable();

    constructor(
        private cookieService: CookieService,
        private http: HttpClient,
        private router: Router,
        private sessionStorageService: SessionStorageService) {
    }

    register(registerCommand: RegisterCommand): Observable<string> {
        return this.http.post<any>('api/auth/register',
registerCommand).pipe(
            map(response => {
                return response;
            }),
            catchError(err => {
                const statusCode = err.status;
                console.error('Error Status Code:', statusCode);

                if (statusCode === 400) {
                    return throwError(() => new Error('Хибна введені дані
реєсатрції!'));
                }

                return throwError(() => new Error('Виникла помилка на
сервері.'));
            })
        );
    }

    authenticate() {
        let token = this.sessionStorageService.getItem('token');
        console.log(token);
        try {
            let claims = decodeToken(token!);
            this.userClaims = claims;
            if (claims.exp <= Math.floor(Date.now() / 1000) - 3600 * 3) {
```

```
                this.sessionStorageService.setItem('sessionExpired',
'true');
                this.sessionExpired();
                return;
            }

            this.authSubject.next(true);
            switch (claims.role) {
                case 'Admin':
                    this.isAdminSubject.next(true);
                    break;
                case 'CommunityAdmin':
                    this.isCommunityAdminSubject.next(true);
                    break;
                case 'Volunteer':
                    this.isVolunteerSubject.next(true);
                    break;
                case 'LocalCitizen':
                    this.isLocalCitizenSubject.next(true);
                    break;
                case 'MilitaryOrFamily':
                    this.isMilitaryOrFamilySubject.next(true);
                    break;
            }
        }
        catch (error) {
            this.sessionStorageService.removeItem('token');
        }
    }

    login(email: string, password: string): Observable<boolean> {

        return this.http.post<Token>('api/auth/login', {
            email: email,
            password: password
        }).pipe(map(response => {
            console.log(response);
            this.sessionStorageService.setItem('token', response.token);
            this.authenticate();
            return true;
        }),
            catchError(err => {
                let errorMessage: string = '';

                switch (err.status) {
                    case 500:
                        errorMessage = "Виникла помилка на сервері,
спробуйте ще раз";
                        break;
                    case 404:
                        errorMessage = "Не було знайдено насупної пошти: "
+ email;
                        break;
                    case 400:
                        errorMessage = "Введений хибний пароль";
                        break;
                    case 403:
                        let errors: string[] = [];

                        if (err.error.detail.includes("Not accepted")) {
                            errors.push("Заявка на реєстрацію за цією
поштою. Зачекайте повідомлення на пошту або спробуйте пізніше.");
                        }
```

```
                        if (err.error.detail.includes("Not confirmed")) {
                            errors.push("Дану пошту не було підтверджено,
підтвердіть пошту.");
                        }

                        errorMessage = errors.join('\n');
                        break;
                }

                return throwError(() => new Error(errorMessage));
            }));
    }

    sessionExpired() {
        if (this.sessionStorageService.getItem('token')) {
            this.sessionStorageService.removeItem('token');
        }
        this.authSubject.next(false);
        this.isAdminSubject.next(false);
        this.isCommunityAdminSubject.next(false);
        this.isLocalCitizenSubject.next(false);
        this.isMilitaryOrFamilySubject.next(false);

        console.log(this.router.url);
        this.router.navigate(['/', 'login']);
    }

    logout() {

        if (this.sessionStorageService.getItem('token')) {
            this.sessionStorageService.removeItem('token');
        }

        this.authSubject.next(false);
        this.isAdminSubject.next(false);
        this.isCommunityAdminSubject.next(false);
        this.isLocalCitizenSubject.next(false);
        this.isMilitaryOrFamilySubject.next(false);

        console.log(this.router.url);
        this.router.navigate([this.router.url]);
    }

    sendEmailConfirmation(sendEmailConfirmation: SendEmailConfirmation):
Observable<boolean> {
        return this.http.post(`api/auth/send-email-confirmation`,
sendEmailConfirmation).
            pipe(map(response => {
                return true;
            }),
                catchError(err => {
                    return throwError(() => new Error('Виникла помилка
відправлення підтвердження на пошту'));
                }));
    }

    confirmEmail(token: string, email: string): Observable<boolean> {
        return this.http.post(`api/auth/confirm-
email?token=${token}&email=${email}`, {}).
            pipe(map(response => {
                return true;
            }),
                catchError(err => {
```

```
                    return throwError(() => new Error('Виникла помилка
підтвердження пошти'));
                }));
    }

    emailExists(email: string): Observable<boolean> {
        return this.http.get<boolean>(`api/auth/email-
exists?email=${email}`);
    }

    phoneNumberExists(phoneNumber: string): Observable<boolean> {
        return this.http.get<boolean>(`api/auth/phonenumber-
exists?phoneNumber=${phoneNumber}`);
    }

    isAllowedToEditAnnouncement(id: number): Observable<boolean> {
        return this.http.get<boolean>(`api/announcements/${id}/is-allowed-
to-edit`).
            pipe(map(response => {
                return response;
            }),
                catchError(err => {
                    return of(false);
                }));
    }

    isAllowedToEditVolunteerEvent(id: number): Observable<boolean> {
        return this.http.get<boolean>(`api/volunteer/events/${id}/is-
allowed-to-edit`).
            pipe(map(response => {
                return response;
            }),
                catchError(err => {
                    return of(false);
                }));
    }
}
```

Додаток 3

Копії презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО"

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

ВЕБЗАСТОСУНОК ДЛЯ ДОПОМОГИ ВІЙСЬКОВИМ В ЗАПАСІ ТА ЇХНІМ СІМ'ЯМ З ПОШУКОМ РОБОТИ, ЖИТЛА ТА МІСЦЯ НАВЧАННЯ, І ДОПОМОГИ ОРГАНІЗАЦІЇ ВОЛОНТЕРСЬКИХ ЗАХОДІВ

Виконав: Попович Юрій Петрович

Керівник: доц. кафедри ПЗКС, к.т.н., доц. Ткаченко Костянтин Олександрович

Київ – 2025

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: забезпечення ефективної інтеграції військових у запасі та їхніх сімей у нові громади шляхом створення програмного забезпечення, яке дозволяє здійснювати пошук житла, роботи, можливостей для освіти та перекваліфікації, а також організовувати волонтерські заходи з метою зміцнення зв'язків між ветеранами та місцевою спільнотою.

Завдання:

1. Виконати аналіз ринку ПЗ для підтримки та інтеграції військових.
2. Порівняти конкурентів з виявленням їхніх сильних та слабких сторін.
3. Визначити функціональні та нефункціональні вимоги системи.
4. Підготувати дизайн вебзастосунку.
5. Обґрунтувати вибір стеку технологій для розробки ПЗ.
6. Розробити вебзастосунок для допомоги військовим.
7. Протестувати вебзастосунок на відповідність вимогам.
8. Навести шляхи подальшого покращення.

АКТУАЛЬНІСТЬ

Станом на 2024 рік в Україні понад 1 мільйон осіб пройшли військову службу, з яких десятки тисяч щорічно повертаються до цивільного життя.

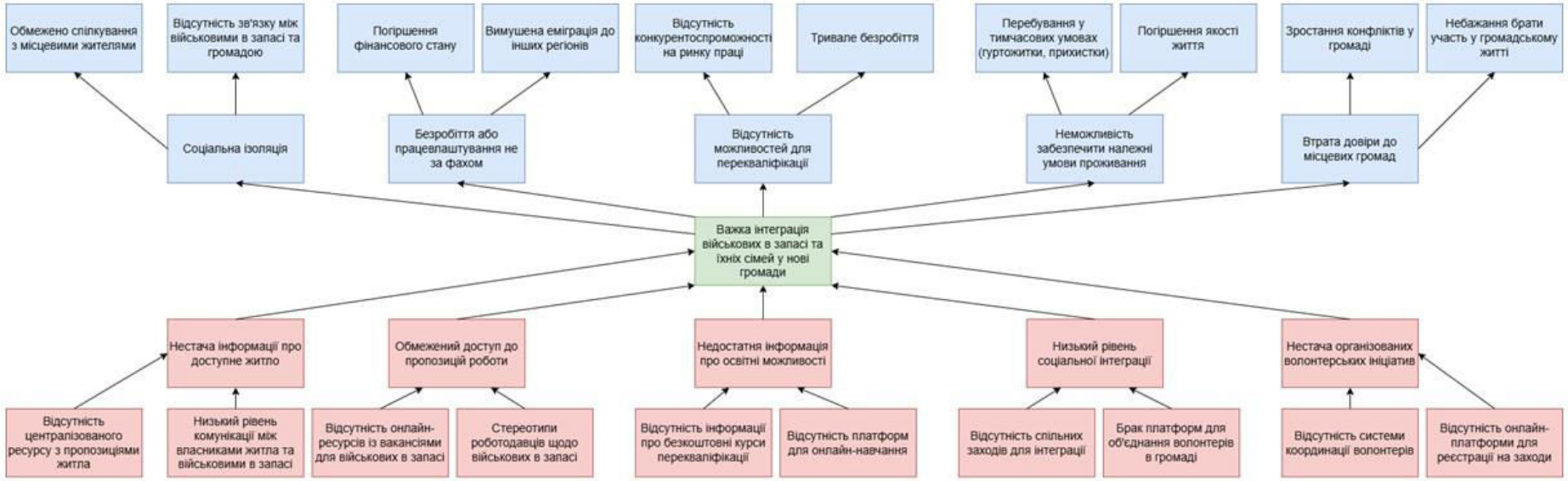
Лише близько 30% ветеранів успішно працевлаштовуються протягом першого року після демобілізації

Відсутність єдиного цифрового ресурсу, який би об'єднував інформацію про доступне житло, вакансії, навчальні програми та волонтерську підтримку

ФУНКЦІОНАЛЬНІ ВИМОГИ

- Реєстрація та вхід користувача в систему.
- Створення, перегляд, редагування та видалення оголошень відповідно до прав доступу.
- Можливість взяти участь у волонтерських заходах.
- Можливість перегляду волонтерських заходів
- Можливість відгукнутися на оголошення та перегляду власних відгуків.
- Можливість перегляду власного профілю.
- Можливість перегляду статей з психологічної підтримки.
- Можливість перегляду контактів спеціалістів з психологічної допомоги.
- Можливість перегляду волонтерських груп.
- Можливість слідкувати за волонтерськими групами.

ДЕРЕВО ПРОБЛЕМ



ВИБІР ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ

ASP.NET CORE

Fluent Validation

Angular

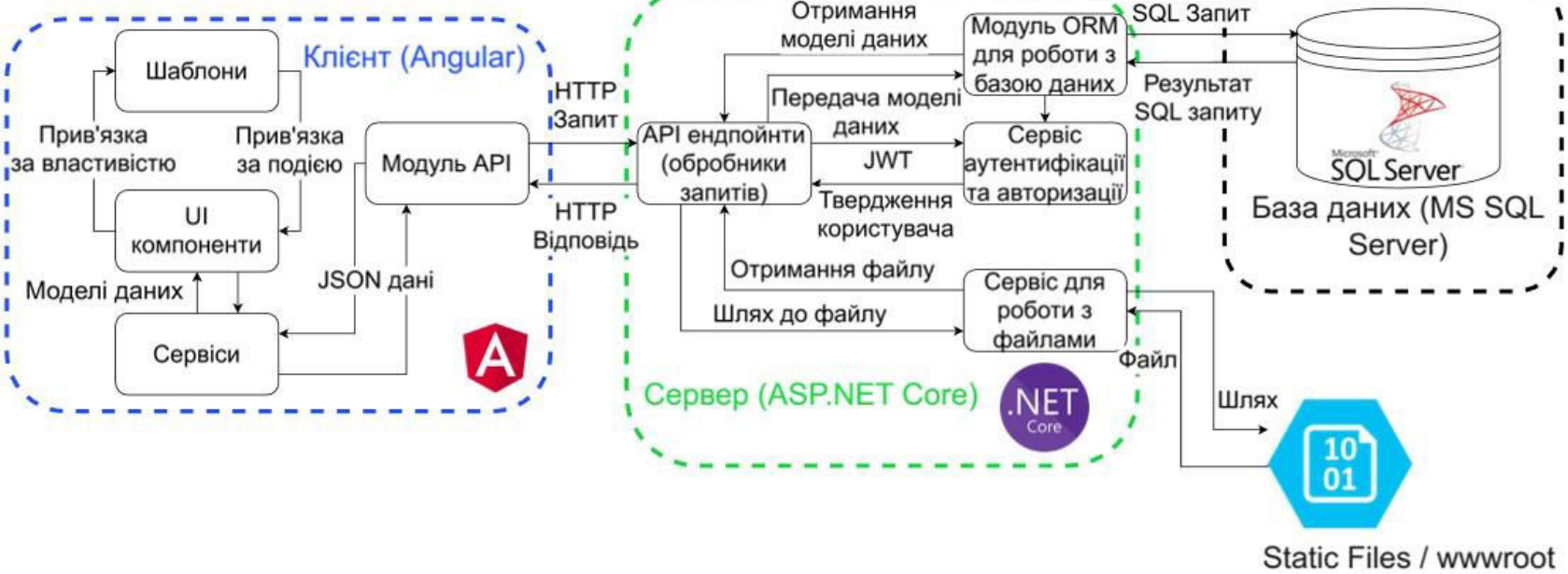
TypeScript

MS SQL Server

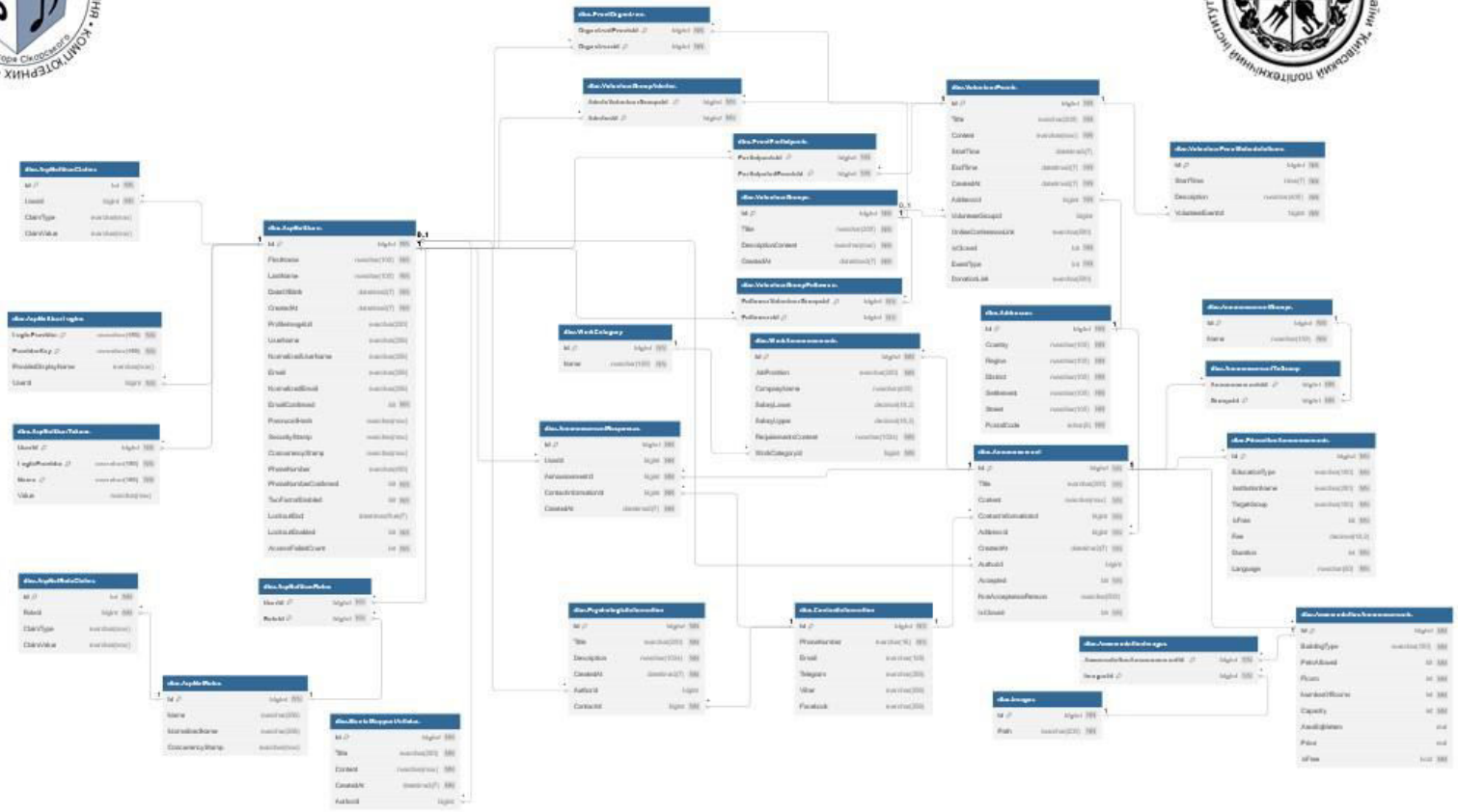
Entity Framework Core

C#

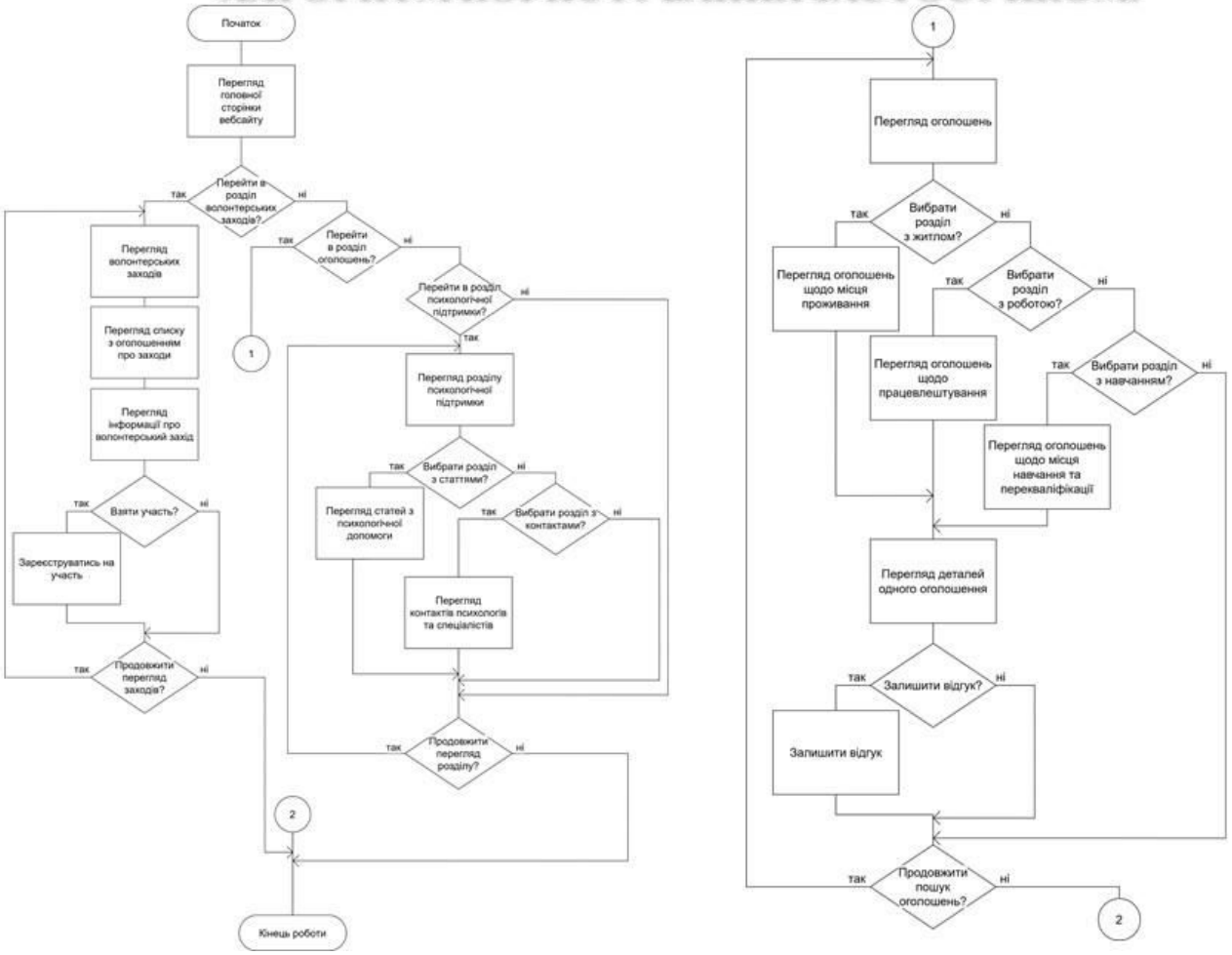
АРХІТЕКТУРА СИСТЕМИ



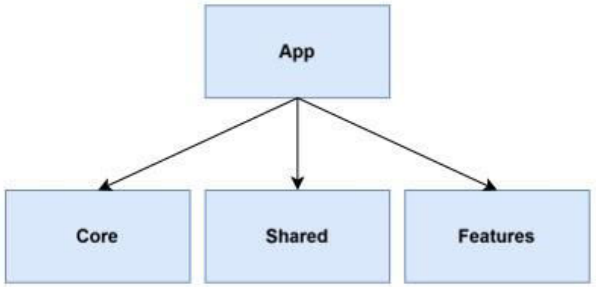
СТРУКТУРА БД



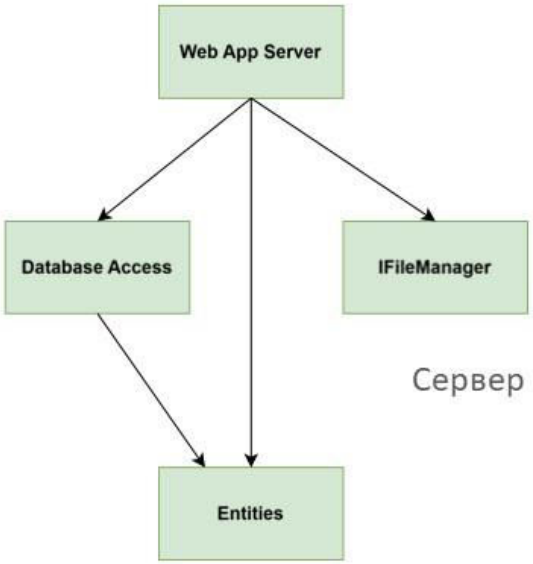
ВЕБЗАСТОСУНОК ДЛЯ ДОПОМОГИ ВІЙСЬКОВИМ. АЛГОРИТМ КОРИСТУВАННЯ ЗАСТОСУНКОМ.



ДІАГРАМА СТРУКТУРУ КЛІЄНТСЬКОЇ ТА СЕРВЕРНОЇ ЧАСТИН

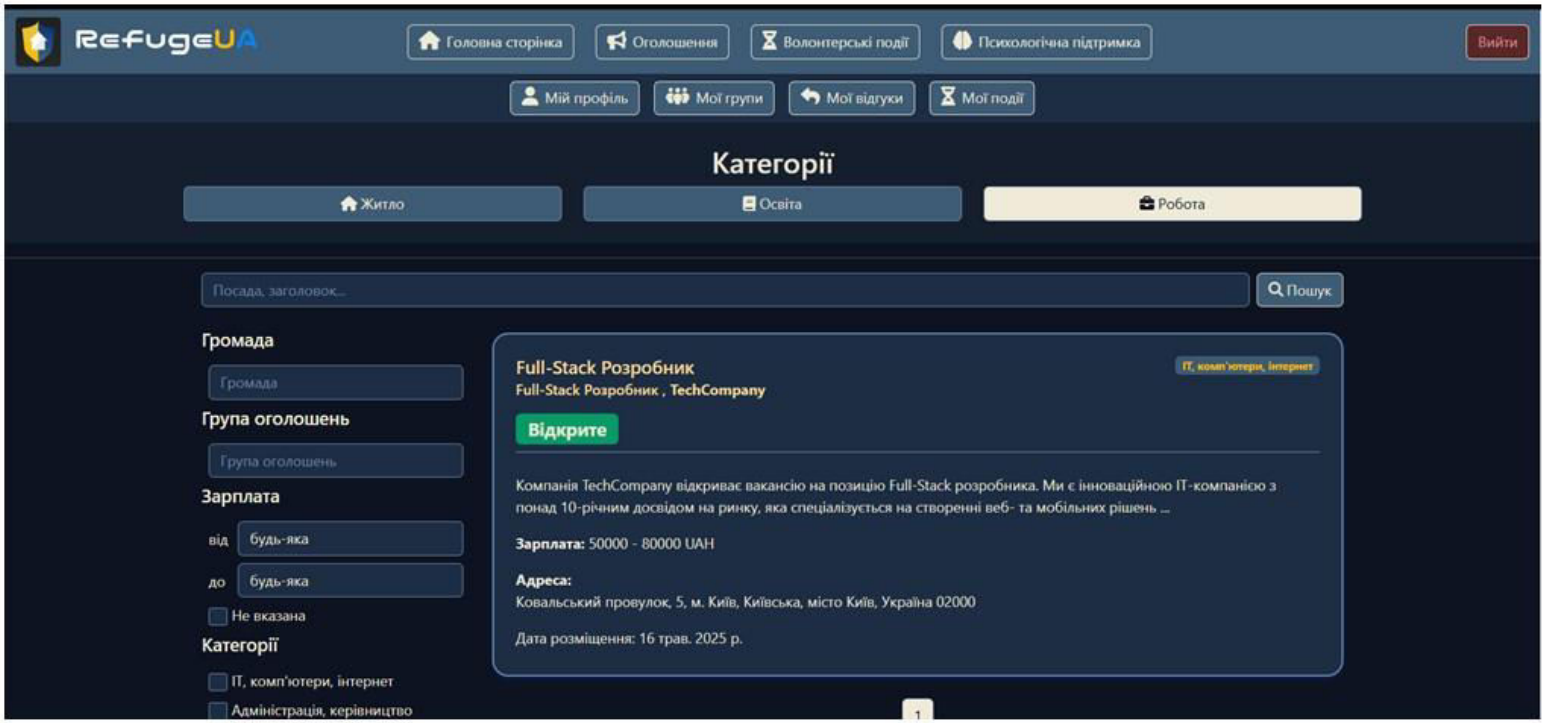


Клієнт

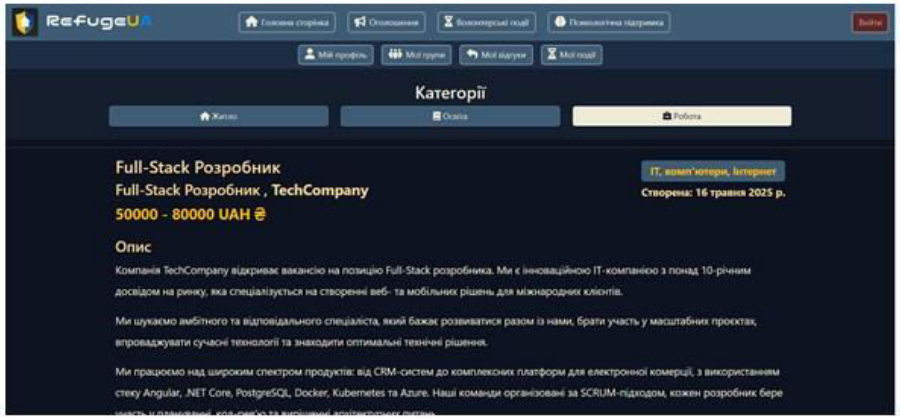


Сервер

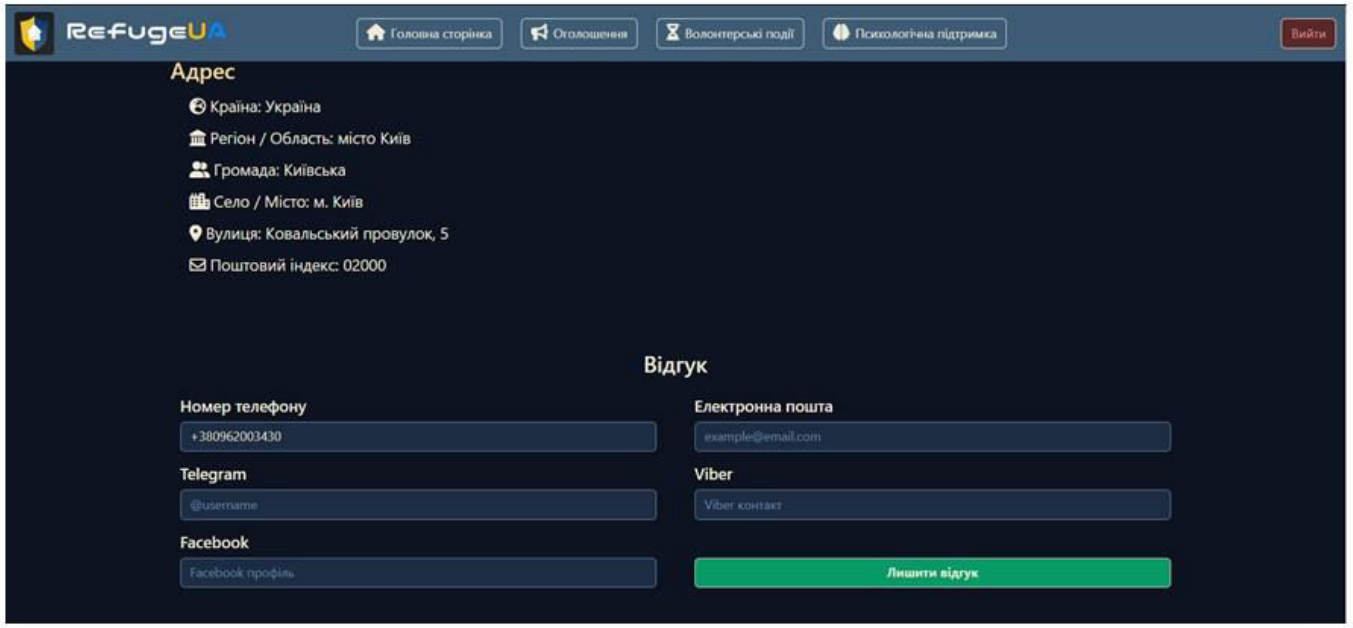
СТОРІНКА ПОШУКУ ОГОЛОШЕННЯ



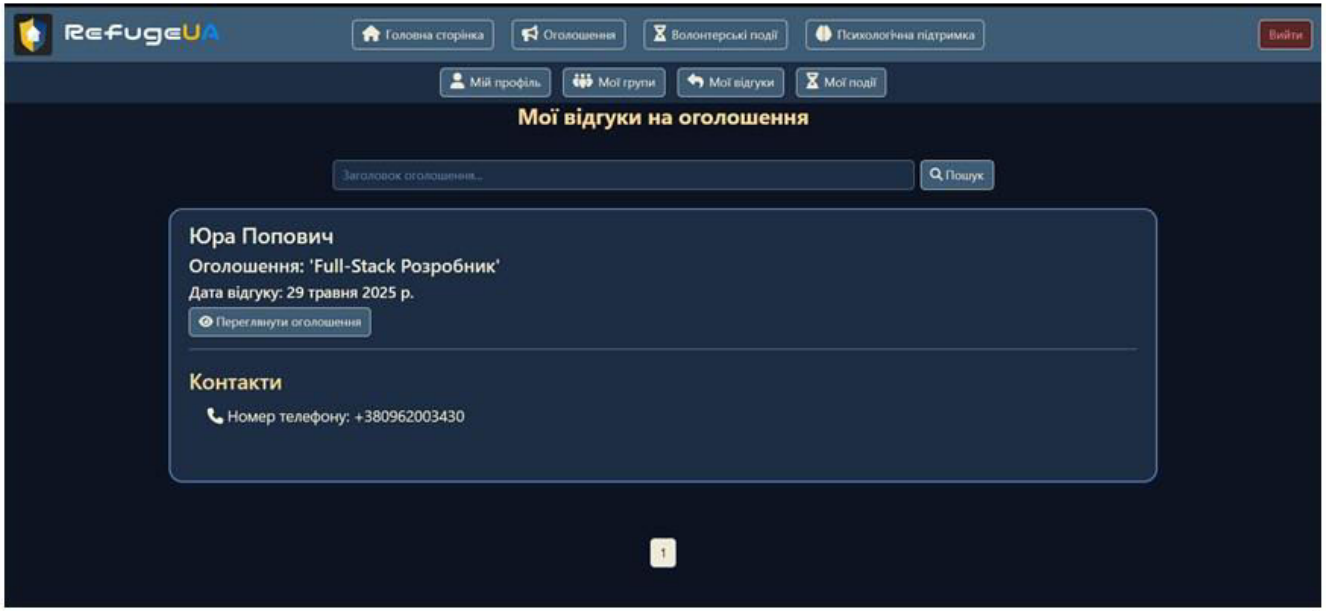
СТОРІНКА ОГОЛОШЕННЯ



ФОРМА ДЛЯ СТВОРЕННЯ ВІДГУКУ

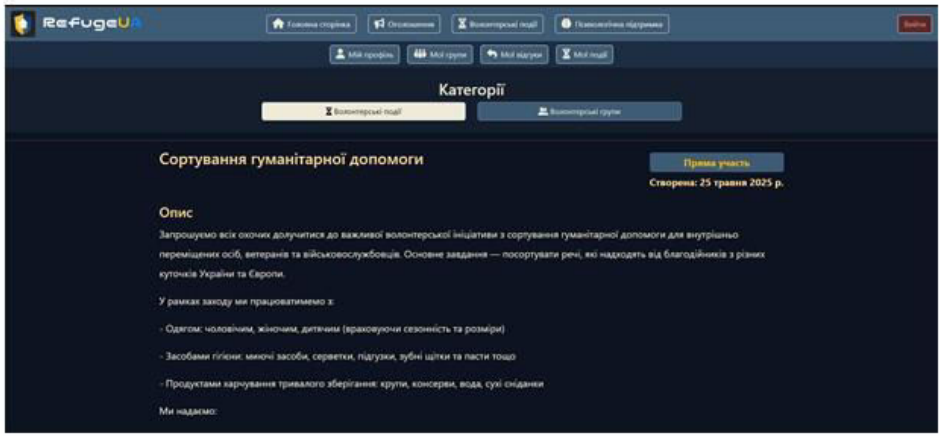


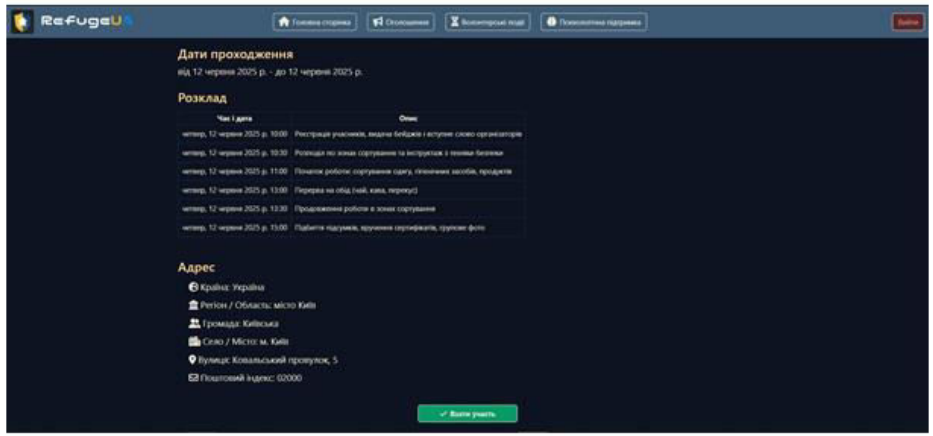
СТОРІНКА ПЕРЕГЛЯДУ СВОЇХ ВІДГУКІВ



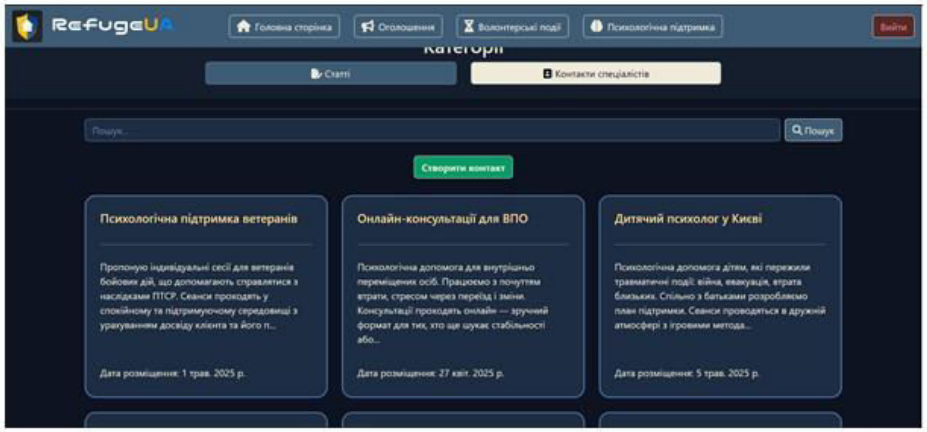
СТОРІНКА ПЕРЕГЛЯДУ ВОЛОНТЕРСЬКИХ ЗАХОДІВ

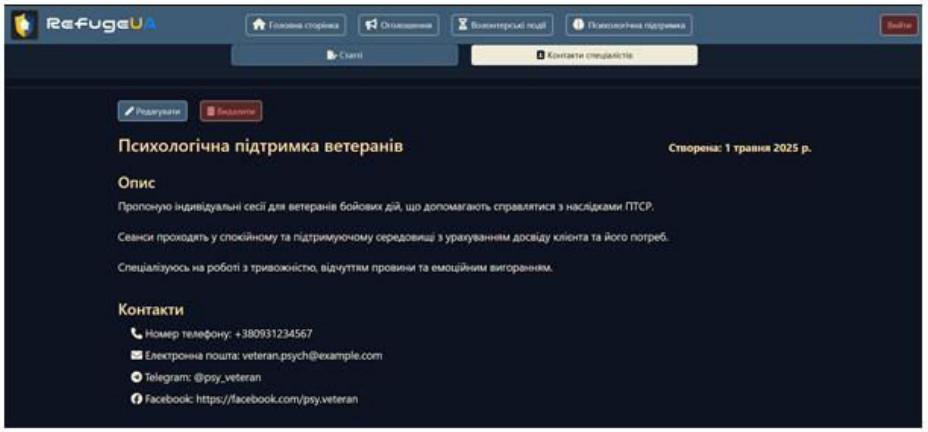
СТОРІНКА ПЕРЕГЛЯДУ ВОЛОНТЕРСЬКОГО ЗАХОДУ



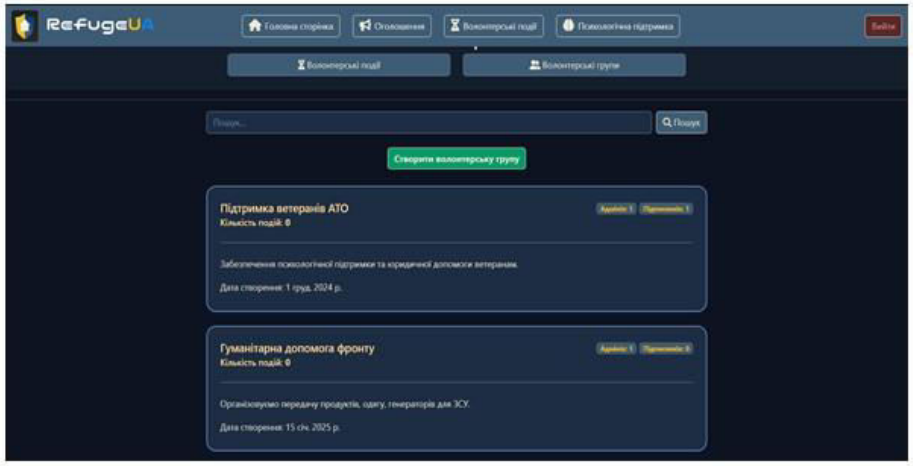


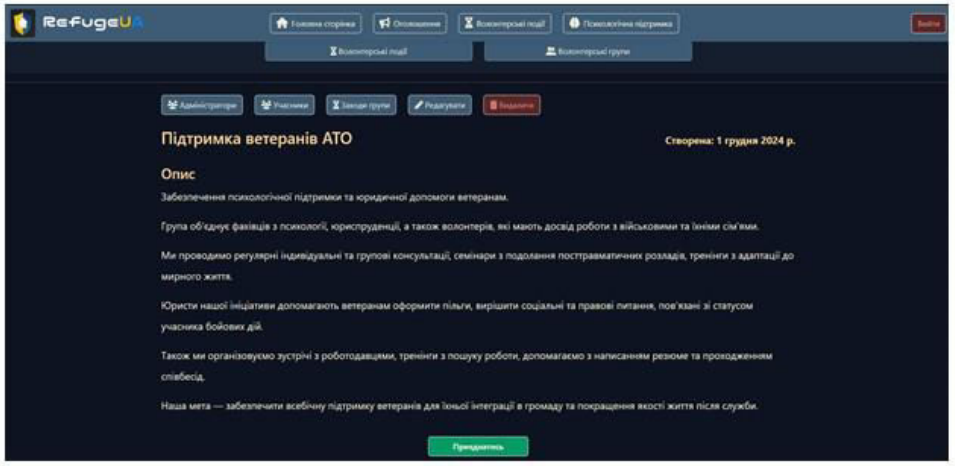
СТОРІНКА ПЕРЕГЛЯДУ РОЗДІЛУ ПСИХОЛОГІЧНОЇ ПІДТРИМКИ



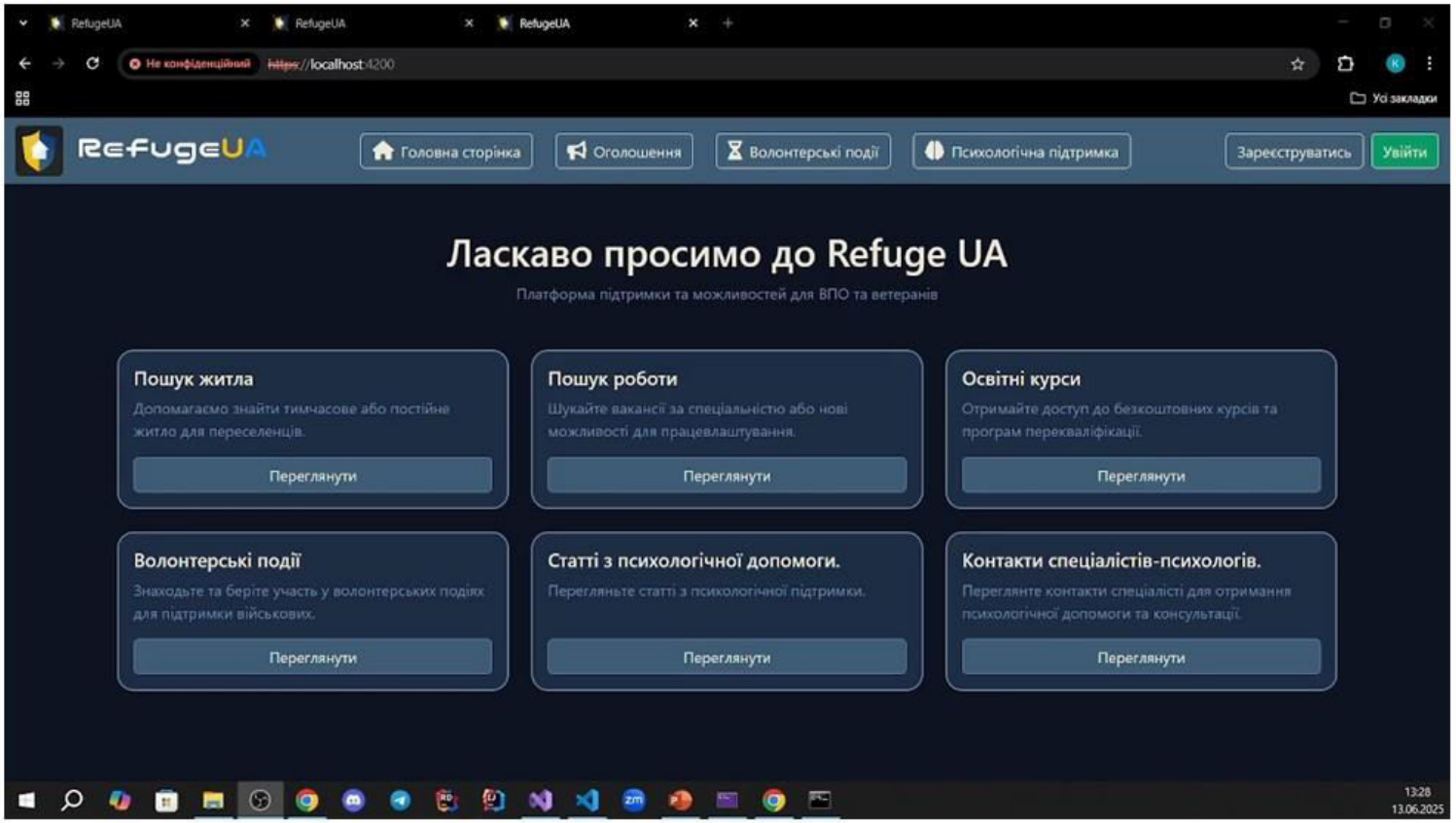


СТОРІНКА ПЕРЕГЛЯДУ РОЗДІЛУ ВОЛОНТЕРСЬКИХ ГРУП





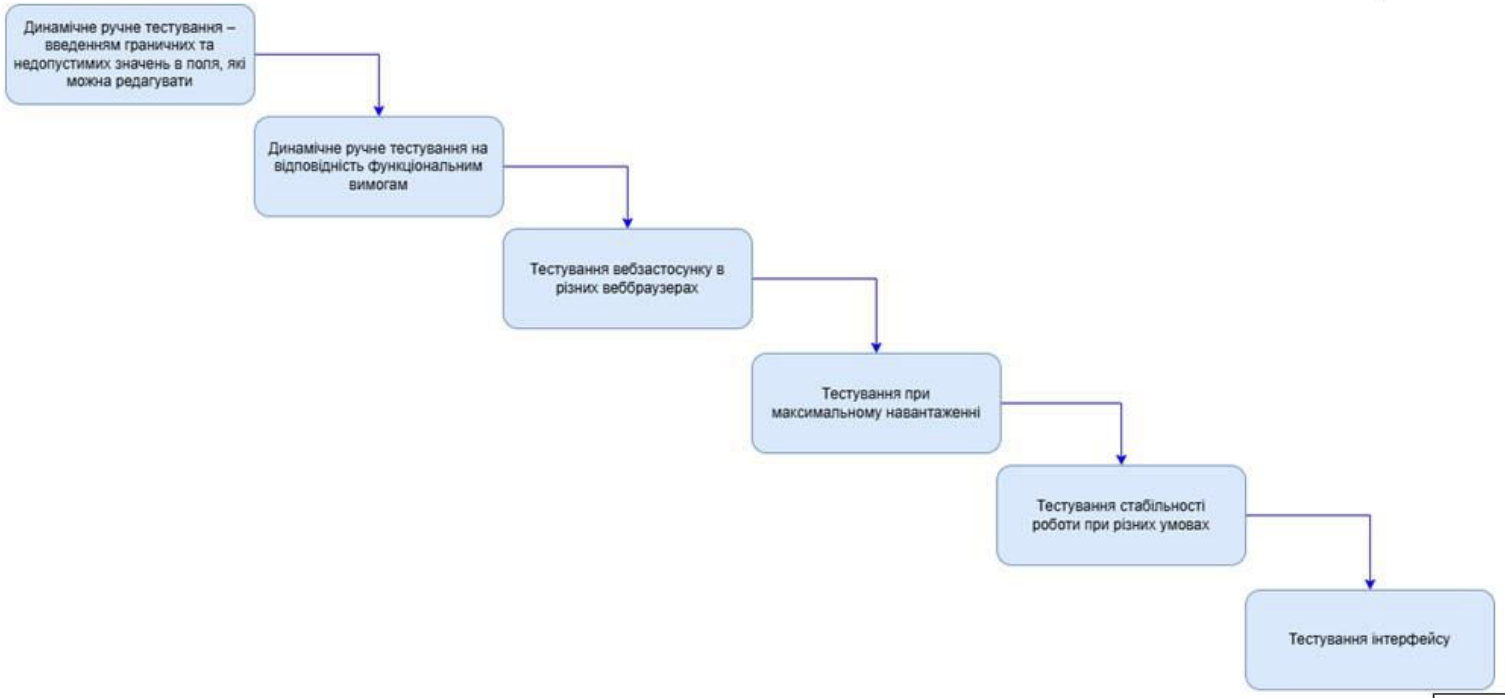
ВІДЕО КОРИСТУВАННЯ ВЕБЗАСТОСУНКОМ



КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Можливість перегляду інформації про консультаційні заходи та контакти психологів для звернення	Публікація оголошень про доступне житло для ветеранів	Створення оголошень про заходи від громади для підтримки ветеранів
Каталог навчальних закладів та програм перекваліфікації	Можливість розміщення вакансій і подання відгуків на них	Створення груп волонтерів та управління їх діяльністю
Публікація оголошень про волонтерські заходи		Можливість перегляду оголошень

ЗАСОБИ ТА ПОРЯДОК ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО ПЗ



РЕЗУЛЬТАТИ ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПЗ ТА АНАЛОГІВ

Критерій	Український фонд ветеранів	Veteran Hub	RefugeUA
Можливість перегляду інформації про консультаційні заходи та контакти психологів для звернення	+	+	+
Публікація оголошень про доступне житло для ветеранів	–	–	+
Можливість розміщення вакансій і подання відгуків на них	+/–	–	+
Каталог навчальних закладів та програм перекваліфікації	+/–	–	+
Створення груп волонтерів та управління їх діяльністю	–	–	+
Публікація оголошень про волонтерські заходи	+/–	–	+
Можливість перегляду оголошень	–	–	+
Створення оголошень про заходи від громади для підтримки ветеранів	–	–	+

ШЛЯХИ ПОДАЛЬШОГО РОЗВИТКУ ПРОЄКТУ

Створення календаря подій

Створення мультимовного інтерфейсу

Розробка стратегії поширення вебзастосунку за межами України

Розробка механізму нагадувань про майбутні події

Впровадження механізмів зворотного зв’язку

ВИСНОВКИ

1. Проведено аналіз потреб і труднощів, з якими стикаються військовослужбовці та члени їхніх родин.
2. Досліджено існуючі програмні рішення, виявлено їхні переваги та недоліки.
3. Проведено аналіз інструментів для реалізації застосунку й обрано найбільш відповідні.
4. Розроблено програмне забезпечення відповідно до архітектури та визначених вимог.
5. Здійснено тестування та порівняння розробленого ПЗ з аналогами за основними критеріями, підтверджено відповідність вимогам.
6. Визначено напрямки подальшого розвитку проєкту.

Дякую за увагу!

URL, ЛОГІНИ ТА ПАРОЛІ

URL: https://refugeuawebapi-bda2bzbneegxafhf.canadacentral-01.azurewebsites.net

Користувачі:

- owner111@example.com – Адміністратор громади.
- cool.mai11111@gmail.com - Військовий, член сім'ї військового.
- cool.mail@gmail.com – Волонтер.
- petzik.popovidsdsoch@gmail.com – Місцевий житель.
- petzik.popovich@gmail.com – Адміністратор.

Паролі для всіх однакові (латиниця, 8 символів) – **1234Aa__**

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ДОПОМОГИ ВІЙСЬКОВИМ В ЗАПАСІ ТА ЇХНІМ СІМ'ЯМ З ПОШУКОМ РОБОТИ, ЖИТЛА ТА МІСЦЯ НАВЧАННЯ, І ДОПОМОГИ ОРГАНІЗАЦІЇ ВОЛОНТЕРСЬКИХ ЗАХОДІВ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Костянтин ТКАЧЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Юрій ПОПОВИЧ

2024

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Вебзастосунок для допомоги військовим в запасі та їхнім сім’ям з пошуком роботи, житла та місця навчання, і допомоги організації волонтерських заходів, який являє собою вебсайт, створений з використанням СУБД Microsoft SQL Server та фреймоворків ASP.NET Core, Angular.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) можливість створення, редагування, видалення та перегляду оголошень стосовно навчання, житла, робота та волонтерских заходів.
2) можливість залишити відгук на оголошення;
3) можливість зареєструватись на участь у волонтерському заходів;
4) можливість переглянути власні події, на які зареєстрований;
5) можливість створення, редагування, видалення та перегляду статей з та контактів спеціалістів з психологічної підтримки;
6) можливість переглянути власні відгуки на оголошення;
7) можливість переглянути відгуки на своє оголошення;
8) можливість створення, редагування, видалення та перегляду волонтерських груп;
9) можливість приєднатись у волонтерську групу;
10) можливість переглянути свій профіль;
11) забезпечення належного рівня безпеки даних;
12) зручність роботи з вебзастосунком;
13) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.

Працездатність вебзастосунку перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування вебзастосунку в різних веббраузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ДОПОМОГИ ВІЙСЬКОВИМ В ЗАПАСІ ТА ЇХНІМ СІМ'ЯМ З ПОШУКОМ РОБОТИ, ЖИТЛА ТА МІСЦЯ НАВЧАННЯ, І ДОПОМОГИ ОРГАНІЗАЦІЇ ВОЛОНТЕРСЬКИХ ЗАХОДІВ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Костянтин ТКАЧЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Юрій ПОПОВИЧ

2025

ЗМІСТ

1. Опис структури вебзастосунку

Вебзастосунок для допомоги військовим в запасі та їхнім сім’ям з пошуком роботи, житла та місця навчання, і допомоги організації волонтерських заходів було розроблено у вигляді односторінкового вебзастосунку, який складається наступних розділів:

1) головна сторінка;
2) сторінка реєстрації та входу;
3) сторінка з оголошеннями;
4) сторінка оголошення;
5) сторінка створення, редагування оголошення;
6) сторінка відгуків на оголошення;
7) сторінка волонтерських подій;
8) сторінка волонтерської події;
9) сторінка створення, редагування волонтерської події;
10) сторінка волонтерських груп;
11) сторінка волонтерської групи;
12) сторінка створення, редагування волонтерської групи;
13) сторінка статей з психологічної підтримки;
14) сторінка статті з психологічної підтримки;
15) сторінка створення, редагування статті з психологічної підтримки;
16) сторінка контактів психологів;
17) сторінка контакту психолога;
18) сторінка створення, редагування контакту психолога;
19) сторінка профілю користувача;
20) сторінка з своїми оголошеннями;
21) сторінка модерації оголошень;
22) сторінка модерації користувачів та заявок на реєстрацію;
23) сторінка груп оголошення;
24) сторінка з подіями, в яких користувач бере участь.

Сторінки редагування та створення об’єктів є однаковими, тобто мають одну й ту саму форму, з тією різницею, що на сторінці редагування поля вже заповнені даними, які користувач бажає редагувати. Сторінки редагування доступно відповідно автору створеної сутності або адміністраторам. Сторінки модерації оголошень та користувачів доступні лише адміністраторам.

2. Головна сторінка

Після входу на вебзастосунок для підтримки військових користувач потрапляє на головну сторінку (рис. 1). У верхній частині інтерфейсу розташована навігаційна панель, яка містить кнопки-посилання: «Головна сторінка», «Оголошення», «Волонтерські події» та «Психологічна підтримка». Кожна з них перенаправляє користувача до відповідного розділу.

Праворуч від навігації розміщена панель авторизації. Якщо користувач ще не авторизований, відображаються кнопки «Зареєструватись» (перехід до форми реєстрації) та «Увійти» (перехід до сторінки входу). У випадку успішної авторизації замість них показується кнопка «Вийти», яка дозволяє завершити сеанс. Ця навігаційна панель доступна на всіх сторінках вебзастосунку.

На головній сторінці розміщені інформаційні картки, кожна з яких містить короткий опис розділу та кнопку «Переглянути» для переходу. Наприклад, картка «Пошук житла» веде до розділу з оголошеннями категорії «Житло», «Пошук роботи» – до оголошень про працевлаштування, а «Освітні курси» – до відповідного розділу з інформацією про навчальні можливості.

Крім того, картки з заголовками «Волонтерські події», «Статті з психологічної підтримки» та «Контакти спеціалістів-психологів» ведуть до відповідних тематичних розділів.

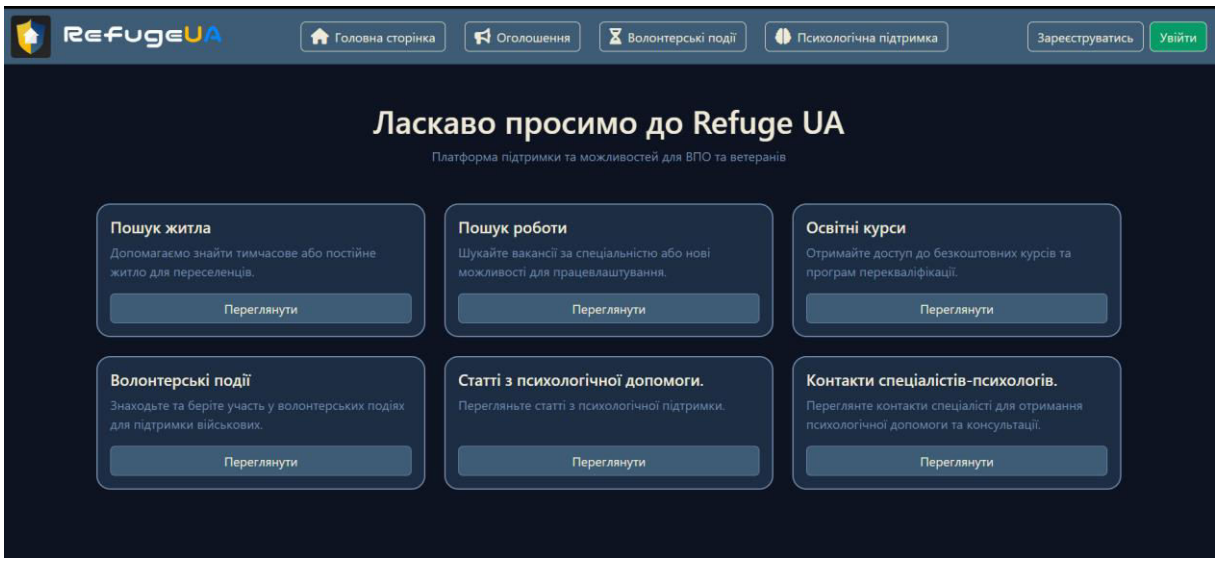


Рис. 1. Головна сторінка анонімного користувача

Залежно від ролі авторизованого користувача, на сторінці з'являється додаткова навігаційна панель. Вона містить посилання на розділи, доступ до яких має лише користувач із відповідною роллю.

В системі є 5 ролей: «Адміністратор», «Представник громади», «Волонтер», «Місцевий житель» та «Військовий, член сім'ї військового».

Усі навігаційні панелі користувачів містять спільні елементи – посилання «Мій профіль» та «Мої групи».

«Мій профіль» дає змогу перейти на сторінку власного профілю, де користувач може переглянути особисту інформацію та за потреби відредагувати її, наприклад змінити ім'я, контактні дані або пароль.

«Мої групи» відкриває сторінку зі списком волонтерських груп, у яких бере участь користувач.

Користувачі з ролями «Адміністратор» та «Представник громади» бачать однакові пункти навігаційної панелі (рис. 2), однак кожне з посилань веде до розділів, адаптованих відповідно до їхньої ролі. Наприклад, у розділі «Модерація оголошень» для користувача з роллю «Представник громади» відображаються лише ті оголошення, які були створені із зазначенням адреси, що належить до його громади. Посилання «Користувачі та реєстрація» дозволяє перейти в розділ модерації користувачів та заявок на

реєстрацію. Посилання «Модерація оголошення» дозволяє перейти на сторінку з списком оголошень, який містить можливості для дозволу та заборони публікації оголошення інших користувачів. Посилання «Мої оголошення» дозволяє перейти на сторінку з списком власних оголошень. Посилання «Мої створені подій» дозволяє перейти на сторінку з списком власних створених подій. Посилання «Мої події» дозволяє перейти на сторінку з списком подій, на які користувач зареєструвався.

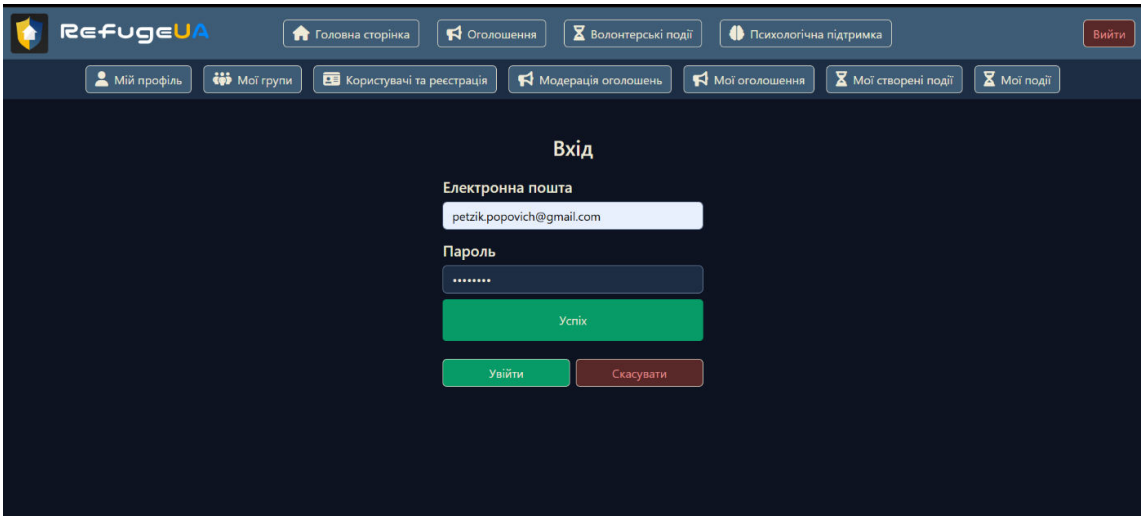


Рис. 2. Навігаційна панель адміністратора та представника громади

Навігаційна панель місцевого жителя (рис. 3) містить посилання, які вже було описані вище.

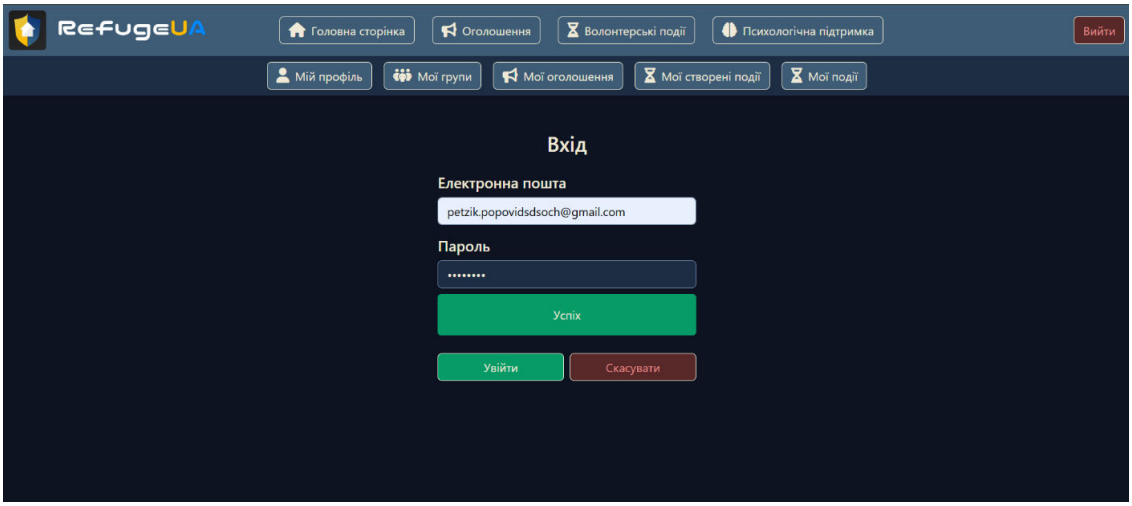


Рис. 3. Навігаційна панель місцевого жителя

Навігаційна панель військового або члена сім’ї військового (рис. 4) містить одне унікальне посилання, а саме «Мої відгуки», яке дозволяє перейти на сторінки з власними відгуками на оголошення. Всі інші посилання описано вище.

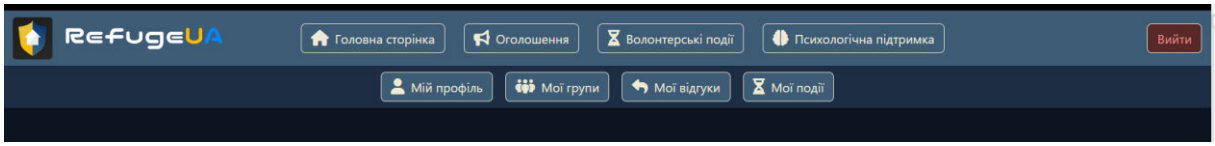


Рис. 4. Навігаційна панель військового або члена сім’ї військового

3. Процедура авторизації користувачів

Щоб зареєструватися в системі, користувачу необхідно перейти на сторінку з формою реєстрації, натиснувши на посилання «Зареєструватись», яке розташоване на верхній навігаційній панелі.

На сторінці реєстрації (рис. 5) користувач заповнює всі необхідні поля: ім’я, прізвище, статус (роль), дата народження, електронна пошта, номер телефону, громада (необов’язково), пароль, підтвердження паролю. Залежно від правильності введених даних стає доступною кнопка «Зареєструватись». Після натискання, у разі успішного заповнення форми, відображається повідомлення про успішну реєстрацію (рис. 6).

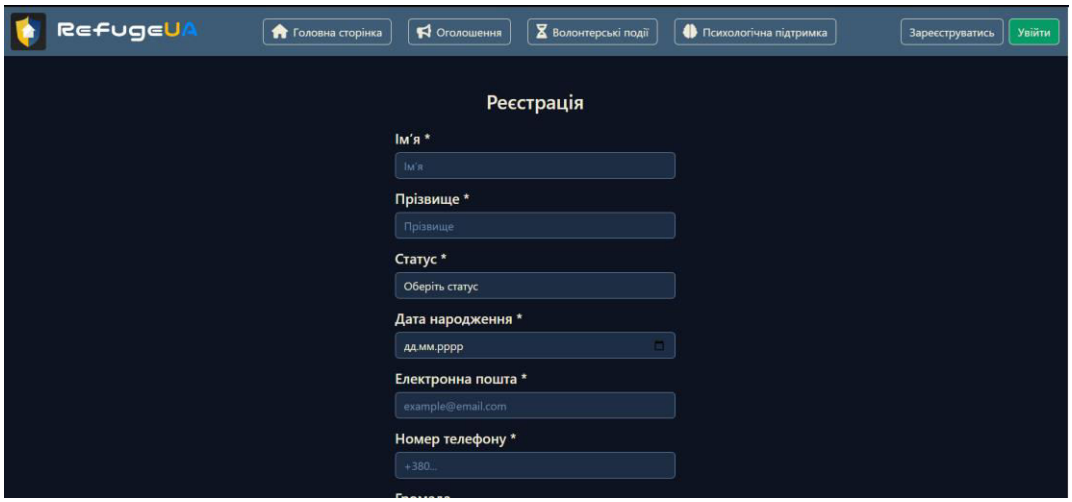


Рис. 5. Форма для реєстрації

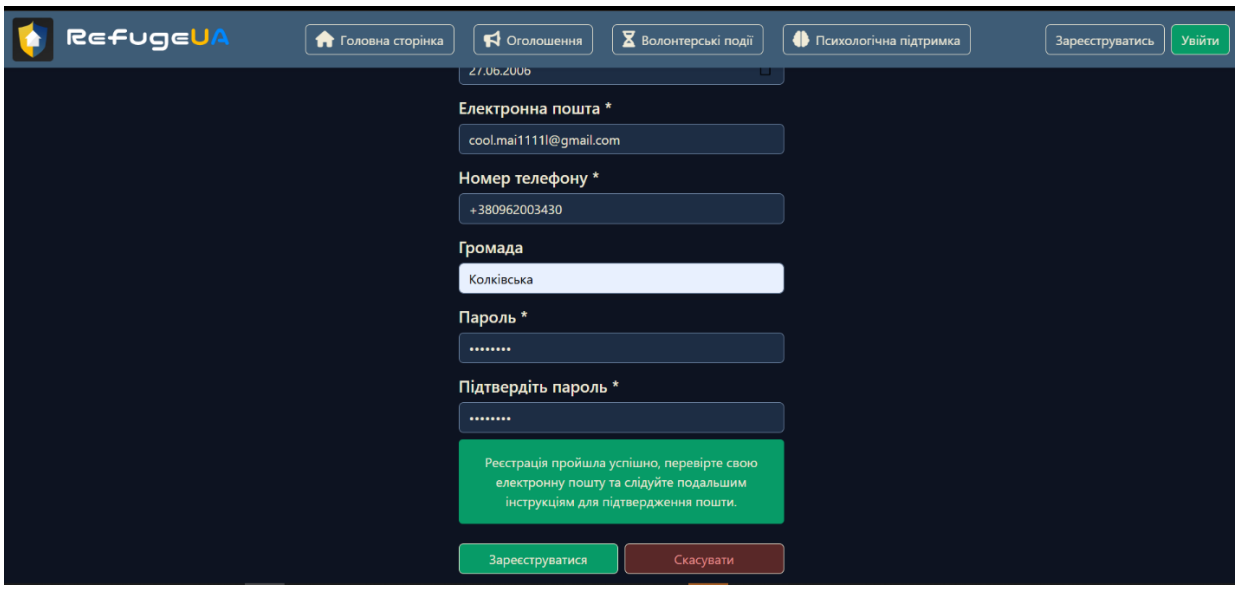


Рис. 6. Повідомлення про успішну реєстрацію

Щоб увійти в систему, користувач повинен спочатку підтвердити свою електронну пошту через надіслане посилання. Після цього адміністратор має схвалити його реєстрацію. Лише після підтвердження пошти та схвалення заявки користувач зможе авторизуватись через сторінку входу, доступну за посиланням «Увійти» на навігаційній панелі.

Для входу користувач вводить свою пошту, пароль та натискає на кнопку «Увійти», після чого виводиться повідомлення про успішність входу в систему (рис. 7).

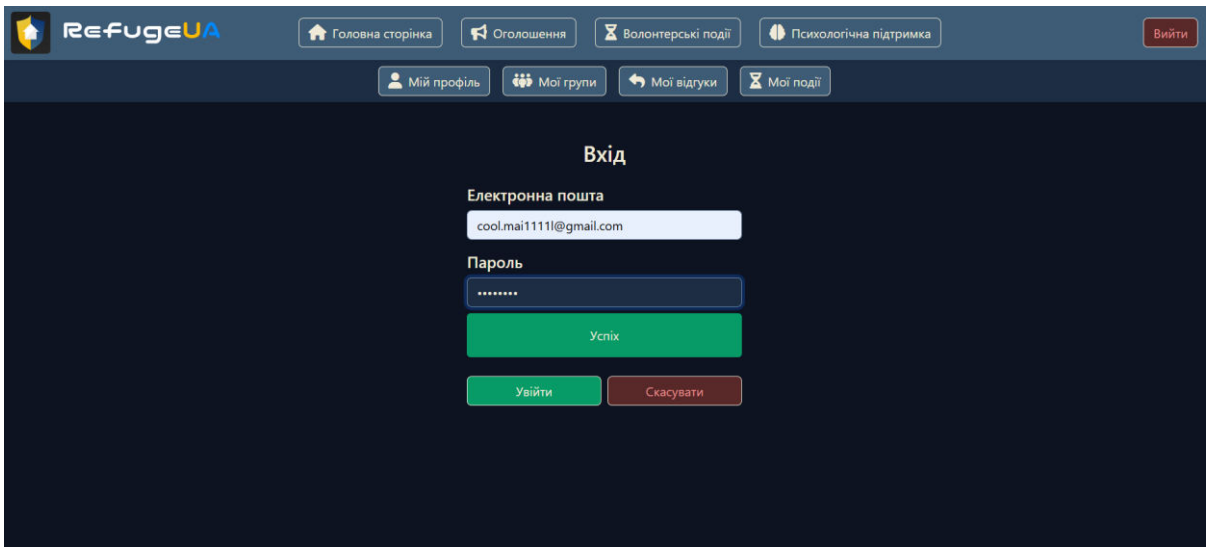


Рис. 7. Вхід в систему

4. Сторінка з оголошеннями

Для переходу до розділу з оголошеннями, користувач натискає на посилання «Оголошення». Для перегляду оголошень непотрібно бути авторизованим в системі, але деякі операції будуть недоступними.

Після переходу до відповідного розділу вебзастосунок спочатку відображає (рис. 8) порожню сторінку з трьома кнопками, що відповідають категоріям оголошень: «Житло», «Навчання» та «Робота». Щоб переглянути оголошення, користувачу необхідно натиснути на одну з цих кнопок. Після цього з’являється інтерфейс з пошуковим рядком, панеллю фільтрів, списком оголошень та нумерацією сторінок для навігації.

Рис. 8. Сторінка оголошень

Кожна панель фільтрів на сторінках оголошень відповідної категорії містить два основні поля для фільтрації: «Група оголошень», що дозволяє вибрати тематичну або функціональну категорію всередині основної, та «Громада», яка дає змогу обмежити результати лише оголошеннями, що стосуються певної територіальної громади.

На сторінці вкладки «Робота» (рис. 9) наявні також такі фільтри як: зарплата (від та до), зарплата не вказана та категорії (сфери) роботи.

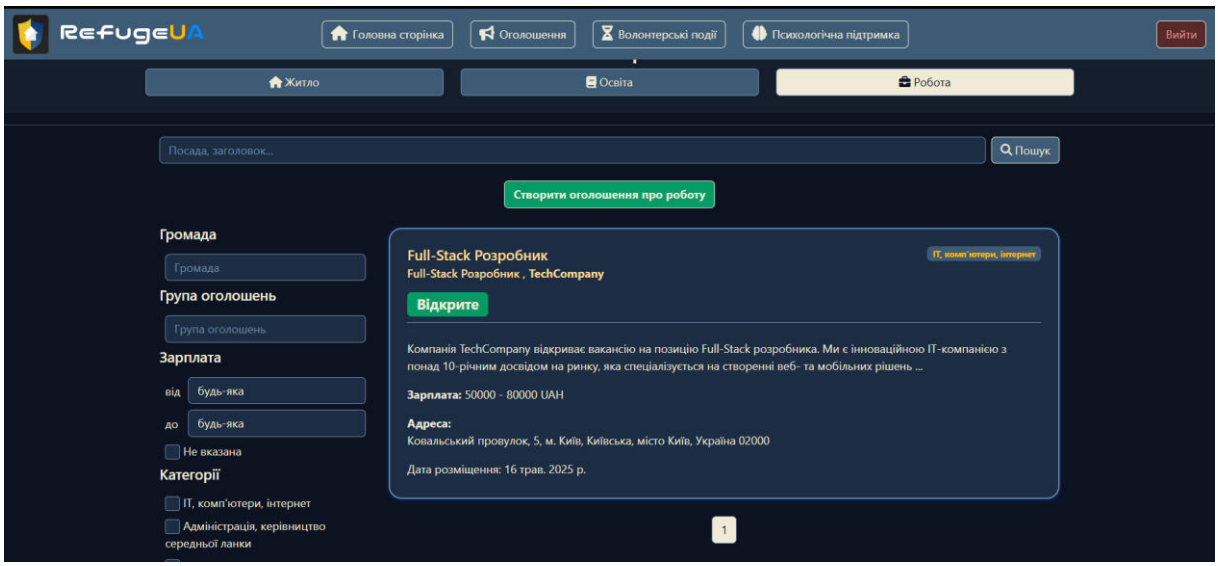


Рис. 9. Сторінка з оголошеннями про роботу

На сторінці вкладки «Житло» (рис. 10) наявні фільтри: оплата (від-до), лише безплатно, площа (від-до), місткість, поверхи, кількість кімнат, тип будівлі та чи дозволено домашні улюбленці.

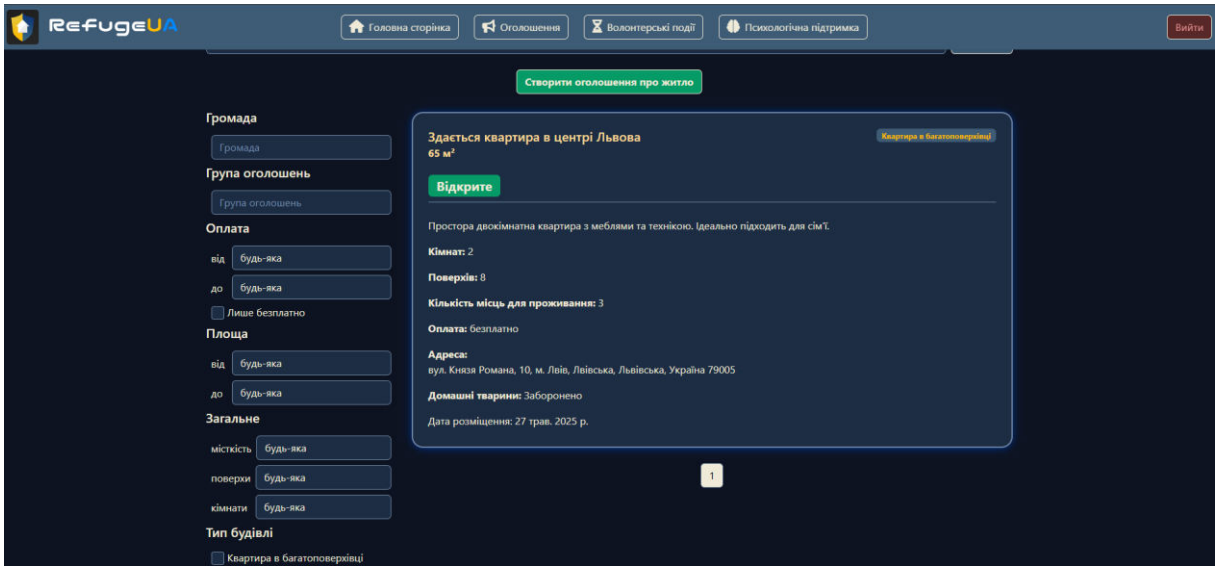


Рис. 10. Сторінка з оголошеннями про житло

На сторінці вкладки «Освіта» (рис. 11) наявні фільтри: оплата (від-до), лише безплатно, тривалість (від-до), тип освіти та цільові групи.

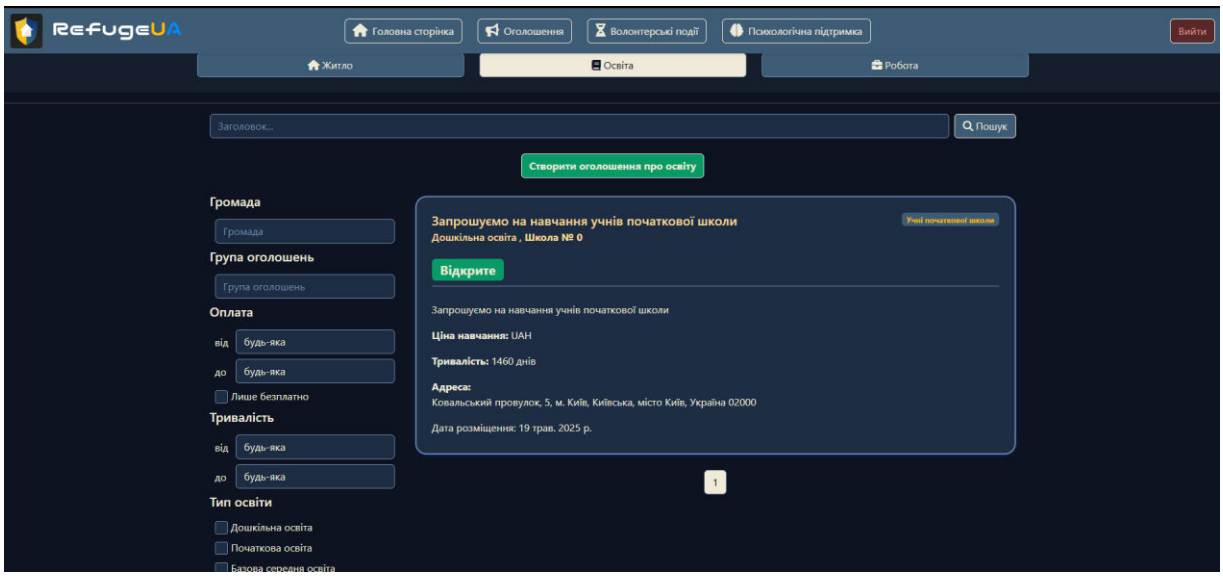


Рис. 11. Сторінка з оголошеннями про освіту

5. Сторінка оголошення

Для детального перегляду оголошення користувач може натиснути на картку попереднього перегляду у списку, після чого відкривається сторінка з повною інформацією про вибране оголошення.

На цій сторінці відображаються заголовок, загальний опис, дата створення, адреса (якщо вона вказана) та контактні дані особи або організації, що розмістила оголошення. Окрім основної інформації, в залежності від категорії оголошення, сторінка може містити додаткові специфічні дані.

Зокрема, для оголошень про роботу виводяться вимоги до кандидата, розмір заробітної плати (якщо вказано), посада, назва компанії та сфера її діяльності.

Для освітніх оголошень вказується тривалість навчання, форма оплати (у разі платного навчання), тип освіти, назва навчального закладу та цільова аудиторія.

У житлових оголошеннях зазначаються площа житла, тип будівлі, кількість кімнат, максимальна кількість мешканців, поверховість, наявність дозволу на проживання з домашніми тваринами та зображення житла.

Наведемо приклад деталей оголошення про роботу (рис. 12-14).

Рис. 12. Оголошення про роботу

Залежно від того, чи є користувач адміністратор або автором оголошення, він може виконувати додаткові дії, такі як перегляд відгуків на оголошення, редагування, перегляд груп оголошення, закриття або відкриття оголошення, видалення, відхилення або прийняття оголошення (рис. 12-14). Якщо оголошення відхилене, адміністратор може вказати причину відхилення у відповідному полі, після чого користувачу відображається повідомлення з поясненням причини відхилення.

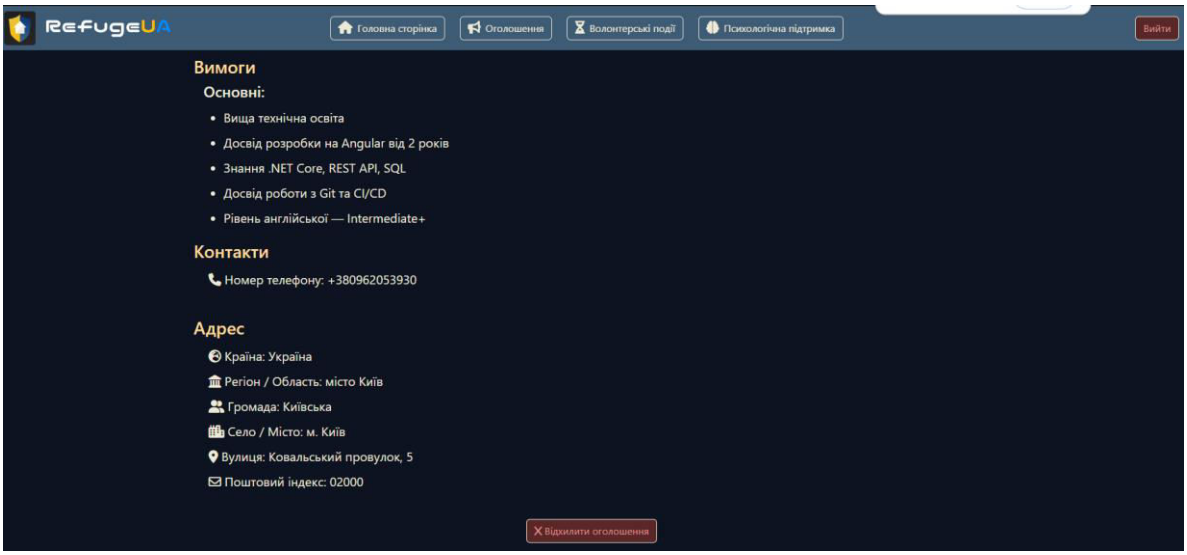


Рис. 13. Деталі оголошення, коли оголошення прийняте

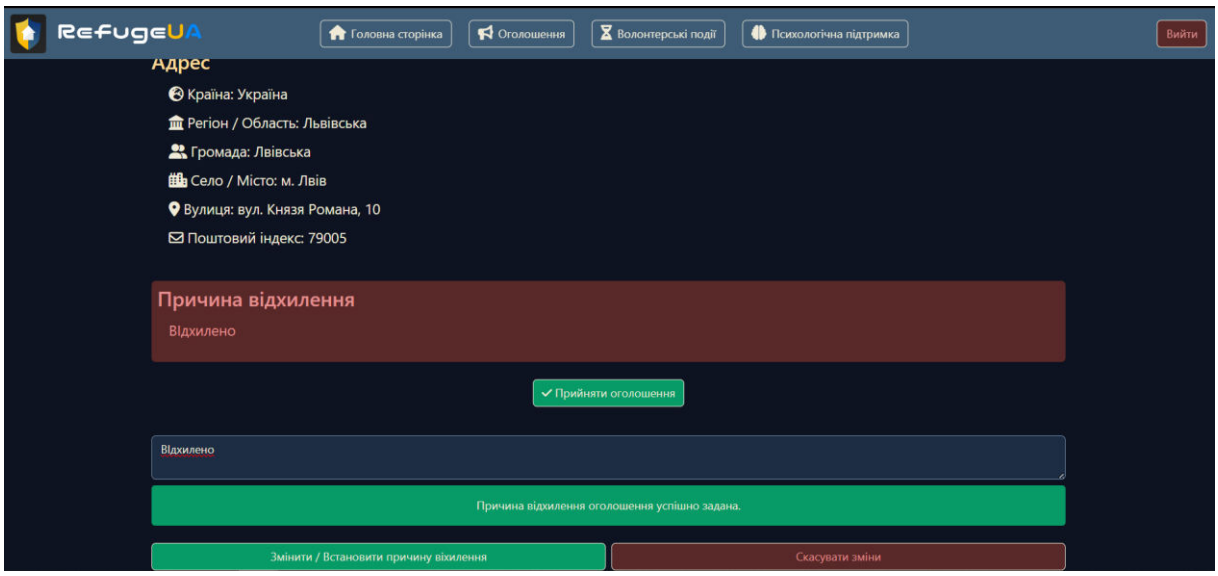


Рис. 14. Деталі оголошення, коли оголошення відхилене

Якщо користувач має статус військового або члена родини військового, йому відкривається доступ до форми відгуку, розміщеної внизу сторінки оголошення (рис. 15). У цій формі користувач може ввести свої контактні дані, після чого натиснути кнопку «Лишити відгук», щоб надіслати повідомлення автору оголошення. Користувач за бажанням може також видалити надісланий відгук (рис. 16).

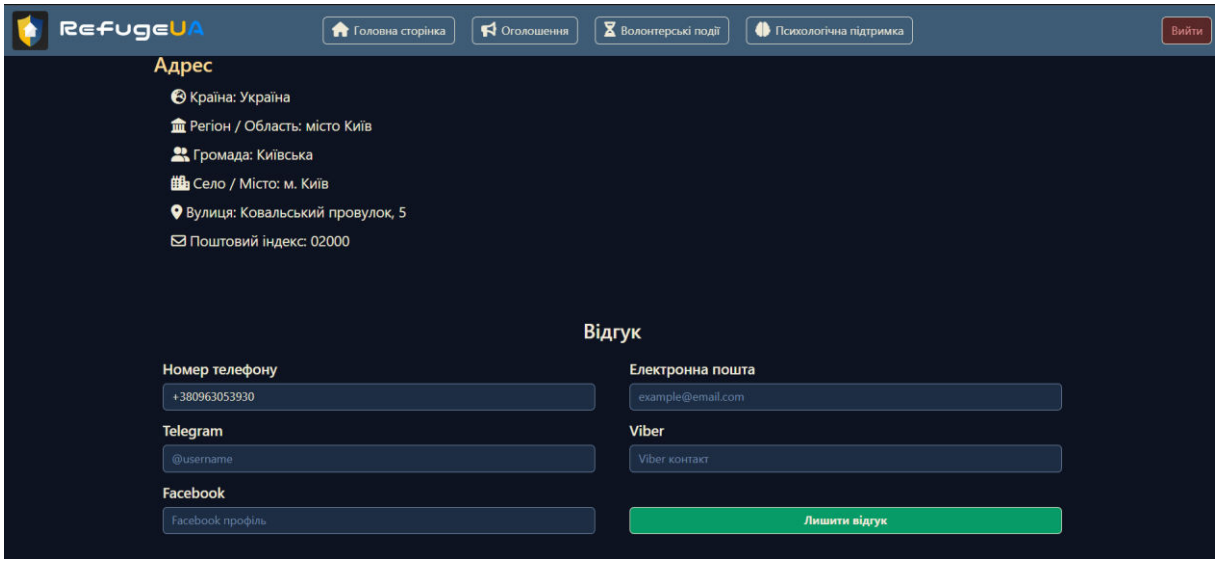


Рис. 15. Форма для відправки відгуку на оголошення

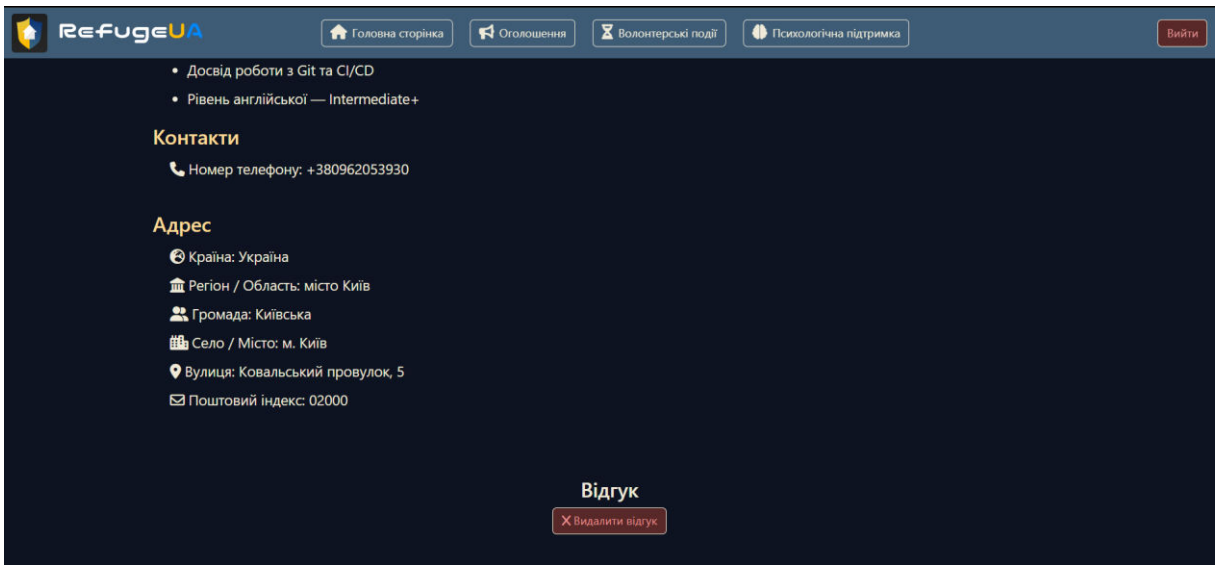


Рис. 16. Кнопка «Видалити відгук» на сторінці оголошення

6. Сторінка створення, редагування оголошення

Для створення оголошення про роботу, житло або навчання, адміністратор, волонтер або місцевий житель має натиснути кнопку «Створити оголошення про [тип оголошення]» у відповідній вкладці категорії оголошень. Після цього користувач буде перенаправлений на сторінку з формою створення обраного типу оголошення. У кожній формі обов’язково потрібно ввести заголовок, опис оголошення, адресу (рис. 22) та контактну інформацію (рис. 21). Усі інші поля відображаються та заповнюються відповідно до типу оголошення.

Для оголошення про роботу заповнюються такі полі (рис. 17): посада, назва підприємства, мінімальна зарплата, максимальна зарплата, вимоги та категорія (сфера) роботи.

Для оголошення про освіту заповнюються такі полі (рис. 18-19): назва закладу, тип освіти, мова введення навчання, тривалість в днях, цільові групи, оплата.

Для оголошення про житло заповнюються такі полі (рис. 20): тип будівлі, поверхи, кількість кімнат, кількість мешканців, площа, оплата та фото житла.

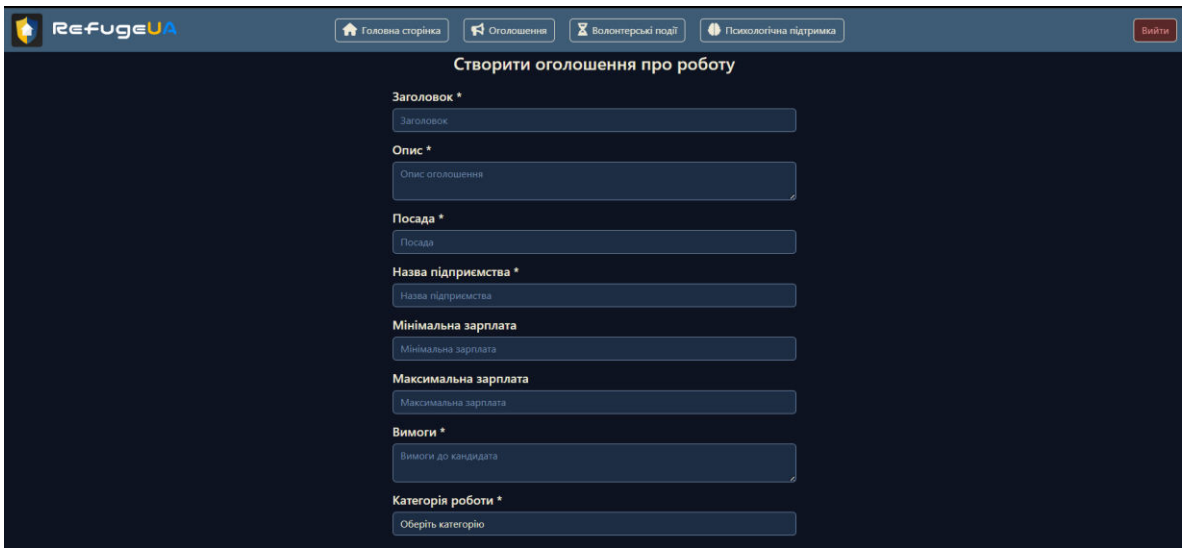


Рис. 17. Поля форми створення оголошення про роботу

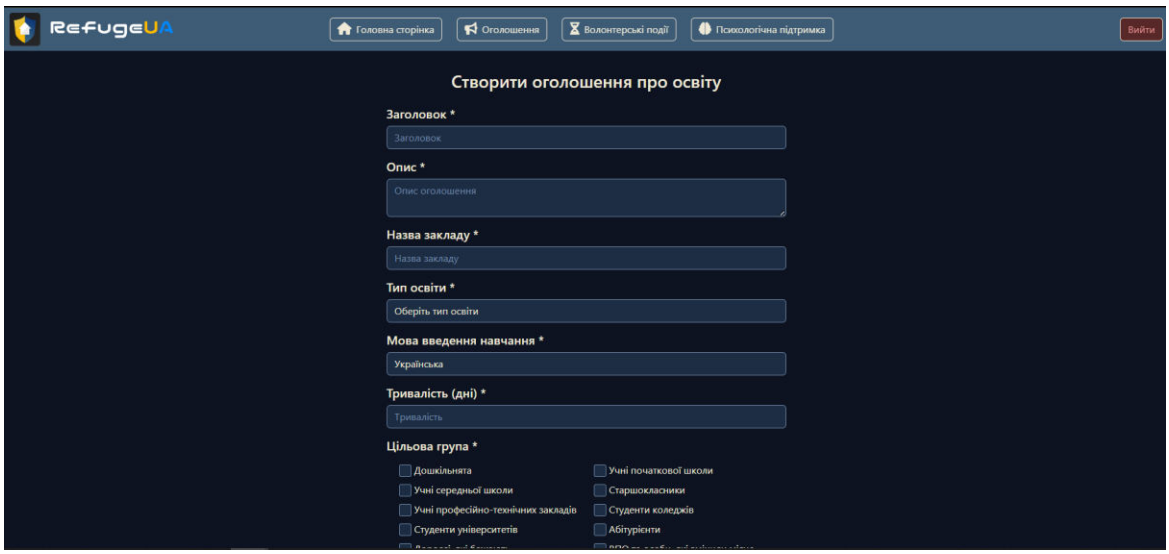


Рис. 18. Поля форми створення оголошення про освіту

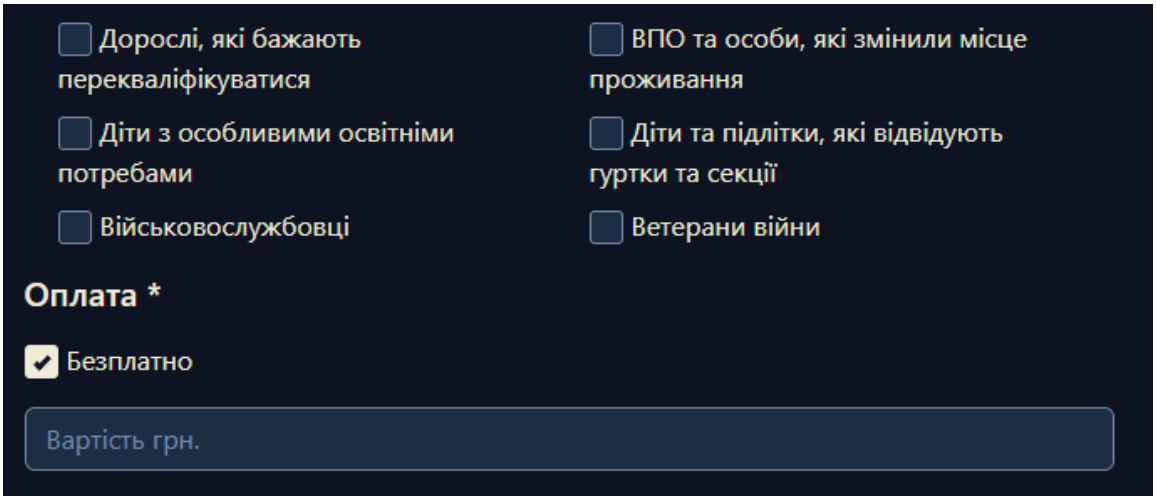


Рис. 19. Продовження форми створення оголошення про освіту

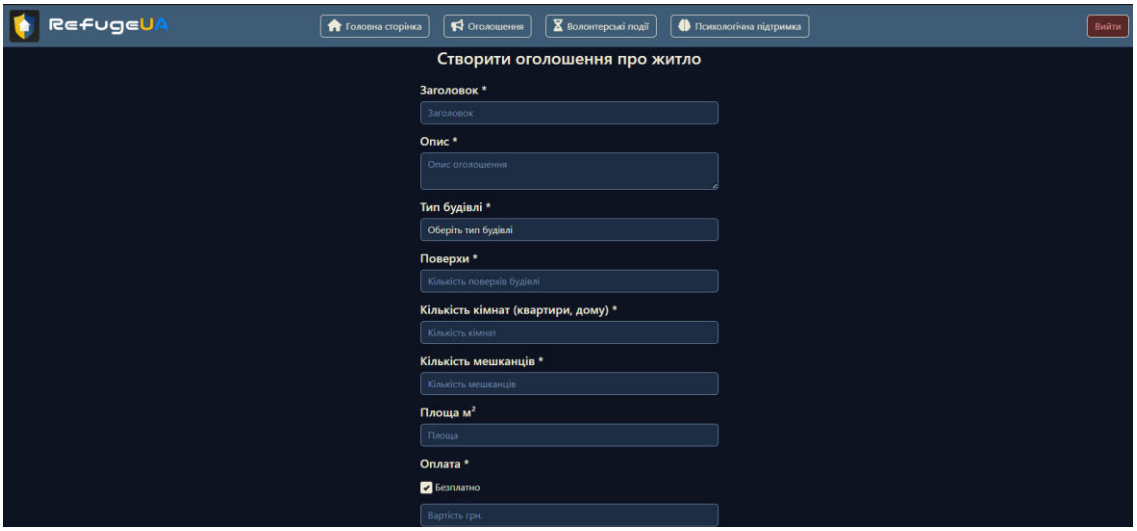


Рис. 20. Поля форми створення оголошення про житло

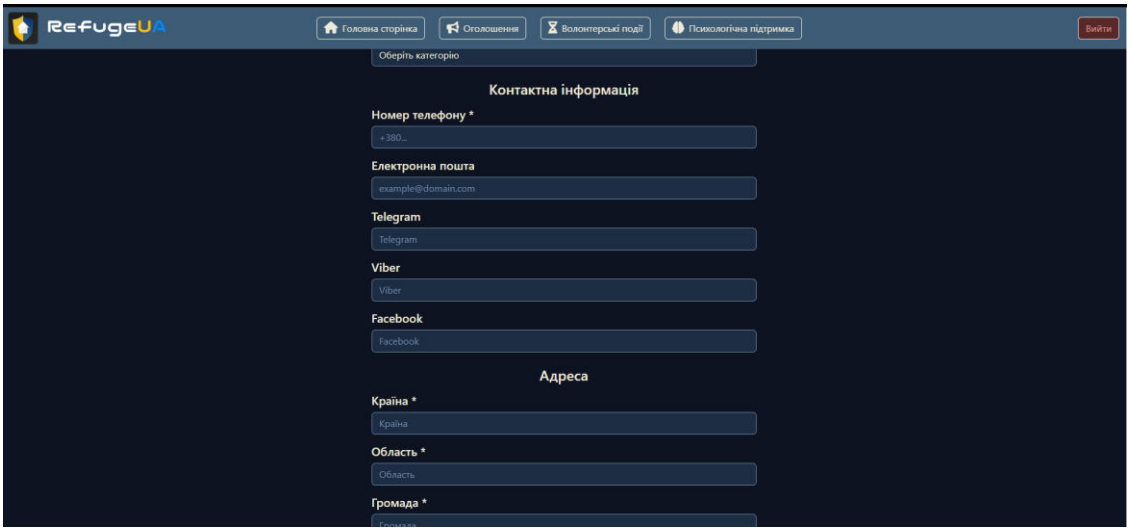


Рис. 21. Поля контактної інформації оголошення

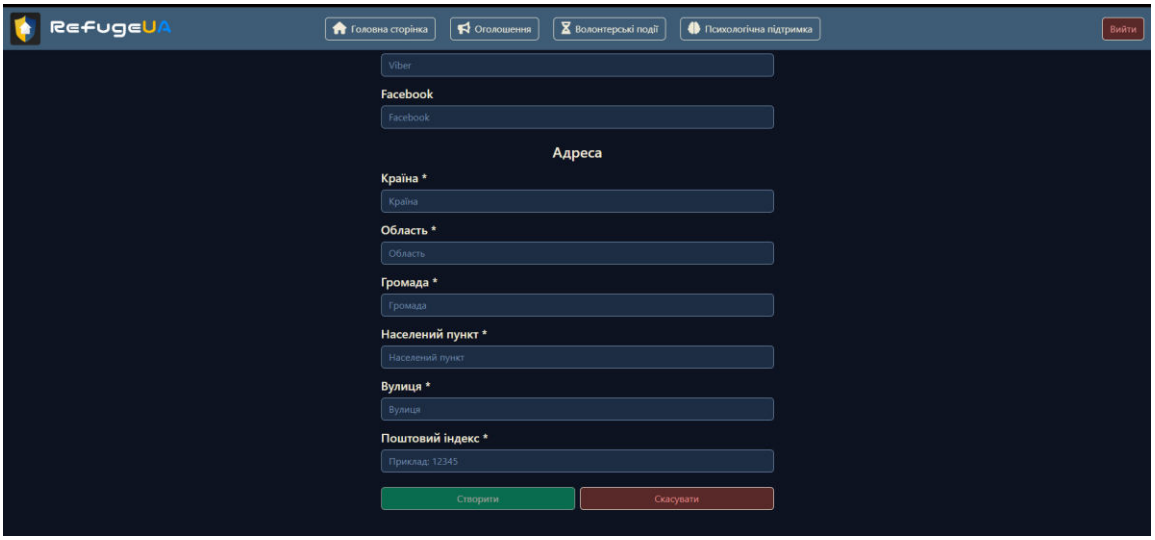


Рис. 22. Поля адреси, кінець форми

Для переходу на сторінку редагування оголошення користувачу необхідно натиснути кнопку «Редагувати», яка розміщується зверху опису оголошення. Сторінка редагування є ідентичною сторінці створення оголошення, за винятком того, що всі поля у формі вже автоматично заповнені відповідно до даних редагованого оголошення. Користувач може змінити будь-яку з цих інформацій та зберегти оновлену версію.

7. Сторінка перегляду відгуків на оголошення

Щоб переглянути відгуки на оголошення, автор оголошення або адміністратор може натиснути на посилання «Переглянути відгуки», яке знаходиться перед описом оголошення. Після натискання користувач буде перенаправлений на окрему сторінку, де відображається список усіх відгуків, залишених на це оголошення. На сторінці з відгуками відображаються контактні дані осіб, які залишили відгук, а також дата та час його створення. Крім того, на сторінці доступний пошуковий рядок, що дозволяє здійснювати пошук відгуків за повним ім’ям або контактними даними користувачів (рис. 23).

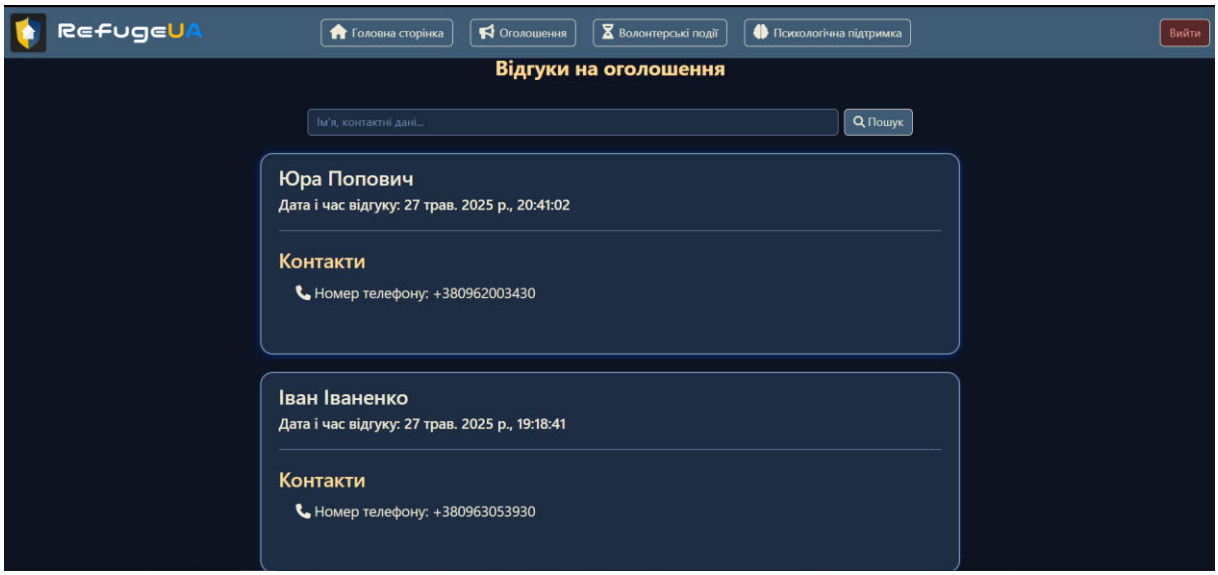


Рис. 23. Сторінка з відгуками на оголошення

Щоб переглянути свої відгуки користувачу необхідно перейти на відповідну сторінку натиснувши на посилання «Мої відгуки» на панелі навігації.

На сторінці з відгуками виводиться пошуковий рядок для фільтрації, а також список відгуків. Кожен відгук містить контактні дані користувача, дату та час залишення, а також посилання на відповідне оголошення, до якого цей відгук був залишений (рис. 24).

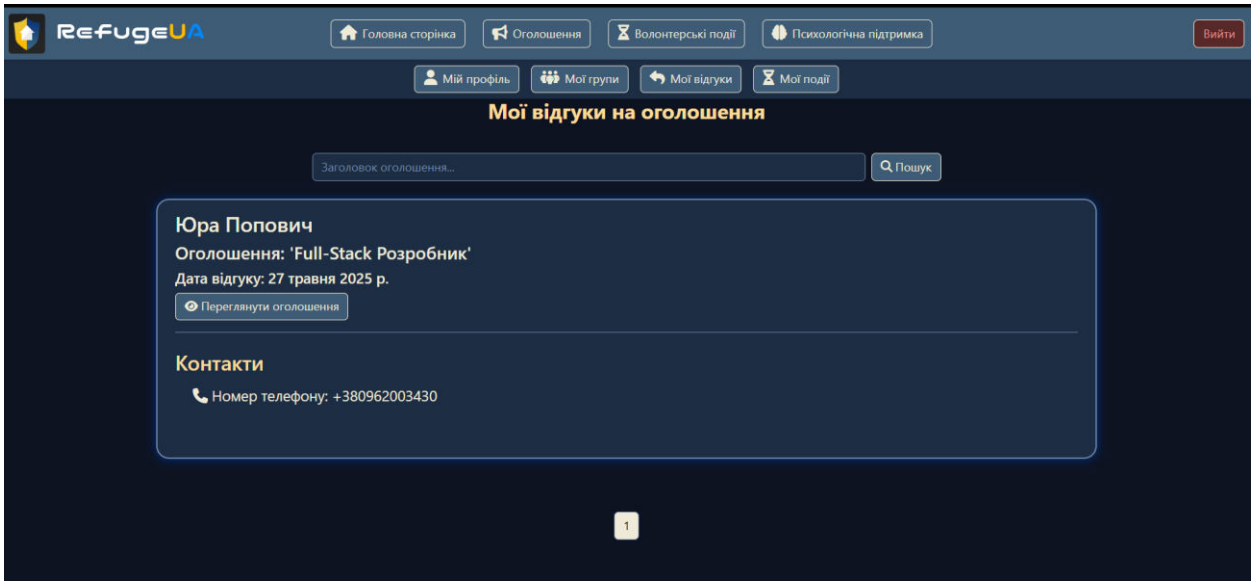


Рис. 24. Список з власними відгуками

8. Сторінка волонтерських подій

Для переходу до розділу з оголошеннями про волонтерство, користувач натискає на посилання «Волонтерські події» у навігаційній панелі. Після цього він переходить до розділу волонтерства, який містить дві вкладки: «Волонтерські події» та «Волонтерські групи». Щоб переглянути список доступних подій, користувачу необхідно натиснути на вкладку «Волонтерські події», після чого відображається перелік заходів (рис. 25).

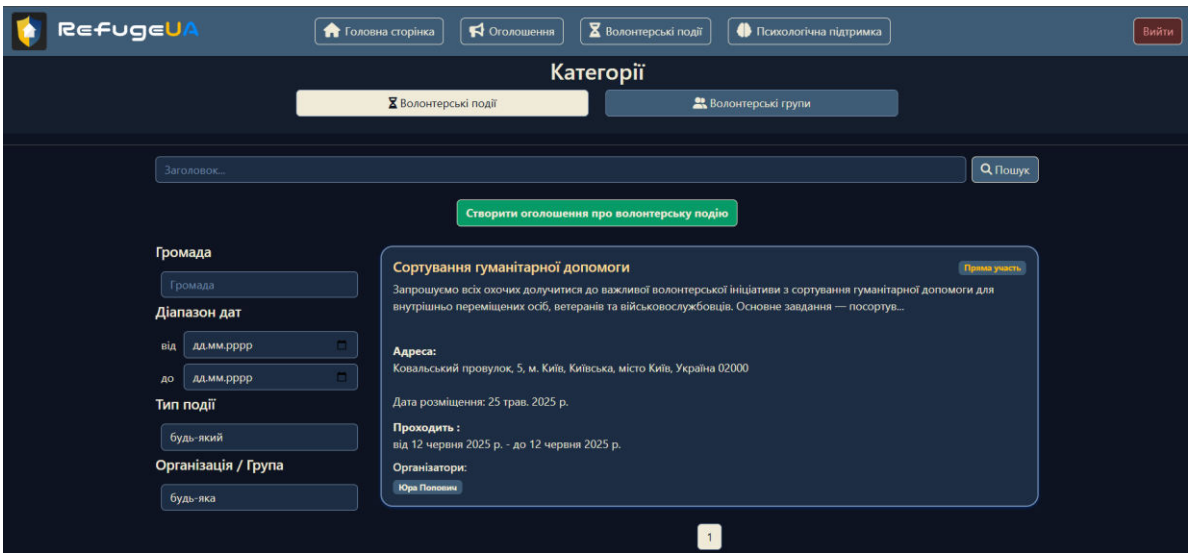


Рис. 25. Список з волонтерськими подіями

На сторінці волонтерських заходів користувач бачить пошуковий рядок для швидкого пошуку подій за ключовими словами та панель фільтрів, яка містить поля громада, діапазон дат проведення заходу, тип події (пряма участь пожертвування) та група, до якої належить оголошення про проведення волонтерського заходу. Після зміни значень форми, пошук здійснюється автоматично. Також наявні нумерації сторінок для навігації внизу панелі для виведення оголошення про волонтерські події.

9. Сторінка волонтерської події

Для детального перегляду волонтерського заходу, користувач може натиснути на картку попереднього перегляду заходу зі списку та перейти на сторінку заходу, де виводиться заголовок, загальний опис, дата створення, адреса, при наявності посилання на конференцію, дати проходження (від та до) та розклад (рис. 26-27).

Залежно від того, чи є користувач адміністратор або автором оголошення, він може виконувати додаткові дії, такі як редагувати, закрити подію та видалити її. Для цього доступні відповідні кнопки «Редагувати»,

«Закрити подію» та «Видалити». Сторінка з переглядом учасників доступна всім користувачам.

Для переходу на цю сторінку є відповідна кнопка «Учасники».

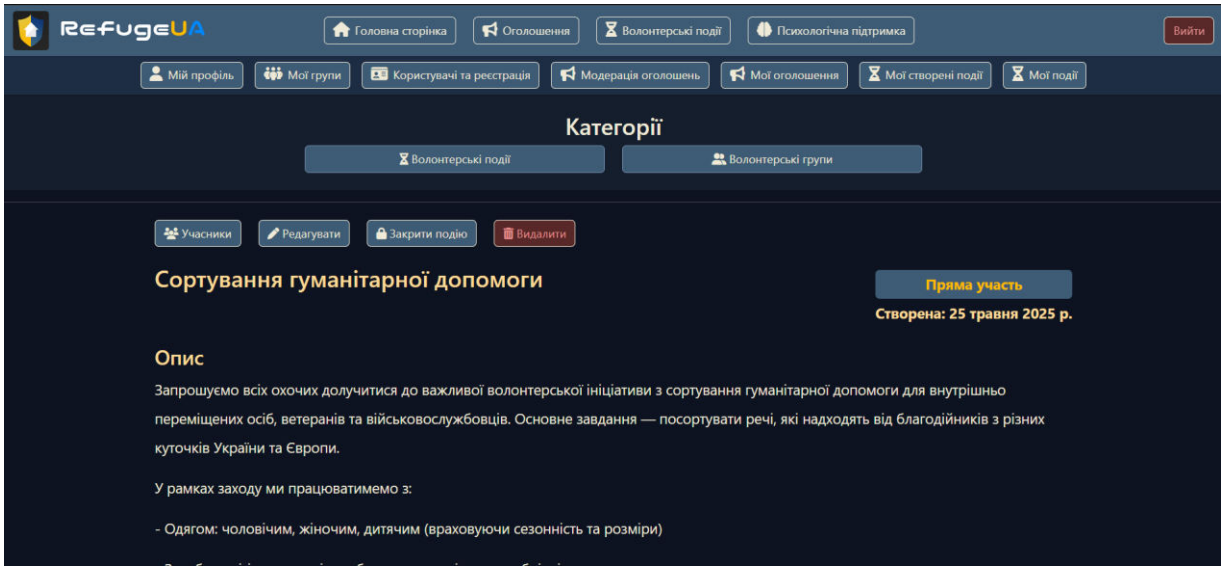


Рис. 26. Сторінка оголошення волонтерського заходу 1

Рис. 27. Сторінка оголошення волонтерського заходу 2

Користувач може взяти участь в заході, натиснувши на кнопку «Взяти участь» внизу сторінки (рис. 28).

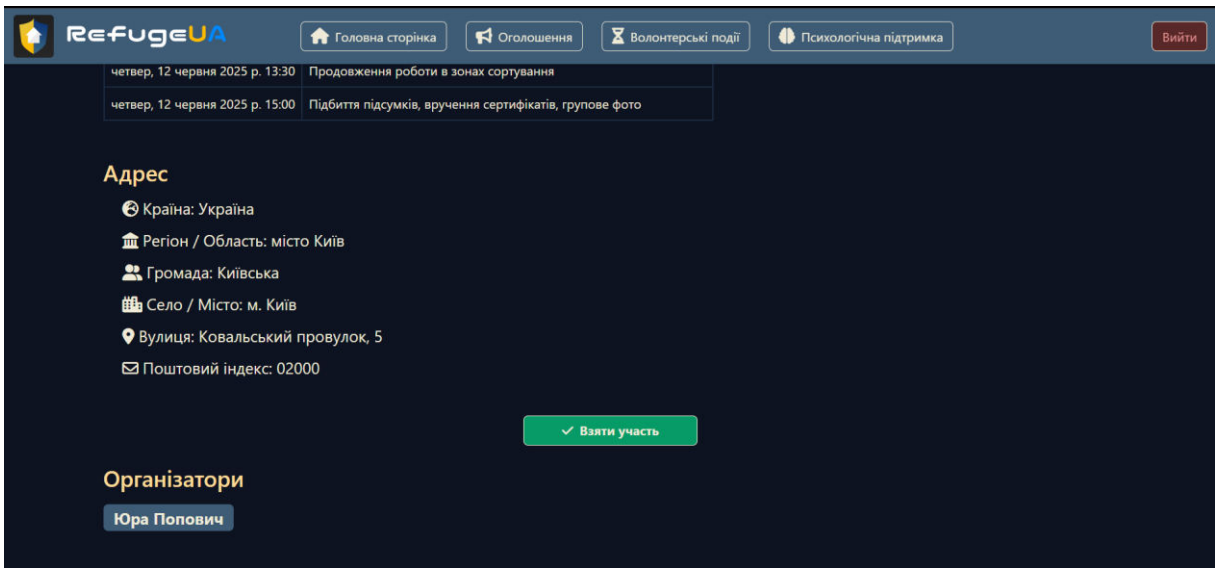


Рис. 28. Кнопка «Взяти участь» на сторінці волонтерського заходу

10. Сторінка створення, редагування волонтерської події

Для створення оголошення про волонтерський захід волонтер, місцевий житель або представник громади має натиснути кнопку «Створити оголошення про волонтерську поді» у вкладці категорії «Волонтерські події». Після цього користувач буде перенаправлений на сторінку з формою створення оголошення про подію (рис. 29).

У цій формі необхідно ввести заголовок, опис, тип події (пряма участь пожертвування), волонтерську групу, дати початку та кінця. За потреби користувач може додати розклад до волонтерського заходу. Для цього необхідно натиснути кнопку «Додати розклад», після чого з’явиться новий пункт розкладу, у якому потрібно вказати час початку та короткий опис запланованої активності. Користувач може додати кілька таких пунктів для створення повного розкладу заходу (рис. 30).

Залежно від типу обраної події, а саме пожертвування та пряма участь, на формі з’являються відповідні необов’язкові поля для заповнення – «Посилання на онлайн конференцію» (рис. 32) та «Посилання на пожертвування» (рис. 31).

Користувач за потреби може вказати адресу проведення волонтерського заходу. Для цього у формі передбачено прапорець «Включити адресу». Після його активації з’являються відповідні поля, у які потрібно ввести дані адреси проведення заходу (рис. 33).

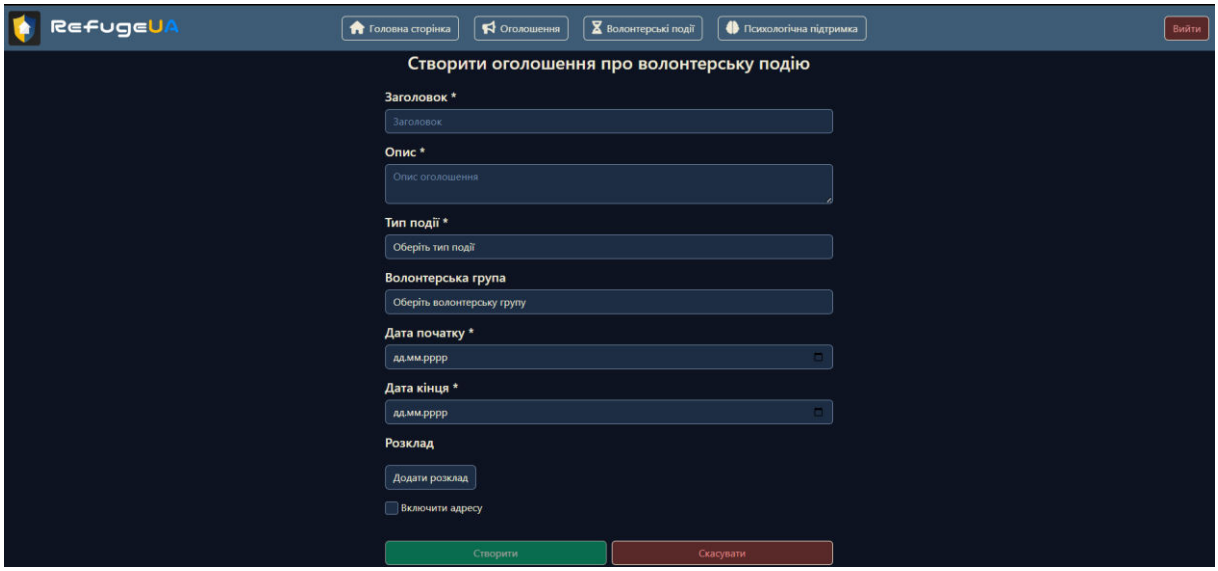


Рис. 29. Сторінка з формою для створення оголошення про волонтерський захід

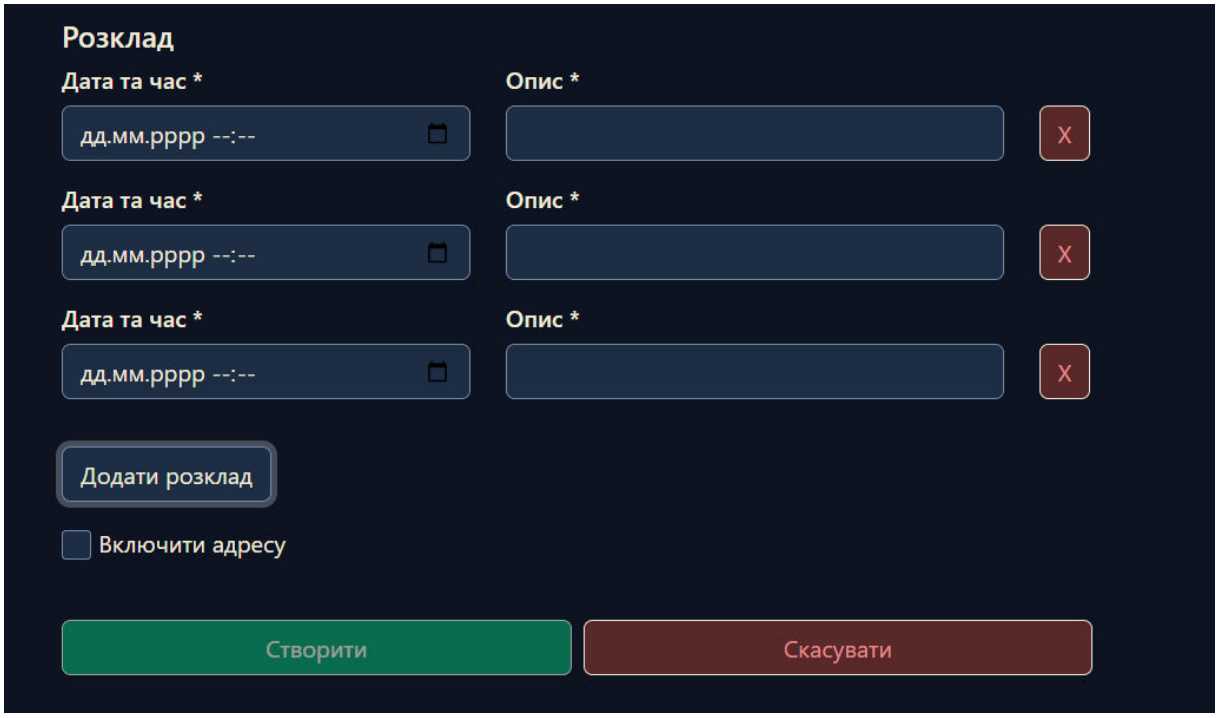


Рис. 30. Форма для створення оголошення про волонтерську подію, заповнення розкладу

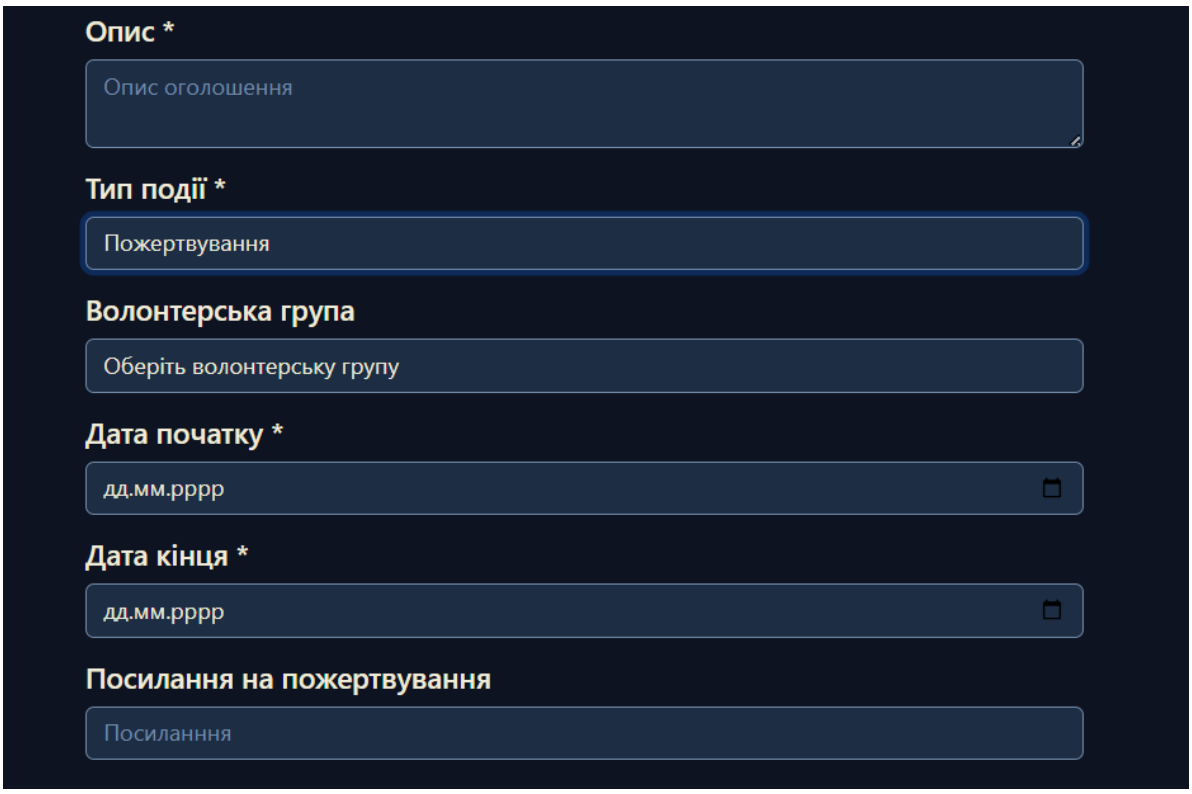


Рис. 31. Форма для створення оголошення про волонтерську подію, обраний тип події «Пожертвування»

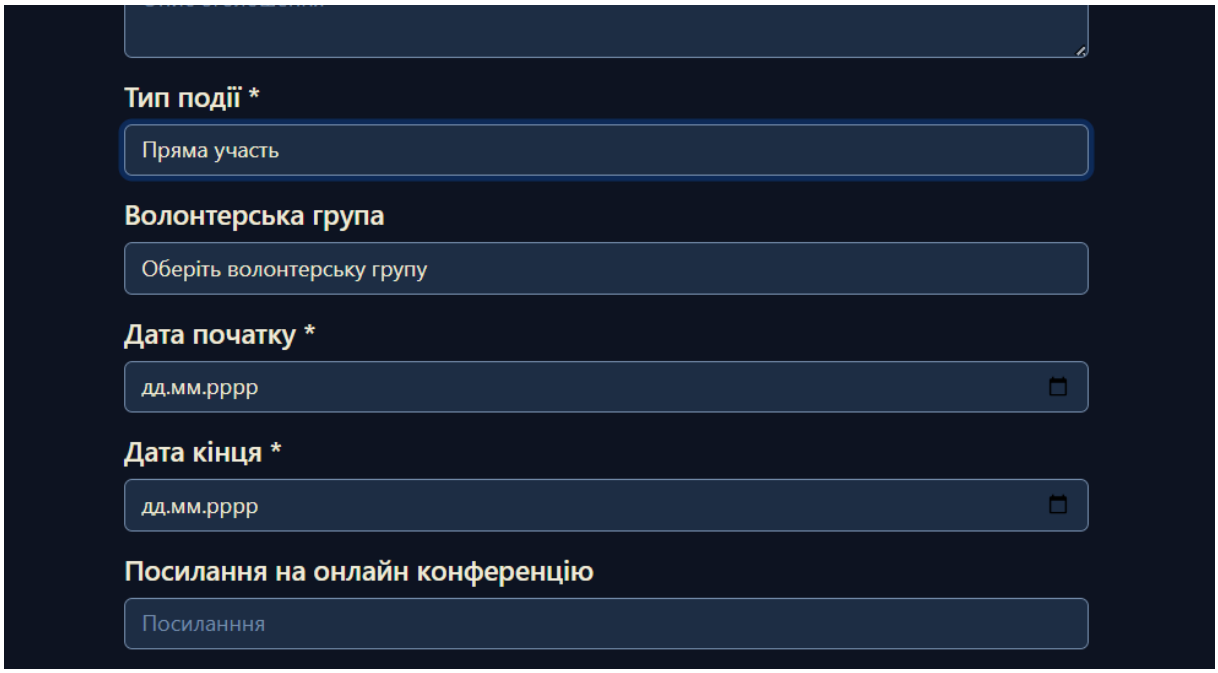


Рис. 32. Форма для створення оголошення про волонтерську подію, обраний тип події «Пряма участь»

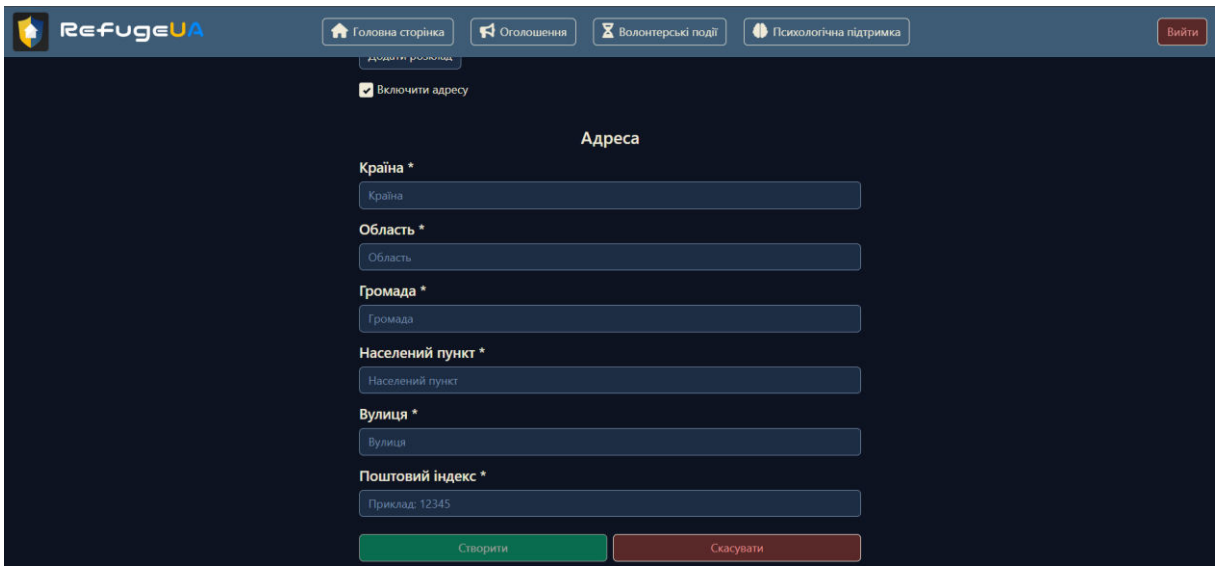


Рис. 33. Форма для створення оголошення про волонтерську подію, заповнення адреси

Для переходу на сторінку редагування оголошення про волонтерську подію користувачу необхідно натиснути кнопку «Редагувати», яка розміщується зверху опису оголошення про волонтерську подію. Сторінка редагування є ідентичною сторінці створення оголошення, за винятком того, що всі поля у формі вже автоматично заповнені відповідно до даних редагованого оголошення. Користувач може змінити будь-яку з цих інформацій та зберегти оновлену версію.

11. Сторінка волонтерських груп

Для переходу до розділу з оголошеннями про волонтерство, користувач натискає на посилання «Волонтерські події» у навігаційній панелі. Після цього він переходить до розділу волонтерства, який містить дві вкладки: «Волонтерські події» та «Волонтерські групи». Щоб переглянути список доступних волонтерських груп, користувачу необхідно натиснути на вкладку «Волонтерські групи», після чого відображається перелік груп. Також з’являється інтерфейс з пошуковим рядком з пошуком

за назвою групи та нумерації сторінок для навігації внизу панелі для виведення волонтерських груп (рис. 34).

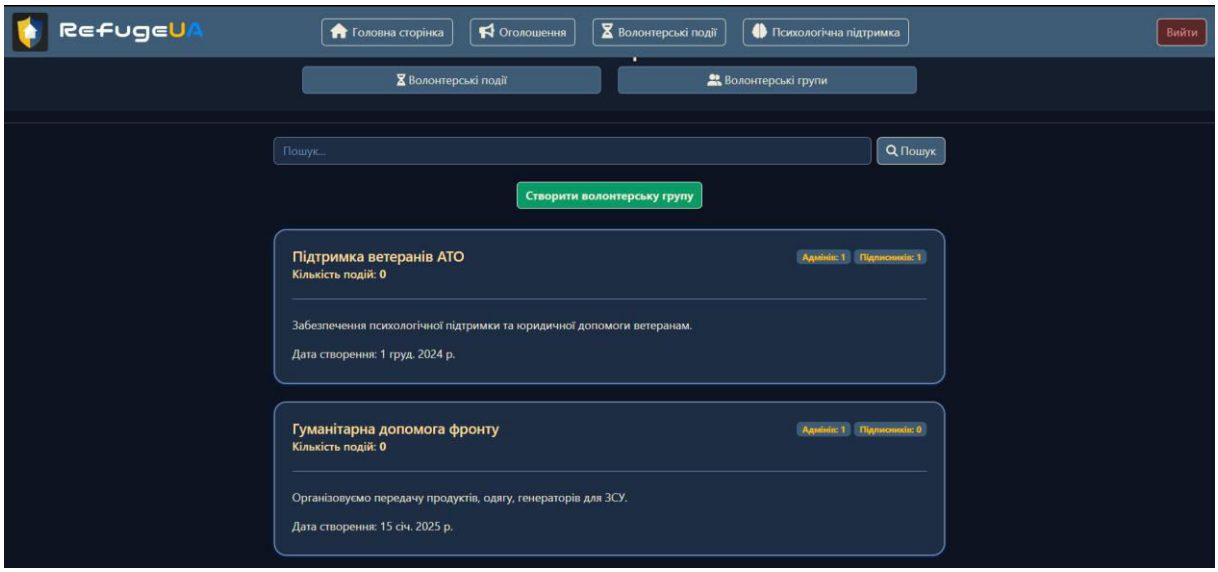


Рис. 34. Сторінка волонтерських груп

12. Сторінка волонтерської групи

Для детального перегляду волонтерської групи, користувач може натиснути на картку попереднього перегляду групи зі списку та перейти на сторінку групи (рис. 35), де виводиться заголовок, загальний опис та дата створення.

Якщо користувач є адміністратором, йому доступні кнопки «Редагувати» та «Видалити» для виконання відповідних дій з оголошенням. Окрім цього, він також має доступ до кнопок «Адміністратори», «Учасники» та «Заходи групи», які дозволяють переглядати список адміністраторів групи, її учасників, а також усі заходи, організовані цією групою.

Користувачу також доступна кнопка «Приєднатись», при натисканні на яку користувач стає учасником групи.

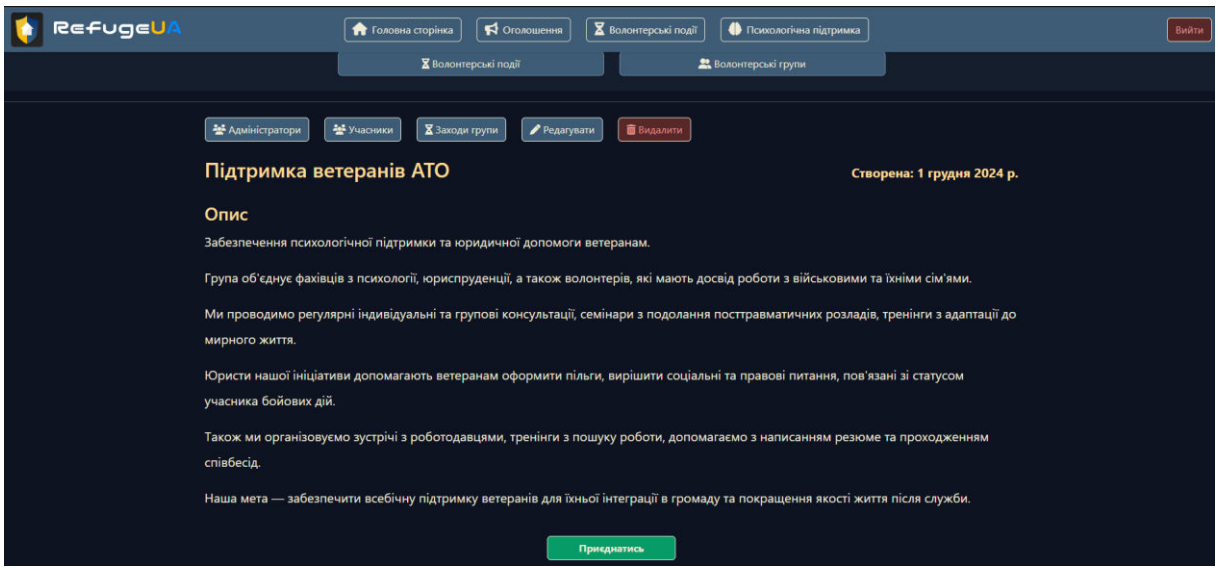


Рис. 35. Сторінка групи

13. Сторінка створення, редагування групи

Для створення волонтерської групи волонтер або адміністратор має натиснути кнопку «Створити волонтерську групу» на сторінці волонтерських груп. Після цього користувач буде перенаправлений на сторінку з формою створення волонтерської групи (рис. 36).

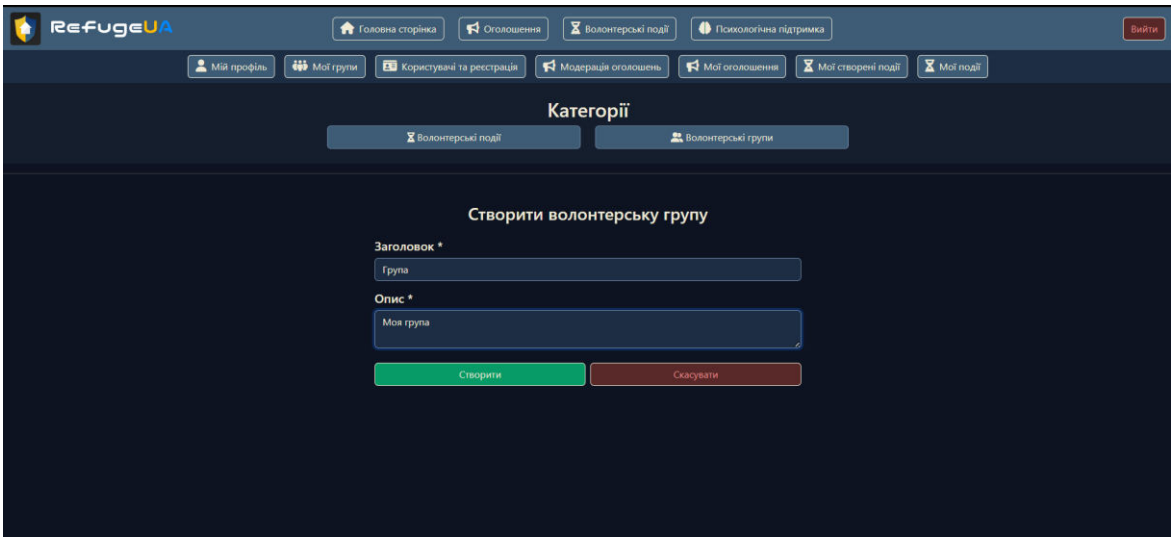


Рис. 36. Сторінка створення групи

На сторінці форми потрібно заповнити необхідну інформацію про групу: заголовок та опис. Після введення всіх необхідних даних, користувачу стає доступна кнопка «Створити», після натискання на яку виводиться повідомлення про успіх створення групи.

Для переходу на сторінку редагування волонтерської групи користувачу необхідно натиснути кнопку «Редагувати», яка розміщується зверху опису групи. Сторінка редагування є ідентичною сторінці створення, за винятком того, що всі поля у формі вже автоматично заповнені відповідно до даних редагованого оголошення. Користувач може змінити будь-яку з цих інформацій та зберегти оновлену версію.

14. Сторінка статей з психологічної підтримки

Для переходу до розділу з матеріалами психологічної підтримки необхідно натиснути кнопку «Психологічна підтримка». Після цього користувач буде перенаправлений на відповідну сторінку, де відображаються дві вкладки: «Статті» та «Контакти спеціалістів». Після натискання вкладки «Статті» відбувається виведення всіх статей з психологічної підтримки (рис. 37) та пошуковий рядок для пошуку статей за заголовком.

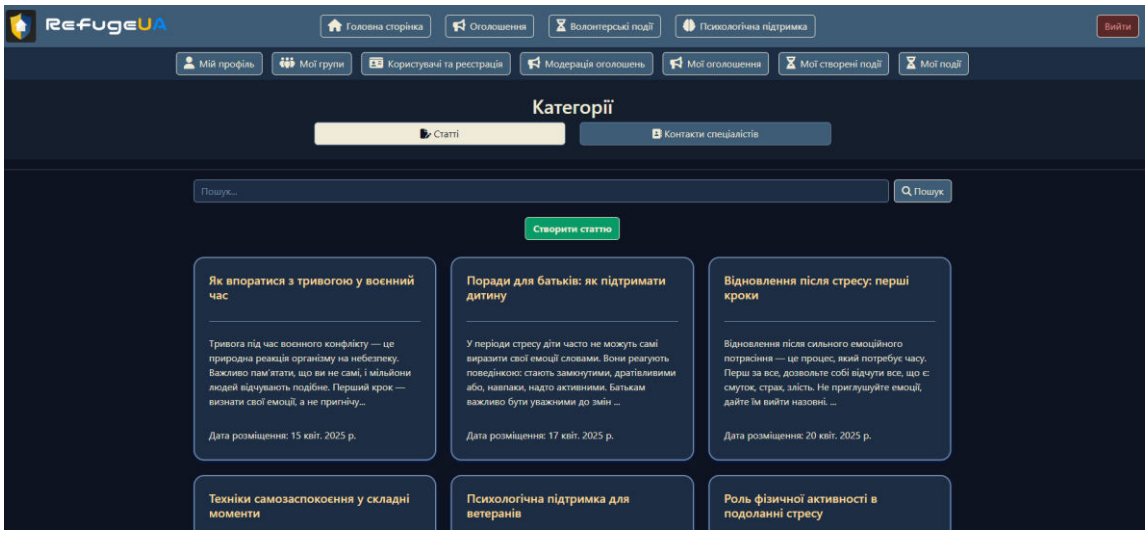


Рис. 37. Виведення списку статей

15. Сторінка статті з психологічної підтримки

Для перегляду статті необхідно натиснути на картку попереднього перегляду зі списку статей з психологічної підтримки та сторінці «Статті». Після натискання відбувається перехід на сторінку огляду статі, де користувач може її прочитати (рис. 38).

Сторінка містить заголовок, опис та дату створення статті. Якщо користувач є адміністратором, йому доступні кнопки «Редагувати» та «Видалити», які дозволяють відповідно перейти на сторінку редагування статті або видалити її зі списку публікацій.

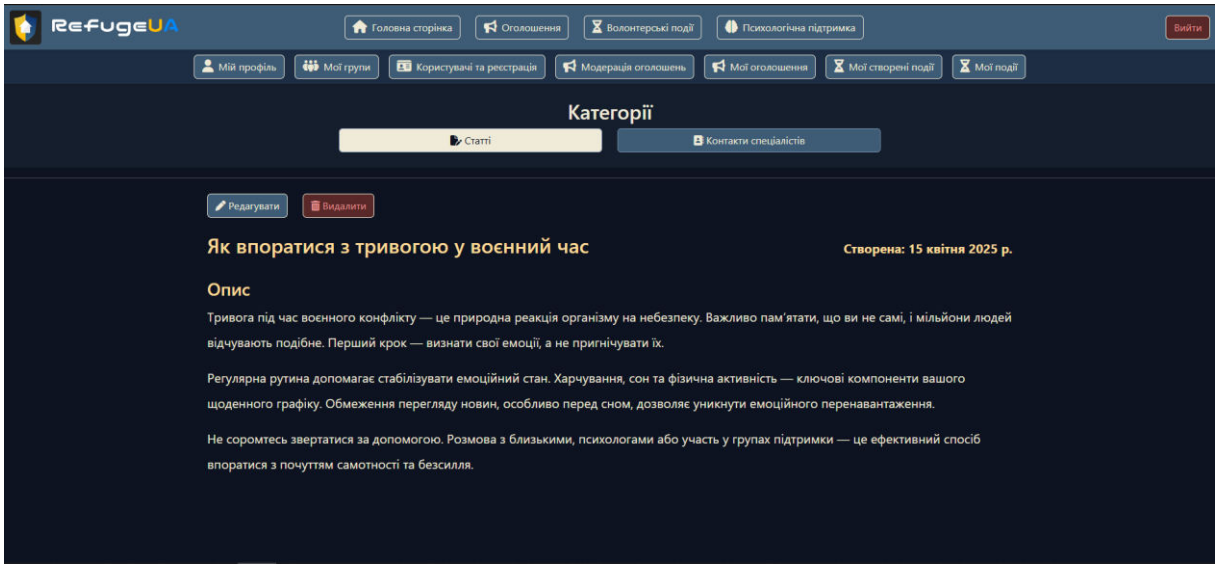


Рис. 38. Сторінка статті з психологічної підтримки

16. Сторінка створення, редагування статті з психологічної підтримки

Для того, щоб перейти на сторінку створення статті, користувачу потрібно відкрити вкладку «Статті» у розділі «Психологічна підтримка», де відображається список уже опублікованих матеріалів. На цій сторінці буде доступна кнопка «Створити статтю», натискання на яку перенаправляє користувача на сторінку з формою створення нової статті (рис. 39).

На сторінці створення статті відображається форма, у якій користувач вводить заголовок та опис статті. Після заповнення обов'язкових полів стає доступною кнопка «Створити статтю». Після натискання на цю кнопку відбувається створення статті, і користувач бачить повідомлення про її успішне додавання.

Для переходу на сторінку редагування статті користувачу необхідно натиснути кнопку «Редагувати», яка розміщується у верхній частині сторінки статті. Сторінка редагування є ідентичною сторінці створення статті, за винятком того, що всі поля форми вже автоматично заповнені відповідно до даних редагованої статті. Користувач може змінити будь-яку з цих даних та зберегти оновлену версію.

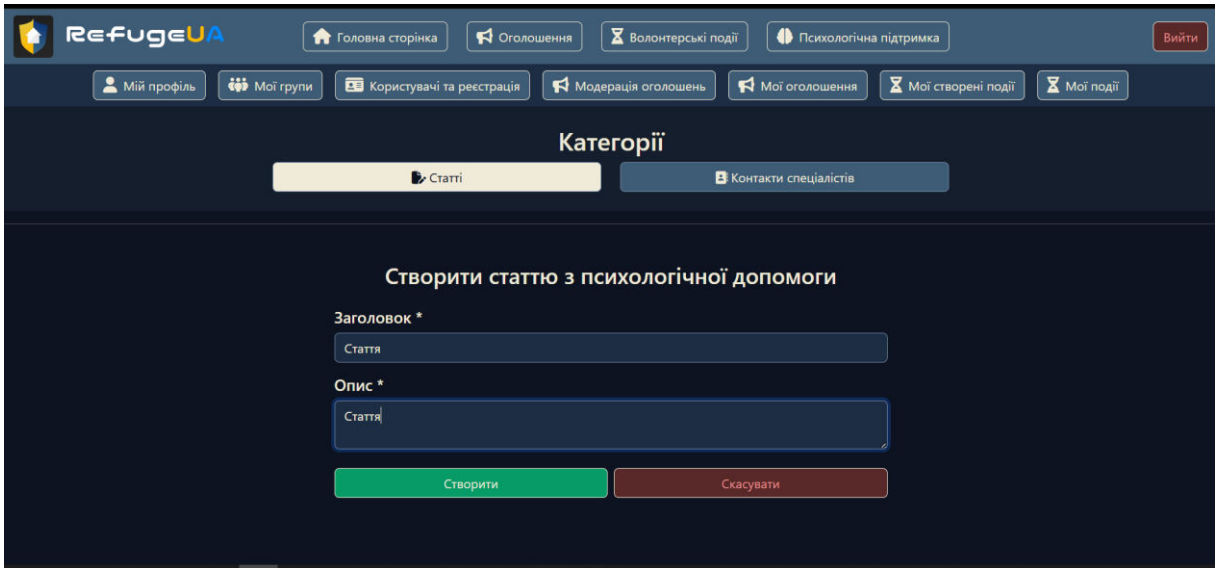


Рис. 39. Форма створення статті з психологічної підтримки

17. Сторінка контактів психологів

Для переходу до розділу з контактів спеціалістів, необхідн натиснути на вкладу «Контакти спеціалістів» в розділі «Психологічна підтримка».

Після натискання вкладки «Контакти спеціалістів» відображається список контактів психологів та пошуковий рядок для пошуку за контактами й заголовком (рис. 40).

Кожна картка містить заголовок, короткий опис і дату створення. В кінці списку є нумерація сторінок для навігації.

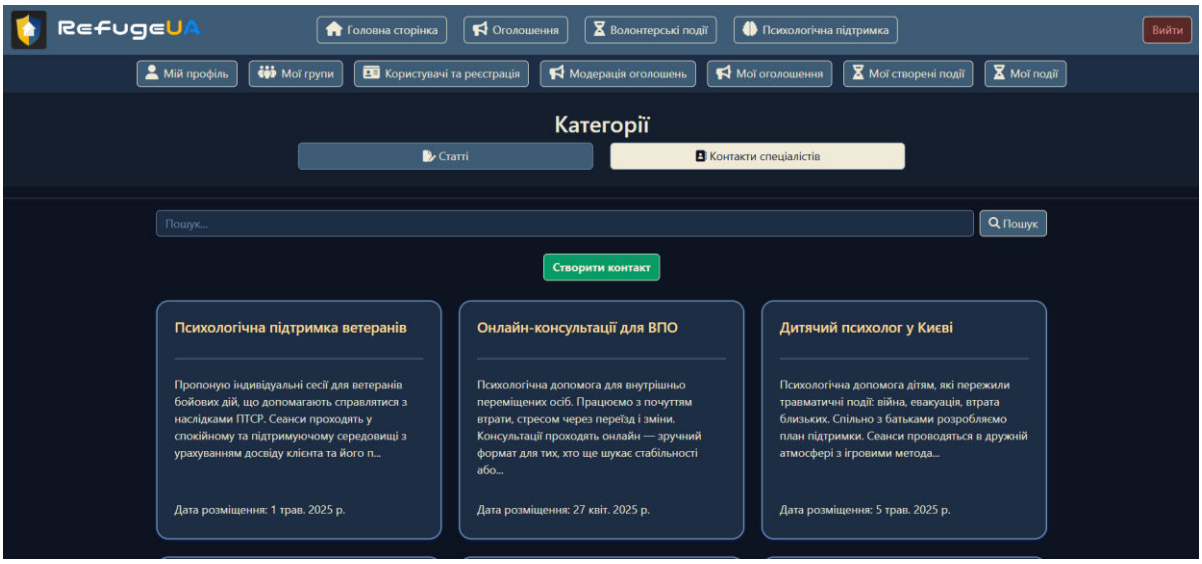


Рис. 40. Сторінка контактів спеціалістів-психологів

18. Сторінка контакту психолога

Для детального перегляду контактів психолога потрібно натиснути на картку передчасного перегляду контакту на сторінці «Контакти спеціалістів». Після чого користувач переходить на сторінку контакту спеціаліста. Сторінка містить заголовок, опис, дату створення та контактні дані (рис. 41).

Контактні дані спеціаліста розміщуються під описом і включають номер телефону, а також за наявності – електронну пошту, Telegram, Viber та Facebook. На рис. 41 показано приклад, у якому відсутнє лише посилання на Viber.

Якщо користувач є адміністратором, йому доступні кнопки «Редагувати» та «Видалити», які дозволяють відповідно перейти на сторінку редагування контакту спеціаліста або видалити його зі списку публікацій.

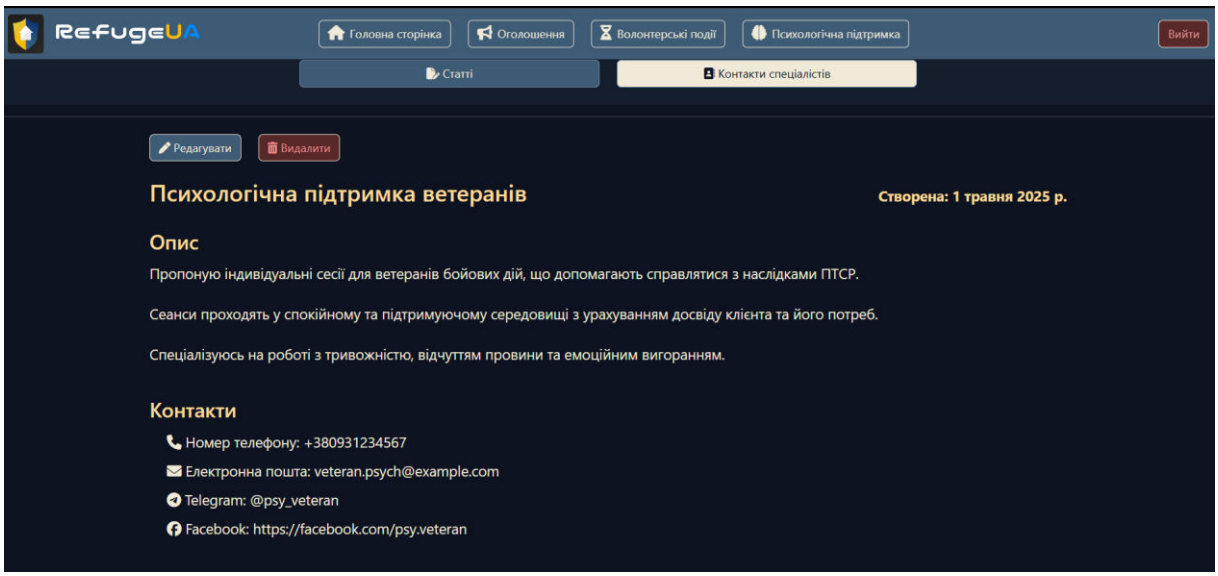


Рис. 41. Сторінка контакту спеціаліста

19. Сторінка створення, редагування контакту психолога

Для того, щоб перейти на сторінку створення контакту психолога, користувачу потрібно відкрити вкладку «Контакти спеціалістів» у розділі «Психологічна підтримка», де відображається список наявних контактів. На цій сторінці доступна кнопка «Створити контакт», натискання на яку перенаправляє користувача на сторінку з формою створення нового контакту психолога (рис. 42).

На сторінці створення контакту відображається форма, у якій користувач вводить заголовок, опис та контактні дані (номер телефону, пошта, телеграм, вайбер, фейсбук). Після заповнення обов’язкових полів стає доступною кнопка «Створити». Після натискання на цю кнопку контакт зберігається, і користувач бачить повідомлення про успішне додавання.

Для переходу на сторінку редагування контакту користувачу необхідно натиснути кнопку «Редагувати», розміщену у верхній частині картки контакту. Сторінка редагування аналогічна сторінці створення, але всі поля форми вже заповнені поточними даними контакту. Користувач може внести зміни та зберегти оновлену інформацію.

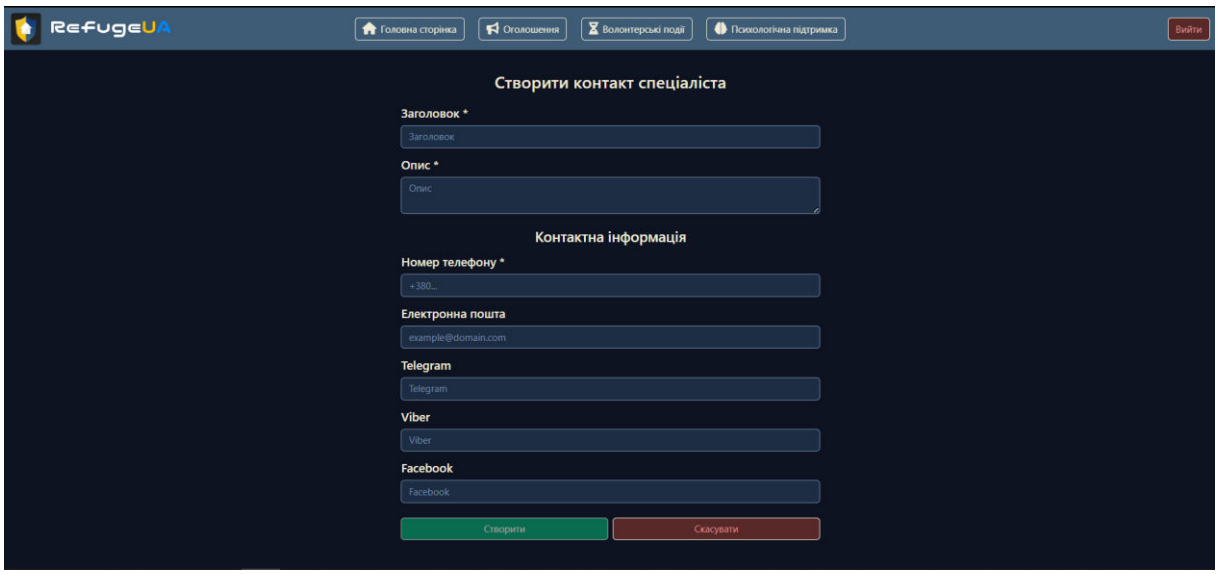


Рис. 42. Сторінка форма для створення контакту психолога

20. Сторінка профілю користувача

Для перегляду власного профілю, користувачу доступне посилання «Мій профіль» в навігаційній панелі. Після натискання користувач переходить на сторінку свого профілю, де він може побачити свої особисті дані та при потребі їх змінити (рис. 43).

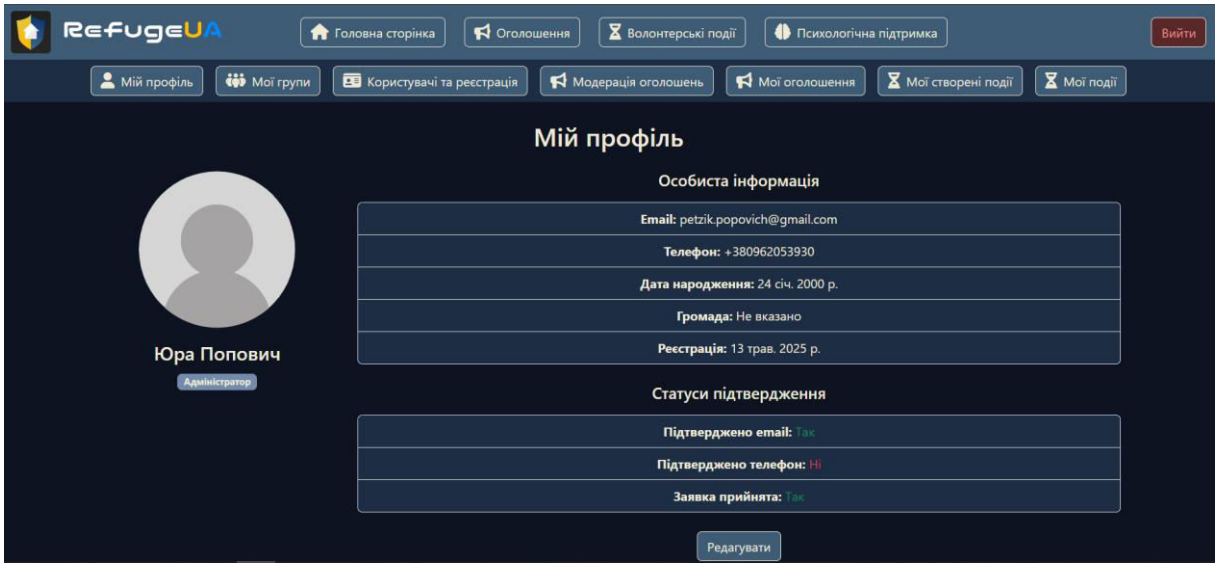


Рис. 43. Профіль користувача

21. Сторінка з своїми оголошеннями

Для перегляду власних створених оголошень користувач може натиснути на посилання «Мої оголошення» в навігаційній панелі. Після цього він переходить на сторінку, де відображаються всі його оголошення та пошуковий рядок (рис. 44).

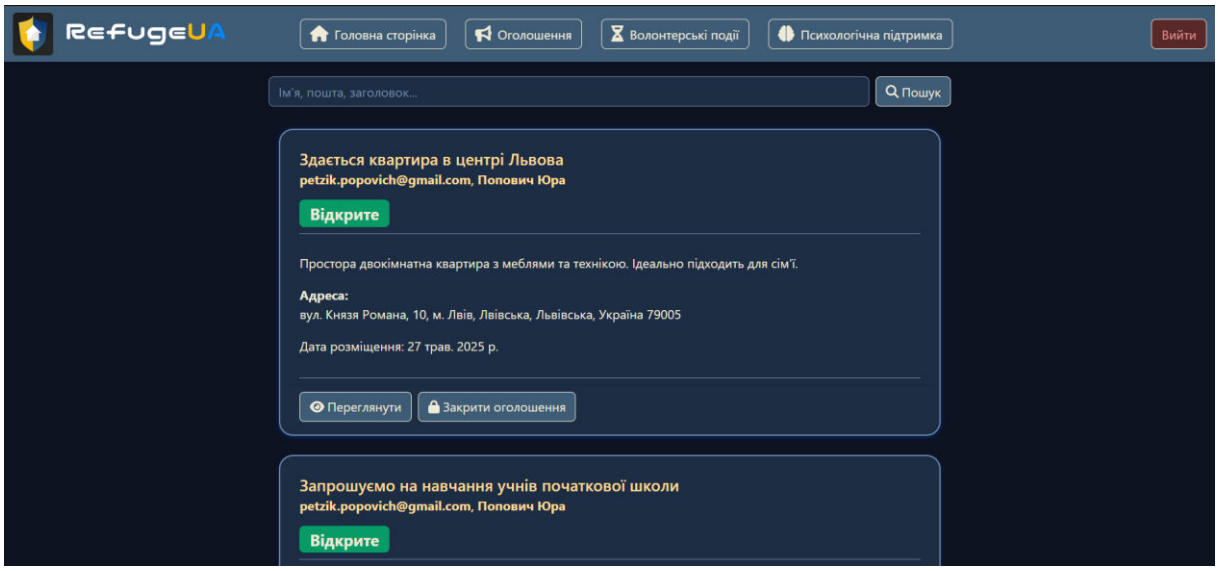


Рис. 44. Сторінка з власними оголошеннями

Кожна картка в списку містить стислу інформацію про оголошення: заголовок, електронну пошту, ім’я автора, короткий опис, адресу та дату розміщення. Поруч з описом розміщені дві кнопки: «Переглянути» – перенаправляє користувача на сторінку оголошення, та «Закрити оголошення» (або «Відкрити оголошення») – дозволяє змінити стан оголошення без переходу на його окрему сторінку.

22. Сторінка модерації оголошень

Для перегляду сторінки модерації оголошень адміністратор має доступ до посилання «Модерація оголошень», після переходу за яким

відкривається сторінка з оголошеннями, з елементами інтерфейсу для модерації (рис. 45).

На сторінці «Модерація оголошень» відображається список карток попереднього перегляду оголошень усіх користувачів та пошуковий рядок, який дозволяє знаходити оголошення за заголовком. На кожній картці оголошення є посилання «Переглянути» для переходу на сторінку детального перегляду, кнопка «Закрити оголошення» (або «Відкрити оголошення»), яка змінює стан публікації оголошення – робить його видимим або невидимим для інших користувачів, а також кнопка «Прийняти» (або «Відхилити»), яка надає автору оголошення дозвіл або відмову в розміщенні оголошення в системі.

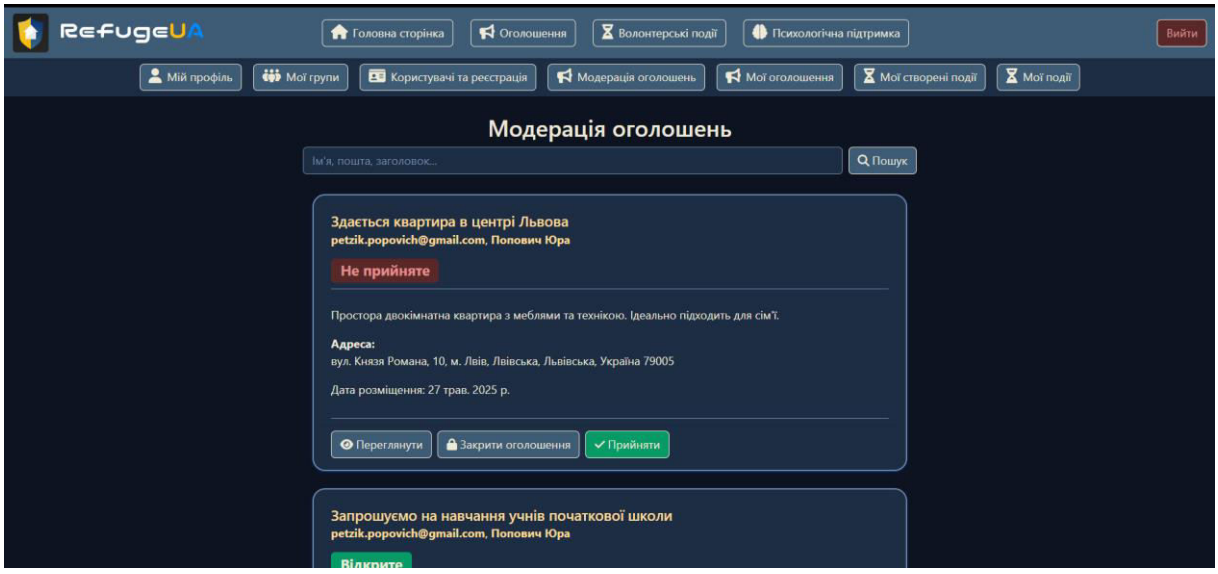


Рис. 45. Сторінка модерації оголошення

23. Сторінка модерації користувачів та заявок на реєстрацію

Для перегляду сторінки модерації користувачів та заявок на реєстрацію адміністратор має доступ до посилання «Користувачі та реєстрація», після переходу за яким відкривається сторінка з списком користувачів в системі, з елементами інтерфейсу для модерації. Якщо

адміністраторам є представник громади, він може переглядати користувачі лише на території своєї громади (рис. 46).

На сторінці «Користувачі та реєстрація» відображається список карток попереднього перегляду зареєстрованих користувачів, а також пошуковий рядок, який дозволяє знаходити користувачів за іменем або електронною поштою. На кожній картці користувача вказано його ім’я, електронну пошту, номер телефону, роль та дату народження. Додатково доступні кнопки: «Переглянути» – для переходу до детального перегляду профілю; «Дозволити доступ» – для надання користувачеві доступу до системи; «Підтвердити пошту» – для ручного підтвердження електронної адреси; та «Видалити» – для видалення користувача зі списку.

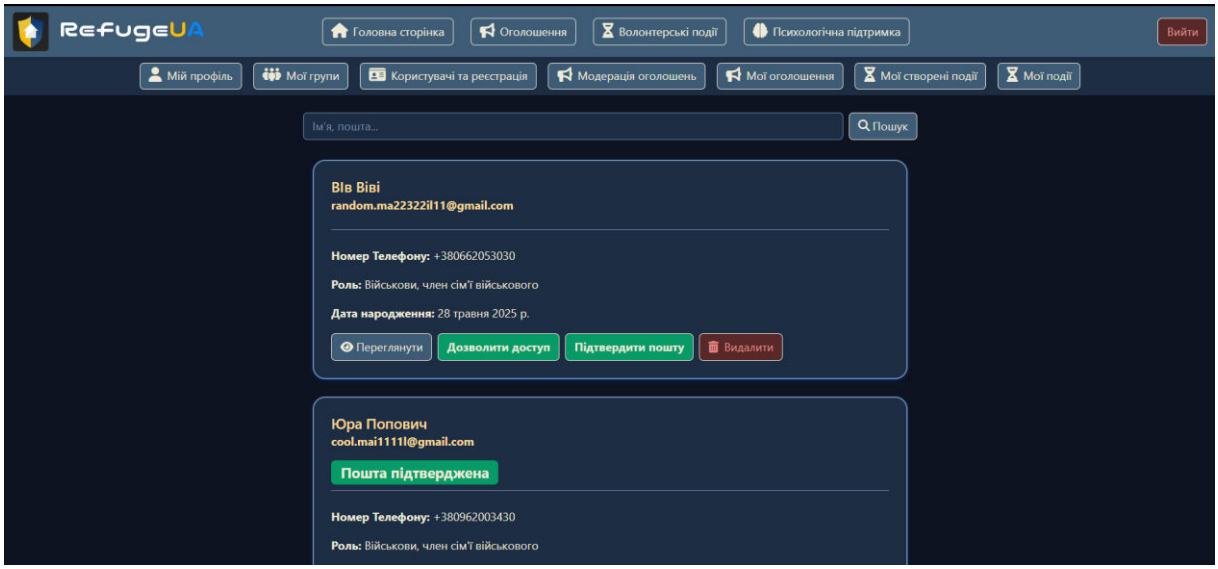


Рис. 46. Сторінка модерації користувачів та заявок на реєстрацію

24. Сторінка груп оголошення

Зі сторінки деталей оголошення адміністратор має доступ до посилання «Переглянути групи», яке веде на сторінку керування групами, пов’язаними з конкретним оголошенням.

Інтерфейс поділено на дві основні панелі: «Поточні групи» (ліворуч) – відображає перелік груп, які вже пов’язані з оголошенням, та «Групи на

вибір» (праворуч) – список усіх доступних груп, які можна прив’язати. Натискання на назву групи переміщує її з однієї панелі до іншої: якщо натиснути на групу в панелі «Групи на вибір», вона додається до оголошення і з’являється у списку «Поточні групи»; аналогічно, при натисканні на групу з «Поточних груп» вона вилучається та повертається до доступних. Нижче розташований блок для створення нової групи – він містить поле введення назви (до 100 символів) і кнопки «Створити» (рис. 47).

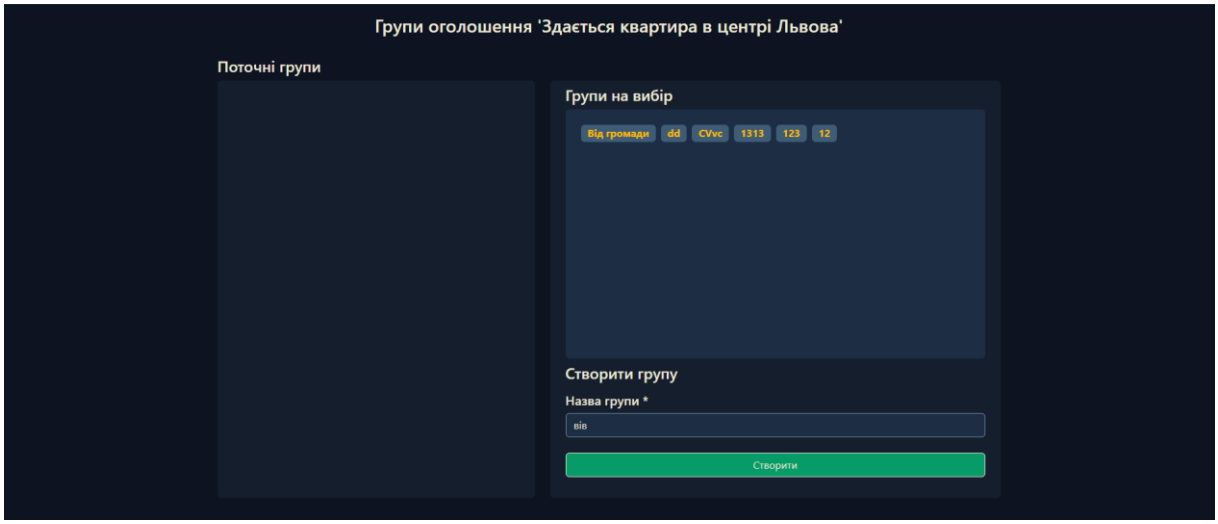


Рис. 47. Сторінка груп оголошення

25. Сторінка з подіями, в яких користувач бере участь

У навігаційній панелі користувач має доступ до посилання «Мої події», яке веде на сторінку зі списком подій, на які він зареєстрований (рис. 48). Після переходу на цю сторінку відображається перелік зареєстрованих подій у вигляді списку карток із основною інформацією: назвою події, типом, коротким описом, датами та місцем проведення. Для того, щоб переглянути детальну інформацію про подію, поруч із кожною подією в списку розміщене посилання «Переглянути». Після натискання на це посилання користувач перенаправляється на сторінку події.

У разі відсутності зареєстрованих подій відображається повідомлення про те, що користувач наразі не має активних реєстрацій.

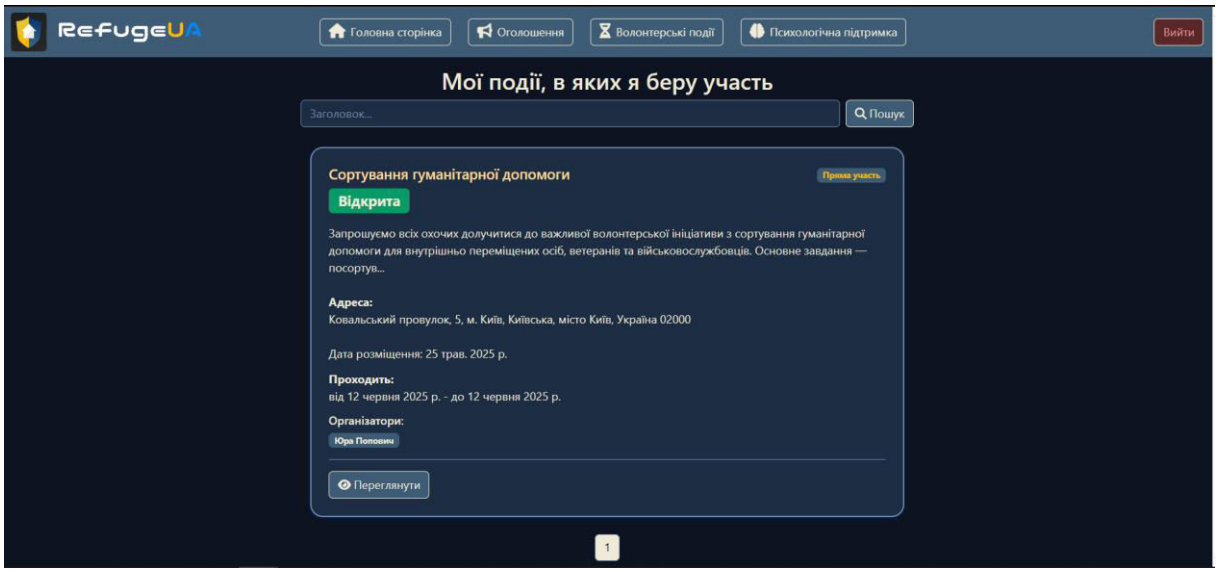


Рис. 48. Сторінка подій, в яких користувач бере участь