Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти *другий (магістерський)*

Розробка та дослідження інформаційної системи організації та координації волонтерської діяльності
(тема)

Виконав:
студент *II* курсу, групи *СПРм-22-2*
Рибаков М.Д.
(прізвище, ініціали)

Спеціальність *122 Комп'ютерні науки*
(код і повна назва спеціальності)
Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Системне проектування*
(повна назва освітньої програми)

Керівник *доц. Петрова Р. В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Гребеннік І. В.*
(підпис) (прізвище, ініціали)

2024 р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Дата 19.06.24 Рибаков М. Д.

 Підпис

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.
Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведено «19» червня 2024 р.

Керівник кваліфікаційної роботи доц. Петрова Р.В 

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*

Кафедра *Системотехніки*

Рівень вищої освіти *другий (магістерський)*

Спеціальність *122 Комп'ютерні науки*
(код і повна назва)

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Рибакову Максиму Дмитровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розробка та дослідження інформаційної системи організації та координації волонтерської діяльності»*
затверджена наказом університету від *«01» квітня 2024 р.* № *259 Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *«20» червня 2024 р.*
3. Вихідні дані до роботи *Об'єкт дослідження – інформаційна система організації та координації волонтерської діяльності. Предмет дослідження – Методи координації та моніторингу волонтерської діяльності. Об'єкт розробки – додаток для оптимізації ефективності організації та координації волонтерської діяльності. Технічне забезпечення: IBM-сумісний персональний комп'ютер. Перелік використовуваних програмних засобів: ОС Microsoft Windows 10, мова або пакет імітаційного моделювання.*
4. Перелік питань, що потрібно опрацювати в роботі *4.1 Перелік скорочень та термінів. 4.2 Вступ. 4.3 Аналіз предметної області. 4.4 Аналіз аналогічних систем. 4.5 Постановка задачі. 4.6 Архітектура додатку. 4.7 Вимоги до БД. 4.8 Вимоги до користувацького Інтерфейсу. 4.9 Опис алгоритмів головних бізнес-процесів розроблюваного веб-застосунку 4.10 Обгрунтування вибору мови програмування. 4.11 Обгрунтування вибору фреймоврку. 4.12 Імплементація безпеки застосунку 4.13 Взаємодія з БД. 4.14 Експериментальні дослідження. 4.15 Розробка дизайну 4.16 Інтеграція з хмарним середовищем 4.17 Висновки. 4.18 Переліе джерел 4.19 Керівництво користувача. 4.36. Текст програми.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п. 5 включається до завдання за рішенням випускової кафедри) *5.1 Схема бази дани. 5.2 Діаграма bpmn реєстрації. 5.3 bpmn створення запиту. 5.4 Архітектура додатку. 5.5 Мікросервісна архітектура*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання та аналіз завдання, уточнення плану роботи*	*13.01.24*	виконано
2	*Аналіз формування заявки на допомогу як об'єкта*		виконано
	автоматизації	*26.01.24*	виконано
3	*Огляд існуючих моделей та методів моніторингу та*		виконано
	координації гуманітарної допомоги	*01.02.24*	виконано
4	*Розробка вимог до інформаційної системи*	*20.02.24*	виконано
5	*Розробка програмного забезпечення задачі*	*14.03.24*	виконано
6	*Проведення експериментальних досліджень*	*26.03.24*	виконано
7	*Оформлення пояснювальної записки*	*04.04.24*	виконано
8	*Підготовка презентації*	*01.05.24*	виконано
9	*Подання закінченої роботи науковому керівникові*	*02.06.24*	виконано
10	*Подання роботи на рецензування*	*16.06.24*	виконано
11	*Попередній захист*	*19.06.24*	виконано
12	*Подання роботи до екзаменаційної комісії*	*18.06.24*	виконано

Дата видачі завдання *«13» січня 2024 р.*

Студент _____
(підпис)

Керівник роботи _____ *доц. Петрова Р. В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Кваліфікаційна робота містить: 67 с., 35 рис., 1 табл., 25 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, ЖИТТЄВИЙ ЦИКЛ, ВОЛОНТЕР, КОРИСТУВАЧ, ВОДІЙ, UML, DFD, ERD, БАЗА ДАНИХ, СЕРВЕР, СИСТЕМА, ДОДАТОК.

Об’єкт дослідження – інформаційна система, що призначена для організації, обробки, аналізу та координації даних, пов'язаних з волонтерською діяльністю.

Предмет дослідження – методи координації та організації волонтерської діяльності з використанням інформаційної системи.

Мета роботи – розробка та дослідження окремих елементів і функцій інформаційної системи для підвищення ефективності організації та координації волонтерської в реальному часі, з використанням інтегрованого підходу.

Система буде розроблятись з використанням мікросервісної архітектури, де за сервіс бекенду відповідає FastApi Python, за сервіс бекенду відповідає react.js, а за сервіс аутинтефікації Firebase Auth. В якості системи управління базами даних було обрано SQL Server Management Studio – система управління базами даних, яка розробляється корпорацією Microsoft.

Об’єктом розробки є додаток для оптимізації ефективності організації та координації волонтерської діяльності, який може бути використаний волнтерськими організаціями або звичайними користувачами у яких є потреба розширити сферу використання не тільки в живу, а і у Інтернеті.

ABSTRACT

The report on qualification work contains: 67 pages, 35 figures, 1 table, 25 sources.

INFORMATION SYSTEM, LIFE CYCLE, VOLUNTEER, USER, DRIVER, UML, DFD, ERD, DATABASE, SERVER, SYSTEM, APP.

The object of the study is an information system designed for monitoring, processing, analysis and coordination of data related to volunteering.

The subject of the research is methods of coordination and organization of volunteer activities using an information system.

The purpose of the work is the development and research of individual elements and functions of the information system to increase the effectiveness of the organization and coordination of volunteer work in real time, using an integrated approach.

The system will be developed using a microservice architecture, where FastApi Python is responsible for the backend service, react.js is responsible for the backend service, and Firebase Auth is responsible for the authentication service. SQL Server Management Studio was chosen as the database management system, a database management system developed by Microsoft Corporation.

The object of development is an application for optimizing the effectiveness of the organization and coordination of volunteer activities, which can be used by volunteer organizations or ordinary users who need to expand the scope of use not only live, but also on the Internet.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

SQL (Structured Query Language) - мова структурованих запитів

API (Application Programming Interface) – Прикладний програмний інтерфейс

GSC (Google Cloud Storage) – хмарний сервіс для зберігання і керування даними.

БД – База даних

Транзакція – Група послідовних операцій.

Фреймворк – Набір інструментів, бібліотек та правил, який використовується для створення програмних додатків.

ІС – Інформаційна система.

СУБД – Система управління базами даних.

HTTP (HyperText Transfer Protocol) – Протокол передачі гіпертекстуUI (User Interface) – користувацький інтерфейс

OAuth (Open Authorization) - Відкритий стандарт авторизації

OWASP (Open Web Application Security Project) - Відкритий проєкт з безпеки вебзастосунків

JSON (JavaScript Object Notation) - запис об'єктів JavaScript

JWT (JSON Web Token) – JSON веб токен

ВСТУП

Реалії сьогодення з якими зіткнулася Україна та українське суспільство сприяли згуртуванню населення та фахівців різноманітних сфер. Щоб підтримати людей, що потребують допомоги, багато українців почали займатись волонтерською діяльністю. За незначний проміжок часу волонтерство в Україні перетворилось на потужний національний громадянський рух, як окремих осіб, так і громадських організацій, які спрямовують свою діяльність на надання різнопланової допомоги за окремими напрямами. Волонтерська діяльність в Україні стала не лише важливим чинником у підтримці різних сфер, але й символом солідарності та активності громадян у вирішенні соціальних проблем.

Незважаючи на існуючі ініціативи та проекти, волонтерська діяльність в Україні стикається з різноманітними труднощами у плануванні, координації та виконанні завдань. Однією з ключових проблем є відсутність централізованого засобу комунікації та обміну інформацією між волонтерами, організаціями та координаторами. Це ускладнює співпрацю між учасниками волонтерських проектів, відстеження прогресу та взаємодію між різними волонтерськими групами. Дана проблема може призвести до дублювання зусиль, неправильного розподілу ресурсів та навіть до невиконання деяких важливих завдань.

Інформаційна система організації та координації волонтерської діяльності має на меті полегшити та оптимізувати процес взаємодії між волонтерами, координаторами та організаціями. Актуальність такого застосунку очевидна в сучасному світі, де волонтерська діяльність відіграє важливу роль у підтримці різних сфер.

Така система дозволить ефективно координувати роботу волонтерів, спрощувати процес реєстрації та взаємодії з ними, підтримувати спільність дій та обмін досвідом між учасниками. Також вона буде сприяти відстеженню

виконання завдань, забезпеченню волонтерів необхідною інформацією та ресурсами, а також підвищенню мотивації і залучення нових учасників до волонтерської діяльності.

Така система забезпечить підвищення ефективності ресурсного управління, дозволяючи краще розподіляти матеріальні та фінансові ресурси між проектами та потребуючими групами. Додатково, система сприятиме створенню прозорого та ефективного механізму моніторингу та оцінки результативності волонтерських проектів. Вона дозволить збирати та аналізувати дані про виконання завдань, витрати ресурсів та вплив проектів на громади. Також, за допомогою цієї системи буде забезпечена можливість взаємодії з іншими подібними платформами, що дозволить уникнути дублювання зусиль та забезпечить більшу інтеграцію та спільність дій волонтерських організацій. Такий підхід сприятиме підвищенню ефективності та забезпечить більшу згуртованість громадського сектору в Україні.

Таким чином, ця інформаційна система стане незамінним інструментом для досягнення мети волонтерських програм та реалізації їхнього потенціалу в наданні допомоги та підтримки для потребуючих категорій населення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Через складну ситуацію в Україні зараз спостерігається значна потреба у волонтерах. Волонтери відіграють важливу роль у наданні допомоги військовим, постраждалим від бойових дій, переселенцям, а також у підтримці медичної та соціальної сфери в умовах кризи. Їхня безкоштовна праця допомагає зменшити соціальну напругу, забезпечує необхідну допомогу тим, хто зазнає складнощів, і сприяє зміцненню громадської солідарності та взаємодопомоги в українському суспільстві.

Волонтерство – це діяльність, яка здійснюється добровільно для суспільства чи окремих соціальних груп, без розрахунку на винагороду. Французьке слово volontaire (“волонтер”) походить від латинського voluntarius (“добровільний”).

Волонтер – це людина, яка добровільно надає безоплатну допомогу людям, які потребують особливої підтримки та соціального захисту, або державі, некомерційній чи громадській організації. Таким чином, волонтерами стають не з фінансових спонукань, а заради добровільної допомоги іншим та набуття безцінного життєвого досвіду. Адже регулярно та усвідомлено допомагаючи іншим людина дарує їм свій час, вміння та ресурси. І натомість отримує щось значуще: знання, навички, досвід, коло соратників, нові знайомства.

Волонтерство стає все більш популярним у сучасному суспільстві. Люди різного віку та соціального становища обирають цей шлях, щоб внести свій вклад у розвиток громади та допомогти тим, хто цього потребує.

Волонтерство відкриває безліч можливостей для особистісного розвитку. Воно допомагає набути нових навичок, зрозуміти свої сильні сторони та виявити свої справжні інтереси. Волонтерство також дає

можливість відчути радість від допомоги іншим, від почуття власної корисності та значимості.

Серед нормативних актів, які регулюють волонтерську діяльність в Україні, виокремлюють:

- Закон України «Про волонтерську діяльність» – основний документ, який регулює відносини, пов'язані з наданням волонтерської допомоги. Закон визначає, хто може надавати волонтерську допомогу, у яких напрямах та за яких умов;
- Закон України «Про благодійну діяльність та благодійні організації» – основний документ, який регулює відносини у сфері благодійності. Зокрема, Закон визначає поняття благодійної діяльності, благодійної організації, бенефіціара, а також виокремлює сфери благодійності та умови збору благодійних пожертв.

Основні цілі і мета волонтерської діяльності:

- Сприяння у проведенні соціально вагомих заходів і акцій (не політичного характеру), спрямованих об'єднати зусилля для координації дій у підвищенні ефективності надання благодійної допомоги, забезпечення максимальної ефективності матеріальної, організаційної та правової допомоги, здійснення заходів для досягнення спільних цілей, які полягають у багатосторонній допомозі Збройним силам України, внутрішньо переміщеним особам, дитячим будинкам (дітям сиротам), інвалідам війни, ветеранам війни, спортивним та навчальним закладам, притулкам для тварин, медичним закладам;
- Систематична комунікація та координація спільних дій між волонтерами, громадськими організаціями, благодійними організаціями, а також їх комунікація та координація з державними органами та органами місцевого самоврядування, міжнародними організаціями, спрямованих на створення прозорості, довіри, партнерських відносин та поліпшення співпраці;

- Придбання нових знань і навичок, обмін інформацією і досвідом, що пов’язані з реалізацією програм та проектів у сфері соціального захисту волонтерів, участь у тренінгах для волонтерів, на форумах, круглих столах, конференціях, засіданнях;
- Можливість повноцінного розвитку особистого потенціалу, прагнення до покращення своїх професійних навичок, висока якість роботи та послуг, побудованих на соціальній відповідальності перед суспільством.

Ключовим аспектом системи, що досліджується та розроблюється є моніторинг діяльності волонтерів та організацій. Моніторинг діяльності - це процес систематичного спостереження, аналізу та оцінки різних аспектів або етапів виконання певної діяльності, проекту або процесу. Основна мета моніторингу діяльності полягає в тому, щоб забезпечити ретельне відстеження прогресу, визначити відхилення від поставлених цілей або стандартів, ідентифікувати можливі проблеми або ризики. Важливою частиною моніторингу є збір та аналіз даних, а також звітність про результати спостережень. Моніторинг діяльності у системі буде здійснюватися за допомогою відкритої звітності зборів та розподілу коштів.

Для успішного проектування та розробки даної системи, перш за все, необхідно визначити, для кого буде функціонувати дана система:

Ті, хто шукають допомогу – особи або організації, що потребують допомоги від суспільства. Для них буде реалізовано можливість подання заявок, в яких будуть відображені їх потреби.

Волонтери - особи або організації, які на безоплатній основі пропонують свою допомогу. Ці користувачі матимуть змогу збирати кошти на потреби, що відображатимуться у поданих заявках, а також публікувати звітність, щодо процесу виконання запитів.

Адміністратори – категорія користувачів, котрі будуть обробляти звернення та скарги, а також здійснювати модерацію створюваних заявок.

1.2 Аналіз аналогічних систем

На сьогоднішній день існує велика кількість інформаційних систем, призначених для підтримки та координації волонтерської діяльності. Ці системи можуть мати різний функціонал, від простих до складних, і включати в себе такі можливості, як реєстрація волонтерів, взаємодія з організаціями, розподіл завдань, відстеження прогресу проектів, обмін інформацією та документами, а також аналіз ефективності діяльності.

Ці інформаційні системи спрощують процес волонтерства, роблять його більш організованим та ефективним, дозволяючи координаторам краще керувати ресурсами та волонтерами. Вони допомагають уникнути дублювання зусиль, покращують комунікацію між учасниками, забезпечують швидкий доступ до необхідної інформації та сприяють збору даних для подальшого аналізу та вдосконалення стратегій волонтерських програм.

Деякі інформаційні системи є великими проектами з великою спільнотою користувачів, які постійно вдосконалюються та розширюються. Інші можуть бути створені конкретними організаціями або групами волонтерів для вирішення конкретних завдань або потреб.

Одними з найвідоміших таких інформаційних систем є Prozorro, Palyanytsya.Інфо, Волонтер.Орг та Helplist.

Електронна система публічних закупівель Prozorro – це онлайн-платформа, де державні та комунальні замовники оголошують тендери на закупівлю товарів, робіт і послуг, а представники бізнесу змагаються на торгах за можливість поставити це державі.

Якщо порівнювати функціонал систем Prozorro та інформаційної системи для волонтерства, то в обох випадках можна відзначити наявність звітності. У Prozorro це може бути звіт про проведений тендер, в якому фіксуються всі деталі процесу закупівлі, від інформації про учасників до умов угоди. У волонтерській інформаційній системі звітність може включати

інформацію про кількість волонтерів, задіяних у конкретному проекті, виконані завдання, витрачений час і ресурси, а також результати діяльності

Головну сторінку системи зображено на рисунку 1.1.

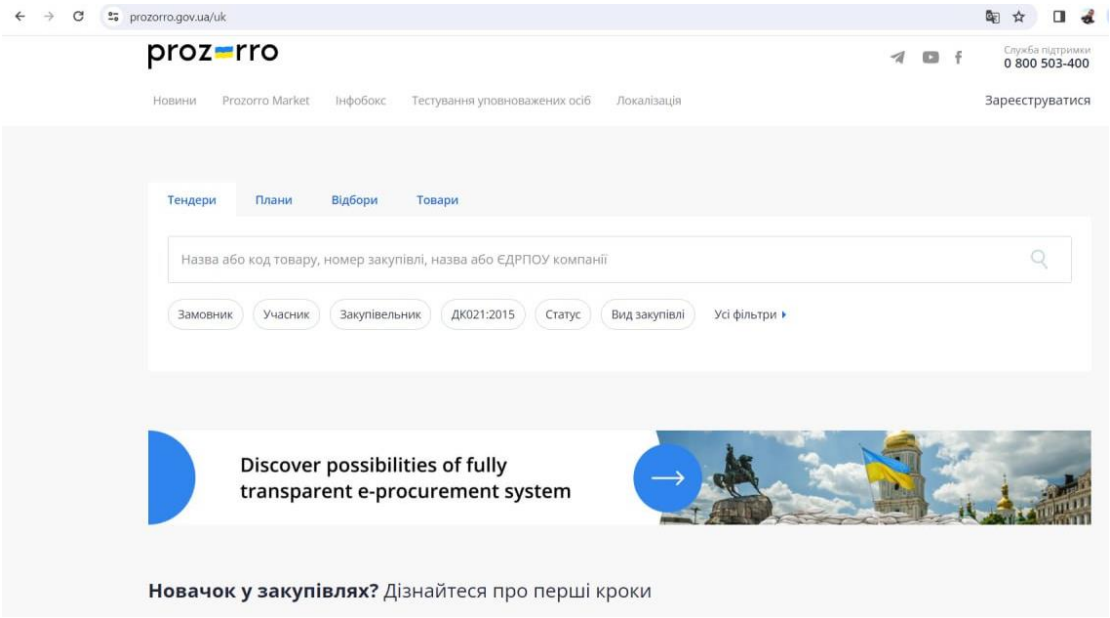

Рисунок 1.1 – Головна сторінка системи Prozorro

Паляниця.Інфо — це відкрита база організацій, які надають гуманітарну та волонтерську допомогу населенню по всій Україні.

Платформа створена Українською Волонтерською Службою спільно з IT-компанією SoftServe для допомоги людям у час повномасштабної війни.

Паляниця.Інфо налічує понад 950 організацій та ініціатив, які регулярно оновлюються. Портал створений для того, аби спростити пошук допомоги та зробити його ефективнішим. Відтак організації розподілені за 15 категоріями, актуальними під час війни та розподіляються за локаціями в Україні.

Кожна організація, додана на платформу, верифікується Українською Волонтерською Службою.

Якщо ви потребуєте гуманітарної чи волонтерської допомоги — ви маєте змогу скористатися спеціальним фільтром на головній сторінці. Фільтр

дозволяє за лічені хвилини знайти потрібну організацію, залежно від вашого місцеперебування та категорії запиту.

Після цього, із переліку потрібних ініціатив можна обрати ту, яка найбільше відповідає вашому запиту. На окремій сторінці організації зберігаються актуальні дані про неї — опис, контакти, офіційний сайт.

Якщо ви є представником/представницею волонтерської організації — ви можете скористатися кнопкою «Додати інформацію». Після заповнення спеціальної форми представник команди зв'яжеться з вами для підтвердження інформації.

Головну сторінку системи зображено на рисунку 1.2.

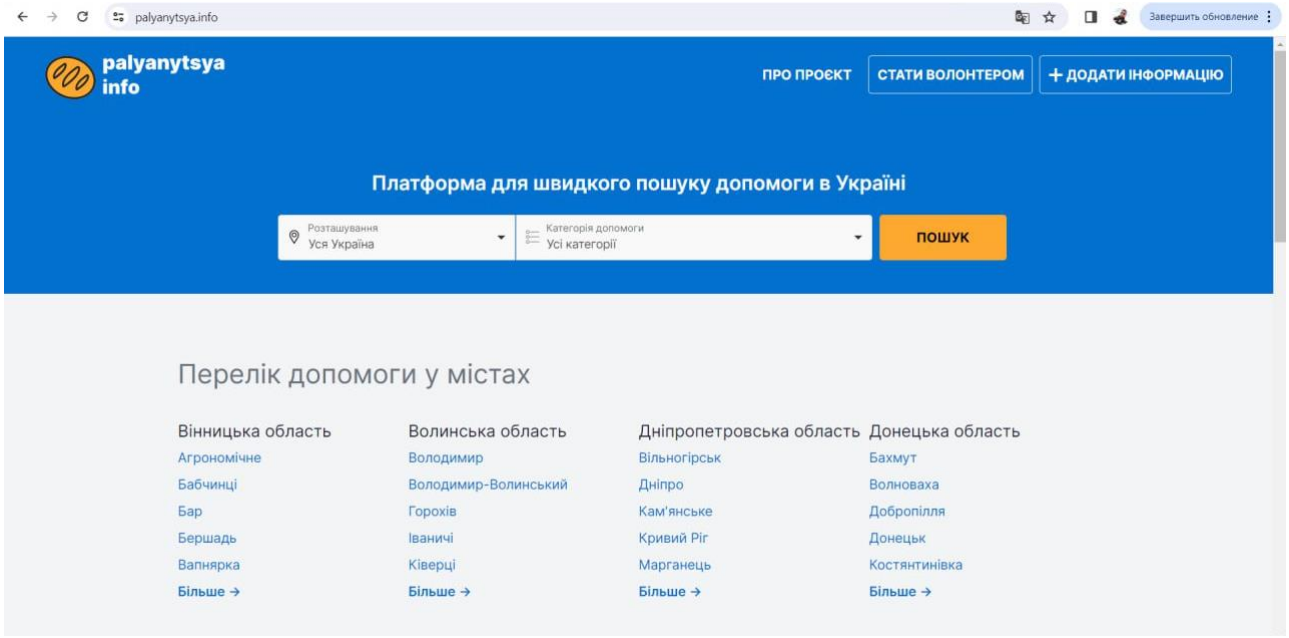

Рисунок 1.2 – Головна сторінка системи Паляниця.Інфо

Сервіс Паляниця.Інфо пропонує різноманітні можливості користувачів, які потребують допомоги або інформації під час війни. Користувачі можуть знайти інформацію про те, де і як отримати гуманітарну допомогу, включаючи їжу, одяг, медикаменти та інші необхідні речі. Сервіс також допомагає знайти тимчасове житло для. Паляниця.Інфо надає інформацію про медичні заклади, де можна отримати допомогу, включаючи контакти та адреси лікарень і клінік.

Користувачі можуть отримати юридичну допомогу та консультації щодо різних питань, пов'язаних з евакуацією, соціальним захистом та іншими юридичними аспектами.

Пошук організацій за категоріями у системі Паляниця.Інфо зображено на рисунку 1.3.

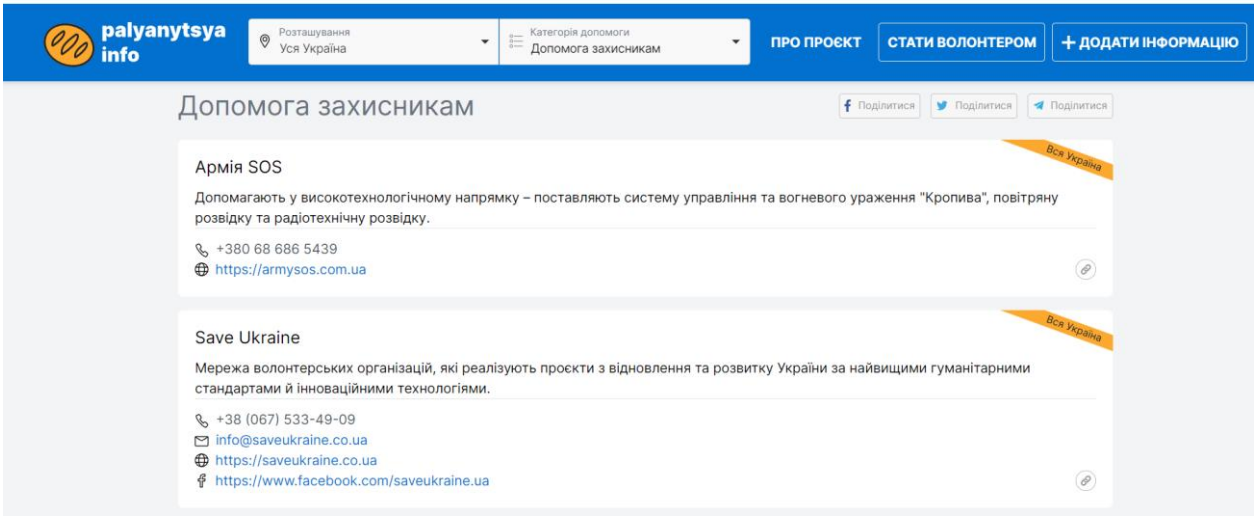

Рисунок 1.3 – Приклад пошуку за категоріями

Платформа volonter.org – це веб-сервіс, який допомогає громадським ініціативам знаходити для своїх проектів волонтерів, а волонтерам - допомагати в реалізації проектів. Завдання платформи – побудувати горизонтальні зв'язки та встановити ефективну комунікацію між волонтерами, волонтерськими групами та громадськими організаціями.

Головну сторінку платформи volonter.org зображено на рисунку 1.4.

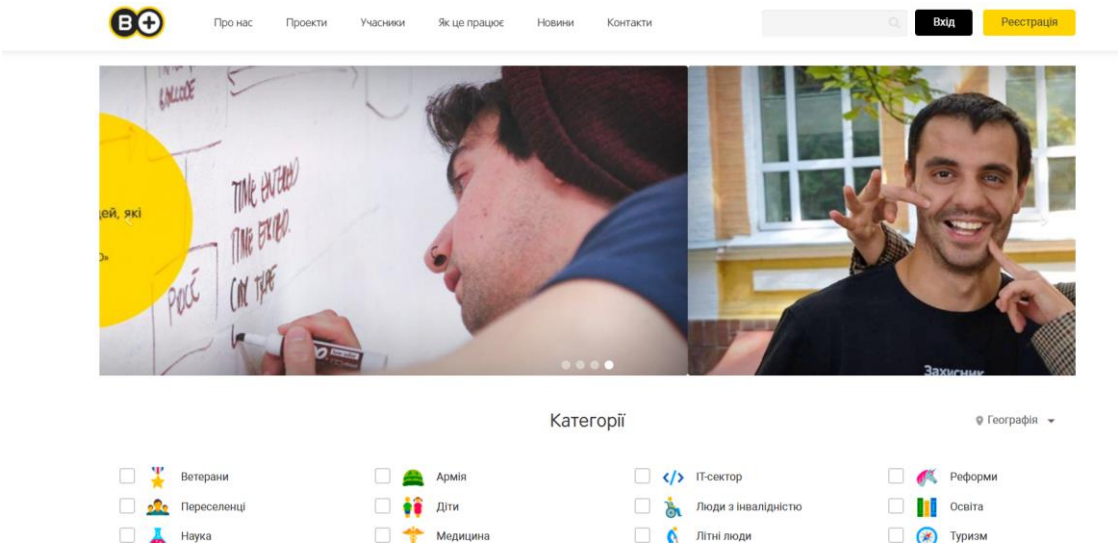

Рисунок 1.4 – Головна сторінка платформи volonter.org

В системі існує можливість пошуку проектів за містом або тематикою, а також перегляду усіх відкритих проектів. Користувач може обирати проекти, які він бажає підтримати, та підписуватися на оновлення від організаторів, щоб завжди бути в курсі подій. Приєднавшись до проекту, користувач може спілкуватися з організаторами через чат, щоб обговорити деталі та способи допомоги. Участь у проекті дозволяє користувачу підняти свій статус на сайті, що відображає його активність та внесок у реалізацію різноманітних ініціатив.

На платформі ви маєте можливість створювати власні події або проекти, або приєднатися до вже існуючих. Для реалізації вашої ідеї вам можуть знадобитися різні фахівці, такі як дизайнер, копірайтер, художник або звукорежисер. Ви можете знайти цих людей, описавши свої потреби та запрошуючи до співпраці. На платформі ви також матимете спеціальний чат із користувачами, які зацікавлені у вашому проекті, де ви зможете обговорювати ідеї та ділитися планами. Незалежно від того, яку роль ви оберете при реєстрації, ви матимете можливість долучитися до проектів або створити власний.

Сторінку учасників зображено на рисунку 1.5.

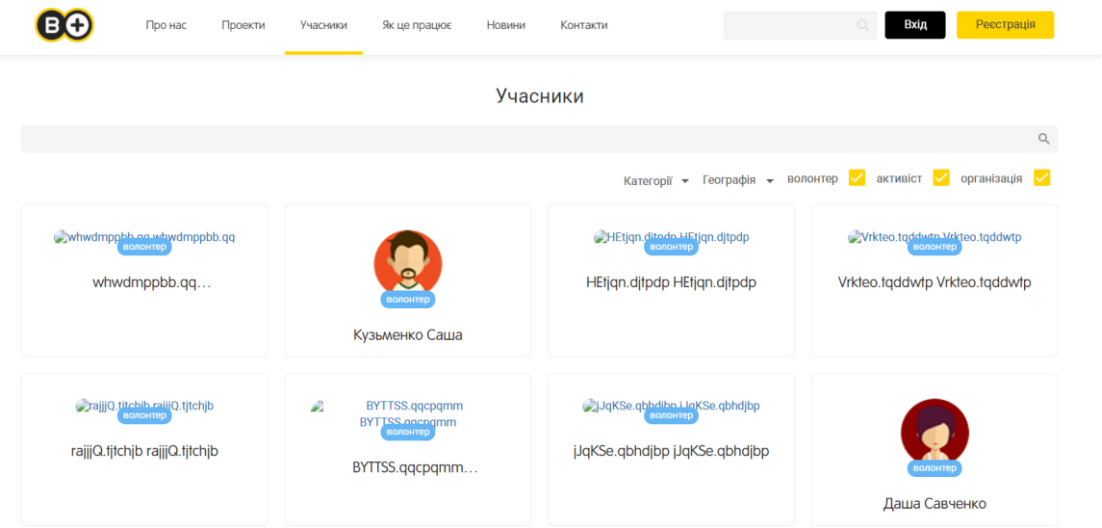

Рисунок 1.5 – Сторінка учасників на платформі volonter.org

Сторінку реєстрації на платформі зображено на рисунку 1.6.

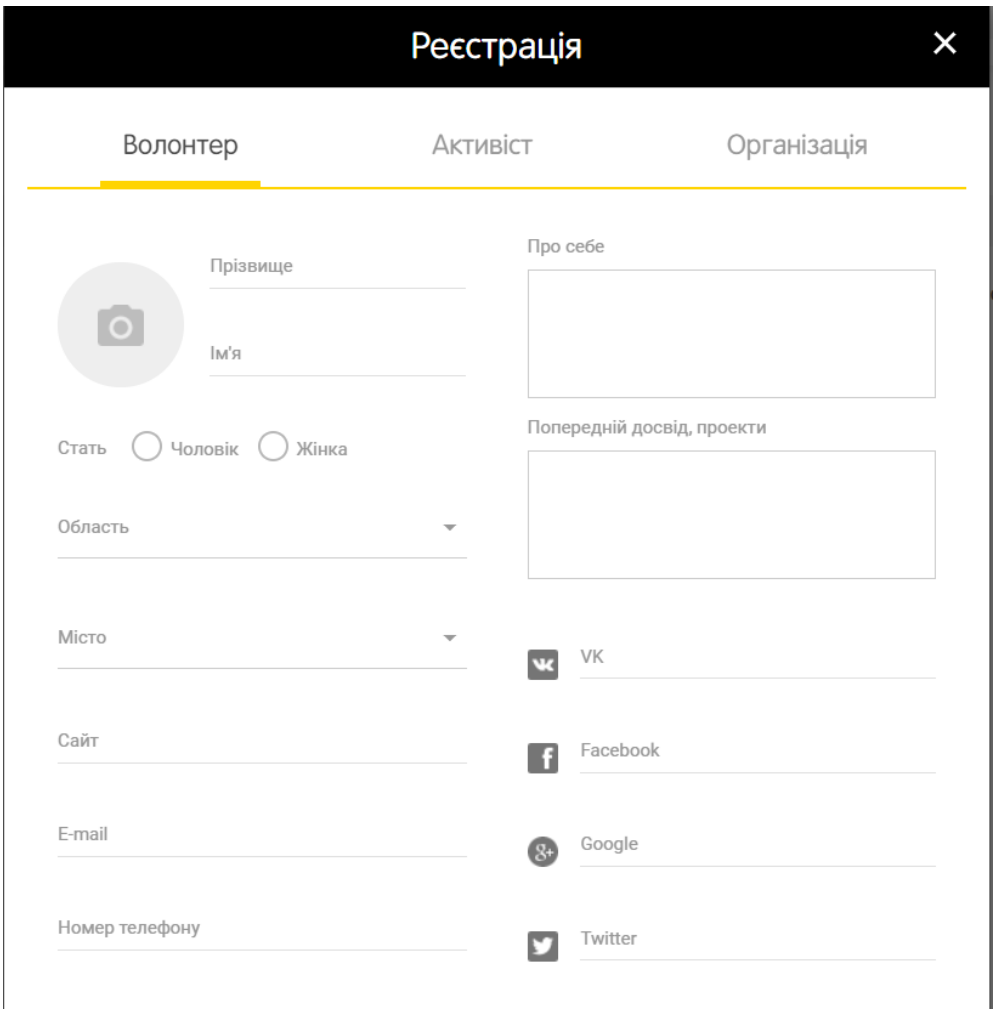

Рисунок 1.6 – Сторінка реєстрації на платформі volonter.org

Helplist.io – це платформа, де є можливість звертатися за допомогою, натискаючи кнопку "Створити запит" і повідомляти, що вам необхідно. Якщо ви волонтер або фонд, ви можете вибрати місто та категорію, щоб побачити, хто потребує вашої допомоги у вашому місті. Це дозволяє зв'язати тих, хто надає допомогу, з тими, хто її потребує, забезпечуючи ефективну організацію та координацію волонтерської діяльності.

Головну сторінку платформи Helplist.io зображено на рисунку 1.7.

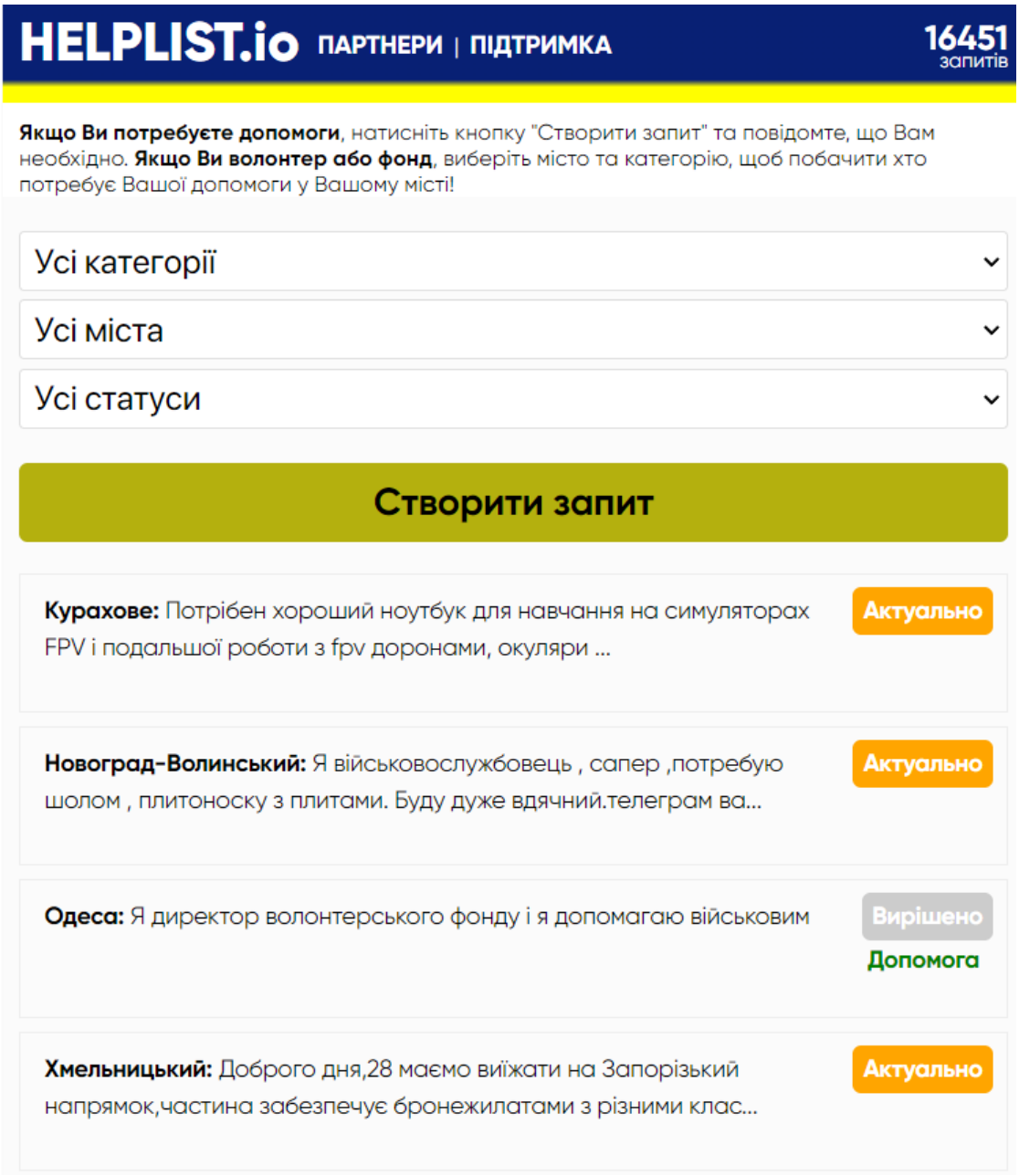

Рисунок 1.7 – Головна сторінка платформи Helplist.io

1.3 Постановка задачі

Задля досягнення вищезазначеної мети кваліфікаційної роботи необхідно:

- визначити та проаналізувати функціональні та нефункціональні вимоги до інформаційної системи;
- обрати методи та засоби реалізації;
- визначити структуру інформаційної системи та розробити її прототип;
- розробити модель бази даних;
- здійснити верстку та програмування;
- провести тестування інформаційної системи.

Використання розробленої інформаційної системи спрямовано на автоматизацію виконуваних бізнес-процесів.

До основних можливостей, що мають бути реалізовані при розробці належать:

– Реєстрація користувачів та волонтерів: Можливість створення облікових записів для організацій, користувачів та волонтерів з унікальними ідентифікаторами та обліковими даними;

– Створення запитів на допомогу: Функціональність, що дозволяє користувачам ініціювати прохання про допомогу через інформаційну систему, надаючи детальну інформацію про потрібну допомогу;

– Управління профілями користувачів: Функціонал для зміни, оновлення та управління особистою інформацією користувачів та волонтерів, включаючи контактні дані, навички, інтереси тощо;

– Система комунікації: Можливість взаємодії та спілкування між користувачами, волонтерами та організаторами за допомогою внутрішньої системи повідомлень, чату або електронної пошти;

– Аналітика та звітність: Функціональність для аналізу та візуалізації даних про волонтерську діяльність.

З огляду на кількість заявленого функціоналу, неможливо уникнути бізнес-ризиків, а також існує потреба виділити критерії успіху, за яких проект можна вважати прибутковим.

Бізнес-ризики - це потенційні негативні події або обставини, які можуть виникнути у процесі діяльності підприємства і призвести до збитків, втрат чи інших небажаних наслідків для бізнесу. Ці ризики можуть походити з різних джерел і охоплюють широкий спектр аспектів бізнесу, включаючи фінансові, організаційні, правові, технологічні, репутаційні та інші. В початковій версії застосунку існують наступні бізнес ризики:

– крадіжка даних;

– невідповідність кількості запитів та тих, хто спроможний і прагне їх виконати;

– розміщення скаму;

– зловживання службовим становищем адміністраторами;

– підробка особи при реєстрації.

В подальших версіях окреслені ризики мають бути усунені або мають бути додані механізм їх запобігання.

Проект може вважатись успішним за умов:

– покриття базових витрат на розгортку, підтримку проекту за рахунок розміщеної реклами;

– відсоткова перевага закритих запитів над відкритими.

З метою уникнення подальших судових позовів важливо зазначити, що проект має власні обмеження. До таких обмежень належать:

– проект не забезпечує потреби реципієнтів безпосередньо;

– проект не має поширюватись на такі країни як росія та Білорусь.

Ні в початковій, ні в подальших версіях програмного продукту дані обмеження не будуть усунені.

Додатково до перерахованих ризиків і умов успішності проекту, можна розглянути деякі аспекти, що стосуються функціональності та ефективності самого продукту:

- Вдосконалення функціоналу;
- Оптимізація продуктивності;
- Покращення безпеки;
- Збільшення аудиторії та розширення географічного покриття;
- Підтримка та взаємодія з користувачами.

2 РОЗРОБКА ФУНКЦІОНАЛЬНИХ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Архітектура додатку

Розроблювана система архітектурно має бути:

– гнучкою;
– розподіленою;
– надійною з високою доступністю відповідей сервісів;
– легкою для розгортання.

Гнучка архітектурно система – це система, яка володіє властивостями відкритості та адаптивності до змін. Основні характеристики гнучкої архітектури включають:

– модульність;
– легкість розширення;
– гнучкість використання технологій;
– гнучкість використання технологій.

Розподілена архітектурна система - це система, яка складається з різних компонентів або модулів, які розгорнуті на різних фізичних або логічних вузлах мережі та взаємодіють між собою через мережу. Кожен компонент може виконуватися на власному сервері або вузлі і мати власний процес чи потік виконання.

З урахуванням вимоги потреби в розподіленій системі, при розробці архітектури важливо уникати надмірної деталізації, побудови сплутаних взаємозв’язків, спільних БД, порушення правил єдиної відповідальності, каскадних викликів, створення монолітів.

Надійна система з високою доступністю відповідей сервісів - це система, яка забезпечує неперервний доступ до своїх сервісів і функціональностей

навіть у випадку виникнення помилок або збоїв. Для досягнення цієї мети використовуються різні стратегії та методики, серед яких:

– Резервне копіювання: Дублювання компонентів системи та резервних каналів зв'язку, щоб забезпечити продовження роботи навіть у випадку відмови або втрати одного з елементів;

– Балансування навантаження: Розподіл навантаження між різними екземплярами або регіонами для запобігання перевантаженню будь-якого конкретного компонента;

– Автоматичне масштабування: Автоматичне збільшення або зменшення кількості ресурсів системи в залежності від поточного навантаження, щоб забезпечити стабільну продуктивність;

– Моніторинг та реагування: Постійний моніторинг стану системи та автоматичне сповіщення про можливі проблеми або відхилення в роботі;

– Запасні резервні копії даних: Регулярне створення резервних копій даних та розробка планів відновлення під час втрати даних.

Легка для розгортання система - це система, яка має простий та швидкий процес встановлення та налаштування на цільовому середовищі. Для досягнення цієї характеристики системи часто використовують такі методи та підходи:

– Контейнеризація: Використання контейнерних технологій, таких як Docker, для упаковки програмного забезпечення та всіх його залежностей у відокремлені контейнери, що полегшує процес розгортання;

– Інфраструктура як код: Використання інструментів автоматизації, таких як Ansible, Terraform або Chef, для опису інфраструктури системи у вигляді коду, що дозволяє легко та швидко створювати та налаштовувати потрібне середовище;

– Хмарні сервіси: Використання платформи хмарних обчислень, такої як Amazon Web Services (AWS), Microsoft Azure або Google Cloud

Platform (GCP), для автоматизації розгортання та керування інфраструктурою безпосередньо з хмарного середовища;

– Мікросервісна архітектура: Розбиття системи на невеликі та незалежні мікросервіси дозволяє розгортати їх окремо та швидко, оскільки кожен мікросервіс може бути розгорнутий незалежно один від одного;

– Автоматизовані сценарії розгортання: Розробка автоматизованих сценаріїв розгортання, які дозволяють автоматично встановлювати та налаштовувати систему без необхідності вручного втручання.

З огляду на вищезазначені умови в постановці завдання до архітектури застосунку, на рисунку 2.1 представлена загальна архітектурна діаграма.

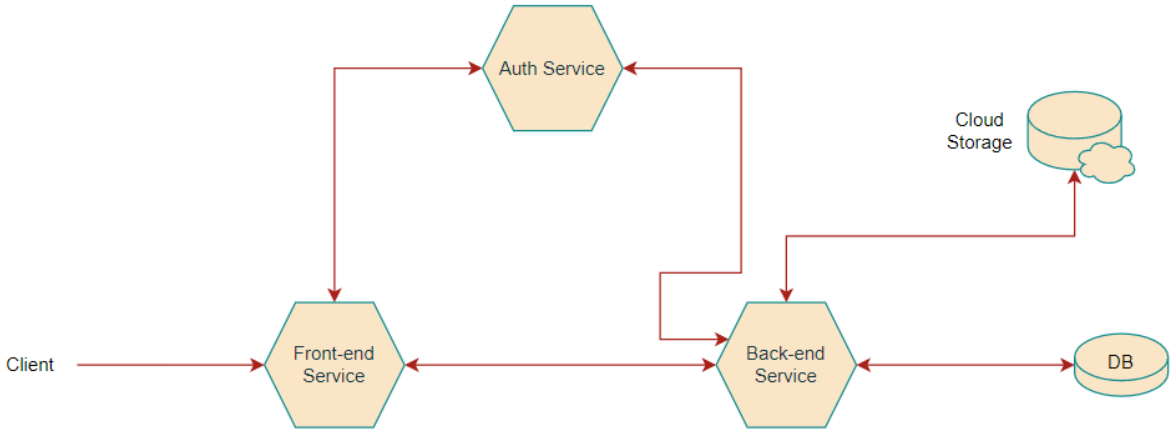

Рисунок 2.1 – Загальна архітектура системи

Як показано на діаграмі та з урахуванням вимог, архітектура застосунку є мікросервісною з розподіленим фронт-енд та бекенд-енд додатку.

Мікросервісна архітектура є підходом до структурування програмного забезпечення, де додатки розбиваються на невеликі і незалежні компоненти, відомі як мікросервіси. Кожен мікросервіс функціонує як окремий процес, що може бути розгорнутий, масштабований і керований незалежно від інших. Це дозволяє розробникам концентруватись на конкретних функціях або послугах, замість того щоб вирішувати всі аспекти великого монолітного додатку.

Кожен мікросервіс має власний набір функцій і може використовувати різні технології і мови програмування в залежності від його специфічних потреб. Це сприяє полегшенню розробки, вдосконаленню тестування і розвитку мікросервісів незалежно один від одного. Кожен мікросервіс може бути розгорнутий в окремому контейнері, що забезпечує ізольованість та надійність роботи.

Окремим сервісом в інформаційні системі також представлений сервіс автентифікації, який відповідає за перевірку та підтвердження ідентичності користувачів, які намагаються отримати доступ до системних ресурсів чи функціоналу. Дана система має бути реалізована у вигляді веб-додатку.

2.2 Вимоги до БД

В базі даних має міститися інформація про користувачів системи, організації, запити про допомогу, участь користувачів або в запитах про допомогу та дані про виконання запиту.

Er-діаграму бази даних зображено на рисунку 2.2.

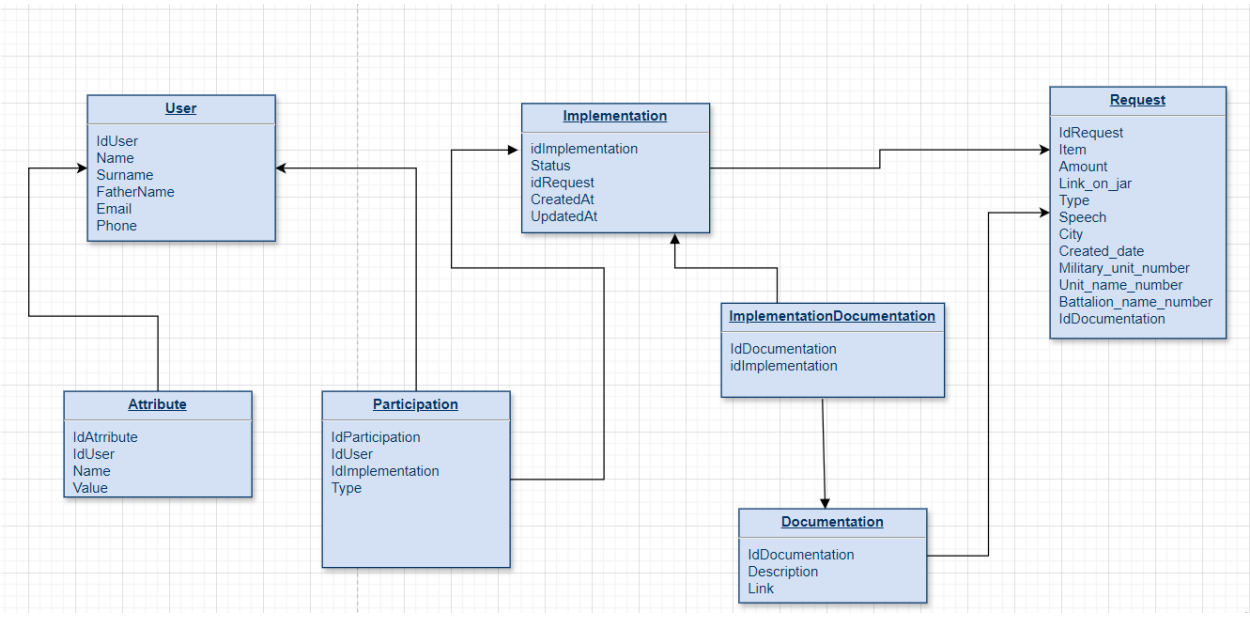

Рисунок 2.2 – Er-діаграма

Дані користувачів та організацій мають включати:

– idUser – ідентифікатор користувача;

– Name – ім’я користувача;

– Email – пошта користувача, що має слугувати додатковими індексом та бути унікальною;

– Phone – номер телефону користувача.

Всі інші додаткові дані користувачів та організацій зберігатимуться в атрибутах. Атрибути можуть включати в себе:

– IdAtribute – ідентифікатор атрибуту;

– Name – назва атрибуту;

– Value – значення атрибуту.

До атрибутів можуть входити такі дані:

– Surname – прізвище користувача;

– Fathername – по-батькові користувача;

– IPN – ідентифікаційний код користувача;

– Date_registred – дата реєстрації організації;

– EDRPOU – ЄДРПОК код організації.

Дані про запит на допомогу мають включати:

– IdRequest – ідентифікатор запиту;

– Item – предмети, що включені до запиту;

– Amount – кількість предметів;

– Link_on_jar – посилання на банку для збору;

– Type – тип запиту;

– Speech – звернення від того, хто створив запит;

– City – зручне місце для отримання предметів запиту;

– CreatedDate – дата створення запиту;

– Military_unit_number – номер військової частини;

– Unit_name_number – назва підрозділу;

– Battalion_name_number – номер або назва батальйону;

Дані про виконання запитів мають включати:

– idImplementation – ідентифікатор виконання;

– Status – статус виконання запиту;

– idRequest – ідентифікатор запиту про допомогу;

– Created_date – дата створення;

– Updated_date – дата останнього оновлення;

Дані про документи мають включати:

– idDocument – ідентифікатор документу;

– Description – опис документу;

– Link– посилання на документ;

Розроблювана база даних має забезпечувати:

– Надійність та цілісність даних: Гарантувати, що дані зберігаються та обробляються надійно, з урахуванням усіх необхідних механізмів транзакцій та обмежень цілісності;

– Ефективність: Забезпечувати швидкий доступ до даних та оптимізовані запити, щоб база даних працювала ефективно, навіть при великому обсязі інформації;

– Масштабованість: Бути готовою до масштабування, щоб забезпечити обробку зростаючого обсягу даних та користувачів без втрати продуктивності.

– Безпека: Гарантувати захищеність даних від несанкціонованого доступу та зловживання, використовуючи механізми автентифікації, авторизації та шифрування;

– Легкість розширення та змін: Бути легкою для розширення та змін, щоб вона відповідала змінюваним потребам бізнесу та могла включати нові функціональні можливості;

– Сумісність: Бути сумісною з існуючим програмним забезпеченням та середовищем розробки для безпроблемної інтеграції та взаємодії з іншими компонентами системи.

2.3 Вимоги до користувацького інтерфейсу

Інтерфейс користувача повинен відповідати наступним вимогам:

– Зручність. Використання сформованими та звичними користувачеві стандартами користування веб додатками. Посилання, кнопки і форми повинні бути виділені, мати оптимальний розмір, щоб користувач міг з легкістю переходити по ним;

– Архітектура інформації. Структура зображених елементів повинна бути максимально простою та зручною. На екрані має бути оптимальна кількість позначень і кнопок які будуть вказувати користувачу куди треба натиснути, вони роблять інтерфейс інтуїтивно зрозумілим та легким;

– Дизайн. Дизайн веб-додатка повинен бути простим, сучасним, мінімалістичним та інтуїтивно зрозумілим. Кольорове рішення має бути в м’яких тонах, приємних для сприйняття. Усі сторінки повинні бути виконані в єдиному стилі. Кольори та конструкція елементів повинні бути вибрані так, щоб забезпечити чітку видимість та контрастність, особливо для людей з обмеженим зором;

– Користувацький досвід. Користувач повинен мати змогу дістатися до бажаного місця чи сторінки сайту у «два кліки». Необхідно зменшити кількість ситуацій які можуть спровокувати користувача довго думати або приймати рішення що він повинен робити далі;

– Швидкість. Стримане використання анімацій та ефектів для забезпечення швидкості завантаження та реакції на дії користувачів;

– Інтеграцію з іншими сервісами. Можливість легкої інтеграції з іншими сервісами та платформами для зручного обміну даними та функціональністю.

3 ВИЗНАЧЕННЯ АЛГОРИТМІВ РОЗРОБЛЮВАНОЇ СИСТЕМИ

3.1 Опис алгоритмів головних бізнес-процесів розроблюваного веб-застосунку

У контексті розробки та дослідження інформаційної системи організації та координації волонтерської діяльності, бізнес-процеси визначають послідовність дій, необхідних для успішного функціонування та досягнення цілей системи. Основні бізнес-процеси в цьому контексті включають:

- Реєстрація користувачів: процес реєстрації нових користувачів у системі, включаючи збір та перевірку особистих даних;
- Створення запитів про допомогу: процес створення нових запитів про допомогу або проектів, які потребують волонтерської підтримки;
- Пошук волонтерів: процес пошуку та залучення волонтерів для участі у різних ініціативах та проектах;
- Координація діяльності: процес забезпечення ефективної комунікації та спільної роботи між учасниками проектів та організаторами;
- Моніторинг та звітність: процес відстеження прогресу виконання проектів, збір даних та генерація звітів для аналізу та вдосконалення роботи системи.

Для візуалізації процесів було обрано нотацію bpmn. BPMN (Business Process Model and Notation) є інструментом для моделювання та оптимізації бізнес-процесів. Ця нотація використовується не лише для документування процесів, але й для їх аналізу з метою підвищення ефективності. Вона забезпечує єдину мову, зрозумілу як бізнес-аналітикам, так і технічним фахівцям, що сприяє кращій співпраці між різними відділами в організації.

BPMN дозволяє детально описувати складні процеси, включаючи їхні етапи, ролі та взаємодії. Використання стандартних символів і елементів допомагає уникнути неоднозначностей та помилок, що можуть виникнути при

використанні менш формалізованих методів опису процесів. Це сприяє кращому розумінню і спільному використанню моделей процесів у великих командах.

Однією з ключових переваг BPMN є можливість інтеграції з іншими системами та інструментами управління бізнес-процесами. Це дозволяє автоматизувати певні аспекти процесів, що в свою чергу підвищує продуктивність і зменшує ризик людських помилок. Зокрема, BPMN використовується для створення специфікацій, які можуть бути безпосередньо імплементовані в системах управління бізнес-процесами (BPMS).

BPMN підтримує різні види подій і шлюзів, які допомагають моделювати як прості, так і складні логічні умови та потоки. Це включає в себе обробку виняткових ситуацій, умовні розгалуження і паралельні потоки, що робить BPMN надзвичайно гнучким для опису будь-яких бізнес-процесів.

Процес реєстрації зображено на рисунку 3.1.

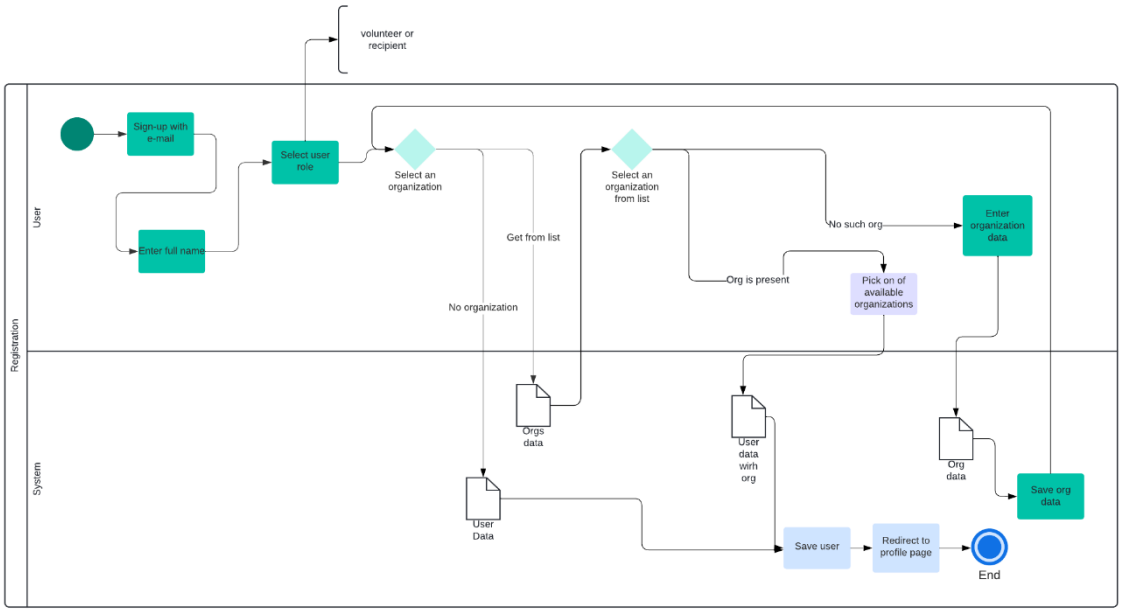

Рисунок 3.1 – Діаграма bpmn 2.0 реєстрації

Процес створення запиту про допомогу зображено на рисунку 3.2

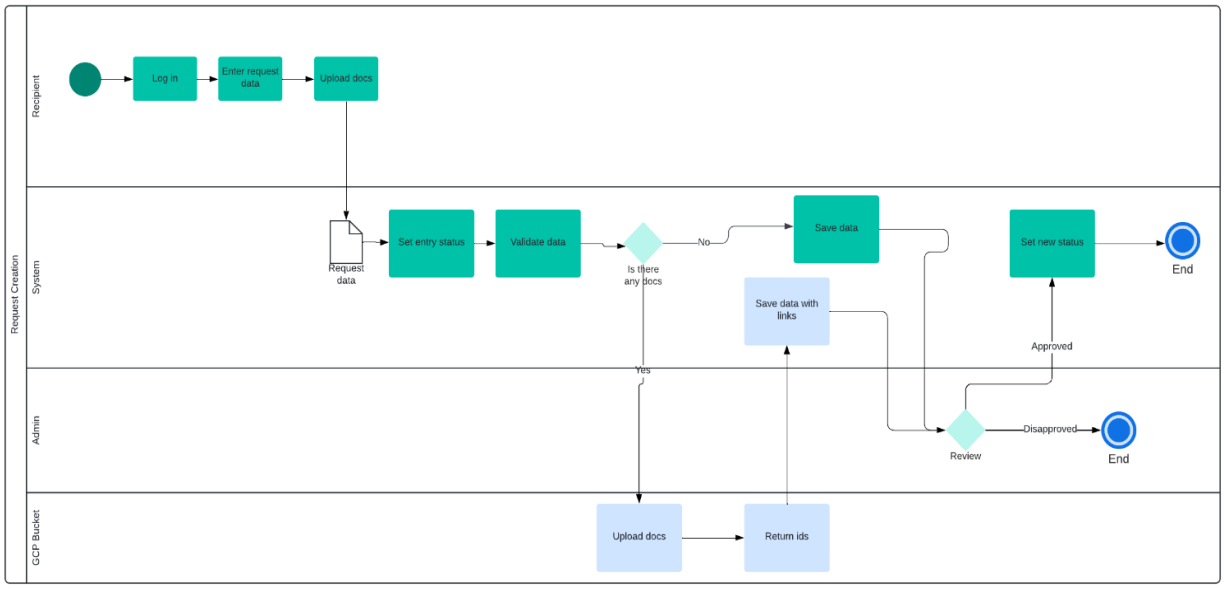

Рисунок 3.2 – Діаграма bpmn 2.0 створення запиту про допомогу

Ці бізнес-процеси спрямовані на забезпечення ефективного управління та розвитку волонтерських ініціатив, сприяючи координації зусиль та досягненню мети системи - організації та підтримки волонтерської діяльності.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Обгрунтування вибору мови програмування

Для написання бекенду інформаційної системи було обрано мову програмування Python.

Python - це потужна і популярна мова програмування, яка використовується для різноманітних цілей, від веб-розробки до наукових обчислень і штучного інтелекту. Вона має чистий і простий синтаксис, що дозволяє швидко вчитися новачкам і підтримує об'єктно-орієнтоване, процедурне і функціональне програмування. Python активно використовується у великих корпораціях, таких як Google, Facebook, Dropbox і інших, завдяки своїй ефективності і простоті в розробці і є платформенно незалежною. Мова має велику стандартну бібліотеку, яка включає інструменти для роботи з рядками, числами, списками, файлами і багато чого іншого. Через свою строгість у використанні пробілів і відступів Python робить код більш читабельним.

Python підтримує модульність, що дозволяє розширювати функціональність програм і має вбудовану підтримку для створення тестів і одиниць коду, що сприяє стабільності програмного забезпечення. Вона активно використовується для розробки веб-додатків за допомогою фреймворків і має великий вибір інтегрованих розробничих середовищ і текстових редакторів. Python підтримує паралельне програмування і многопотоковість для прискорення обчислень і має вбудовану підтримку для обробки винятків і відлагодження. Мова активно використовується в наукових обчисленнях і обробці даних завдяки бібліотекам для математики, фізики і біології.

Python має великий вибір інструментів для візуалізації даних і побудови графіків, що спрощує аналіз і представлення інформації. Мова підтримує

обробку регулярних виразів для зручного пошуку і обробки текстової інформації і має вбудовану підтримку для взаємодії з базами даних. Python підтримує розробку мобільних додатків і має велику спільноту розробників і користувачів, що активно допомагає вирішувати проблеми і обмінюватися досвідом.

Python підтримує інтеграцію з багатьма іншими мовами програмування, такими як C, C++, і Java, що дозволяє розширювати його можливості і використовувати в різноманітних проектах. Завдяки цьому Python може ефективно взаємодіяти з іншими системами та використовувати вже існуючий код, що спрощує процес розробки та знижує витрати на його реалізацію. Вона також підтримує автоматизацію рутинних завдань і скриптінг, що робить її популярною серед системних адміністраторів і DevOps-інженерів.

Python відомий своєю здатністю до швидкої розробки прототипів, що дозволяє розробникам тестувати ідеї і створювати MVP (мінімально життєздатні продукти) з мінімальними зусиллями. Це особливо важливо для стартапів і невеликих компаній, які прагнуть швидко виходити на ринок. Завдяки активному розвитку і частим оновленням, Python залишається актуальним і відповідає сучасним вимогам індустрії програмного забезпечення. Мова має багатофункціональні фреймворки для машинного навчання та штучного інтелекту, такі як TensorFlow і PyTorch, що робить її незамінною у сучасних технологічних розробках.

Для написання фронтенду інформаційної системи було обрано React JS, що буде розгорнутий на Node JS.

Node.js - це середовище виконання JavaScript, яке використовує рушій V8, розроблений компанією Google. Воно дозволяє розробникам використовувати JavaScript для серверного програмування, що робить можливим створення повного стеку додатків на одній мові. Node.js відомий своєю асинхронною архітектурою і обробкою подій, що забезпечує високу продуктивність і масштабованість при обробці великої кількості запитів. Це

робить його ідеальним для створення мережевих додатків, веб-серверів, API і різноманітних серверних додатків.

Node.js володіє розширеними можливостями, що дозволяють розробникам швидко і ефективно розгортати застосунки, орієнтовані на події. Це досягається завдяки моделі подій, яка дозволяє обробляти запити асинхронно та реагувати на події платформи без блокування і використання зайвих ресурсів. Такий підхід особливо важливий у високонавантажених застосунках, де кожна мілісекунда може мати значення для швидкодії і масштабованості системи.

Ще однією перевагою Node.js є активна спільнота розробників, яка підтримує фреймворк і постійно вносить в нього нові ідеї та покращення. Це забезпечує швидкий розвиток екосистеми Node.js, розширення функціональності і підтримку новітніх стандартів JavaScript. Розширеність пакетного менеджера npm також сприяє швидкому поширенню корисних модулів і бібліотек, що спрощує розробку застосунків і дозволяє зосередитися на основній функціональності.

Node.js підтримує широкий спектр інструментів для розробки, включаючи різноманітні фреймворки, що дозволяють швидко створювати серверні додатки з необхідною функціональністю, такою як маршрутизація, обробка запитів і відповідей, обробка помилок та інші аспекти веб-розробки. Це дозволяє розробникам вибирати оптимальний інструментарій для вирішення конкретних завдань і забезпечує гнучкість у виборі технологій.

Однією з істотних особливостей Node.js є його підтримка модульності і відкритих стандартів, що сприяє перевикористанню коду і підтримці чистоти архітектури додатків. Модулі Node.js можуть бути легко інтегровані в існуючі системи або використовуватися для розробки незалежних компонентів, що спрощує розширення функціоналу і підтримку складних проектів.

Node.js є інструментом високої продуктивності для розробки сучасних серверних додатків, який поєднує ефективність розробки і широкі можливості для інтеграції з іншими технологіями.

React.js - це бібліотека JavaScript для створення інтерфейсів користувача, що зосереджується на компонентах. Вона розроблена компанією Facebook і використовується для побудови односторінкових додатків інтерфейсів, які змінюються динамічно без перезавантаження сторінки. React базується на концепції "однієї проблеми - один компонент", що сприяє чіткості і легкості розробки.

Однією з ключових особливостей React є використання віртуального DOM (Document Object Model), який дозволяє ефективно оновлювати лише змінені елементи на сторінці, замість повного перерендеринга. Це забезпечує високу швидкодію і плавність роботи додатків, особливо у великих проектах зі складним інтерфейсом.

React пропонує можливості для компонентного підходу до розробки, що дозволяє використовувати повторно визначені компоненти для побудови складних інтерфейсів. Компоненти можуть бути класовими або функціональними, залежно від потреб проекту, і кожен компонент може мати власний стан, що дозволяє відслідковувати зміни і взаємодіяти з користувацькими діями.

React також підтримує використання JSX (JavaScript XML), який є розширенням синтаксису JavaScript і дозволяє вбудовувати HTML-подібний код безпосередньо в JavaScript. Це полегшує створення компонентів і підвищує читабельність коду, зберігаючи його на одному рівні.

React надає широкі можливості для управління станом додатків і компонентів, включаючи передачу пропсів (props) для передачі даних між компонентами і використання власного стану для локального зберігання даних. Це дозволяє створювати інтерактивні інтерфейси, які реагують на дії користувачів і взаємодіють з серверними данними асинхронно.

React також відомий своєю активною спільнотою розробників, яка постійно вносить нові ідеї і покращення у фреймворк. Ця спільнота активно співпрацює над вдосконаленням інструментів і бібліотек, що дозволяє React залишатися сучасним і відповідати потребам ринку програмного забезпечення.

Крім того, React активно використовується такими компаніями, як Facebook, Instagram, Airbnb і багатьма іншими великими гравцями індустрії. Його популярність у великих корпораціях свідчить про його надійність, масштабованість і здатність вирішувати складні завдання.

Окрім стандартних інструментів і бібліотек, у React є розширена підтримка для тестування, що дозволяє розробникам забезпечувати якість свого коду і впевненість у відповідності до вимог. Велика увага до тестування дозволяє покращувати стабільність і надійність додатків на основі React у будь-якому середовищі.

React є добре документованим фреймворком з багатьма прикладами і ресурсами для самостійного вивчення. Це дозволяє новим розробникам швидко освоювати його основи і впроваджувати рішення на практиці. Доступність якісної документації сприяє популяризації React і забезпечує стабільність в його використанні в професійних проектах.

Деталізовану архітектуру системи зображено на рисунку 2.2.

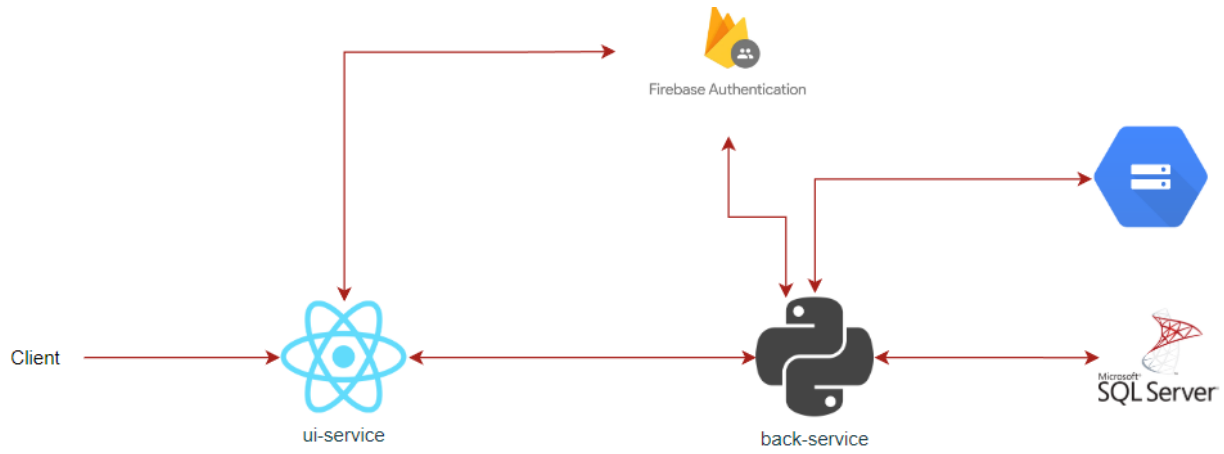

Рисунок 4.1 – Деталізована архітектура системи

4.2 Обгрунтування вибору фреймворку

В якості фреймворку було обрано FastApi.

FastAPI - це сучасний, високопродуктивний веб-фреймворк для побудови API на Python, який створено з урахуванням найновіших стандартів і найкращих практик. Він дозволяє розробникам швидко і ефективно створювати веб-додатки і RESTful API з мінімальними зусиллями, завдяки своїй простоті та інтуїтивно зрозумілому інтерфейсу. Використовуючи асинхронну модель програмування, FastAPI забезпечує високий рівень продуктивності та масштабованості додатків, що робить його ідеальним для високонавантажених сервісів.

Однією з головних переваг FastAPI є його підтримка автоматичної генерації документації для API за допомогою інтеграції з OpenAPI та JSON Schema. Це дозволяє розробникам легко створювати та підтримувати актуальну документацію, що спрощує взаємодію з іншими командами та зовнішніми користувачами. FastAPI також підтримує типізацію на основі анотацій типів Python, що сприяє зменшенню кількості помилок та підвищенню якості коду.

FastAPI забезпечує високу продуктивність завдяки використанню асинхронних можливостей Python та Uvicorn як ASGI-сервера. Це дозволяє обробляти велику кількість запитів одночасно без значного зниження продуктивності. Крім того, фреймворк підтримує інтеграцію з різними інструментами та бібліотеками, такими як SQLAlchemy, Pydantic і Alembic, що полегшує роботу з базами даних та валідацією даних.

Важливою особливістю FastAPI є його здатність працювати з WebSockets, що дозволяє створювати реального часу додатки, такі як чати, онлайн-ігри та інші інтерактивні сервіси. Фреймворк також підтримує OAuth2 та JWT для аутентифікації і авторизації, що забезпечує високий рівень безпеки для користувачів.

FastAPI надає гнучкі можливості для налаштування маршрутизації, що дозволяє легко організувати і обробляти різні HTTP-запити. Завдяки підтримці асинхронних завдань, розробники можуть виконувати фонові задачі без блокування основного потоку, що значно покращує загальну продуктивність додатка. Крім того, FastAPI підтримує middleware, що дозволяє додавати додаткові функції, такі як логування, обробка помилок та безпека, без зміни основного коду додатка.

Фреймворк також забезпечує високу ступінь модульності, що дозволяє розділяти великий код на менші, легкі для управління модулі. Це спрощує розробку та тестування окремих компонентів додатка, підвищуючи якість кінцевого продукту. FastAPI інтегрується з інструментами тестування, такими як pytest, що дозволяє легко писати і виконувати тести для забезпечення надійності і стабільності коду.

Однією з сильних сторін FastAPI є його підтримка для міжнародних стандартів, таких як JSON

і GraphQL, що дозволяє створювати API, сумісні з різноманітними клієнтськими додатками та сервісами. Фреймворк також підтримує інтеграцію з різними системами черг, такими як RabbitMQ і Kafka, що дозволяє створювати високонадійні системи з обробкою подій у реальному часі.

FastAPI пропонує зручні засоби для конфігурації та управління середовищем виконання, що дозволяє легко налаштовувати додаток під різні умови використання. Завдяки інтеграції з контейнеризаційними платформами, такими як Docker, розробники можуть швидко розгортати і масштабувати свої додатки у хмарних середовищах. Це забезпечує високу гнучкість і ефективність у розробці, тестуванні та розгортанні веб-додатків.

4.3 Імплементація безпеки застосунку

Як вже зазначалося раніше, інформаційна система має мікросервісну архітектуру та складається з фронтенд сервісу, бекенд сервісу та сервісу авторизації. В якості сервісу авторизації було обрано вже готовий сервіс Firebase Auth.

Firebase Authentication (Firebase Auth) - це інструмент для аутентифікації користувачів, розроблений компанією Google. Він забезпечує простий і безпечний спосіб управління автентифікацією користувачів для веб-додатків і мобільних додатків на різних платформах, таких як Android, iOS і веб. Firebase Auth підтримує кілька методів входу, включаючи електронну пошту та пароль, телефонний номер, а також інтеграцію з популярними постачальниками ідентифікації, такими як Google, Facebook, Twitter та GitHub.

Однією з головних переваг Firebase Auth є його простота у використанні, що дозволяє розробникам швидко додавати функції автентифікації до своїх додатків без необхідності самостійно розробляти складні системи безпеки. Firebase Auth також підтримує одноразові паролі (OTP) для аутентифікації через SMS, що забезпечує додатковий рівень безпеки для користувачів.

Firebase Auth надає вбудовані механізми для управління сесіями користувачів, включаючи підтримку токенів доступу і оновлення, що дозволяє створювати безпечні і надійні додатки. Крім того, він інтегрується з іншими сервісами Firebase, такими як Firebase Firestore і Firebase Realtime Database, що спрощує управління даними користувачів і їх авторизацію у ваших додатках.

Firebase Auth також забезпечує підтримку анонімного входу, що дозволяє користувачам взаємодіяти з додатком без необхідності реєстрації. Це корисно для додатків, де користувачі можуть захотіти спробувати функціонал перед тим, як створити повноцінний обліковий запис. Крім того, Firebase Auth підтримує зручні засоби для налаштування та локалізації сторінок входу, що забезпечує зручний користувацький досвід.

Важливою особливістю Firebase Auth є його здатність до безшовної інтеграції з іншими сервісами Google Cloud, що дозволяє розробникам використовувати єдину платформу для управління всіма аспектами своїх додатків. Firebase Auth також підтримує роботу з фреймворками для побудови інтерфейсів, такими як Angular, React і Vue.js, що спрощує інтеграцію автентифікації в сучасні веб-додатки.

Firebase Auth забезпечує високу продуктивність і надійність завдяки використанню інфраструктури Google, що гарантує стабільну роботу додатків навіть при високих навантаженнях. Крім того, Firebase Auth надає регулярні оновлення і підтримку від Google, що забезпечує актуальність і безпеку вашого додатка.

Завдяки своїм можливостям та інтеграції з іншими сервісами, Firebase Auth є відмінним вибором для розробників, які шукають простий і ефективний спосіб додавання функцій аутентифікації до своїх додатків. Він дозволяє зосередитися на розробці основного функціоналу, забезпечуючи при цьому високий рівень безпеки і зручності для користувачів.

Для інтеграції Firebase Auth з інформаційною системою потрібно створити власний проект та налаштувати його. Створений проект зображено на рисунку 4.2.

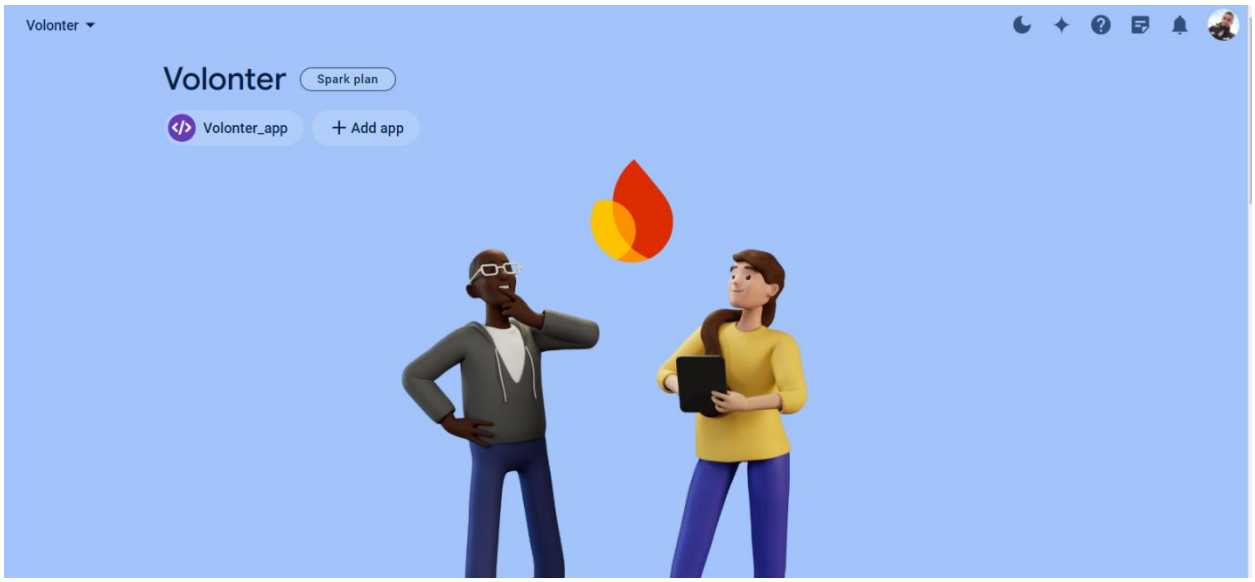

Рисунок 4.2 – Створений проект в Firebase Auth

Серед провайдерів для авторизації, було обрано Google провайдер. З’єднання з ним здійснюється через протокол OAuth 2.0.

OAuth 2.0 (Open Authorization 2.0) - це стандартний протокол авторизації, який дозволяє додаткам отримувати обмежений доступ до користувацьких облікових записів на веб-сервісах без необхідності передачі паролів. Це забезпечує більш безпечний і зручний спосіб авторизації, дозволяючи користувачам надавати і відкликати доступ до своїх даних на сторонніх сервісах. OAuth 2.0 використовується багатьма великими компаніями, такими як Google, Facebook, Microsoft та інших, для забезпечення безпечної авторизації в їх додатках і сервісах.

Однією з головних переваг OAuth 2.0 є його здатність забезпечувати доступ до ресурсів без необхідності розкривати облікові дані користувачів. Це досягається шляхом використання токенів доступу, які представляють собою тимчасові ключі, що надають обмежений доступ до ресурсів користувача. Ці токени можуть бути відкликані або оновлені без необхідності зміни основних облікових даних.

OAuth 2.0 підтримує кілька типів грантів (grant types), які визначають різні способи отримання токенів доступу. Основні типи грантів включають Authorization Code Grant, Implicit Grant, Resource Owner Password Credentials Grant і Client Credentials Grant. Кожен з цих типів грантів підходить для різних сценаріїв використання, що робить OAuth 2.0 гнучким рішенням для різних типів додатків.

Authorization Code Grant є найбільш безпечним типом гранту і зазвичай використовується для веб-додатків, де клієнтська частина не може безпосередньо обробляти облікові дані користувача. В цьому сценарії користувач перенаправляється на сторінку авторизації постачальника ідентифікації, де вводить свої облікові дані. Після успішної авторизації користувач отримує код авторизації, який клієнтський додаток обмінює на токен доступу.

Implicit Grant зазвичай використовується для односторінкових додатків (SPA), де безпека не є критичною, оскільки токен доступу повертається безпосередньо в URL-адресу редиректу. Resource Owner Password Credentials Grant дозволяє клієнтським додаткам, які довіряють користувачам, отримувати токени доступу, використовуючи облікові дані користувача безпосередньо. Client Credentials Grant використовується для серверних додатків, де клієнт отримує доступ до власних ресурсів або до ресурсів інших користувачів за допомогою своїх облікових даних.

Однією з важливих особливостей OAuth 2.0 є його здатність працювати з різними типами клієнтів, включаючи веб-додатки, мобільні додатки та десктопні додатки. Це забезпечує універсальність протоколу і дозволяє використовувати його в широкому спектрі додатків і сервісів. OAuth 2.0 також підтримує розширення, що дозволяє адаптувати протокол до специфічних вимог і сценаріїв використання.

OAuth 2.0 має потужні засоби для управління сесіями користувачів і забезпечення безпеки, такі як обмеження терміну дії токенів і можливість відкликання токенів у будь-який момент. Це забезпечує високий рівень безпеки і контролю над доступом до ресурсів користувача. Крім того, протокол OAuth 2.0 підтримує сучасні засоби шифрування і захисту даних, що забезпечує надійний захист від несанкціонованого доступу і зловживань.

Процес приєднання Google провайдера до Firebase Auth зображено на рисунку 4.3.

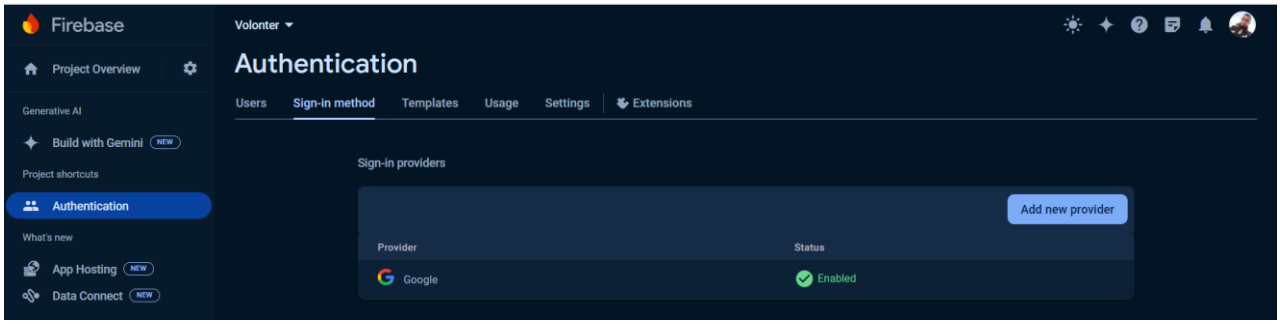

Рисунок 4.3 – Приєднання Google провайдера до Firebase Auth

Наступним кроком є налаштування проекту в Firebase Auth. Даний процес зображено на рисунку 4.4

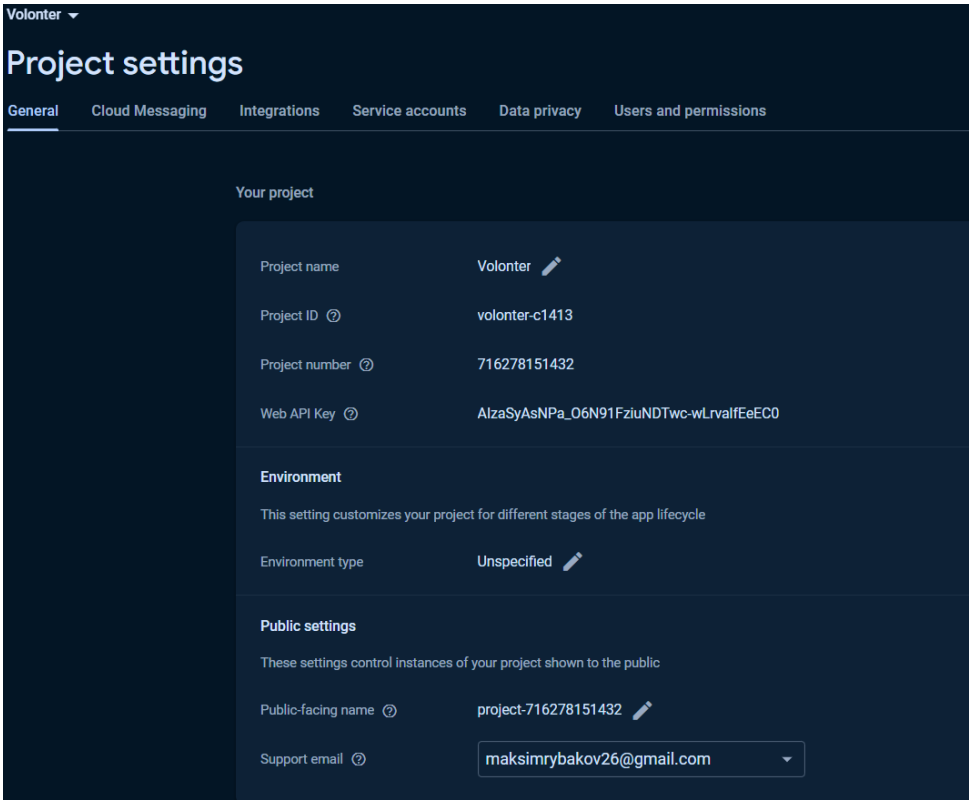

Рисунок 4.4 – Налаштування проекту Firebase Auth

Аутентифікаційний запит здійснюється через UI сервіс. Приклад такого запиту наведено на рисунку 4.5.

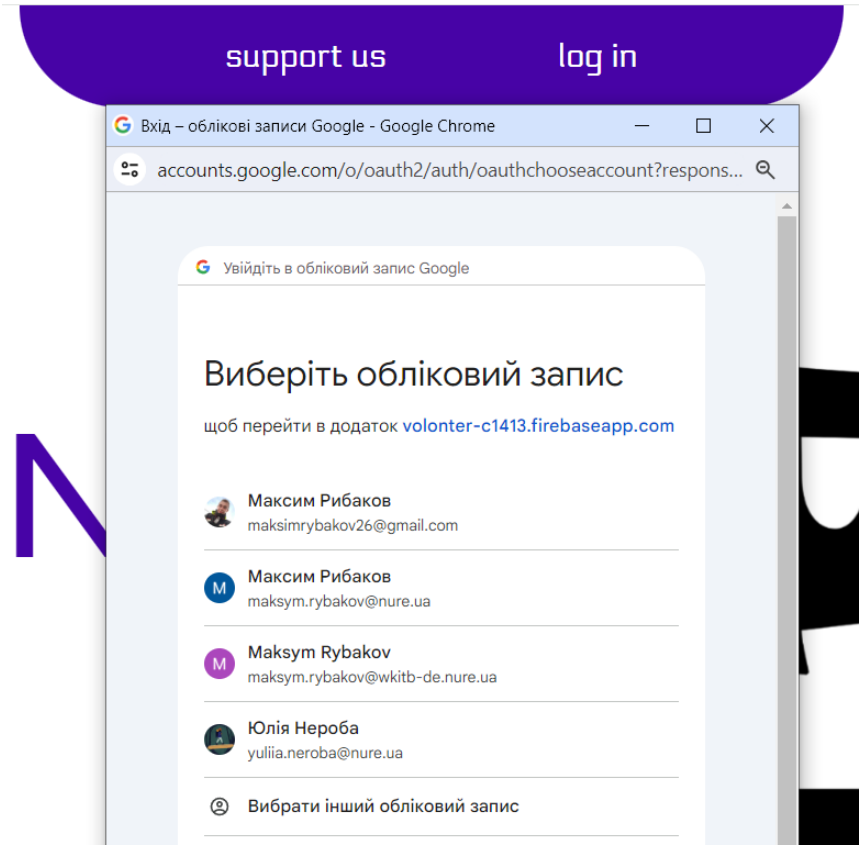

Рисунок 4.5 – Аутентифікаційний запит

Після вдалої аутентифікації, сервіс зберігає JWT токен.

JWT токен - це стандарт для безпечної передачі інформації між сторонами у вигляді JSON об'єкта. Його часто використовують для аутентифікації та авторизації в веб-додатках. Токен складається з трьох частин, розділених крапками: заголовка, корисного навантаження та підпису.

Заголовок містить тип токена (JWT) і алгоритм підпису, наприклад, HMAC SHA256 або RSA. Корисне навантаження включає інформацію про користувача і додаткові метадані, такі як ім'я користувача, термін дії токена і ролі користувача. Підпис забезпечує цілісність і автентичність токена. Він створюється шляхом хешування заголовка і корисного навантаження з використанням секретного ключа або пари ключів, якщо використовується асиметричне шифрування.

JWT токен підписується секретним ключем для симетричного шифрування або приватним ключем для асиметричного шифрування. Це гарантує, що токен не може бути змінений без відповідного ключа. Однак, дані в токені не шифруються, тому їх можна прочитати без спеціальних ключів.

JWT токени мають термін дії, який вказується у полі exp(expiration). Після закінчення цього терміну токен стає недійсним, і користувачеві потрібно отримати новий токен для продовження роботи. Усі необхідні для аутентифікації та авторизації дані містяться в самому токені, що робить його самодостатнім і зручним для використання. Для захисту токенів під час передачі важливо використовувати HTTPS, щоб запобігти їх перехопленню.

На сервер токен надсилається через заголовок Authorization для перевірки його автентичності і визначення, чи слід обробляти запит на дані. Приклад JWT токену зображено на рисунку 4.6.

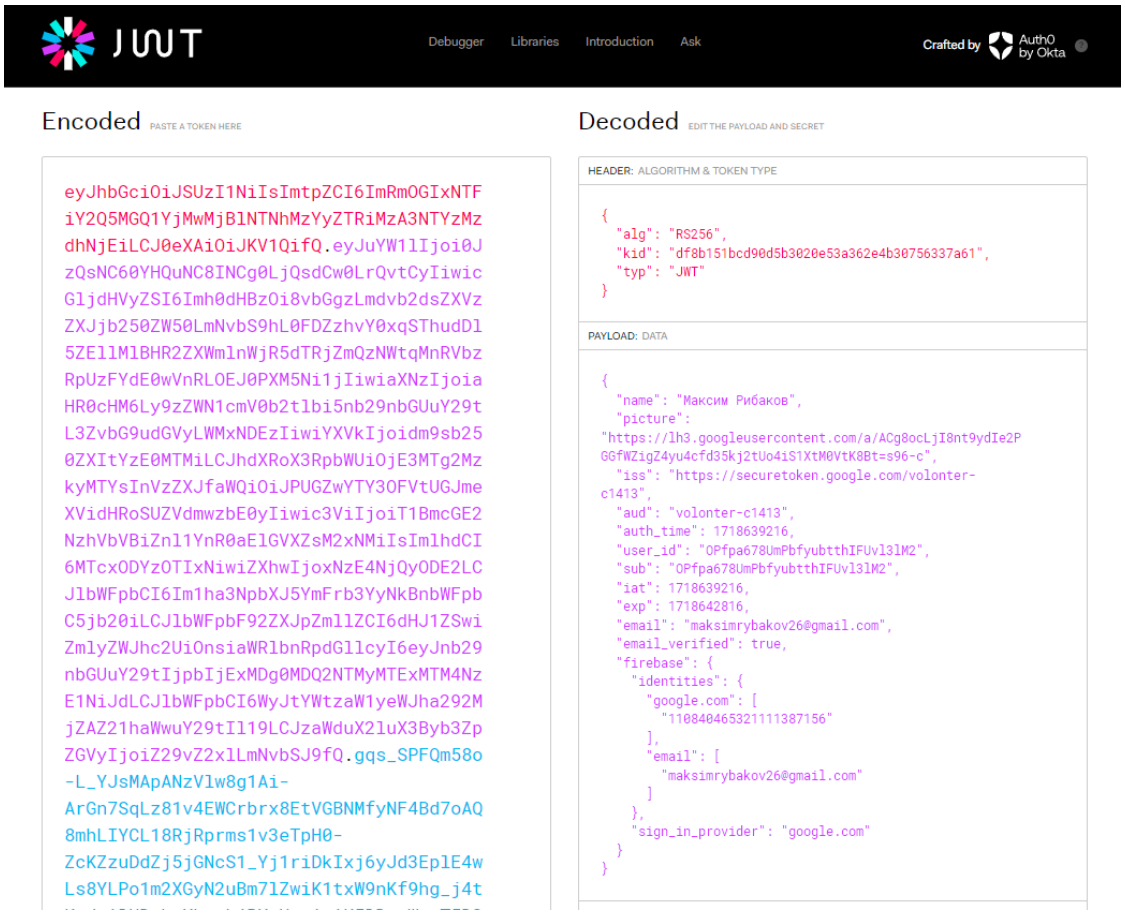

Рисунок 4.6. – JWS токен

Для взаємодії севрвера з Firebase Auth використовуєть бібліотека firebase_admin.

Firebase Admin SDK для Python - це інструментарій, що дозволяє розробникам з легкістю інтегрувати функціональність Firebase у свої Python додатки. Цей SDK забезпечує доступ до широкого спектру Firebase сервісів, включаючи базу даних Firestore, аутентифікацію користувачів, зберігання файлів у Cloud Storage, відправлення повідомлень через Cloud Messaging та інші сервіси Firebase.

Основні функції Firebase Admin SDK для Python включають створення, оновлення та видалення даних у реальному часі в базі даних Firestore, що забезпечує потужний інтерфейс для зберігання структурованої інформації. Це особливо корисно для додатків, які потребують зберігання та синхронізації даних між різними клієнтськими пристроями.

Firebase Admin SDK також дозволяє керувати ідентифікацією користувачів Firebase, включаючи створення нових облікових записів користувачів, оновлення їхніх атрибутів, перевірку підтвердження електронної пошти та інші операції пов'язані з управлінням аутентифікацією.

Додатково, Firebase Admin SDK для Python забезпечує гнучкість вибору технологій та методів розробки, що дозволяє розробникам інтегрувати Firebase функціонал у будь-які типи додатків і сервісів. Цей підхід сприяє швидкому впровадженню і розширенню функціональності Firebase у реальних проектах, що спрощує розробку та підтримку додатків.

Фрагмент ініціалізації Firebase Admin SDK зображено на рисунку 4.7.

```
cred = credentials.Certificate("C:\Projects\Firebase-adminsdk.json")
firebase_admin.initialize_app(cred)
```

Рисунок 4.7 – Ініціалізація Firebase Admin SDK

Фрагмент сервісу для отримання токену зображений на рисунку 4.8, а обробка токену на бекенд сервісі на рисунку 4.9.

```
import { initializeApp } from 'firebase/app';
import { getAuth, signInWithPopup,  GoogleAuthProvider, signOut } from 'firebase/a

const firebaseConfig = {
  apiKey: "AIzaSyAsNPa_O6N91FziuNDTwc-wLrvalfEeEC0",
  authDomain: "volonter-c1413.firebaseapp.com",
  projectId: "volonter-c1413",
  storageBucket: "volonter-c1413.appspot.com",
  messagingSenderId: "716278151432",
  appId: "1:716278151432:web:40dc8ffaf4a77718d894bd",
  measurementId: "G-ZZBCQ6RR8E"
};

const app = initializeApp(firebaseConfig);
const auth = getAuth(app);

const UserService = {
  // Метод для входу через Google акаунт
  login() {
    const provider = new GoogleAuthProvider();
    return signInWithPopup(auth, provider);
  },

  // Метод для виходу з системи
  logout() {
    return signOut(auth);
  },

  // Метод для отримання поточного токена
  async getToken() {
    return auth.currentUser ? await auth.currentUser.getIdToken(true) : null;
  }
};

export default UserService;
```

Рисунок 4.8 - Запит на отримання токену

```
async def verify_firebase_token(token: str):
    try:
        decoded_token = auth.verify_id_token(token)
        return decoded_token
    except auth.InvalidIdTokenError as e:
        raise HTTPException(status_code=401, detail="Invalid authorization token")
    except auth.ExpiredIdTokenError as e:
        raise HTTPException(status_code=401, detail="Expired authorization token")

@app.get("/is_registered")
async def protected_route(request: Request):
    auth_header = request.headers.get("Authorization")
    if auth_header is None:
        raise HTTPException(status_code=401, detail="Missing authorization header")
    bearer_token = auth_header.split(" ")[1]
    decoded_token = await verify_firebase_token(bearer_token)
    print(decoded_token)
    email = decoded_token.get('email')
    check_email = check_email_exists(server, database,email=email)
    return check_email
```

Рисунок 4.9 – Верифікація та обробка токену на бекенд сервісі

4.4 Взаємодія з БД

В якості СУБД було обрано реляційну базу даних Microsoft SQL Server.

Реляційні бази даних є основоположним типом баз даних, який використовується для зберігання та управління даними за допомогою реляційної моделі. У цій моделі дані представляються у вигляді таблиць, де кожен рядок відповідає окремому запису, а кожен стовпчик - атрибуту цього запису. Це дозволяє ефективно організовувати та структурувати інформацію, спрощуючи її обробку та доступ до неї.

Однією з ключових переваг реляційних баз даних є їх стандартизований підхід до управління даними за допомогою мови запитів SQL. SQL є ефективним інструментом для маніпуляції даними, що дозволяє виконувати різноманітні операції, такі як вибірка, вставка, оновлення та видалення даних. Це робить РБД особливо корисними для розробників і адміністраторів баз даних, оскільки спрощує розробку програмного забезпечення і підтримку систем.

Ще однією важливою характеристикою реляційних баз даних є їх здатність до нормалізації даних. Нормалізація дозволяє уникати дублювання даних і забезпечує стійкість інформації в базі. Це важливо для забезпечення цілісності даних і підтримки ефективної роботи системи в умовах змінюваної архітектури додатку і вимог бізнесу.

Окрім того, реляційні бази даних підтримують транзакційні операції, що є ще однією ключовою перевагою. Транзакції дозволяють гарантувати атомарність, надійність, ізольованість і стійкість (ACID), що є критичними вимогами для багатьох бізнес-процесів і додатків. Це робить реляційні бази даних надійним інструментом для забезпечення доступності і цілісності даних в умовах високих вимог до надійності і безпеки.

Microsoft SQL Server - це реляційна система управління базами даних, розроблена компанією Microsoft. Вона використовується для зберігання та

управління даними, забезпечуючи високу продуктивність, надійність і безпеку. MsSQL Server підтримує SQL як основну мову запитів, що дозволяє користувачам виконувати складні операції з даними. Завдяки своїй масштабованості, він може ефективно працювати як на невеликих серверах, так і на великих кластерних системах.

Однією з ключових особливостей MsSQL Server є його інтеграція з іншими продуктами Microsoft, такими як Azure, Visual Studio та Power BI. Це забезпечує безшовну роботу в екосистемі Microsoft і дозволяє користувачам легко розгортати, управляти і аналізувати дані. Крім того, MsSQL Server пропонує розширені можливості для аналітики та бізнес-інтелекту, включаючи інтеграцію з інструментами для аналізу великих даних і побудови звітів.

MsSQL Server підтримує високий рівень безпеки даних завдяки вбудованим засобам шифрування, аутентифікації та авторизації. Він також має функції для забезпечення відповідності нормативним вимогам, таким як аудит і моніторинг доступу до даних. Це робить його надійним вибором для компаній, що працюють з конфіденційною інформацією.

Функція Always On Availability Groups забезпечує високу доступність і аварійне відновлення, дозволяючи створювати репліки баз даних на різних серверах. Це значно підвищує надійність системи і забезпечує безперервність бізнес-процесів. Крім того, MsSQL Server підтримує транзакції, які забезпечують цілісність даних навіть у випадку збоїв системи.

MsSQL Server пропонує широкий спектр інструментів для адміністрування та управління базами даних, таких як SQL Server Management Studio (SSMS) та SQL Server Data Tools (SSDT). Ці інструменти забезпечують зручне середовище для розробки, тестування і налагодження баз даних, що полегшує роботу адміністраторів і розробників.

MsSQL Server підтримує різні типи даних, включаючи структуровані, неструктуровані і просторові дані. Це дозволяє зберігати і обробляти різноманітні типи інформації, від текстових і числових даних до географічних

координат і мультимедійних файлів. Завдяки цьому, система підходить для широкого спектра застосувань, від корпоративних додатків до великих інформаційних систем.

MsSQL Server також забезпечує високу продуктивність завдяки використанню передових технологій оптимізації запитів, індексації та кешування. Це дозволяє ефективно обробляти великі обсяги даних і забезпечувати швидкий доступ до інформації. Крім того, система підтримує паралельну обробку запитів і багатопотоковість, що підвищує її продуктивність і масштабованість.

Однією з важливих функцій MsSQL Server є підтримка інтеграції з хмарними сервісами, такими як Microsoft Azure. Це дозволяє користувачам розгортати бази даних у хмарі, використовуючи переваги хмарної інфраструктури, такі як автоматичне масштабування, висока доступність і зниження витрат на підтримку апаратного забезпечення. Крім того, MsSQL Server підтримує гібридні сценарії, що дозволяє поєднувати локальні і хмарні ресурси для оптимального управління даними.

MsSQL Server включає потужні засоби для автоматизації задач, такі як SQL Server Agent, який дозволяє планувати і виконувати регулярні завдання, такі як резервне копіювання, оновлення статистики та обслуговування індексів. Це допомагає знизити операційні витрати і забезпечує безперебійну роботу системи.

Підключення до MsSQL Server здійснюється завдяки бібліотеці PyODBC.

PyODBC - це бібліотека для Python, яка надає доступ до баз даних через інтерфейс Open Database Connectivity. ODBC є стандартом для доступу до різних типів баз даних, таких як Microsoft SQL Server, PostgreSQL, MySQL, Oracle та інші, що дозволяє розробникам працювати з різними системами за допомогою єдиної API.

PyODBC забезпечує Python інтерфейс для взаємодії з базами даних через ODBC, що робить її універсальним інструментом для розробки додатків, які потребують роботи зі структурованою інформацією у базах даних. Вона підтримує як Python 2.x, так і Python 3.x, що робить її доступною для широкого кола проектів та систем.

Основні можливості PyODBC включають підтримку розширених опцій конфігурації з'єднань, таких як налаштування кодування, обмеження часу очікування і буферизації запитів, що дає розробникам гнучкість у керуванні взаємодією з базою даних. PyODBC також підтримує використання параметризованих запитів, що сприяє зменшенню ризику SQL-ін'єкцій та покращенню загальної безпеки додатків.

Бібліотека PyODBC дозволяє виконувати різні типи SQL-запитів до баз даних, такі як SELECT, INSERT, UPDATE і DELETE, а також виконувати збережені процедури та функції, що робить її потужним інструментом для роботи з даними. Вона підтримує обробку винятків і відлагодження, що дозволяє ефективно вирішувати проблеми з взаємодією з базою даних та управлінням помилками.

Фізична модель бази даних у SQL Server Management Studio зображена на рисунку 4.10.

Фізична модель є результатом процесу, що описує та визначає деяку предметну область. Дані в моделі представлені у вигляді сутностей, які пов’язані між собою певними зв’язками, що виражають залежності і вимоги між ними.

Опис основних сутностей наведений в таблиці 4.1

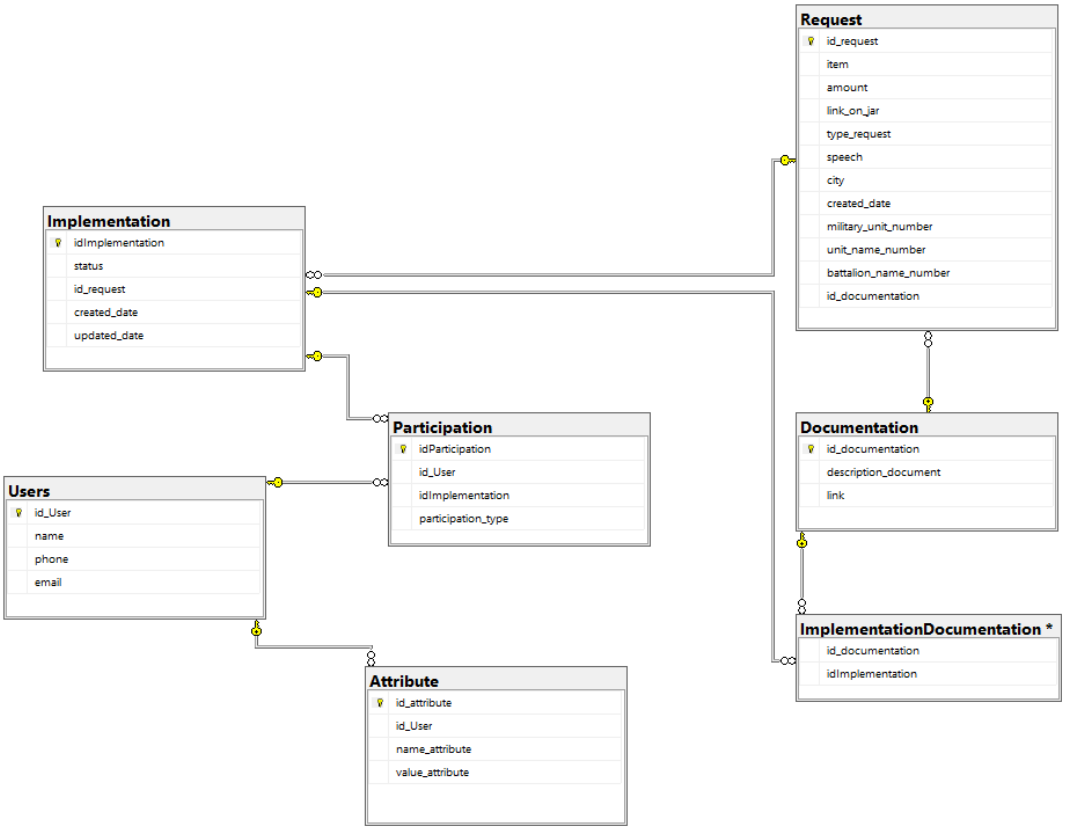

Рисунок 4.10 – Фізична модель даних

Таблиця 4.1 – Сутності фізичної моделі даних

1	2	3	4	5
№	Назва сутності	Назва атрибуту	Тип даних	Призначення
1	Users	Id_User	INT	первинний ключ
		name	NVARCHAR	ім’я користувача
		phone	NVARCHAR	телефон
		email	NVARCHAR	електронна пошта
2	Attribute	id_attribute	INT	первинний ключ

		id_user	INT	зовнішній ключ
		name_attribute	NVARCHAR	назва атрибуту
		value_attribute	NVARCHAR	значення атрибуту
3	Documentation	id_documentation	INT	первинний ключ
		description_document	NVARCHAR	опис документу
		link	NVARCHAR	посилання на документ
4	Request	id_request	INT	первинний ключ
		item	NVARCHAR	назва предмету
		amount	FLOAT	кількість
		link_on_jar	NVARCHAR	посилання на банку
		type_request	NVARCHAR	тип запиту
		speech	NVARCHAR	звернення в запиті
		city	NVARCHAR	місто отримання
		created_date	DATE	дата створення

		military_unit_number	INT	номер військової частини
		unit_name_number	NVARCHAR	номер/назва підрозділу
		battalion_name_number	NVARCHAR	номер/назва батальйону
		id_documentation	INT	зовнішній ключ
5	Implementation	idImplementation	INT	первинний ключ
		status	NVARCHAR	статус запиту
		id_request	INT	зовнішній ключ
		created_date	DATE	дата прийняття запиту
		updated_date	DATE	дата останнього оновлення
6	Participation	idParticipation	INT	первинний ключ
		id_user	INT	зовнішній ключ
		idImplementation	INT	зовнішній ключ
		participation_type	NVARCHAR	тип учасника

7	Implementation Documentation	id_documentation	INT	зовнішній ключ
		id_implementation	INT	зовнішній ключ

4.5 Експериментальні дослідження

Суть дослідження полягає у вибірці з бази даних інформації про користувача за полем email при великій кількості записів. Для генерування SQL скрипту на вставку даних було використано сервіс SQL Data Generator.

SQL Data Generator - це інструмент від компанії Redgate, призначений для створення великих обсягів тестових даних у базах даних Microsoft SQL Server. Він допомагає розробникам, тестувальникам і адміністраторам баз даних швидко і ефективно заповнювати бази даних реалістичними даними для проведення тестування, відладки і демонстрацій.

SQL Data Generator дозволяє користувачам налаштовувати типи даних для кожного стовпця в таблицях, забезпечуючи відповідність створюваних даних реальним умовам. Ви можете визначати шаблони для імен, адрес, дат, чисел та інших типів даних, що дозволяє отримати різноманітні і реалістичні набори даних. Крім того, інструмент підтримує взаємозв'язки між таблицями, що дозволяє зберігати цілісність даних при генерації.

Однією з важливих функцій SQL Data Generator є можливість інтеграції з існуючими базами даних, що дозволяє автоматично використовувати схему бази даних для генерації відповідних даних. Це спрощує процес налаштування і забезпечує точне відображення структури бази даних. Інструмент також підтримує створення спеціальних сценаріїв і умов для генерації даних, що дозволяє точно налаштувати процес відповідно до вимог конкретного проекту.

SQL Data Generator забезпечує високу продуктивність і може обробляти великі обсяги даних, що робить його незамінним для тестування продуктивності і навантаження на систему. Інструмент також має зручний інтерфейс користувача, що полегшує процес налаштування і управління генерацією даних, дозволяючи користувачам швидко освоїти його функціонал і почати використовувати для своїх проектів.

За допомогою SQL Data Generator було згенерована скрипт на вставку близько 10 мільйонів записів до таблиці Users. Таблиця має 3 поля та ідентифікатор, що створюється автоматично. Вставка відбувалася через Python та її виконання зайняло 2206 секунд. Фрагмент функції для вставки зображено на рисунку 4.11.

```
def insert_data (server: str, database: str) -> bool:
    connection_string = ("Driver={ODBC Driver 17 for SQL Server};"
                         f"Server={server};"
                         f"Database={database};"
                         "Trusted_Connection=yes;")

    conn = pyodbc.connect(connection_string)
    cursor = conn.cursor()
    with open('C:/Projects/data.sql', 'r') as file:
        sql_script = file.read()
    sql_commands = sql_script.split(';')

    try:
        for command in sql_commands:
            if command.strip():  # Перевіряємо, чи команда не порожня
                cursor.execute(command)
                conn.commit()

    except pyodbc.Error as ex:
        print(f'Помилка при виконанні SQL-запиту: {ex}')

    # Закриття з'єднання
    cursor.close()
    conn.close()
    print("Вставка даних завершена.")
```

Рисунок 4.11 – Вставка даних до таблиці Users

Для виконання запиту на вибір користувача за його електронною поштою був написаний скрипт на мові SQL. Фрагмент скрипту наведений на рисунку 4.12.

```
DECLARE @StartTime DATETIME;
DECLARE @EndTime DATETIME;
DECLARE @DurationMilliseconds INT;
SET @StartTime = GETDATE();
SELECT *
FROM Users
WHERE email LIKE 'Mireya_Rau@gmail.com';
SET @EndTime = GETDATE();
SET @DurationMilliseconds = DATEDIFF(MILLISECOND, @StartTime, @EndTime);
SELECT 'Час виконання запиту: ' + CAST(@DurationMilliseconds AS VARCHAR) + ' мс';
```

Рисунок 4.12 – Пошук користувача за email

Час виконання даного запиту складає від 330 до 1700 мс. Наступним кроком є індексація поля email. Після індексування поля, час вибору користувача за email становить від 10 до 30 мс, що значно перевершує минулий результат.

4.6 Розробка дизайну

Розробка прототипів дизайну відбувалася за допомогою сервісу Figma.

Figma - це онлайн інструмент для дизайну і прототипування інтерфейсів, який здобув велику популярність серед дизайнерів та розробників і зарекомендував себе як потужний інструмент для спільної роботи над проектами. Основна перевага Figma полягає в тому, що він працює в браузері, що дозволяє користувачам працювати над проектами без необхідності установки спеціального програмного забезпечення.

Одним з ключових функціональних можливостей Figma є можливість створювати векторні графічні елементи, такі як іконки, кнопки, картинки, а також цілі макети веб-сторінок або мобільних додатків. Користувачі можуть

легко маніпулювати формами, кольорами і стилями безпосередньо в інтерфейсі Figma, що спрощує процес створення і редагування дизайну.

Figma підтримує роботу в реальному часі для командної роботи, що робить його ідеальним інструментом для спільної розробки ітеративних проектів. Користувачі можуть взаємодіяти одночасно з одним проектом, спостерігаючи за змінами і вносячи свої коментарі без затримок і необхідності встановлення спеціалізованих програм.

Крім того, Figma підтримує можливість створення прототипів, що дозволяє дизайнерам і розробникам створювати і відтворювати інтерактивні прототипи веб-сторінок або додатків. Це дозволяє команді легко тестувати ідеї і зміни в інтерфейсі до фінальної імплементації, що сприяє покращенню користувацького досвіду і відповідності функціональних вимог.

Figma також підтримує інтеграцію з різними інструментами для розробки, такими як Zeplin, Avocode, Jira, що дозволяє спрощувати робочий процес і підвищувати продуктивність команди. Інтеграція з цими інструментами дозволяє автоматизувати процес передачі інформації і спільної роботи між дизайнерами і розробниками.

Загальний огляд прототипів наведено на рисунку 4.13

В інформаційній системі присутні декілька сторінок, а саме:

- Головна сторінка;
- Сторінка вибору типу користувача;
- Сторінка реєстрації для фізичної особи;
- Сторінка реєстрації для організації;
- Сторінка профілю користувача;
- Сторінка вибору типу запиту для створення;
- Сторінка створення військового запиту;
- Сторінка створення цивільного запиту;
- Сторінка пошуку актуальних запитів;

На головній сторінці є можливість перейти на сторінки реєстрації чи авторизації, а також до створення або пошуку вже створених запитів. Головна сторінка зображена на рисунку 4.13.

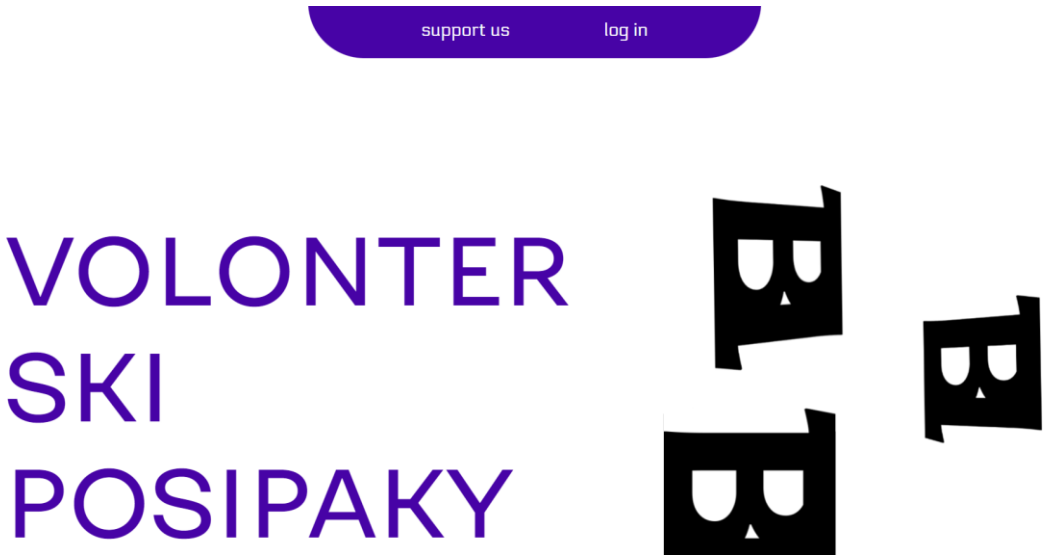

Рисунок 4.13 – Головна сторінка

Сторінку реєстрації для фізичної особи зображено на рисунку 4.14, а для юридичної на рисунку 4.15.

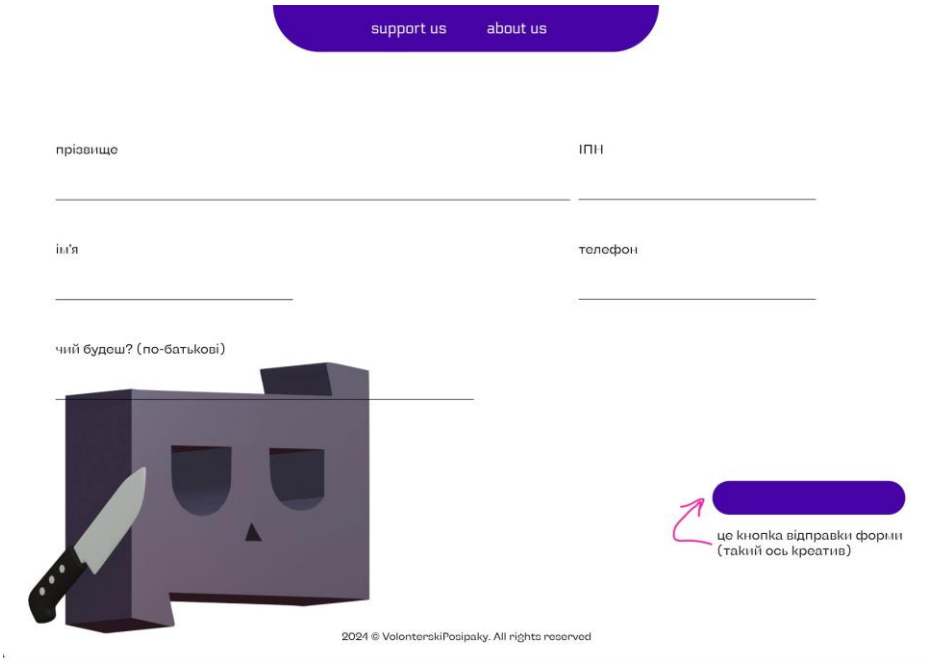

Рисунок 4.14 – Реєстрація фізичної особи

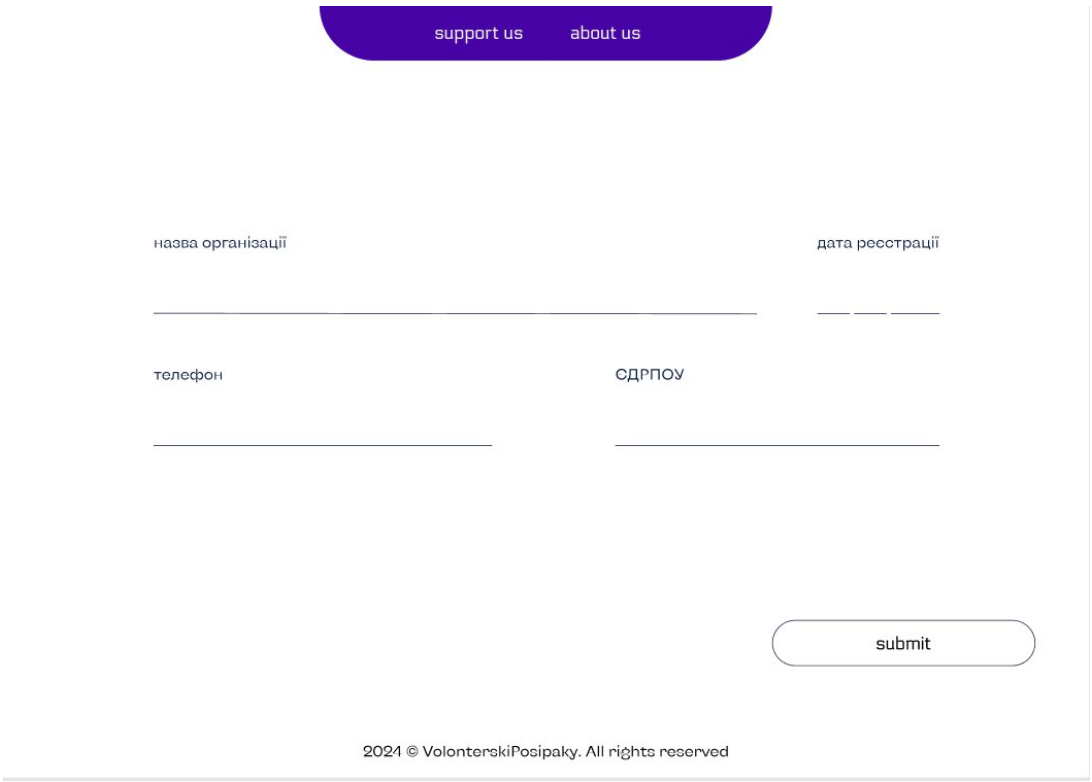

Рисунок 4.15 – Реєстрація організації

Сторінка вибору типу користувача зображена на рисунку 4.16.

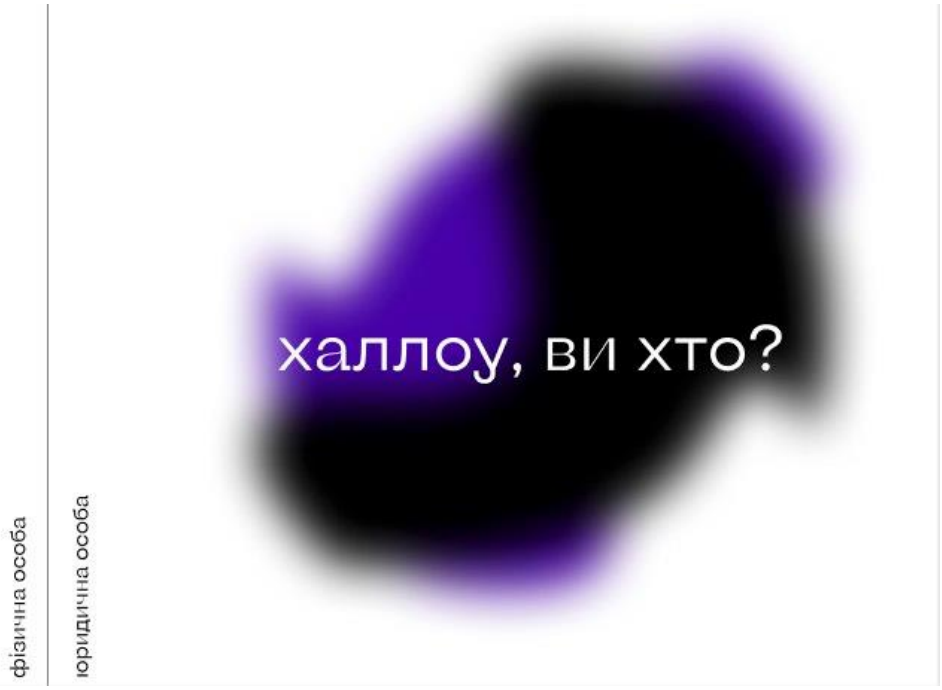

Рисунок 4.16 – Сторінка вибору типу користувача

На сторінці реєстрації фізичної особи присутня 3D анімація, що створена за допомогою сервісу Spline.

Spline є веб-платформою для створення 3D-анімацій та інтерактивних візуальних ефектів без програмування. Цей інструмент дозволяє користувачам легко створювати складні 3D-сцени за допомогою інтуїтивно зрозумілого інтерфейсу, не вимагаючи глибоких знань в графіці або програмуванні.

За допомогою Spline, 3D тепер можна створювати так само легко, як макети у Figma. Spline — це новий інструмент для 3D-моделювання та анімації, який орієнтований саме на веб-дизайнерів.

Spline надає широкий вибір інструментів і можливостей для розгортання та анімації об'єктів у тривимірному просторі. Він підтримує різноманітні ефекти та переходи, що дозволяють створювати живі та динамічні сцени для веб-сайтів, презентацій або інших цифрових продуктів.

Користувачі Spline можуть імпортувати власні моделі та ресурси, що спрощує інтеграцію з існуючими проектами та розширює можливості творчого процесу. Платформа підтримує експорт готових анімацій у форматах, які підходять для веб-розробки або інтеграції з іншими програмами для обробки відео та мультимедіа.

Сторінка профілю користувача зображена на рисунку 4.17.

Рисунок 4.17 – Сторінка профілю користувача

На сторінці вибору типу запиту для створення є два варіанти для вибору: цивільний запит та військовий.

Для військового запиту від користувача вимагається інформація про військовий підрозділ, бригаду та батальйон, а також документ з офіційним запитом від командира. Також потрібно вказати зручне місце для отримання допомоги, посилання на банку для збору коштів, якщо така існує та що саме потрібно та в якій кількості.

Для цивільних запитів поля дублюються з військовим запитом, окрім інформації про військову організацію. Також присутнє поле, в якому можна написати своє звернення до волонтерів, аргументуючи важливість речей, на які цей запит створений.

Задля безпеки, для створення будь-якого запиту обов’язкова авторизація в інформаційній системі, що здійснюється за допомогою Google сервісу. В майбутньому буде додана авторизація за телефоном та за допомогою сервісу Дія Підпис.

Всі документи, що пов’язані з запитами, в інформаційній системі зберігатися не будуть, а будуть автоматично відправляти на пошту особі чи організації, що прийняла запит. Сторінка з вибором типу запиту зображена на рисунку 4.18.

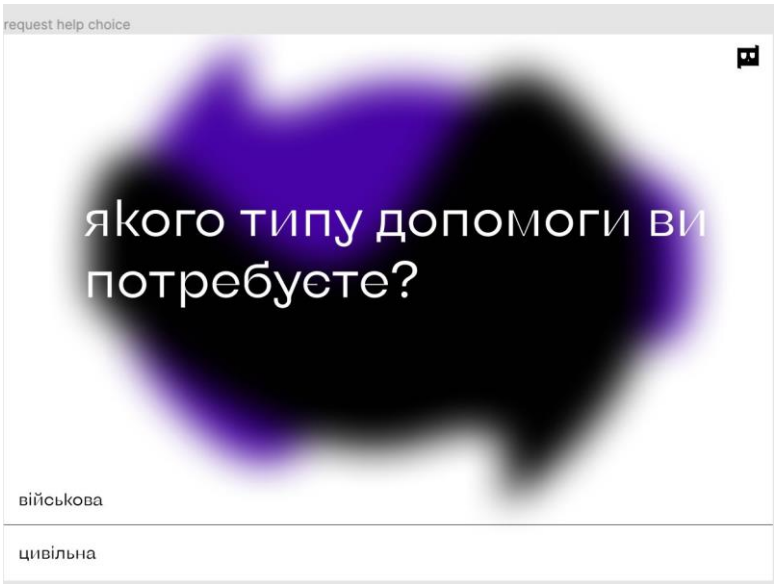

Рисунок 4.18 – Сторінка вибору типу запиту

Сторінка створення цивільного запиту зображена на рисунку 4.19.

Рисунок 4.19 – Створення цивільного запиту

Сторінка створення цивільного запиту зображена на рисунку 4.20.

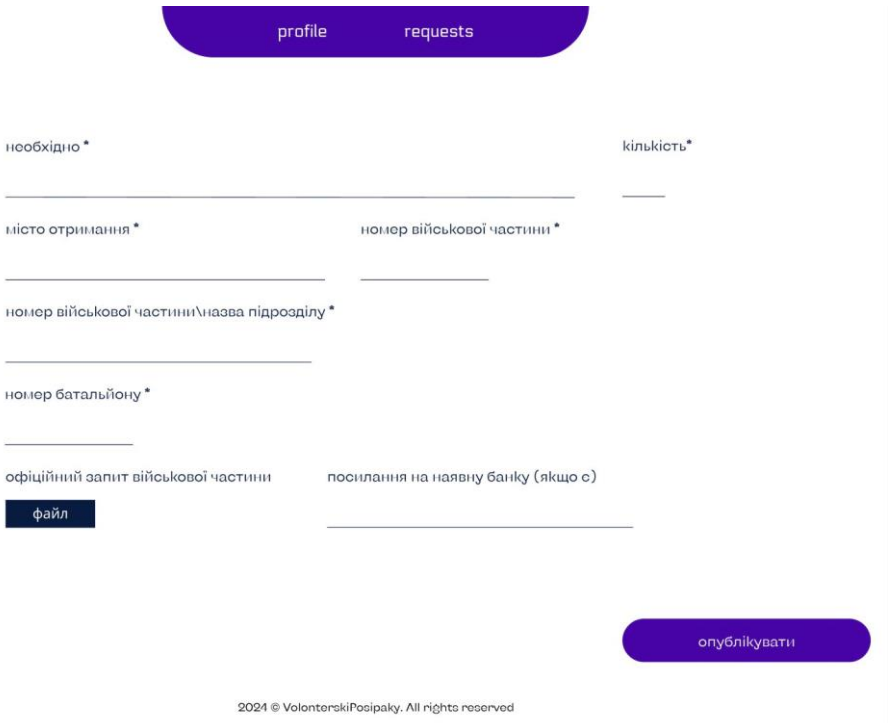

Рисунок 4.20 – Створення військового запиту

Сторінка для пошуку запитів теж з’являється тільки після авторизації. Дану сторінку зображено на рисунку 4.21.

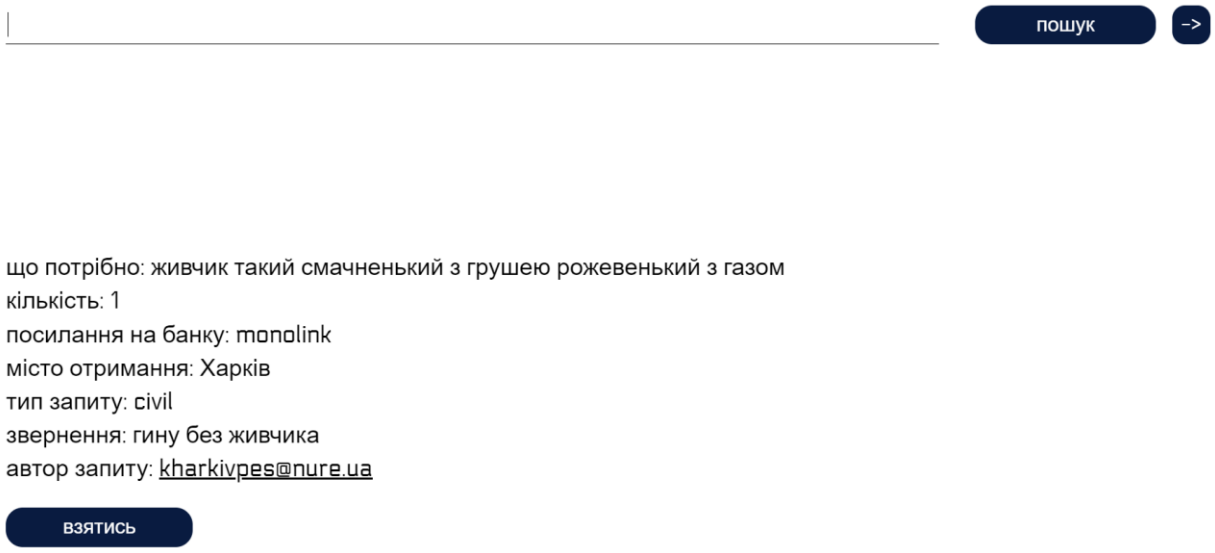

Рисунок 4.21 – Сторінка пошуку запитів

4.7 Інтеграція з хмарним сховищем

Для зберігання документів було вирішено використовувати хмарне сховище.

Хмарне сховище (cloud storage) є сучасним рішенням для зберігання та управління даними в інтернет-обчисленні. Воно надає користувачам можливість зберігати дані в децентралізованому, віддаленому від комп'ютера способі, що забезпечує доступ до даних з будь-якого місця з інтернет-з'єднанням. Основна ідея хмарних сховищ полягає в тому, щоб користувачі могли зберігати свої дані на серверах хмарних провайдерів, які забезпечують надійність, безпеку та доступність.

Однією з ключових переваг хмарного сховища є масштабованість: користувачі можуть збільшувати або зменшувати обсяги збережених даних в залежності від своїх потреб без необхідності інвестувати в власну інфраструктуру. Це робить хмарні сховища ефективними для як

індивідуальних, так і корпоративних користувачів, що потребують різних рівнів масштабування та доступності.

В якості хмарного сховища було обрано Google Cloud Storage.

Google Cloud Storage - це хмарний сервіс для зберігання і керування даними, який надається компанією Google. Основною одиницею організації даних в GCS є бакети (buckets). Бакети є контейнерами для зберігання об'єктів, таких як файли чи дані. Кожен бакет має унікальне глобально ідентифікуюче ім'я і може бути створений в конкретному регіоні або мультирегіонально, що визначає область фізичного розташування даних.

Об'єкти (Objects) є конкретними елементами даних, які зберігаються в бакетах. Кожен об'єкт може мати унікальне ім'я в межах свого бакета і включати в себе саме дані разом з метаданими. GCS підтримує широкий спектр можливостей для керування об'єктами, включаючи версіонування, життєвий цикл, доступ до даних через HTTP або HTTPS протоколи, а також механізми автентифікації і авторизації для забезпечення безпеки даних.

Крім того, GCS надає інтеграцію з іншими сервісами Google Cloud Platform, що дозволяє зручно використовувати збережені дані для обчислень, аналізу даних, розгортання додатків та інших бізнес-процесів. Цей сервіс є частиною широкої екосистеми інструментів Google Cloud, яка підтримує різні сценарії використання, від індивідуальних розробок до корпоративних вирішень великих масштабів.

Google Cloud Storage (GCS) також відомий своєю високою доступністю і надійністю. Сервіс автоматично реплікує дані через кілька фізичних місць для забезпечення надійності і стійкості до відмов. Це робить GCS відмінним вибором для зберігання критичних для бізнесу даних і додатків, які вимагають високого рівня доступності.

Керування доступом до об'єктів в GCS здійснюється через гнучкі механізми управління правами доступу. Ви можете налаштовувати доступ до бакетів і об'єктів за допомогою ролей і прав доступу, що забезпечує контроль

над тим, хто і як може отримувати доступ до даних. Це особливо важливо для забезпечення відповідності з внутрішніми і зовнішніми регуляторними вимогами.

Google Cloud Storage підтримує інтеграцію з іншими сервісами Google, такими як BigQuery для аналізу даних, Cloud Pub/Sub для потокової обробки даних, і Cloud AI для машинного навчання. Це дозволяє легко і ефективно використовувати дані, збережені в GCS, для різних цілей, від бізнес-аналітики до розробки штучного інтелекту.

Google надає офіційний SDK для Python, який спрощує взаємодію з GCS. Цей SDK дозволяє розробникам створювати, завантажувати, оновлювати та видаляти об'єкти у хмарному сховищі безпосередньо зі свого Python коду.

Для роботи з GCS з Python, спочатку необхідно ініціалізувати клієнта GCS. Це можна зробити за допомогою сервісних облікових записів Google Cloud і налаштування з'єднання через ключ або автоматично за допомогою вбудованих у Google Cloud інструментів автентифікації.

За допомогою GCS SDK для Python можна виконувати різні операції з об'єктами, такі як завантаження файлів у сховище, читання, запис та видалення файлів.

Фрагмент функції для завантаження документів в GCS зображено на рисунку 4.22.

```
def upload_file_to_gcs(bucket_name, local_file_path, destination_blob_name):
    client = storage.Client()
    bucket = client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)
    with open(local_file_path, "rb") as f:
        blob.upload_from_file(f)
    print(f"Файл {local_file_path} завантажено в {destination_blob_name}.")
```

Рисунок 4.22 – Завантаження файлу до GCS

Фрагмент функції для завантаження документів з GCS зображено на рисунку 4.23.

```
def download_file_from_gcs(bucket_name, source_blob_name, destination_file_name):
    client = storage.Client()
    bucket = client.bucket(bucket_name)
    blob = bucket.blob(source_blob_name)
    blob.download_to_filename(destination_file_name)
    print(f"Файл {source_blob_name} завантажено як {destination_file_name}.")
```

Рисунок 4.23 – Завантаження файлу з GCS

Фрагмент функції для видалення документів з GCS зображено на рисунку 4.24.

```
def delete_file_from_gcs(bucket_name, blob_name):
    client = storage.Client()
    bucket = client.bucket(bucket_name)
    blob = bucket.blob(blob_name)
    blob.delete()
    print(f"Файл {blob_name} видалено з {bucket_name}.")
```

Рисунок 4.24 – Видалення файлу з GCS

ВИСНОВКИ

В ході роботи був проведений аналіз предметної області. Під час аналізу предметної області були визначені існуючі системи, спрямовані на подібні цілі, та проведено порівняльний аналіз їх функціональності. На основі цього було сформульовано Vision and Scope для розроблюваної інформаційної системи, що описує її мету та обсяг функціоналу.

В результаті було розроблено інформаційну систему, що відповідає всім заданим вимогам. Інформаційна система має мікросервісну архітектуру, що складається з основних сервісів: фронтенд, бекенд та сервіс аутинтефікації. Користувацький інтерфейс зручний у використанні, а дизайн інтуїтивно зрозумілий.

Інформаційна система впроваджує прозорий механізм звітності, що дозволяє збирати та аналізувати дані про ефективність та вплив волонтерських ініціатив. Ця система забезпечує ефективне управління ресурсами та задачами, що дозволяє максимально використовувати потенціал учасників. Вона включає інструменти для планування та виконання проектів, моніторингу їх прогресу і результативності, а також для спілкування та співпраці між учасниками. Гнучка архітектура системи дозволяє швидко адаптуватися до змінних потреб та вимог спільноти.

Експериментальною частиною дослідження стала перевірка швидкості виконання запиту до бази даних для індексованого та неіндексованого поля. В результаті визначено, що для індексованого поля швидкість відбору в рази швидша, ніж для неіндексованого.

Розроблена система пройшла апробацію, взявши участь у 10-й міжнародній студентській науковій конференції «Сучасні аспекти та перспективні напрямки розвитку науки». В рамках цієї участі було опубліковано тези доповіді на тему «Інформаційна система організації та координації волонтерської діяльності».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання / Нац. стандарт України. – Вид. офіц. – [Чинний від 2017-07-01]. – Київ: ДП «УкрНДНЦ», 2016. – 26 с.

2. ДСТУ 7.1:2006. Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання / Нац. стандарт України. – Вид. офіц. – [Чинний від 2007-07-01]. – Київ : Держспоживстандарт України, 2007. – 47 с.

3. Методичні вказівки до організації виконання та захисту кваліфікаційної роботи на здобуття другого (магістерського) рівня вищої освіти спеціальності 122 Комп'ютерні науки, освітньо-професійна програма «Інформаційні технології проєктування» / Упорядники: І.В. Гребеннік, — В.Г. Іванов, А.І. Коваленко, О.Б. Колесник, Ю.В. Міщеряков, І.А. Урняєва, С.І. Чайніков. Харків: ХНУРЕ, 2021. 54 с.

4. «Microservices Patterns: With examples in Java 1st Edition» [Електронне видання] / Автор: Chris Richardson – 2018. – 520с.

5. Microservice architecture patterns [Електронний ресурс] URL: https://microservices.io/patterns/microservices.html Дата звернення (15.05.2024)

6. «Екстремальне програмування: розробка через тестування» [Електронне видання] / Автор: Кент Бек – 2019. – 224с.

7. Python documentation [Електронний ресурс] URL: https://docs.python.org/3/ Дата звернення (16.05.2024)

8. Firebase Authentication [Електронний ресурс] URL: https://firebase.google.com/docs/auth Дата звернення (16.03.2024)

9. Python FastAPI documentation [Електронний ресурс] URL: https://fastapi.tiangolo.com/ Дата звернення (16.05.2024)

10. What is Sequence Diagram? [Електронний ресурс] URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/ Дата звернення (17.05.2024)

11. React docs [Електронний ресурс] URL: https://uk.reactjs.org/docs/getting-started.html (Дата звернення: 16.03.2024)

12. UI/UX Starter Guide [Електронний ресурс] URL: https://www.figma.com/community/file/1019904134655257869 (Дата звернення: 18.05.2024)

13. «Чиста архітектура. Мистецтво розробки програмного забезпечення» [Електронне видання] / Автор: Роберт Мартін – 2018. – 352 с.

14. Automate infrastructure [Електронний ресурс] URL: https://www.terraform.io/ (Дата звернення: 21.05.2024)

15. Google Cloud Platform [Електронний ресурс] URL: https://cloud.google.com/docs/storage (Дата звернення: 15.05.2024)

16. Петрова Р.В., Морозова А.І. Іноваційний реінжинірінг бізнес-процесів в інформаційному середовищі // Вісник Хмельницького національного університету. Серія: Технічні науки. –2019, № 1 (269). – С. 211−215.

17. Томка Ю.Я., Ушенко Ю.О. Основи ASP.NET MVC. Навчальний посібник.- К.: / Томка Ю.Я., Ушенко Ю.О. – Чернівці, Чернівецький нац. ун-т, 2018. – 730 с.

18. Gerardus Blokdyk. Microsoft Sql Server Management Studio a clear and concise reference, 2021. – 312 с.

19. Alan Beaulieu. Learning SQL: Generate, Manipulate, and Retrieve Data, 2020. – 342 с.

20. Mathias Wesk. Business Process Management, 2007. – 426 с.

21. РД 50-680-88 - Керівний документ по стандартизації. Методичні вказівки. Автоматизовані системи. Основні положення.

22. ДСТ 34.003-90 - Автоматизовані системи. Терміни й відділення.

23. Figma [Електронний ресурс] URL: https://www.figma.com/ (Дата звернення: 21.05.2024)

24. Reactjs – popup [Електронний ресурс] URL: https://www.npmjs.com/package/reactjs-popup (Дата звернення: 21.05.2024)

25. Google Cloud Storage [Електронний ресурс] URl: https://cloud.google.com/storage?hl=uk#object-storage-for-companies-of-all-sizes (Дата звернення: 21.05.2024)