Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента *Свириденка Ігоря Євгеновича*
(ПІБ)

академічної групи *121М-21-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка та дослідження ефективності впровадження серверної частини волонтерської платформи засобами Java, SpringBoot, Postgresql, OpenAPI 3.0.*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Бердник М. Г.*			
економічний	*Проф. Вагонова О.Г*			
Рецензент	*Доц. Шедловський І.А.*			
Нормоконтролер	*Проф. Лактіонов І.С.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2022 року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *121 Інженерія програмного забезпечення*
студента *121м-21-1* (група) *Свириденка Ігоря Євгеновича* (прізвище та ініціали)
тема дипломного проекту *Розробка та дослідження ефективності впровадження серверної частини волонтерської платформи засобами Java, SpringBoot, Postgresql, OpenAPI 3.0*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 31.10.2022 р. № 1200 –с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес розробки серверної частини волонтерської платформи.

Предмет досліджень – моделі та методи створення та вибору архітектур серверної частини волонтерської платформи.

Мета роботи – розробка та оптимізація універсальної серверної частини волонтерської платформи, яка надаватиме їм можливість встановлювати зв'язок між учасниками, підписуватись один на одного, ділитись повідомленнями, створювати оголошення, дивитися оголошення поблизу та комунікувати за допомогою коментарів.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна отриманих результатів полягає в тому, що вперше було розроблено прогграмне забезпечення, яка не тільки об'єднує в собі фунціонал розглянутих систем, а й, привносить нові можливості.

Практична цінність полягає в розробці та оптимізації серверної частини волонтерської платформи, що полегшить взаємодію між волонтерами та допоможе їм будувати ефективну взаємодію.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання серверної частини волонтерської платформи.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2022-30.10.2022
Дослідження існуючих волонтерських застосунків та бібліотек програмування подібних систем	01.11.2022-24.11.2022
Створення бекенд застосунку для автоматизації та полегшення взаємодії у сфері волонтерства	25.11.2022-14.12.2022

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки розробці програмного забезпечення для волонтерської платформи, яка надаватиме можливість встановлювати зв'язок між учасниками, підписуватись один на одного, ділитись повідомленнями, створювати оголошення, дивитися оголошення поблизу та комунікувати за допомогою коментарів.

Соціальний ефект від реалізації результатів роботи очікується позитивним, завдяки полегшенню роботи волонтерів.

Завдання видав _____ *проф. Бердник М.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Свириденко І. Є.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: 12.09.2022р.

Термін подання кваліфікаційної роботи до ЕК: 16.12.2022р.

РЕФЕРАТ

Пояснювальна записка: 103 с., 33 рис., 15 табл., 4 дод., 25 джерел.

Об'єкт дослідження: процес розробки серверної частини волонтерської платформи.

Предмет дослідження: моделі та методи створення та вибору архітектур серверної частини волонтерської платформи.

Мета магістерської роботи: розробка та оптимізація універсальної серверної частини волонтерської платформи, яка надаватиме їм можливість встановлювати зв'язок між учасниками, підписуватись один на одного, ділитись повідомленнями, створювати оголошення, дивитися оголошення поблизу та комунікувати за допомогою коментарів.

Методи дослідження: для вирішення поставлених задач були використані наступні методи: аналіз даних, проектування реляційних баз даних, об'єктно-орієнтоване програмування.

Наукова новизна отриманих результатів полягає в тому, що вперше було розроблено програмне забезпечення, яке не тільки об’єднує в собі фунціонал розглянутих систем, а й, привносить нові можливості.

Практична цінність полягає в розробці та оптимізації серверної частини волонтерської платформи, що полегшить взаємодію між волонтерами та допоможе їм будувати ефективну взаємодію.

Список ключових слів: волонтер, мережа, сайт, браузер, сервер, інформаційна система, пристрій.

Explanatory note: 103 p., 33 figures, 15 tables, 4 appendices, 25 sources.

Object of research: the process of developing server part of the volunteer platform.

Subject of research: models and methods of creation and choosing appropriate architechture of server part of the volunteer platform.

Purpose of Master's thesis: Designing and optimizing a universal backend of a volunteer platform that enables them to connect with members, subscribe one-to-one, share messages, create announcements, place announcements and communicate through comments.

Research methods: the following methods were used to solve the problems: data analysis, database architechturing, object-oriented programming, trigonometric functions, pattern programming.

Originality of research is that for the first time the created program not only combines functionally that presented systems have, but also provides new capabilities.

The practical value of the results consist of developing and optimization of the server part of the volunteer platform, which facilitates the interaction between volunteers and helps them to build effective interaction.

List of keywords: program, volunteer, network, site, browser, server, information system, device.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ 10
1.1. Загальні відомості з предметної галузі 10
1.1.1. Волонтерство в світі 10
1.1.2. Волонтерство в Україні 11
1.2. Призначення розробки та галузь її застосування 17
1.3. Постановка завдання 20
1.4. Вимоги до програми або програмного виробу 21
1.4.1. Вимоги до функціональних характеристик 21
1.4.2. Вимоги до інформаційної безпеки 22
1.4.3. Вимоги до складу та параметрів технічних засобів 22
РОЗДІЛ 2. ОБЧИСЛЕННЯ ВІДСТАНІ ВІД МЖ ТОЧКАМИ НА ЗЕМЛІ. ЕВКЛІДОВА ВІДСТАНЬ ТА СИНУС-ВЕРЗУС МЕТОД 22
2.1. Опис застосованих математичних методів 22
РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ 29
3.1. Опис використаної архітектури та шаблонів проектування 30
3.2. Опис використаних технологій та мов програмування 31
3.3. Опис структури програми та алгоритмів її функціонування 42
3.4. Обґрунтування та організація вхідних та вихідних даних програми 59
3.5. Опис розробленого програмного продукту 60
3.5.2. Використані програмні засоби 60
3.5.3. Опис інтерфейсу користувача 61
ВИСНОВКИ 63
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 63
Додаток А. КОД ПРОГРАМИ 65
Додаток Б. ВІДГУК КЕРІВНИКА 99
Додаток В. РЕЦЕНЗІЯ 101
Додаток Г. ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ 103

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс для програмування застосунків;

БД – база даних;

ООП – об'єктно-орієнтоване програмування;

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології.

ВСТУП

В наш час, волонтерський рух широко розповсюджений серед людей різного віку. Складно уявити, як ми б протримались під час повномаштабної війни, якщо б не допомога волонтерів та небайдужих людей. Водночас, спектр допомоги волонтерів доволі широкий: фінансова, матеріальна, моральна фізична допомога, тощо. Поняття «волонтер» може бути визначений як особа, що здійснює допомогу на добровільних засадах, не отримуючи за це матеріальної винагороди. У наш час в Україні доволі волонтерів, тих, хто допомагає волонтерським фондам, та тих, хто потребує допомоги котрим надзвичайно важливо максимально ефективно комунікувати між собою та викладачами. Іноді це відбувається за допомогою різноманітних месенджерів, таких як Viber, Telegram, де розташовуються актуальні оголошення, іноді за допомогою телевізійних ефірів, де ведучі розповідають про організації, котрі займаються допомогою, також існують онлайн ресурси де ведеться пошук тих кому допомогти, або хто має можливість допомагати.

Водночас, якщо взяти інтернет сайти, то існуючі волонтерські онлайн платформи мають доволі різний функціонал: наприклад, один сайт може мати систему показу новин та не мати функціонал користувачів, а інший - навпаки. Тож, за мету було поставлено: по-перше, об’єднати корисний існуючий функціонал, по-друге, додати нових можливостей до розроблюваної системи.

Отже, в першому розділі проаналізовано предметну галузь, розглянуто наявні волонтерські платформи та порівняно їх з розроблюваним продуктом, також сформульовано вимоги щодо функціональних характеристик, інформаційної безпеки, параметрів технічних засобів. Другий розділ містить опис використаних математичних методів та формул. В третьому розділі детально описаний розроблюваний продукт: використана архітектура, схема бази даних та способи взаємодії з програмним інтерфейсом.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Волонтерство в світі

З давніх-давен люди займалися тим, що зараз називають волонтерством. Створення Міжнародного комітету Червоного Хреста Жаном-Анрі Дюнаном у 1859 році часто вважають початком сучасного волонтерського руху в західних країнах. Його ідеями керуються волонтери в усьому світі[18]. Протягом 20 століття волонтерство активно поширювалося. Особливо це проявляється у допомозі постраждалим від наслідків Першої світової війни. У середині століття під егідою ЮНЕСКО був створений Координаційний комітет з міжнародного волонтерства (CCIVS) зі штаб-квартирою в Парижі. Раніше волонтерами називали тих, хто пішов добровольцем на військову службу. «Волонтер» як окремий термін з’явилось трохи пізніше: у країнах Заходу – з другої третини 20 століття, на пострадянському просторі – з 1990-х років.

З 1970 року Міжнародна асоціація волонтерської служби кожні два роки організовує Всесвітній конгрес волонтерів. На такій зустрічі в Парижі в 1990 році була прийнята перша Загальна декларація волонтерів. Його було переглянуто на зустрічі в Амстердамі в 2001 році. Зокрема, волонтерство може втілювати та зміцнювати громаду, піклування, послуги, соціальну відповідальність, навчання та розвиток, може зміцнювати суспільство та вирішувати соціальні проблеми. Декларація підтримує право на волонтерство для всіх чоловіків, жінок і дітей незалежно від раси, релігії, фізичних характеристик, соціального та економічного статусу. [1]

Рис. 1.1.Логотип координаційного комітету міжнародної волонтерської служби (CCIVS)

1.1.2. Волонтерство в Україні

Уже в перші роки незалежності України волонтерство оживає і починає розвиватись. 1992 рік вважається датою, коли служба бере свій початок, а саме - «Телефон довіри», в якій працювали виключно волонтери[25]. Незабаром виникають організації, які починають реалізовувати найрізноманітніші соціальні проекти, такі як: допомога важкохворим, багатодітним сім'ям, піклування за одинокими людьми похилого віку, інвалідами. Активно розвивається напрям боротьби з розповсюдженням СНІДу та допомоги наркозалежним.

За перших років незалежності особливо ефективно працюють волонтерські напрями під егідою релігійних організацій, в тому числі протестантських церков. Для прикладу, Українська Уніонна конференція церкви адвентистів сьомого дня у середині 1990-х років ініціювала створення Адвентистської медичної асоціації України (продовжує діяти і сьогодні, в її рядах понад 700 фахівців вищої і середньої кваліфікації, які надають медичну допомогу нужденним в різних регіонах країни). Водночас, за допомогою Української лютеранської церкви вже більш ніж 15 років працює "Медична клініка на колесах". Громади Армії спасіння реалізують такі програми, як "Ліга милосердя" (започаткована для обслуговування самотніх хворих людей, прикутих до ліжка), "Відкриті двері" (передбачає різноманітну допомогу будинкам для осіб похилого віку),

“Інтенсивна корекція” (спрямована на кореляцію інтелектуального розвитку неповнолітніх у притулках та дитячих будинках). За трьома проектами – “Домашня опіка”, “Допомога на колесах дітям вулиці”, “Консультативний центр для жінок, постраждалих від торгівлі людьми” – працюють структури “Карітас” Української греко-католицької церкви у Хмельницькій області. У деяких населених пунктах Закарпатського регіону осередками Римсько-католицької церкви було засновано мережу безкоштовних аптек[24].

Варто відзначити, що, незважаючи на те, що волонтерська діяльність в Україні офіційно закріплена законом[16], загалом, до подій 2014 року, волонтерський рух в Україні розвивався досить мляво, у порівнянні з іншими країнами світу. Згідно рейтингу World Giving Index в 2010 році Україна займала лише 150 місце (тільки 5 процентів населення було залучено до волонтерської роботи) (рис 1.2.).

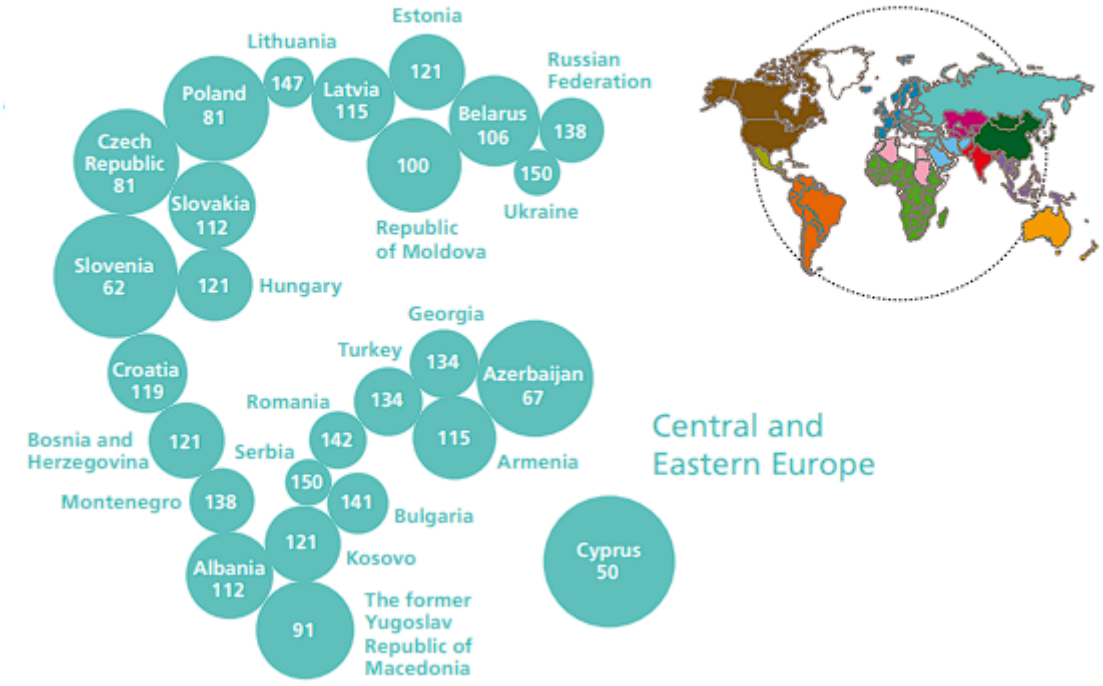

Рис. 1.2. Рейтинг World Giving Index, центральна та східна Європа(2010 р)

Для виявлення переваг й недоліків існуючих онлайн волонтерських платформ, що схожі на розроблюваний додаток, варто розглянути такі приклади, як UA Helpers, volonter.org та zaporuka.org.ua.

UA Helpers

UA Helpers був створений заради полегшення логістики серед волонтерів. Його мета: допомогти волонтерам, активним громадянам, які хочуть долучитися до ініціатив та людям, які потребують допомогу — знайти одне одного.[3]

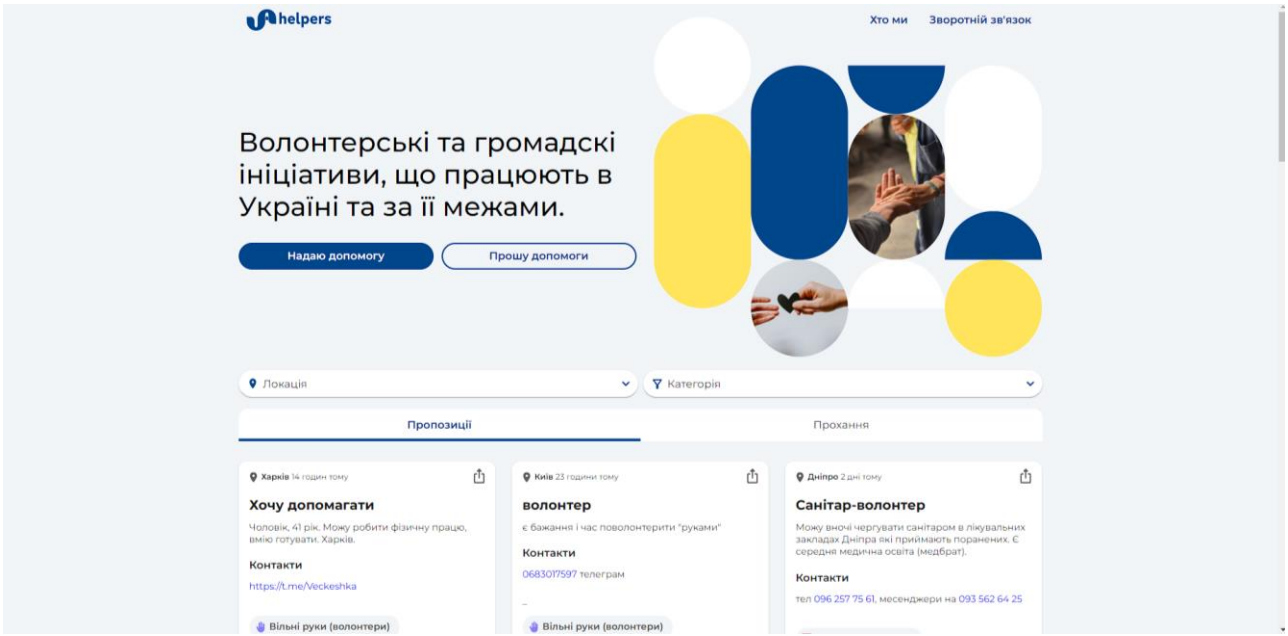

Рис 1.3. Стартова сторінка UA Helpers

За структурою, сайт поділений на два напрямки: пропозиції та прохання.

До пропозицій належить інформація про те, де у місті облаштовані місця для нічлігу, де можна отримати харчі, одяг, медикаменти та медичну допомогу, послуги з перевезення, психологічну допомогу та інше.

Водночас, у категорії «Прохання» можна дізнатися, якої співучасті потребують волонтери чи громадяни. Наприклад: продуктів для приготування їжі, фізичної допомоги для сортування чи розвантажування товарів, пального, допомоги війську, одягу та іншого.

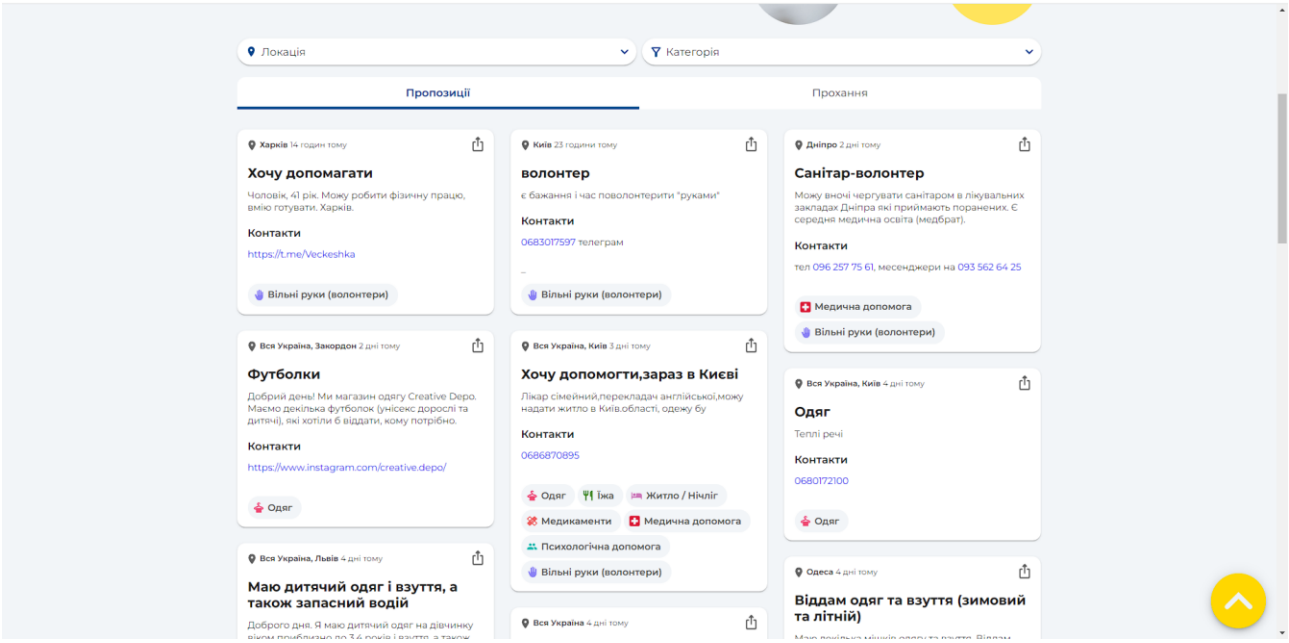

Рис 1.4. Приклад оголошень UA Helpers

Для пошуку інформації потрібно обрати локацію та необхідну категорію. За потреби обрати напрямок - “пропозиція” або “прохання”, таким чином користувач отримає перелік актуальних волонтерських пропозицій або потреб від волонтерів чи приватних осіб.

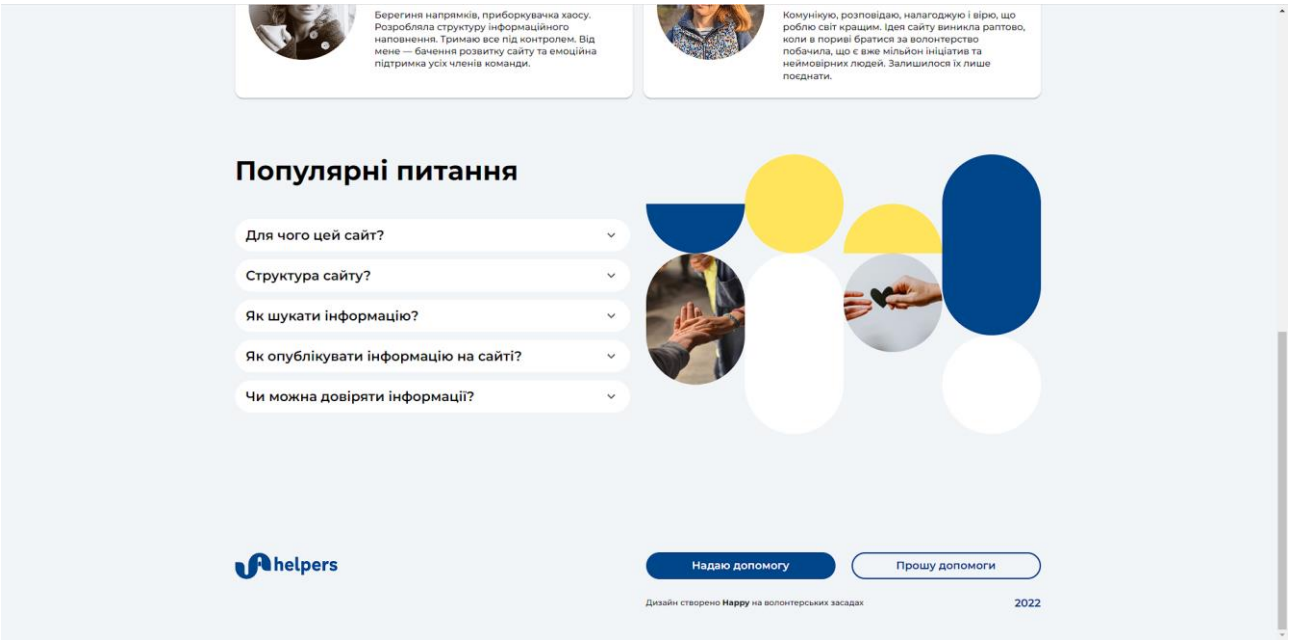

Рис 1.5. Приклад посилань для публікації оголошень.

Для публікації оголошення треба натиснути "Надаю допомогу" або "Прошу допомоги", де відкриється спеціальна анкета, яку необхідно заповнити. Після цього анкету отримають адміністратори сайту, які, після перевірки даних, публікують її.

volonter.org

volonter.org – був створений у 2014 році за сприяння ГО "Центр сприяння волонтерському руху Волонтер.орг" та волонтерській ініціативі "Герої АТО" Завдання платформи - побудувати горизонтальні зв'язки та встановити ефективну комунікацію між волонтерами, волонтерськими групами та громадськими організаціями. [4]

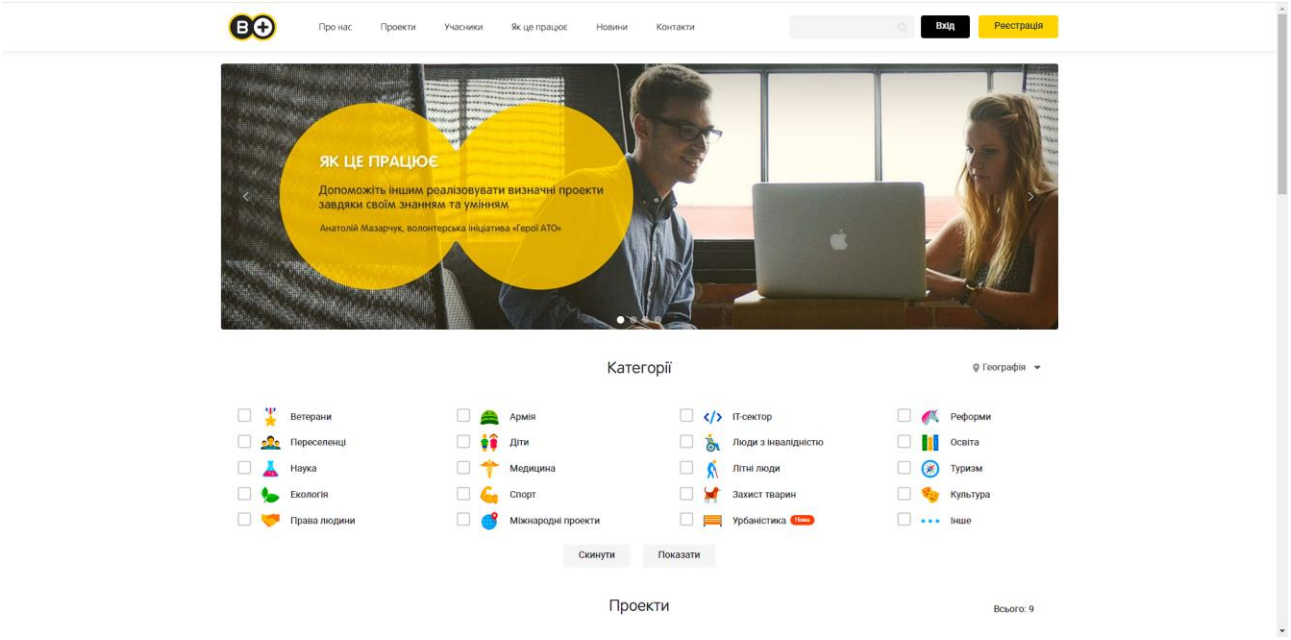

Рис 1.6. Стартова сторінка volonter.org

Під час реєстрації користувачу надається вибір статусу: волонтер, громадська організація чи громадський активіст.

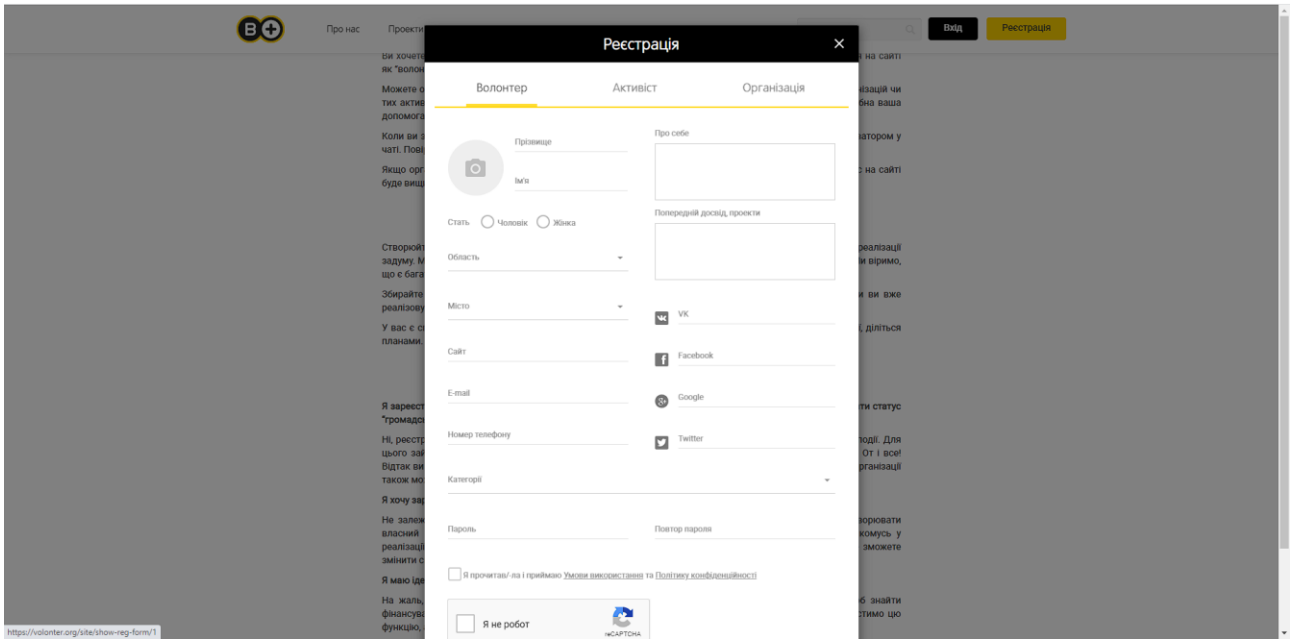

Рис 1.7. Форма реєстрації volonter.org

Роль «Волонтер» підходить більше тим, хто має вільний час, щоб приділити його волонтерській роботі. Користувач з такою роллю може обирати проекти за містом чи тематикою, або перегляньте усі, які зараз відкриті. Також присутня можливість підписки на оновлення організацій чи тих активістів, кому ви довіряєте та завжди бути в курсі, коли їм потрібна допомога.

Ролі «Активіст» або «Організація» для тих , хто створює власні події чи проекти. Таким користувачам доступний спеціальний чат із користувачами платформи, які зацікавлені у створеному проекті.

Водночас серйозний прив’язок ролі то типу створюваних постів немає – таким чином, волонтери можуть також створювати проекти, як організації або активісти, і навпаки - організації або активісти можуть приєднуватись до вже створених проектів.

zaporuka.org.ua

zaporuka.org.ua – волонтерський онлайн-ресурс, що було засновано в 2008 р. Засновницею та головою сервісу є Наталія Оніпко.

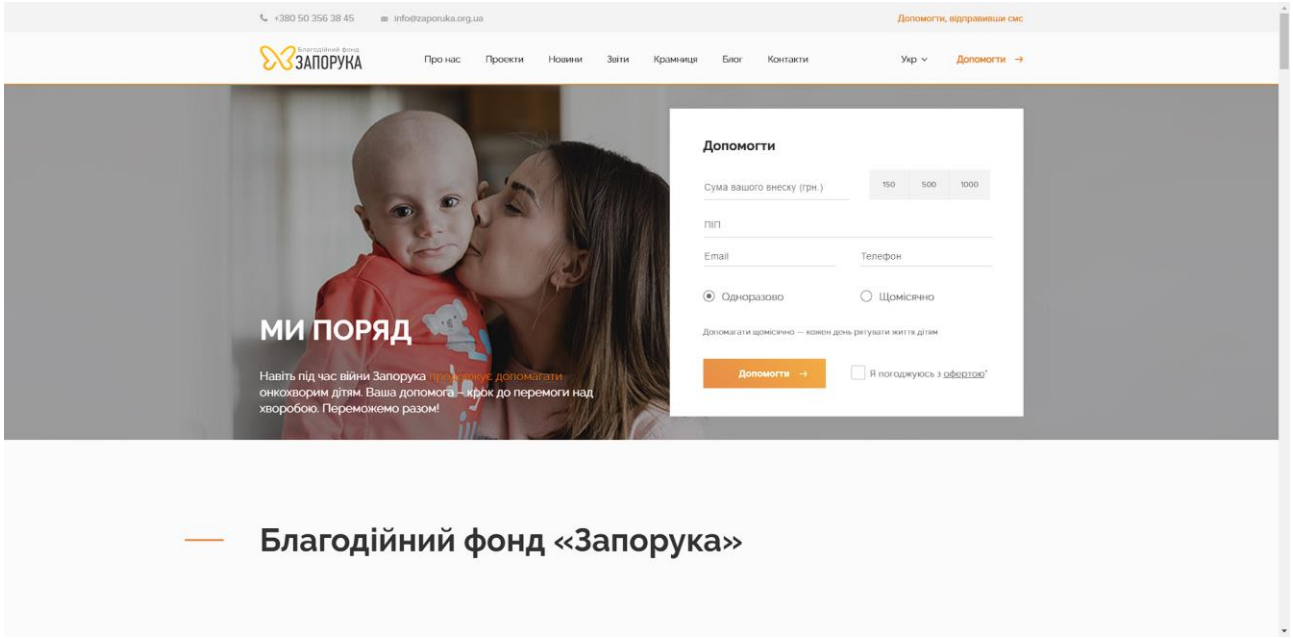

Рис. 1.8. Приклад сторінки zaporuka.org.ua

Головні напрямки фонду сьогодні:

– Забезпечення лікарень, які почали приймати дітей із гарячих точок України, медикаментами, ліками, обладнанням, реанімобілями, фахівцями. даними та іншою особистою інформацією.

– Психологічна підтримка у дитячих відділеннях Національного інституту раку та Інституту нейрохірургії ім. Ромоданова в Києві, лікарні Святого Миколая та Західноукраїнського спеціалізованого дитячого медичного центру у Львові.

– Гуманітарна допомога громадам, які прихистили вимушених переселенців у Київській, Черкаській, Полтавській, Вінницькій та Львівській областях.

– Реконструкція житла для вимушених переселенців.

– Гаряча лінія безкоштовної психологічної підтримки.

Також фонд «Запорука» піклується про онкохворих дітей, а саме: завдяки фонду, був відкритий єдиний в Україні центр для родин з онкохворими дітьми, в якому вони можуть безкоштовно жити разом протягом лікування. До 24 лютого 2022 року тут проживало 1300 дітей різного віку з усіх куточків країни. [5]

Але, на відміну від попередніх сервісів, тут немає системи користувачів, та проекти створюються виключно адміністраторами сайту. Водночас, тут

підключена система оплати, та сторінка зі звітами, яку адміністратори оновлюють кожен квартал.

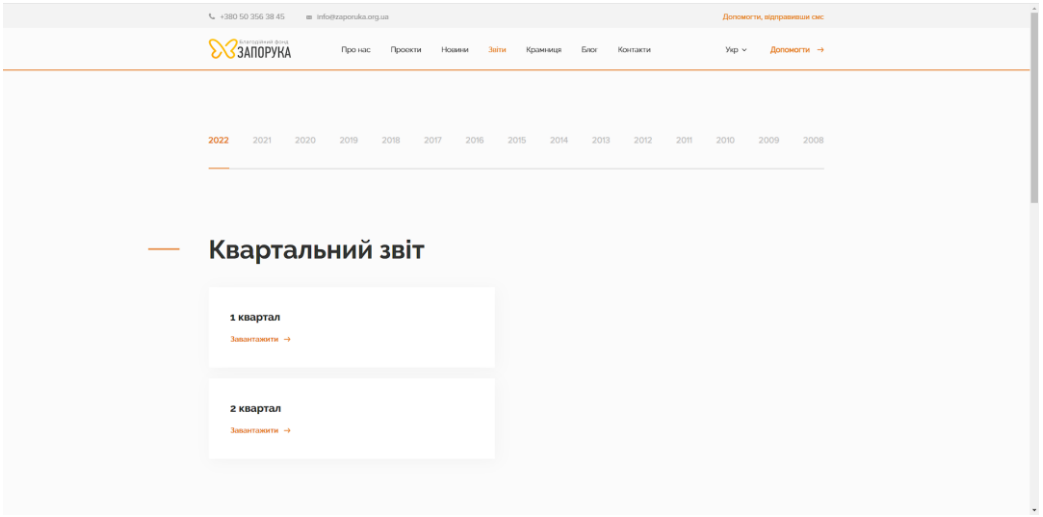

Рис. 1.9. Сторінка з квартальними звітами фонду «Запорука»

Також слід зазначити, що на сайті присутній функціонал новин, який оновлюється адміністраторами, та де користувачі можуть побачити свіжі новини.

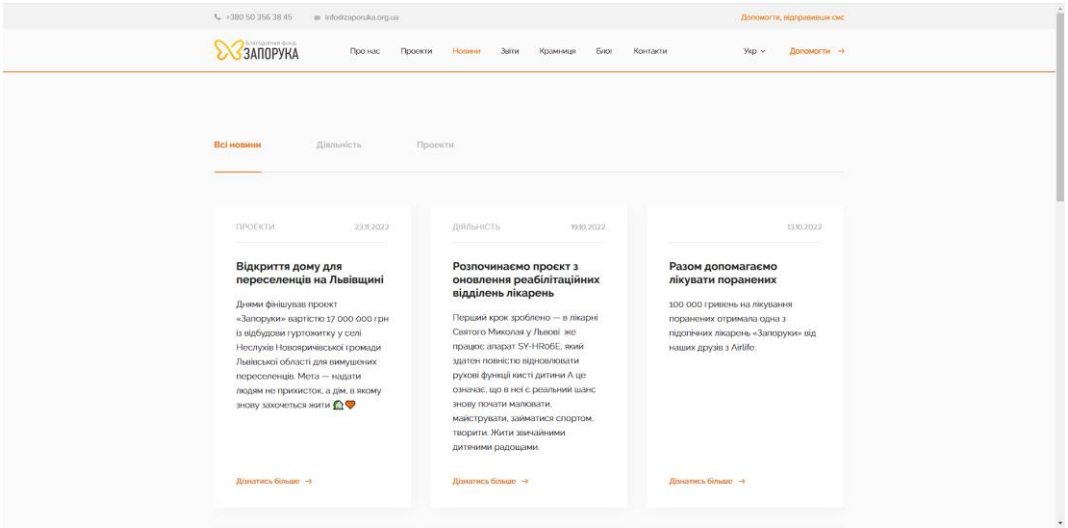

Рис. 1.10. Сторінка з новинами «Запорука»

1.2. Призначення розробки та галузь її застосування

Розробка та дослідження ефективності впровадження серверної частини волонтерської платформи. Розроблений продукт має використовуватися волонтерами/організаціями або тим, хто потребує допомоги.

Призначення розробки полягає наданні потенціальним користувачам наступних переваг автоматизованої системи:

– безперервна робота розроблюваної мережі

– дає можливість зареєструватися та використовувати розроблюваний мережу незалежно від місця перебування;

– дозволяє створювати проекти, долучатись до існуючих;

– дає можливість відстежувати проекти поблизу та швидко до них долучатися;

– дозволяє здійснювати комунікацію за допомогою постів та коментарів;

– дає можливість будувати власне соціально-волонтерське оточення через підписки на інших користувачів.

Таблиця 1.1

Переваги та недоліки програм

	UA Helpers	Volonter.org	zaporuka.org.ua	Розроблювана система
Можливість реєстрації користувачів	Ні	Так	Ні	Так
Можливість підписуватись на інших користувачів	Ні	Так	Ні	Так

Закінч. табл. 1.1

Можливість створювати/редагувати та переглядати проекти	Так	Так	Ні	Так
Можливість фільтрування проектів за локацією, та переглядання проектів поблизу	Так	Ні	Ні	Так
Можливість переглядати новини на сайті	Ні	Так	Так	Так
Можливість створювати публікації, коментувати їх та реагувати на них	Так	Так	Так	Так

З таблиці можливостей кожної з програм, можна зробити висновки, які можливості повинна надавати розроблювальна система, а яких недоліків вона повинна позбутися.

1.3. Постановка завдання

Завданням є розробка серверної частини волонтерської платформи. Обсяг вимог до характеристик даної програми – створити швидкий та універсальний застосунок зі зрозумілим програмним інтерфейсом. Розроблювана система повинна мати наступні можливості: реєстрація користувачів, створення / зміна / видалення категорій, ролей, локацій, реалізована система створення / зміна публікацій, та відповідей на них, можливість підписуватись на обраного користувача. Кожен проект або користувач матиме свою локацію, яка створюватиметься при реєстрації та може бути змінена відовідним запитом на

сервер. Завдяки цьому, програма матиме функціонал розрахунку відстані від однієї точки до іншої, та користувач буде здатний побачити інші об’єкти (проекти або інших користувачів), що знаходяться в заданому радіусі від нього.

1.4. Вимоги до програми або програмного виробу

1.4.1. Вимоги до функціональних характеристик

Кінцевий продукт має дотримуватися наступних функціональними вимог:

– вхідні дані на будь-який запит повинні вводитися користувачем у відповідну форму та передаватися на сервер у форматі JSON;

– при виникненні виняткових ситуацій, застосунок повинен уміти їх обробляти та відповідати коректними повідомленнями з поясненням що саме пішло не так;

– користувач має мати можливість редагувати власні дані;

– має бути реалізована система публікацій(проектів) та коментарів: створення / запиту / редагування / видалення публікації, створення / редагування / видалення коментаря;

– сервер повинен обробляти інформацію та віддавати «відповідь» на кожен HTTP запит;

– повинна бути реалізована можливість підписуватись на інших користувачів;

– для адміністраторів має бути можливість переглядати / створювати / видаляти новини на сайті;

– повинна бути фільтрування проектів за локацією, та переглядання проектів поблизу, у заданому радіусі.

– також має бути фільтрування користувачів за локацією, та знаходження користувачів поблизу, у заданому радіусі;

– повинна бути реалізована можливість створення різних акаунтів користувачів та повинна бути імплементована авторизація та аутентифікація.

1.4.2. Вимоги до інформаційної безпеки

Для забезпечення надійного функціонування системи необхідно реалізувати наступні вимоги:

– валідація введених даних: перевірка на відповідність типів, на введення обов’язкових полів даних;

– обробка виняткових ситуацій;

– виведення повідомлень у разі виникнення помилок;

– захист від несанкціонованого доступу до функціоналу за допомогою JWT (JSON WEB Token).

1.4.3. Вимоги до складу та параметрів технічних засобів

Для коректного функціонування кінцевого продукту, а саме автоматизованої системи управління та підтримки процесів онлайн продажів товарів електроніки необхідними технічними умовами є:

– операційна система Windows 10, Linux, MacOS, Android або IOS;

– обсяг оперативної пам’яті (ОЗУ) не менш 2 ГБ;

– наявність будь-якого сучасного браузера, наприклад, Google Chrome або Opera;

– підтримка високошвидкісного Інтернету.

РОЗДІЛ 2
ОБЧИСЛЕННЯ ВІДСТАНІ ВІД МЖ ТОЧКАМИ НА ЗЕМЛІ. ЕВКЛІДОВА ВІДСТАНЬ ТА СИНУС-ВЕРЗУС МЕТОД.

2.1. Опис застосованих математичних методів

Розроблюваний додаток має функціонал для пошуку проектів або користувачів поблизу (в межах заданого радіусу). Для роботи з координатами буде використовуватися просторовий розширювач бази даних для PostgreSQL – PostGIS, який уміє взаємодіяти з географічними координатами - з його допомогою будуть проводитись обчислення.

Кажучи про географічні координати ми не можемо використовувати прямокутну систему координат (рис. 2.1). Географічні координати не представляють лінійної відстані від початку координат, як нанесено на площину. Це сферичні координати, які описують кутові координати на глобусі. У сферичних координатах точка задається кутом повороту від контрольного меридіана (довгота) і кутом від екватора (широта).

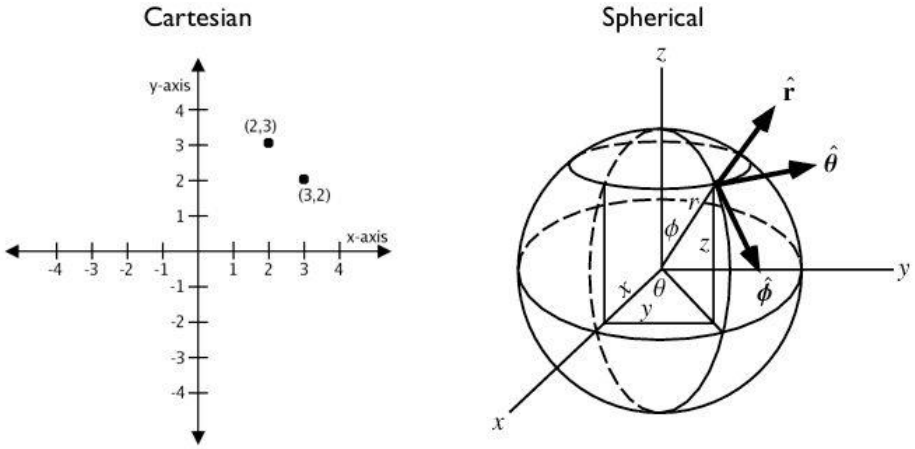

Рис. 2.1. Декартові та сферичні координати

Можна розглядати географічні координати як наближені декартові координати і робити на їх основі просторові обчислення. Однак вимірювання відстані, довжини та площі буде неточним. Оскільки сферичні координати

вимірюють кутову відстань, одиниці вимірювання рахуються в «градусах». Приблизні результати з використанням декартової системи можуть стати доволі неправильними.

Рис. 2.2. Мапа з відстанню між Нью-Йорком та Києвом

Для прикладу, розрахуємо відстань між Нью-Йорком (40° 42′ 51″п. ш. 74° 00′ 21″ з. д.) та Києвом (50° 27′ 16″ п. ш. 30° 31′ 25″с. д.) через Декартову систему координат, тобто відстань напряму, крізь Землю (рис. 2.2).

Технічно, широта і довгота представляють собою частину сферичних координат в тривимірному просторі. Радіус – це радіус Землі, кут нахилу - це широта, і азимут - це довгота. Якщо ми перетворимо сферичні координати в декартові координати x, y, z, ми зможемо легко знайти відстань між цими точками, використовуючи формулу Евклідової відстані (формула 2.1):

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \qquad (2.1)$$

Для перетворення сферичних координат до декартових використовуємо наступні формули 2.2:

$$x = R \cos\theta \cos\emptyset$$

$$y = R \cos\theta \sin\emptyset \qquad (2.2)$$

$$z = R \sin\emptyset,$$

де θ – широта, а $\emptyset$ – довгота.

Тепер пара тонких моментів, які виникають через той факту, що геодезія наближено представляє форму Землі у вигляді сплющенного сфероїда, або еліпсоїда обертання. Тому, коли ми говоримо про координати, ми насправді говоримо про координати на поверхні референц-еліпсоїда використовуваного в конкретній системі координат, в нашому випадку це WGS 84. І розрахована відстань, відповідно, буде відстанню між двома точками на поверхні референц-еліпсоїда.

Це призводить до необхідності врахування таких факторів:

– радіус Землі (радіус еліпсоїда) не є постійною величиною, і залежить від широти даної точки. Таким чином ми повинні розрахувати значення радіуса для кожної точки окремо.

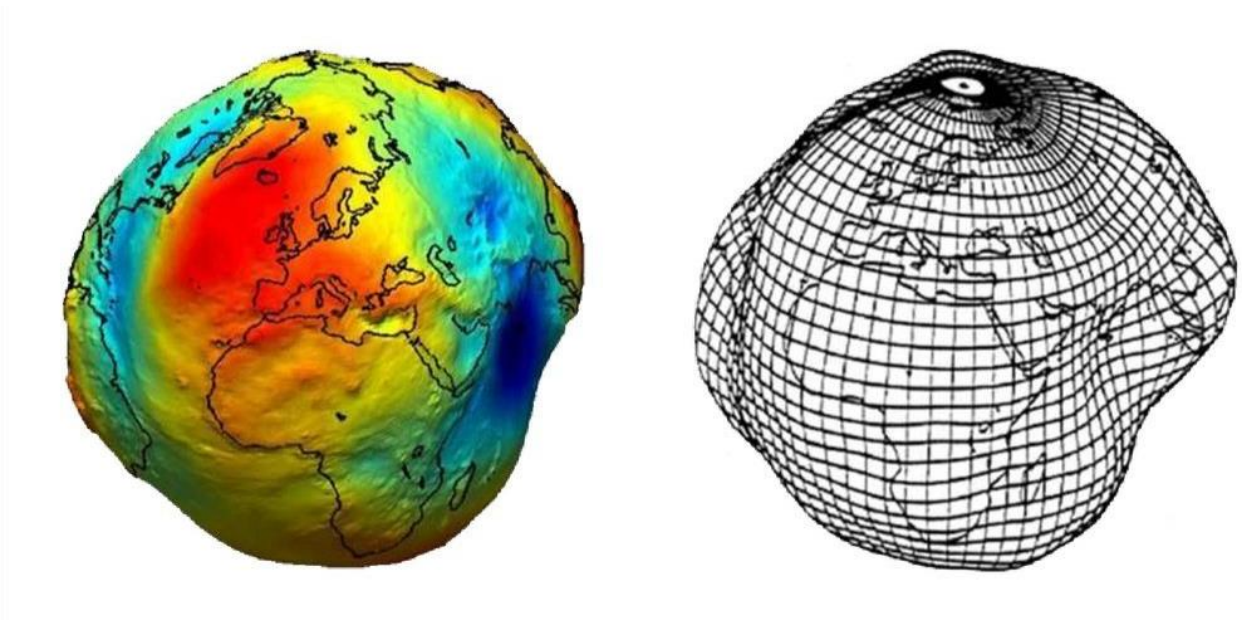

Рис. 2.3. Геоід

– широта зазначена в WGS 84 - це геодезична широта, яка визначається кутом між площиною екватора і нормаллю до поверхні еліпсоїда в даній точці, на відміну від геоцентричної широти, яка визначається кутом між екваторіальній

площиною і центром еліпсоїда. Так як в нашій декартовій системі координат початок координат знаходиться в центрі Землі, ми повинні перейти від геодезичних координат до геоцентричним для обох точок.

Враховуючи ці чинники, треба використати такі формули для перетворення широти і радіусу (формули 2.3):

$$\tan\alpha = \frac{b}{a}\tan\beta \qquad (2.3)$$

$$R = \sqrt{\frac{(a^2\cos\beta)^2 + (b^2\sin\beta)^2}{(a\cos\beta)^2 + (b\sin\beta)^2}},$$

де α – геоцентрічна широта, b – мала піввісь референц-еліпсоїда, a – велика піввісь референц-еліпсоїда, а β – геодезічна широта.

Отже, можна сказати, що для того, щоб розрахувати відстань напряму між двома точками за заданими координатами, треба зробити наступне:

- розрахувати значення радіуса Землі в кожній точці в залежності від широти;
- розрахувати геоцентричну широту в кожній точці;
- перейти від сферичних координат до декартових використовуючи довготу, розрахований радіус і розраховану геоцентричну широту;
- розрахувати відстань використовуючи формулу Евклідової відстані;

Проте, як можна було здогадатися цей підхід не працює для обчислення реальної відстані між двома координатами на планеті, адже евклідова метрика призначена для обчислення відстані на площині, а поверхня Землі - це все-таки фігура, дуже близька до сфери. Для вирішення такого завдання потрібно звернутися до рідко використовуваних тригонометричним функціям.

Одна з таких функцій, називається синус-верзус (формула 2.4), або, по-іншому, версінус. Він являє собою відстань від центральної точки дуги, вимірюваної подвоєним даними кутом, до центральної точки хорди, що стягує дугу. Обчислюється версінус за формулою:

$$\text{versin}\ (\theta) = 2\ \sin^2\left(\frac{\theta}{2}\right) \tag{2.4}$$

Гаверсінус - це просто половина версінуса (формула 2.5), і саме ця функція допоможе нам у вирішенні завдання з пошуком відстані:

$$\text{hav}\ (\theta) = \frac{\text{versin}\ (\theta)}{2} = \sin^2\left(\frac{\theta}{2}\right) \tag{2.5}$$

Для будь-яких двох точок на сфері гаверсінус центрального кута між ними обчислюється за формулою 2.6:

$$\text{hav}\ \left(\frac{d}{r}\right) = \text{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\ \text{hav}(\lambda_2 - \lambda_1), \tag{2.6}$$

де d – це центральний кут між двома точками, що лежать на великому колі, r – радіус сфери, φ_1 та φ_2 – широти першої і другої точок в радіанах, λ_1та λ_2 – довготи першої і другої точок в радіанах.

Позначимо гаверсінус відношення довжини до радіусу як змінну h (формула 2.7):

$$\text{hav}\ \left(\frac{d}{r}\right) = h \tag{2.7}$$

Тоді довжину d можна винести за знак рівності (формула 2.8):

$$d = r\ \text{hav}^{-1}\ (h) \tag{2.8}$$

А для того, щоб позбутися від дробу, висловимо гаверсінус через арксинус (формула 2.9):

$$d = 2r \arcsin\sqrt{h} \tag{2.9}$$

Розкриємо змінну *h* ():

$$d = 2r \arcsin \sqrt{\text{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\,\text{hav}\,(\lambda_2 - \lambda_1)} \qquad (2.10)$$

Підставимо формулу гаверсінуса і отримаємо формулу 2.11 обчислення відстані:

$$d = 2r \arcsin \sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\,\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)} \qquad (2.11)$$

Перш, ніж підставляти координати в формулу, їх потрібно перевести в радіани та не забути, що Земля не є ідеальною сферою і її радіуси трохи варіюються (рис. 2.3). Скористаємося усередненим значенням радіуса, яке, відповідно до стандарту WGS84 приблизно дорівнює 6371 км. [6]

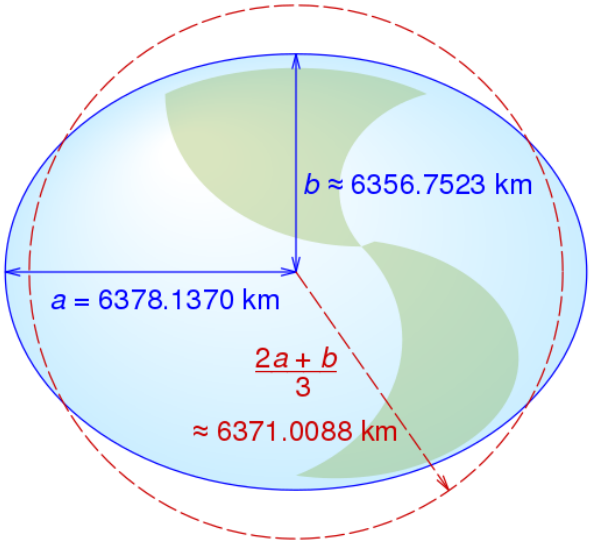

Рис. 2.4. Радіуси Землі

Результати дослідження:

Відстань від Києва до Нью-Йорка:

Через Евклідову відстань – **12386.16** км.

Через синус-верзус - **7510.93** км.

Реальна відстань – **7519** км.

Тобто похибка обчислення через евклідову відстань дорівнює майже **4867** кілометрам, що є дуже великим значенням. Водночас, похибка обчислення через синус-верзус становить лише **8.07** км. Це означає, що для обчислення відстані між віддаленими точками на планеті потрібно використовувати синус-верзус підхід, а коли ми працюємо в масштабах міста або району, можна знехтувати і використати евклідову відстань.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1. Опис використаної архітектури та шаблонів проектування

Шаблони проектування являють собою певний спосіб вирішення певної архітектурної проблеми. Тобто до них можна й потрібно відноситись, як до багаторічного досвіду проектування та побудови складних систем, що вийшов у набір основних прийомів та методів.

Патерни розрізняють за видами[20]:

- породжуючі патерни;
- структурні патерни;
- патерни поведінки;

Для даного додатку була вибрана класична архітектура програмування

REST:

REST (Representational State Transfer) — це стиль архітектури, що використовує HTTP, як протокол передачі даних. Системи, що використовують такий підхід, називаються RESTful – системами (див. рис. 2.1).

Перший, хто використав термін REST, був Рой Філдінг, котрий описав цей архітектурний стиль у своїй докторській дисертації[12]. Згідно цієї роботи, REST – це загальний набір принципів побудови архітектури систем як у веб-просторі, так і у вбудованих систем, оскільки він не задає деталі реалізації.

Існує 6 загальних правил побудови REST-застосунку:

1. Клієнт-серверна модель

Віповідне розмежування обов’язків:

– Фронтенд частина додатку відповідає за візуальне оформлення та безпосередньо взаємодіє з клієнтом, відправляючи відповідні запити до бекенду;

– Бекенд в свою чергу приймає ці запити, оброблює їх та відповідає фронтенду.

2. Кеширування

Збереження даних на стороні клієнта.

Інколи нема потреби кожен раз відсилати запит на бекенд. Для забезпечення швидкої роботи застосунку деякі результати запитів можна зберігати на стороні клієнта.

3. Відсутність стану

Відсутність збереженої інформації про клієнта на сервері. Усі запити супроводжуються шифрованими токенами доступу, які зберігають усю необхідну інформацію про клієнта.

4. Однорідний інтерфейс

Розроблюваний додаток має містити впорядковану структуру інтерфейсу. Енпоїнти мають бути описані в однорідному стилі, формат представлення має бути чітко визначений тощо

5. Шари абстракції

Поділ функціоналу на шари абстракції. Іншими словами – це декомпозиція та спрощення коду: кожен шар має працювати із відповідним йому функціоналом.

6. Запитування коду

Можливість завантаження деяких фрагментів коду у вигляди скриптів. Такий підхід хоч і «розвантажує» серверну частину, але це може бути небажаним с точки зору безпеки.

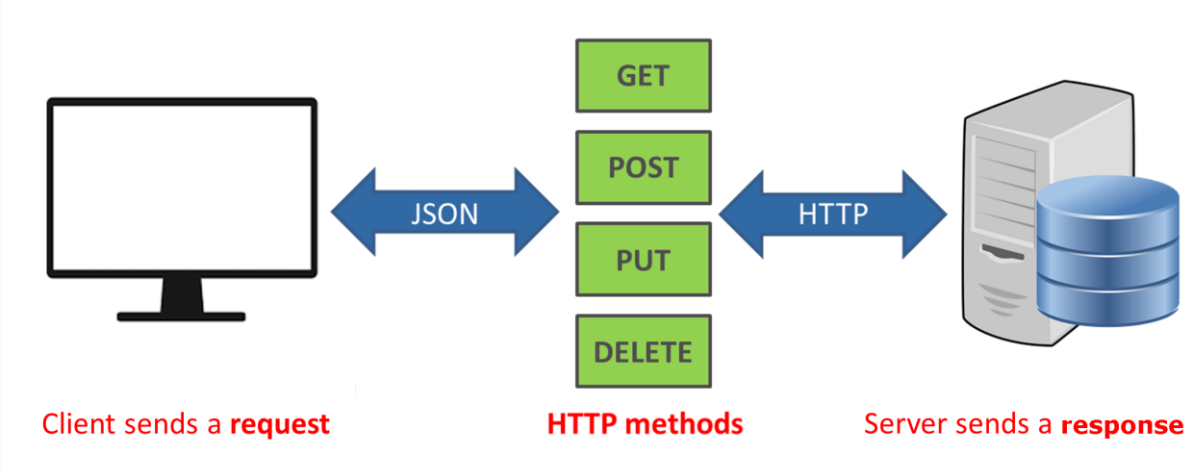

Рис. 3.1. Архітектура REST API

3.2. Опис використаних технологій та мов програмування

Дана інформаційна система була розроблена з використанням таких технологій:

- Java.
- Json Web Token.
- Spring Boot.
- OpenAPI 3.0.
- PostgreSQL.
- PostGIS.
- Liquibase.
- Maven.
- Prometheus.
- Micrometer.

Java

Java — об'єктно-орієнтована мова програмування, випущена В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Використовування даного язика не залежить від платформи. [13]

Ключовою особливістю мови Java є те, що його код спочатку транслюється в спеціальний байт-код, сумісний із різними платформами. А потім цей байт-код виконується віртуальною машиною JVM (Java Virtual Machine). В цьому плані Java відрізняється від стандартних різних мов як Pyrhon або Ruby, код яких відразу ж виконується інтерпретатором. У той же час Java не є і чисто компільовані мовою, як C або C ++.[2]

Подібна архітектура забезпечує кроссплатформенность і апаратну переносимість програм на Java, завдяки чому подібні програми без перекомпіляції можуть виконуватися на різних платформах - Windows, Linux, Mac OS і т.д. Для кожної з платформ може бути своя реалізація віртуальної машини JVM, але кожна з них може виконувати один і той же код.

Ще однією ключовою особливістю Java є те, що вона підтримує автоматичну збірку сміття. А це означає, що не треба звільняти вручну пам'ять від раніше використовувалися об'єктів, як в C ++, так як збирач сміття це зробить автоматично за вас[17].

Рис. 3.2. Логотип Java

Плюси:

– ООП – мова програмування;

– мультифункціональність;

– надійність, завдяки строгій типізації;

– крос-платформенність;

– простий С-подібний синтаксис;

– автоматична збірка сміття(видалення із пам’яті неактивних об’єктів, або об’єктів, на які немає посилань);

– Java має потужний інструментарій для розробки Android-застосунків.

Мінуси:

– використовує багато ОЗУ;

– Java – високорівнева мова програмування, не можна напряму керувати вказівниками;

– швидкість порівняно менша, ніж у С та С++.

Json Web Token

JSON Web Token — це стандарт токена доступу на основі JSON[11]. Використовується для верифікації тверджень.

Такий токен складається з трьох частин: заголовку, вмісту на підпису.

Заголовок(HEADER):

Це JSON елемент, що описує до якого типу належить даний токен і які методи шифрування використовувались.

Таблиця 3.1

Елементи JWT заголовку

Поле	Назва	Значення
typ	Type	JWT завжди мають медіатип ***application/jwt.***
cty	Content type	це поле встановлюється коли JWT містить в собі інший JWT, в противному разі це поле пропускається.
alg	Algorithm	описує використаний алгоритм шифрування. Зазвичай використовують HS256 або RS256. Можна також не використовувати жодного шифрування, вказавши none, але це не рекомендується.

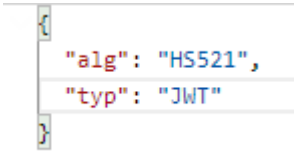

Рис. 3.3. Приклад заголовку JWT

Вміст складається з елемента JSON який описує змінні, що задані користувачем. Вміст також має деякі необов’язкові зарезервовані значення:

Таблиця 3.2

Елементи JWT вмісту

Поле	Назва	Значення
Aud	Audience	цільова аудиторія токена.
Exp	Expiration Time	час закінчення терміну дії токена в форматі Unix (кількість секунд що пройшли від 1970-01-01T00:00:00Z).
Iat	Issued At	коли токен був виданий (в форматі Unix)
Iss	Issuer	той хто видав токен
Jti	JWT ID	унікальна, чутлива до регістру послідовність символів, яка однозначно ідентифікує токен. Може застосовуватись для запобігання повторному використанню токена. Це можуть бути порядкові числа, GUID або Хеш.
Nbf	Not Before	час в форматі Unix до настання якого токен не дійсний.
sub	Subject	визначає якого суб'єкта стосуються твердження, тобто щодо кого або чого робляться твердження.

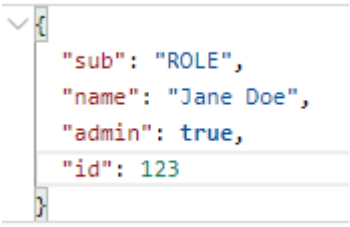

Рис. 3.4. Приклад вмісту JWT.

Підпис – остання частина токену, може бути довільної строкою, що задана користувачем, або ж генерується за допомогою заголовку та вмісту.

Заголовок, вміст і підпис кожен кодуються в Base64 і розділяються в токені крапкою. JWT може виглядати так:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzY290Y2guaW8iLCJleHAiOjEzMMR6MTkzdHJ1ZX0.03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f34533.

Тобто:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 – це закодований заголовок,

eyJpc3MiOiJzY290Y2guaW8iLCJleHAiOjEzMMR6MTkzdHJ1ZX0– закодований вміст, та

03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f34533– відповідно закодований підпис.

Spring Boot

Spring Boot — це веб-фреймворк, що спрощує написання клієнт-серверних веб-застосунків за допомогою потужного вбудованого функціоналу [7].

Даний фреймворк може розглядатись як сукупність інших, взаємно незалежних фреймворків, котрі можуть як працювати окремо один від одного, так і разом, забезпечуючи максимальну ефективність роботи. Складові Spring діляться на наступні структурні елементи, серед яких найважливішими є:

– Inversion of Control-контейнер: конфігурування компонентів застосунку та керування життєвим циклом об’єктів[23].

– Фреймворк, що забезпечує безпеку застосунку: інструментарій конфігурації процесів автентифікації та авторизації, відомий як Spring Security.

– Фреймворк аспектно-орієнтованого програмування: компонент, відомий як AspectJ, за допомогою якого стає можливе так зване наскрізне програмування (можливість запроваджувати схожі блоки коду в різні місця програми під час її виконання) [19].

– Фреймворк доступу даних: у Spring Boot таку роль виконує ORM Spring Data Jpa, за допомогою якої можна будувати інтуїтивно зрозумілі та прості запроси до серверу бази даних.

– Фреймворк для роботи з транзакціями: інструментарій налаштування транзацій.

– Фреймворк MVC: інструментарій для налаштування HTTP – запросів, що надає великий спектр можливостей для розробника.

– Тестування: Spring Boot надає широкі можливості для застосування різноманітних фрейморків тестування, таких як TestNG, Spock Framework, Junit тощо.[9]

Порівняльна характеристика популярних веб-фреймворків:

Таблиця 3.3

Порівняльна таблиця можливостей різних веб – фреймворків.

Частина 1

Проект	Мова	Ajax	MVC - Framework	Інтер - націоналізація	ORM	Testing frameworks
Spring	Java	+	+	+	Hibernate, Jpa, iBatis.	Mockito, JUnit, TestNG
ASP.NET	C#	+	+	+	+	xUnit.net, MS Test, NUnit
Grails	Groovy	+	+	+	Hibernate, JPA, GORM	Spock Framework, JUnit
Django	Python	+	+	+	Django ORM	+

Таблиця 3.4

Порівняльна таблиця можливостей різних веб – фреймворків.

Частина 2

Проект	Мова	DB migration frameworks	Security frameworks	Teamplate frameworks	Caching frameworks	Validation frameworks
Spring	Java	-	Spring Security	JSP, Commons Tiles, Velocity, Thymeleaf	ehcache	Commons validator, Bean Validation
ASP.NET	C#	Entity Framework	ASP.NET Forms Authentication (Default), Pluggable	Razor (Default), ASPX, Pluggable	+	+
Grails	Groovy	multiple plugins: autobase, dbmigrate	Spring Security, Apache Shiro	+	+	+
Django	Python	Built into core	ACL-based	Django Template Language	Cache Framework	+

Swagger

Swagger – це фреймфорк специфікації REST-API, завдяки якому можна формалізувати існуючі ендпоінти та задокументувати створений API[21].

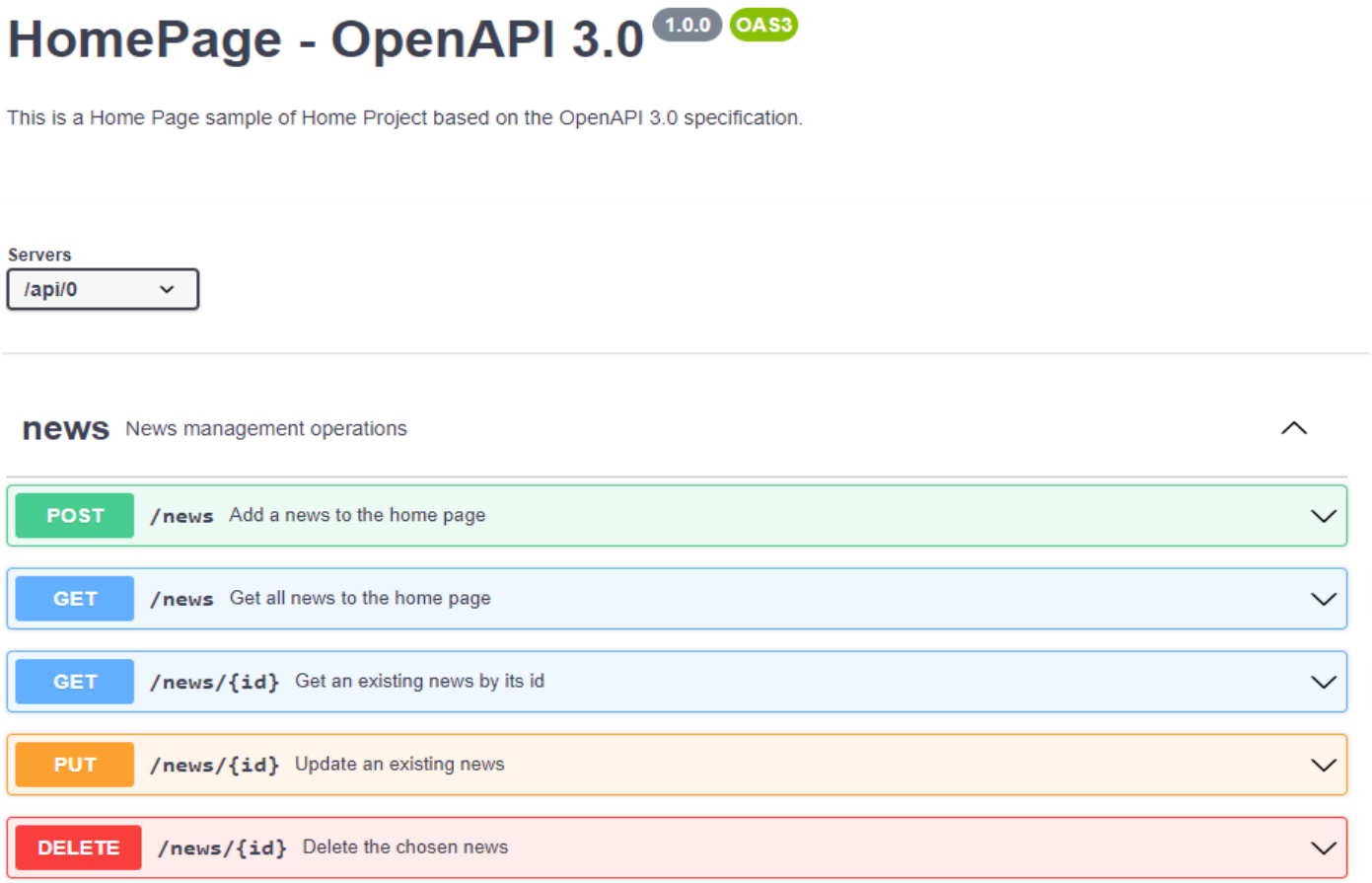

Рис. 3.5. Приклад OpenAPI специфікації.

Як можна бачити, ми можемо не тільки переглядати специфікацію, а й відправляти відповідні запити. Також є можливість сгенерувати сервер або клієнт, на основі існуючого конфігураційного файлу специфікації.

Swagger має два основних підходи написання документації[8]:

– Code-first

Сутність цього підходу полягає у первинному написанні коду з подальшим його покриттям документації.

– API-first

Цей підхід має деякі переваги над попереднім, а саме:

– Уся документація буде суворо формалізована;

– Розробник з самого початку буде знати як правильно йому розробляти API;

Але це не так просто, адже для написання специфікації треба знати синтаксис Swagger Specification та написати її однією з трьох мов (YAML, JSON або ж XML).

Liquibase

Liquibase - це бібліотека з відкритим кодом, незалежна від ядра БД, для відстеження, управління та застосування змін у схемі БД.

Усі зміни в базі даних зберігаються у текстових файлах (XML, YAML, JSON або SQL) та ідентифікуються за допомогою комбінації тегів "id" та "author", а також імені самого файлу. Список усіх застосованих змін зберігається у кожній базі даних, у таблиці DATABASECHANGELOG з якою проводиться консультація щодо всіх оновлень бази даних, щоб визначити, які нові зміни потрібно застосувати. [14]

Liquibase автоматично генерує таблиці DATABASECHANGELOG та DATABASECHANGELOGLOCK під час першого застосування скриптів.

Із відомих аналогів бібліотек для міграції БД, що працюють з системами, розробленими на Java, існує FlywayDB.

Таблиця 3.5

Порівняння FlywayDB та Liquibase

	FlywayDB	Liquibase
Diff(порівняння двух баз даних)	-	+
Можливість ролбеку	+ (за додаткову плату)	+
Формати	SQL	SQL, XML, YAML, JSON

Закінч. табл. 3.5

Можливість генерування SQL	-	+
Можливість застосовувати міграції в залежності від передумов	-	+

Maven

Maven - інструмент для автоматичної збірки проектів. З ним працюють в основному Java-розробники, хоча є плагіни для інтеграції з C / C ++, Ruby, Scala, PHP і іншими мовами.[15]

Одна з головних особливостей фреймворка – декларативний опис проекту. Це означає, що розробнику не потрібно приділяти увагу кожному аспекту збірки, всі необхідні параметри налаштовані за замовчуванням. Зміни потрібно вносити лише в тому обсязі, в якому програміст хоче відхилитися від стандартних налаштувань.

Ще одна перевага проекту - гнучке управління залежностями. Maven автоматично створює на комп’ютері локальний репозиторій та вміє довантажувати в свій нього сторонні бібліотеки, вибирати необхідну версію пакету, обробляти транзитивні залежності.

Також даний фреймворк не залежить від ОС. При роботі з командного рядка параметри залежать від платформи, але Maven дозволяє не звертати уваги на цей аспект.

При необхідності систему збирання можна налаштувати під власні потреби, використовуючи готові плагіни. А якщо нічого підходящого не знайшлося - можна написати свої[22].

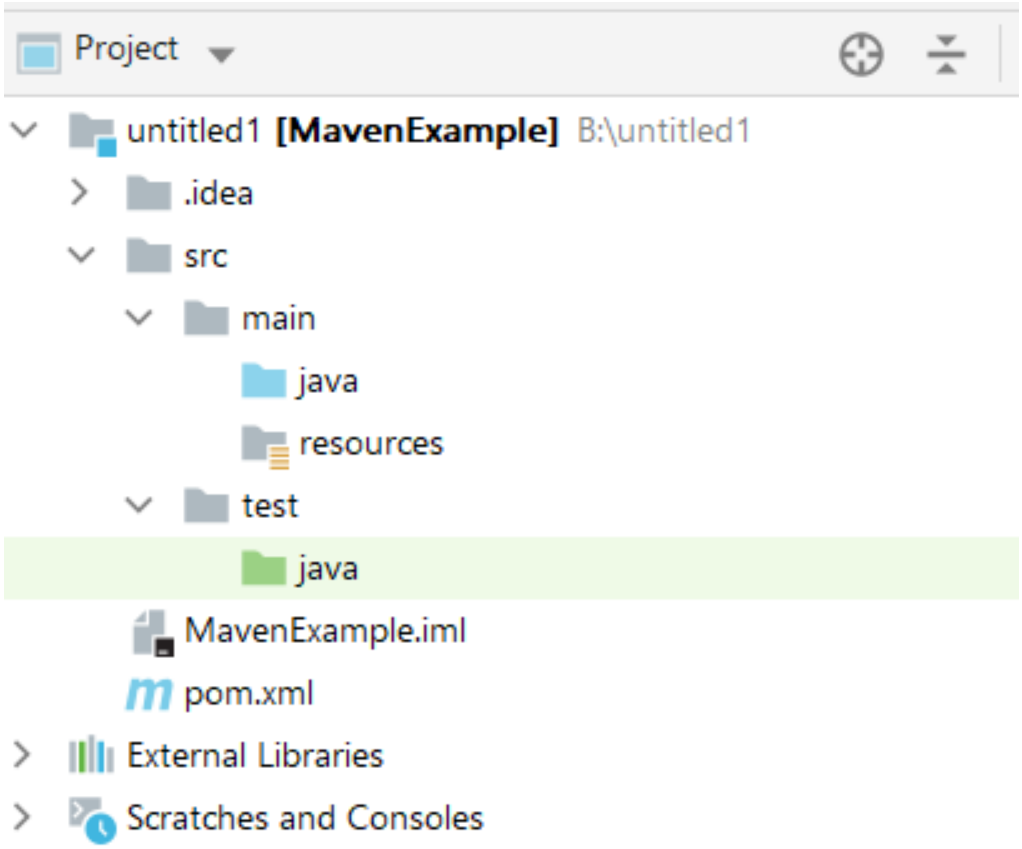

Рис 3.6. Структура проекту Maven

– src/main/java – директорія з файлами, де лежать виконувані Java класи;
– src/main/resources – директорія для конфігураційних файлів;
– src/test/java – директорія де зберігаються тести.

3.3. Опис структури програми та алгоритмів її функціонування

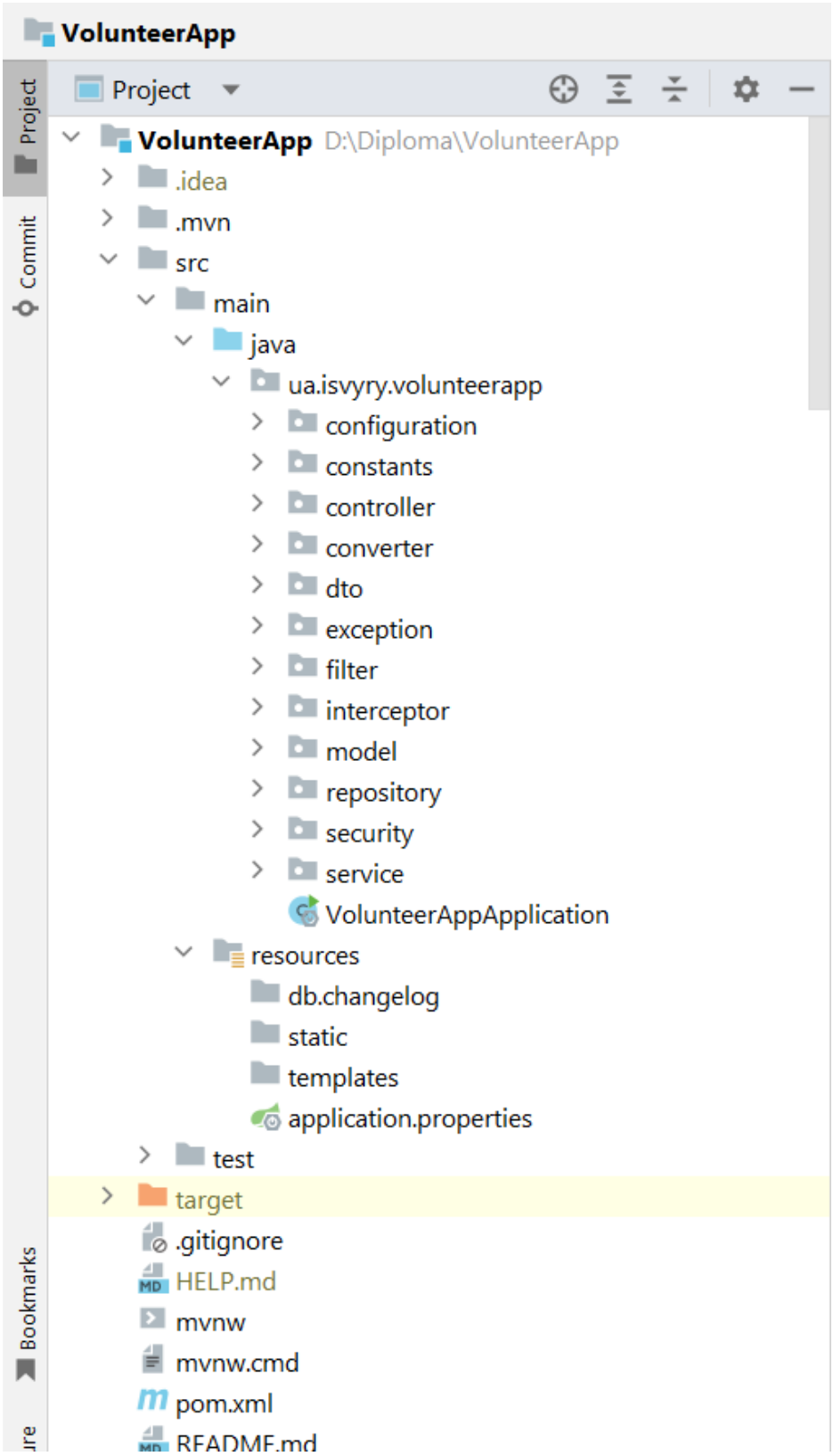

Рис. 3.7. Структура проекту

Дана програма використовує декілька директорій. Це означає додаток ділиться на декілька окремих папок, кожна з яких складається з класів та має власну сферу відповідальності. Давайте розглянемо найважливіші з них:

configuration:

configuration – директорія конфігурацій розроблюваного додатку, тут знаходяться класи конфігурацій.

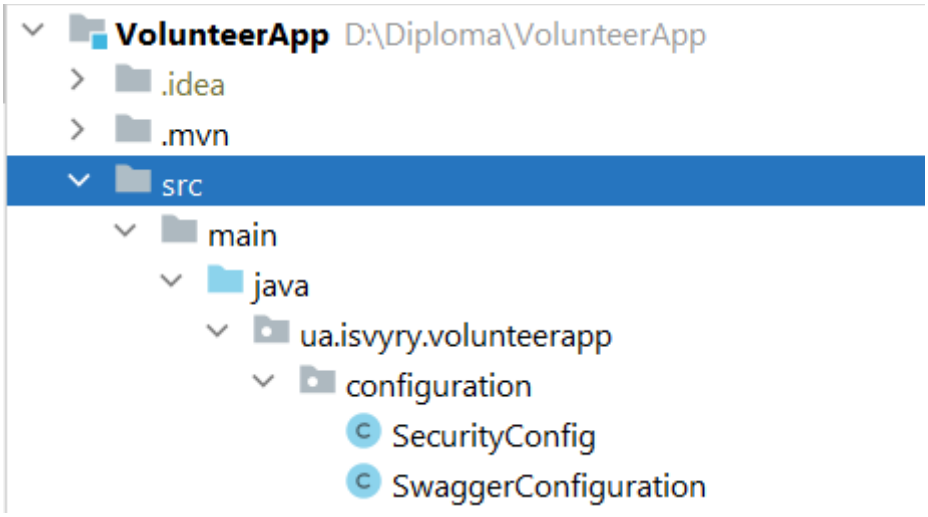

Рис. 3.8. Структура директорії configuration

model, repository:

Дані директорії – це рівень взаємодії з базою даних. Тут знаходяться репозиторії та класи-моделі.

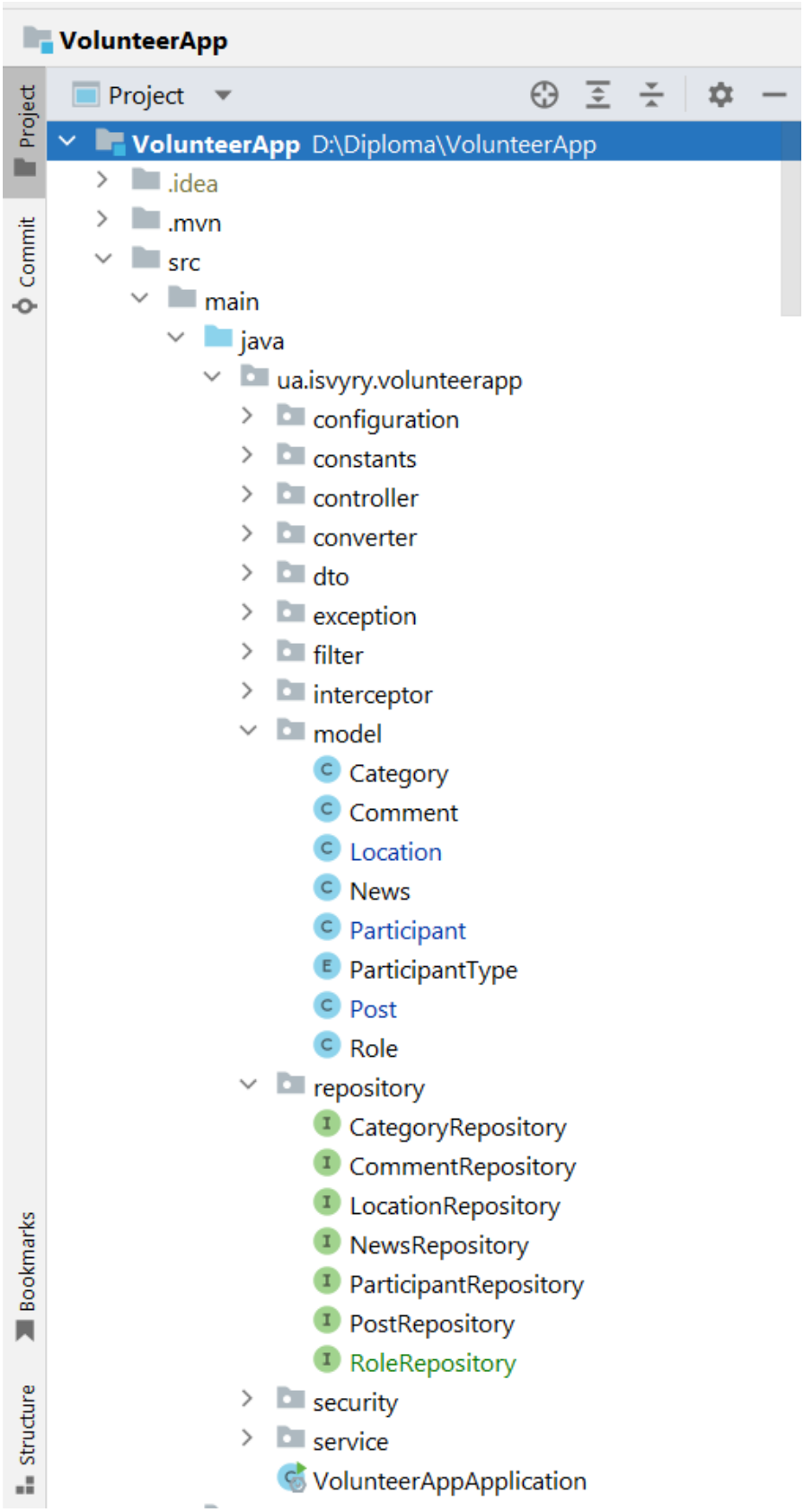

Рис. 3.9. Структура директорій model та repository

Рис. 3.10. Найважливіші репозиторії проекту

Як можна бачити на наведених діаграмах, усі репозиторії успадковують інтерфейс JpaRepository, який включає в себе найнеобхідніші методи для роботи з базою даних, такі як:

findAll(): знаходить усі записи в таблиці, еквівалентний запису «SELECT * FROM `*ENTITY*`»

deleteAll(): видаляє усі записи з таблиці, еквівалентний запису «DELETE FROM `*ENTITY*`»

deleteById(ID): видаляє з таблиці запис, що відповідає вказаному параметру, еквівалентний запису «DELETE FROM `*ENTITY*` WHERE ENTITY_ID = `*ID*`»

findById(ID): знаходить запис в таблиці, що відповідає вказаному параметру, еквівалентний запису «SELECT * FROM `*ENTITY*` WHERE ENTITY_ID = `*ID*`»

findAll(Pageable): знаходить записи в таблиці, що відповідають вказаному параметруPageable, за допомогою якого можна розбивати результат запиту на сторінки та діставати вказану.

findAllById(Collection<ID>): знаходить в таблиці всі записи, що відповідають критеріям пошуку, еквівалентний запису «SELECT * FROM `*ENTITY*` WHERE ENTITY_ID IN (Collection<ID>)»

saveAll(Collection<Entity>): зберігає або оновлює існуючі записи на основі вказаних параметрів.

Також, окрім стандартних реалізацій запитів, які нам надає Spring Data Jpa, існує можливість декларативним способом описати запити, які ми хочемо надіслати до бази даних.

Наприклад, ми хочемо знайти користувача, згідно згідно його е-мейлу. Для цього, у відповідному репозиторії, ми маємо задекларувати метод, у назві якого буде вказано поле, згідно якого буде побудований запит до БД, у нашому випадку це поле - email. Отже, наш метод буде виглядати наступним чином: findByEmail(String email).

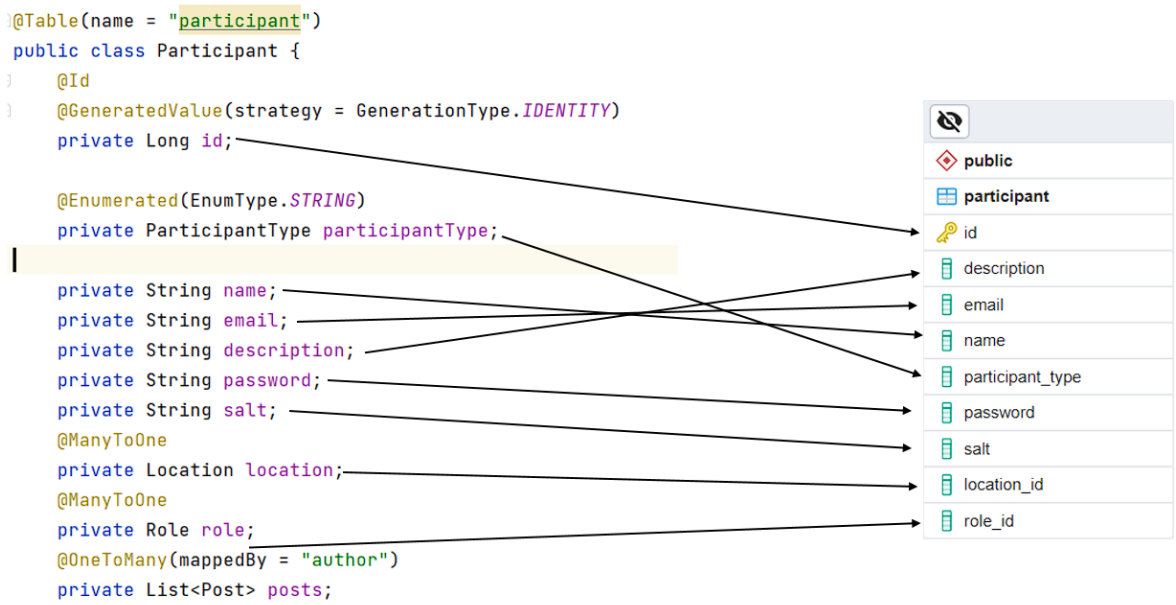

Рис. 3.11. Відповідність між властивостями класу та таблиці БД

service:

Директорія service відповідає за бізнес-логіку застосунку.

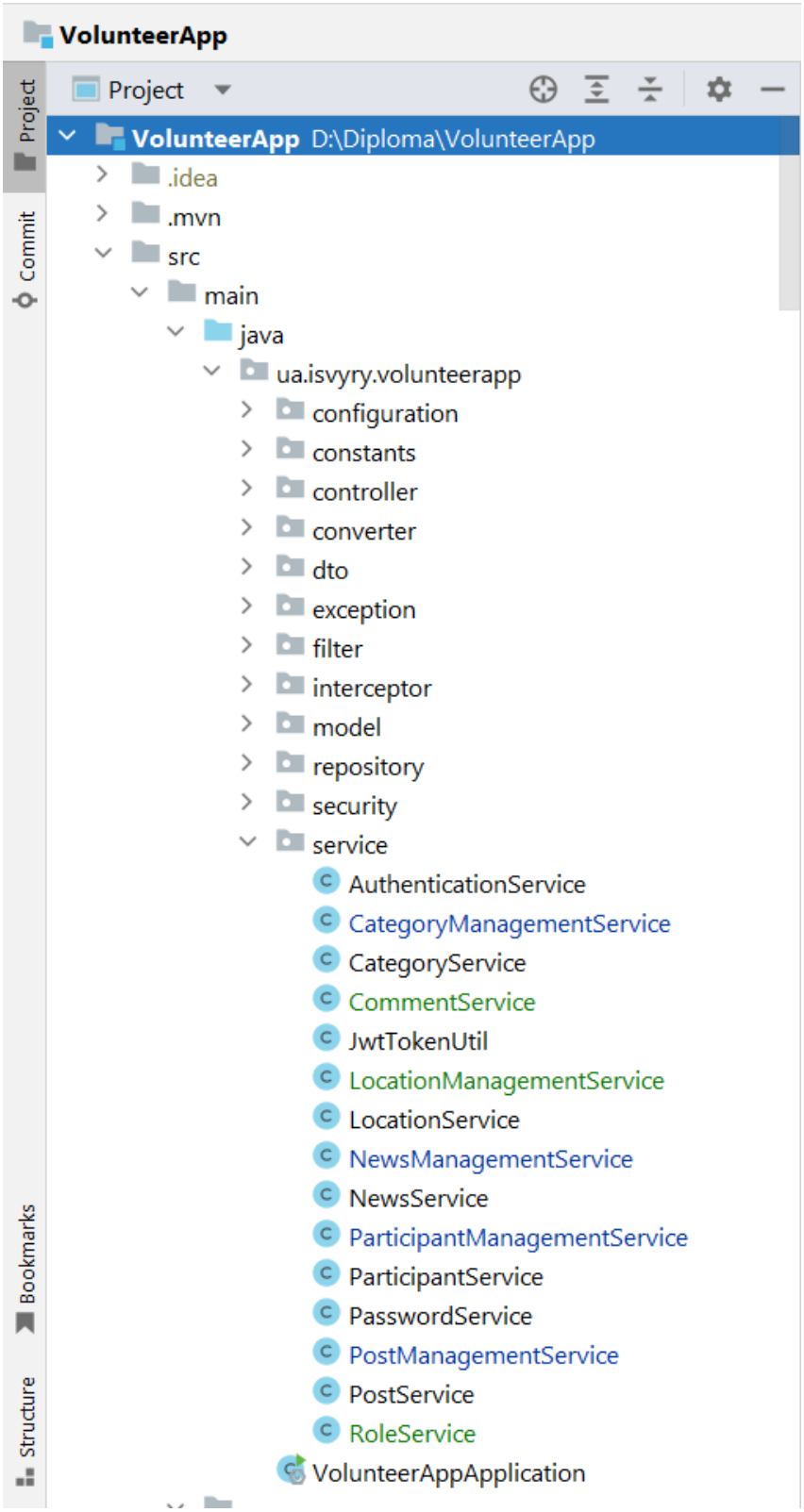

Рис. 3.12. Структура модулю Service

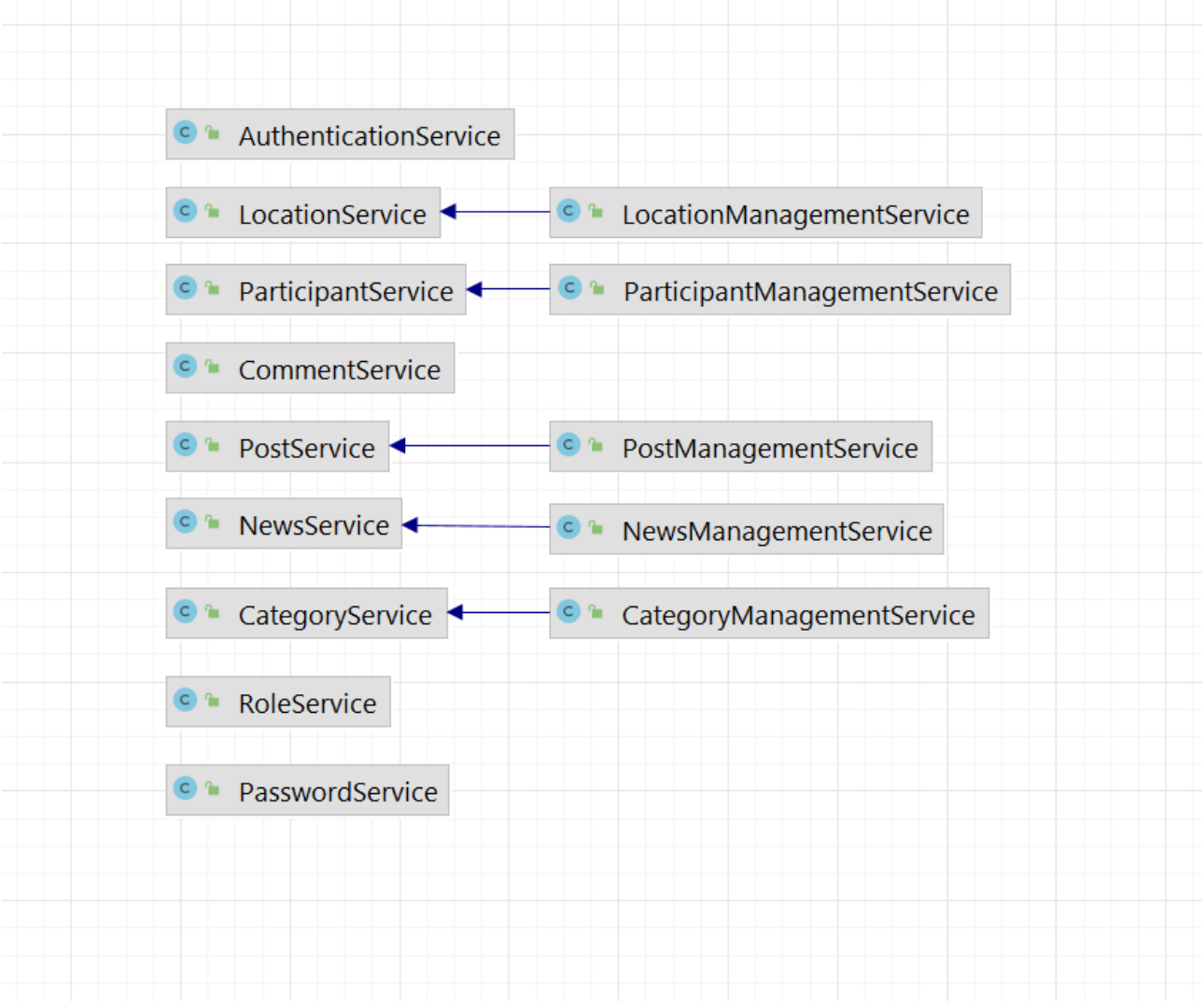

Рис. 3.13. Найважливіші сервіси.

Як можна бачити зі схеми, деякі сервіси мають додатковий шар абстракції, наприклад NewsService – NewsManagementService, PostService – PostManagementService тощо. Такий підхід був зроблений задля сепарації і виокремлення логіки, яка стосується інших типів.

Що мається на увазі: наприклад, для створення посту нам необхідно знати категорію посту та автора, тобто, як мінімум, нам потрібна функціональність від CategoryService та ParticipantService. Але інжектити їх в PostService буде не дуже добре, бо якщо нам десь знадобиться PostService, то разом з ним будуть доступні вищевкащані сервіси також, що може приводити до циклічних залежностей. Тому нам на допомогу приходить PostManagementService, котрий має всі необхідні сервіси для роботи з Post сутністю, таким чином ми маємо PostService,

що цілком чистий(не має зайвих сервісів) та має залежність тільки на відповідний репозиторій. Завдяки цьому написаний код стає набагато чистішим.

Таблиця 3.6

Функції сервісів

Сервіс	Призначення
ParticipantService	Використовується для керування користувачами. А саме: створення, підписування на іншого користувача, знаходження користувачів за певним фільтром та деактивація користувача.
ParticipantManagementService	
RoleService	Даний сервіс використовується для керування ролями. У ньому доступний функціонал додавання нової ролі, отримання всіх доступних ролей, зміна властивостей ролі.
PostService	Має контроль над функціоналом публікацій. Дозволяє створювати, редагувати, читати та видаляти публікації
PostManagementService	
CategoryService	Дає змогу створювати, редагувати, знаходити та видаляти категорії.
CategoryManahementService	
LocationService	Використовується для створення знаходження, та видалення локацій
LocationManagementService	
CommentService	Дають змогу створювати, редагувати, знаходити за певним фільтром та деактивувати відповідно університети, факультети та кафедри.

Закінч. табл. 3.6

NewsService	Використовується для створення, знаходження, редагування та видалення новин.
NewsManagementService	
AuthenticationService	Використовується для автентифікації та виходу з застосунку.
PasswordService	Використовується для хешування паролю

controller:

Даний програмний шар в свою чергу містить контролери та різні конфігураційні класи, що працюють із запитами на сервер та відповідями.

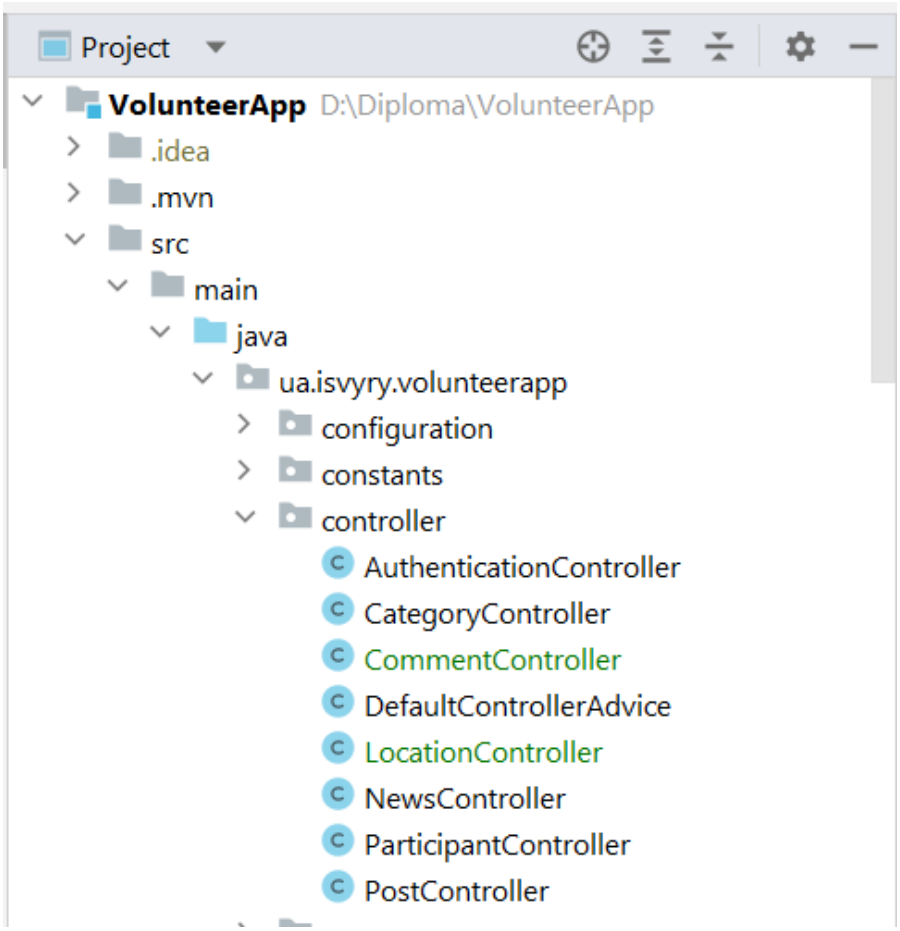

Рис. 3.14. Структура директорії controller

Таблиця 3.7

Найважливіші запити

Конт-роллер	Запит	Тип запиту	Призначення
Auth	/api/v1/auth	POST	Використовується для авторизації користувача
	/api/v1/logout	DELETE	Використовується для виходу користувача із застосунку
Post	/api/v1/posts	POST	Створення публікації
	/api/v1/posts/{id}	GET	Знаходження публікації
	/api/v1/posts/users/{id}	GET	Знаходження публікацій відносно користувача
	/api/v1/posts/{id}/comments	PUT	Редагування публікації(додавання коментаря)
	/api/v1/posts/{id}/subscribe	PUT	Редагування публікації(підписка на оновлення)
	/api/v1/posts/{id}	PUT	Редагування публікації
	/api/v1/posts/users/{id}/subscription	GET	Знаходження публікацій відносно підписок указаного користувача
Comment	/api/v1/comments/{id}	PUT	Редагування коментара
	/api/v1/comments/{id}	DELETE	Видалення коментара

Продовж. табл. 3.7

Participant	/api/v1/participants	POST	Створення користувача
	/api/v1/participants	GET	Знаходження всіх користувачів
	/api/v1/participants/{id}	GET	Знаходження користувача відносно ідентифікатора
	/api/v1/participants/{id}	PUT	Редагування користувача відносно ідентифікатора
	/api/v1/participants/{id}	DELETE	Деактивація користувача
	/api/v1/participants/subscribe/{id}	PUT	Додавання користувача до підписок
Role	/api/v1/roles	POST	Створення ролі
	/api/v1/roles	GET	Знаходження всіх існуючих ролей
	/api/v1/roles/{id}	PUT	Редагування ролі
	/api/v1/roles/{id}	DELETE	Деактивація ролі
Category	/api/v1/categories/	POST	Створення категорії
	/api/v1/categories/{id}	PUT	Редагування категорії відносно ідентифікатора
	/api/v1/categories	GET	Знаходження всіх категорій
	/api/v1/categories/{id}	GET	Знаходження категорії відносно ідентифікатора
	/api/v1/categories/{id}	DELETE	Видалення категорії відносно ідентифікатора

Закінч. табл. 3.7

News	/api/v1/news/	POST	Створення новин
	/api/v1/news/{id}	PUT	Редагування новини відносно ідентифікатора
	/api/v1/news	GET	Знаходження новин
	/api/v1/news/{id}	GET	Знаходження новини відносно ідентифікатора
	/api/v1/news/{id}	DELETE	Видалення новини відносно ідентифікатора
Location	/api/v1/locations/distance	GET	Знаходження відстані між двома локаціями
	/api/v1/locations/nearby /participants	GET	Знаходження користувачів поблизу
	/api/v1/locations/nearby/posts	GET	Знаходження проектів поблизу

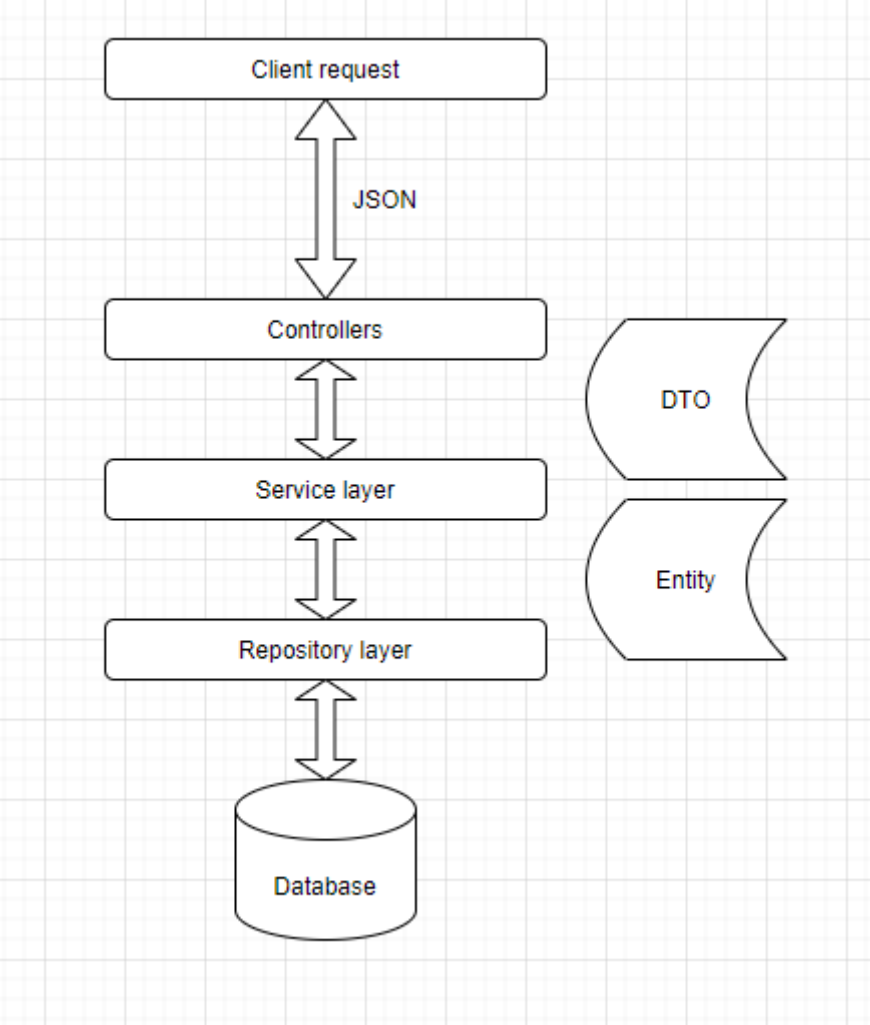

Рис. 3.15. Схема роботи системи

Отже, за схемою, розроблювана система має працювати наступним чином:

1. Клієнтський запит, що був сформований на фронтенді у форматі JSON надходить на шар контролерів. За допомогою класу DispatcherServlet, Spring направляє запит на відповідний метод відповідного класу-контролеру.

2. Після надходження на відповідний метод, керування приймає шар сервісу, куди з контролера перенаправляються тіло запиту та допоміжні змінні.

В контролерах також відбувається первісна валідація надходжених даних.

Для цього нам необхідно у відповідному методі контролера поряд із класом тіла запиту поставити анотацію @Valid.

```
Ihor Svyrydenko *
@ResponseStatus(HttpStatus.CREATED)
@PostMapping("/create")
public ParticipantCreateResponse createParticipant(@RequestBody @Valid ParticipantCreateRequest request) {
    return participantManagementService.createParticipant(request);
}
```

Рис. 3.16. Приклад використання анотації для валідації

А у відповідному класі прописати правила валідації:

```
package ua.isvyry.volunteerapp.dto.request;

import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;

import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
import javax.validation.constraints.Email;
import javax.validation.constraints.Max;
import javax.validation.constraints.NotEmpty;

7 usages  Ihor Svyrydenko *
@Data
@NoArgsConstructor
@AllArgsConstructor
public class ParticipantCreateRequest {
    @Pattern(regexp = "Activist|Organization|Volunteer")
    private String participantType;
    @Size(min = 2, max = 50)
    private String name;
    @Email
    private String email;
    @Pattern(regexp = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)(?=.*[#$@!%&*?])[A-Za-z\\d#$@!%&*?]{8,30}$")
    private String password;
    @Max(500)
    @NotEmpty
    private String description;
}
```

Рис. 3.17. Приклад правил валідації

3. В свою чергу сервіси взаємодіють із шаром-репозиторіїв, формуючи різноманітні SQL-запити відповідно до бізнес логіки.

4. Шар репозиторіїв містить класи-моделі, що є проекціями таблиць у базі даних, та власне репозиторії, за допомогою яких отримується доступ до БД.

5. Також варто не забувати про конфігурації та рівень безпеки. Кожний запит має містити відповідний токен доступу, що зберігає інформацію про користувача та надає тому доступ до ендпоінтів в залежності від його ролі.

База даних:

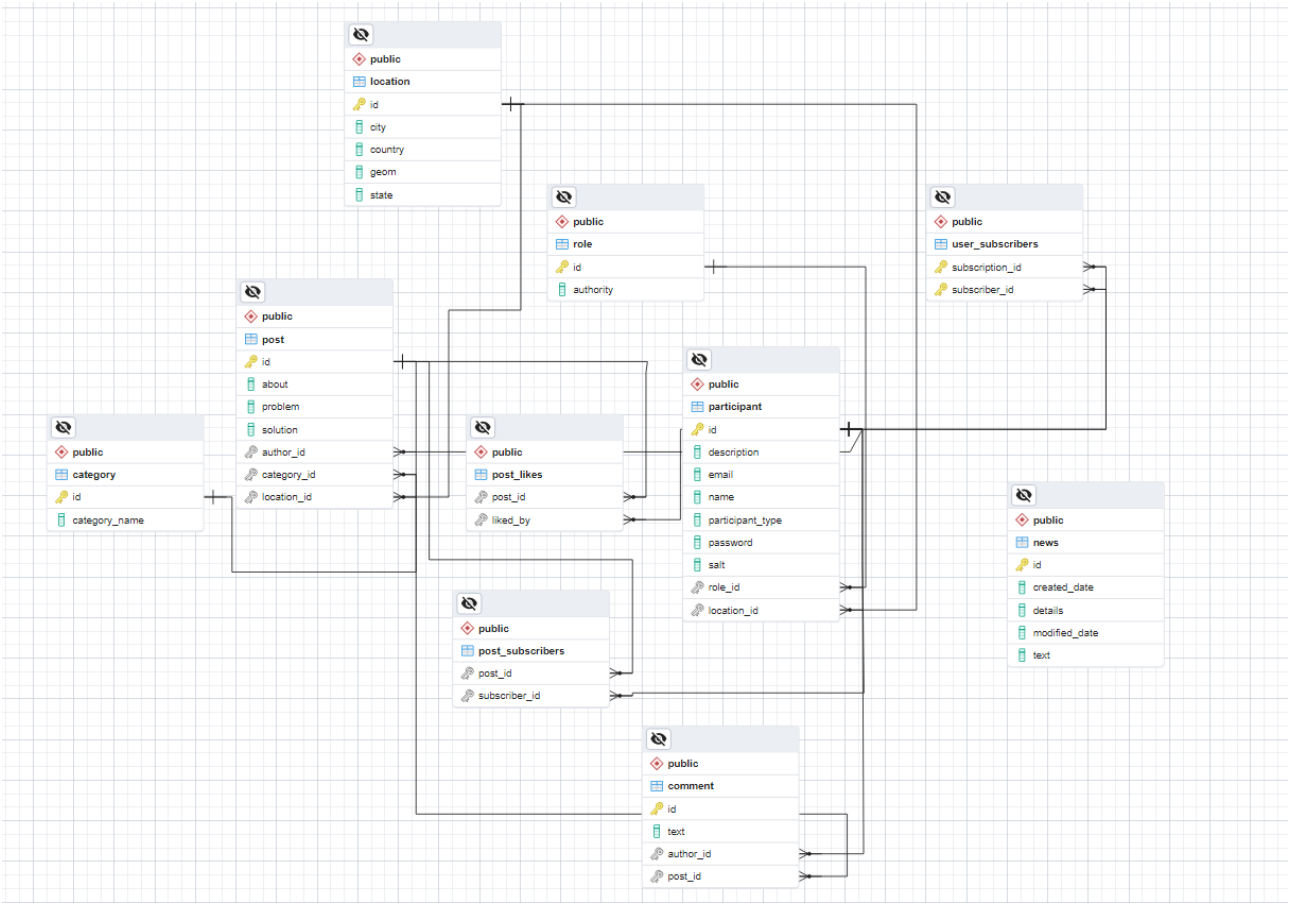

Рис. 3.18. Схема бази даних

Основні таблиці:

Таблиця 3.8

Структура об’єкту Participant

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор користувача
description	varchar	Опис користувача
participant_type	datetime	Тип користувача
name	varchar	Ім’я користувача
email	varchar	Пошта користувача, має бути унікальною
password	varchar	Пароль користувача в зашифрованому вигляді
salt	varchar	Допоміжна строка для шифрування паролю
role_id	bigint	Ідентифікатор ролі, якою володіє користувач. Зовнішній ключ.
location_id	bigint	Ідентифікатор локації. Зовнішній ключ.

Таблиця 3.9

Структура об’єкту Role

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор ролі
authority	varchar	Назва ролі

Таблиця 3.10

Структура об’єкту News

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
details	varchar	Короткі деталі про новину
text	varchar	Текст новини

Таблиця 3.11

Структура об’єкту Comment

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор коментаря
text	varchar	Текст коментаря
author_id	datetime	Ідентифікатор автора. Зовнішній ключ.
post_id	datetime	Ідентифікатор посту. Зовнішній ключ.

Таблиця 3.12

Структура об’єкту Post

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор посту
text	varchar	Текст посту
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису

Закінч. табл. 3.12

enabled	boolean	Показує стан посту
author_id	bigint	Ідентифікатор користувача, що є автором посту. Зовнішній ключ

Таблиця 3.13

Структура об’єкту Category

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор категорії
category_name	varchar	Назва категорії

Таблиця 3.14

Структура об’єкту Location

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор локації
city	varchar	Назва країни
country	varchar	Назва міста
state	varchar	Назва області/штату тощо
geom	geometry	Спеціальній тип PostGIS, який зберігає в собі широту та довготу.

3.4. Обґрунтування та організація вхідних та вихідних даних програми

Вхідні дані надаються системі у форматі JSON, що формуються на фронт-енді в залежності від дій користувача. В залежності від необхідної інформації,

фронт-енд надсилає відповідні запити до розроблюваної системи, котра в свою чергу, оброблює ці запити та відповідає вихідними даними у форматі JSON.

3.5. Опис розробленого програмного продукту

3.5.1. Використані технічні засоби

В процесі тестування та розробки ПЗ були використані наступні технічні засоби:

- оперативна пам’ять обсягом 16 гб;
- процесор Intel Core i7-7700 3.60 GHz;
- накопичувач на жорсткому диску обсягом 250 гб;
- відеоадаптер Radeon RX 6800;
- клавіатура та маніпулятор типу «миш».

Вказані технічні засоби не мають бути обов’язково ідентичних характеристик для безперешкодної роботи системи.

3.5.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

- IntelliJ IDEA.
- Git, GitHub.

IntelliJ IDEA

«IntelliJ IDEA – це особливе середовище програмування або інтегроване середовище розробки (IDE), багато в чому призначене для Java». Це середовище використовується спеціально для розробки програм. Воно розроблене компанією під назвою JetBrains. Воно доступне у двох виданнях: Community Edition, що має ліцензію Apache 2.0, і комерційне видання, відоме як Ultimate Edition. Що робить IntelliJ IDEA настільки відмінним від своїх аналогів, це його легкість у використанні, гнучкість та зручний дизайн.

GIT

Git – це приклад системи управління розподіленими версіями (DVCS), яка зазвичай використовується для розробки відкритого комерційного та некомерційного програмного забезпечення. DVCS дозволяють отримати повний доступ до кожного файлу, гілки та ітерації проекту та дозволяє кожному користувачеві отримати доступ до повної та самодостатньої історії всіх змін. На відміну від популярних колись централізованих систем управління версіями, «DVCS типу Git не потребує постійного підключення до центрального сховища.

GitLab – це платформа для розміщення коду для контролю версій та співпраці. Він дозволяє спільно працювати над проектами з будь-якого місця та зберігати версії проекту на віддаленому сервері.

3.5.3. Опис інтерфейсу користувача

В системі використовується візуалізація розробленого API – OpenAPI Swagger(див. рис. 2.24.). За допомогою цієї сторінки можна дивитись існуючі ендпоїнти та надсилати тестові запити.

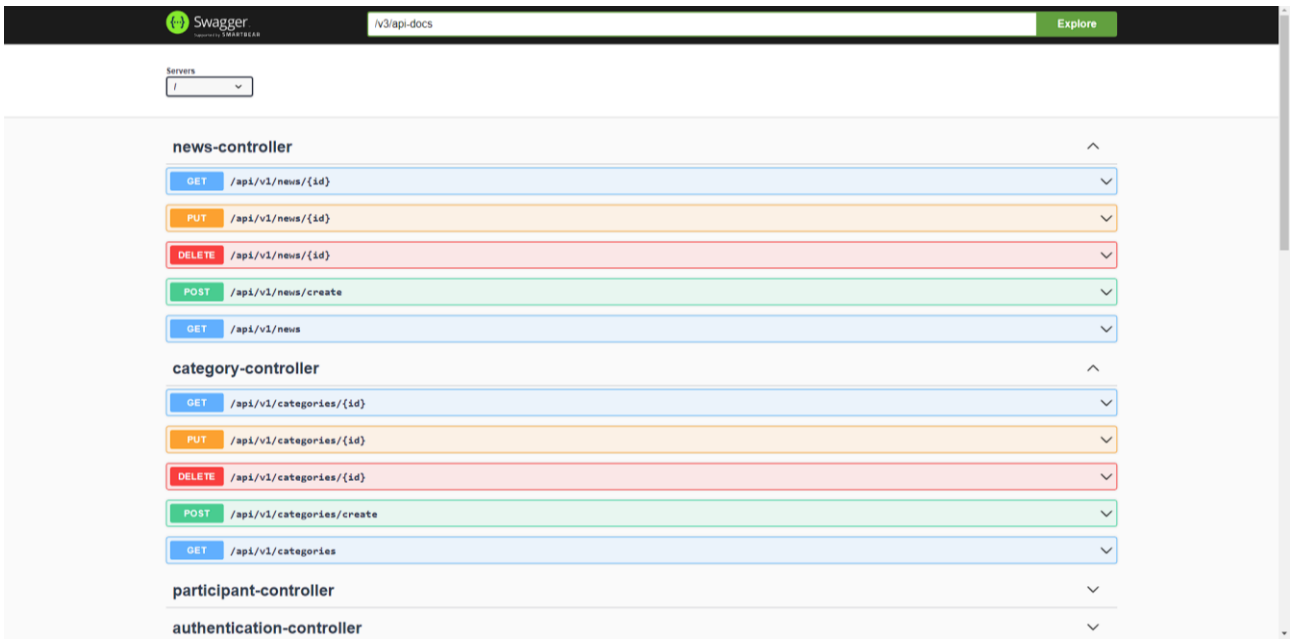

Рис. 3.19. OpenAPI

ВИСНОВКИ

В даній кваліфікаційній роботі була розроблена серверна частина волонтерської платформи. Розроблений додаток призначений полегшити взаємодію між волонтерами. В результаті розробки проекту були реалізовані наступні можливості: реєстрація користувачів, створення / зміна / видалення категорій, ролей, локацій, реалізована система створення / зміна публікацій, та відповідей на них, можливість підписуватись на обраного користувача, також присутня можливість знаходити користувачів/проекти, які знаходять поблизу.

Під час виконання даної кваліфікаційної роботи проекту були виконані наступні задачі:

− був проведений аналіз предметної галузі задачі;

− проведено порівняння можливостей існуючих аналогів розроблюваного продукту;

− спроектовано та створено базу даних;

− написано програмний код застосунку;

− створений API-інтерфейс задокументовано у зручному, для сприйняття, вигляді.

Додаток створений за допомогою мови програмування Java на базі фреймворку Spring. Використовувана база даних – PostgreSQL.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Universal Declaration on Volunteering. International Association for Volunteer Effort. January 2001. URL: https://www.iave.org/advocacy/the-universal-declaration-on-volunteering/

2. Мартін Калін. Java Web Services: Up and Running, Харків : Наука і техніка, 2009. 320 с.

3. Сайт UA Helpers. URL: https://uahelpers.com/

4. Сайт volonter.org. URL: https://volonter.org/pro-nas

5. Сайт zaporuka.org.ua. URL: https://zaporuka.org.ua/pro-nas/

6. Bilal Himite. Distance Between Two Points on Earth URL: https://medium.com/swlh/calculating-the-distance-between-two-points-on-earth-bac5cd50c840

7. Крейг Уолс. Спрінг у Дії, Шосте Видання, Київ : Видавництво Діалектика, 2021. 520 с.

8. Design First or Code First: What's the Best Approach to API Development. URL: https://swagger.io/blog/api-design/design-first-or-code-first-api-development/

9. Козміна Ю., Шефер К. Pro Spring 5: Поглиблений посібник із Spring Framework та його інструментів. Київ : Видавництво Діалектика, 2016. – 1120 с.

10. Фуско М. Сучасна мова Java. Лямбда-вирази, потоки і функціональне програмування. Київ : ДМК Прес, 2019. – 592 с

11. JSON Web Token Introduction. URL: https://jwt.io/introduction

12. Рой Томас Філдінг. Architectural Styles and the Design of Network-based Software Architectures URL:

13. Кішорі Шаран. Особливості мови програмування Java, . Київ : Видавництво Діалектика, 2018 р. 236 с.

14. Документація LiquiBase. URL: https://docs.liquibase.com/home.html

15. Документація Maven. URL: https://maven.apache.org/guides/

16. Закон України про волонтерську діяльність станом на 01.06.2016 № 40 URL: https://zakon.rada.gov.ua/laws/show/3236-17#Text

17. Глушаков С. Програмування на Java 2. Харків: Фоліо, 2003 – 536 с.

18. Тохтарова І. М. Волонтерський рух в Україні: шлях до розвитку громадянського суспільства як сфери соціальних відносин. Теорія та практика державного управління і місцевого самоврядування, 2014, № 2

19. Сайт Eclipce. AspectJ 5 Developer's Notebook. URL: https://www.eclipse.org/aspectj/docs.php

20. Сайт refactoring.guru. URL: https://refactoring.guru/design-patterns

21. Сайт swagger.io. URL: https://swagger.io/specification/

22. Документація Maven Plugins. URL: https://maven.apache.org/plugins/

23. Martin Fowler. Inversion of Control Containers and the Dependency Injection pattern. URL: https://martinfowler.com/articles/injection.html#InversionOfControl

24. Михайло Матяш. Українське волонтерство. URL: https://www.ukrinform.ua/rubric-society/2324579-ukrainske-volonterstvo-avise-unikalne-jomu-zavdacuemo-suverenitetom.html

25. Зміст та організація діяльності центрів екстреної психологічної допомоги "Телефон довіри". Науково-методичний посібник. – М: Державний НДІ сім'ї та виховання, 1999. – 208с .

ДОДАТОК А

КОД ПРОГРАМИ

Category.java

```
package ua.isvyry.volunteerapp.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.GenerationType;
import javax.persistence.GeneratedValue;
import javax.persistence.OneToMany;
import java.util.List;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String categoryName;
    @OneToMany(mappedBy = "category")
    private List<Post> posts;
}
```

Comment.java

```
package ua.isvyry.volunteerapp.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.GenerationType;
import javax.persistence.GeneratedValue;
import javax.persistence.ManyToOne;

@Entity
@Builder
```

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Comment {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    private Participant author;

    private String text;

    @ManyToOne
    private Post post;
}
```

Location.java

```
package ua.isvyry.volunteerapp.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.locationtech.jts.geom.Point;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.GenerationType;
import javax.persistence.GeneratedValue;
import javax.persistence.Column;
import javax.persistence.OneToMany;
import java.util.List;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Location {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(columnDefinition = "GEOMETRY")
    private Point geom;
    private String country;
    private String city;
    private String state;
    @OneToMany(mappedBy = "location")
    List<Participant> users;
```

```
    @OneToMany(mappedBy = "location")
    List<Post> posts;
}
```

News.java

```
package ua.isvyry.volunteerapp.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.Date;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class News {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String text;
    private Date createdDate;
    private Date modifiedDate;
    private String details;

}
```

Participant.java

```
package ua.isvyry.volunteerapp.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.List;
import java.util.Set;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Builder
@Table(name = "participant")
public class Participant {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Enumerated(EnumType.STRING)
    private ParticipantType participantType;

    private String name;
    private String email;
    private String description;
    private String password;
    private String salt;
    @ManyToOne
    private Location location;
    @ManyToOne
    private Role role;
    @OneToMany(mappedBy = "author")
    private List<Post> posts;

    @ManyToMany(mappedBy = "subscribers")
    private List<Post> subscribedPosts;

    @ManyToMany(mappedBy = "likedByUsers")
    private List<Post> likedPosts;

    @OneToMany(mappedBy = "author")
    private List<Comment> comments;

    @ManyToMany
    @JoinTable(name = "user_subscribers",
            joinColumns = @JoinColumn(name = "subscription_id"),
            inverseJoinColumns = @JoinColumn(name = "subscriber_id"))
    private Set<Participant> subscriptions;

    @ManyToMany
    @JoinTable(name = "user_subscribers",
            joinColumns = @JoinColumn(name = "subscriber_id"),
            inverseJoinColumns = @JoinColumn(name = "subscription_id"))
    private Set<Participant> subscribers;

}
```

ParticipantType.java

```
package ua.isvyry.volunteerapp.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

import java.util.Arrays;
import java.util.NoSuchElementException;
```

```
@Getter
@AllArgsConstructor
public enum ParticipantType {
   VOLUNTEER("Volunteer"),
   ACTIVIST("Activist"),
   ORGANIZATION("Organization");

   private final String name;

   public static ParticipantType determineType(String type) {
      return Arrays.stream(ParticipantType.values())
            .filter(s -> s.name.equals(type))
            .findFirst()
            .orElseThrow(() -> new NoSuchElementException(String
                  .format("Provided wrong participant type: %s", type)));
   }
}
```

Post.java

```
package ua.isvyry.volunteerapp.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.List;
import java.util.Set;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Post {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private String problem;
   private String solution;
   private String about;
   @ManyToOne
   private Location location;
   @ManyToOne
   private Category category;
   @ManyToOne
   private Participant author;
   @ManyToMany
   @JoinTable(name = "post_subscribers",
         joinColumns = @JoinColumn(name = "post_id"),
```

```
        inverseJoinColumns = @JoinColumn(name = "subscriber_id"))
    private List<Participant> subscribers;
    @ManyToMany
    @JoinTable(name = "post_likes",
        joinColumns = @JoinColumn(name = "post_id"),
        inverseJoinColumns = @JoinColumn(name = "liked_by"))
    private List<Participant> likedByUsers;
    @OneToMany(mappedBy = "post")
    private List<Comment> comments;
}
```

Role.java

```
package ua.isvyry.volunteerapp.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.security.core.GrantedAuthority;

import javax.persistence.*;
import java.util.List;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Role implements GrantedAuthority {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String authority;

    @Override
    public String getAuthority() {
        return authority;
    }

    @OneToMany(mappedBy = "role")
    List<Participant> participants;
}
```

CategoryRepository.java

```
package ua.isvyry.volunteerapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.model.Category;
```

```
@Component
public interface CategoryRepository extends JpaRepository<Category, Long> {
   boolean existsByCategoryName(String categoryName);
}
```

CommentRepository.java

```
package ua.isvyry.volunteerapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.model.Comment;

@Component
public interface CommentRepository extends JpaRepository<Comment, Long> {
}
```

LocationRepository.java

```
package ua.isvyry.volunteerapp.repository;

import org.locationtech.jts.geom.Point;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.model.Location;

import java.util.List;

@Component
public interface LocationRepository extends JpaRepository<Location, Long> {
   @Query(value="SELECT * from location where ST_DistanceSphere(geom, :p) <
:distanceM", nativeQuery = true)
   List<Location> findNearWithinDistance(Point p, double distanceM);
}
```

NewsRepository.java

```
package ua.isvyry.volunteerapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.model.News;

@Component
public interface NewsRepository extends JpaRepository<News, Long> {
}
```

ParticipantRepository.java

```
package ua.isvyry.volunteerapp.repository;
```

```
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.model.Location;
import ua.isvyry.volunteerapp.model.Participant;
import ua.isvyry.volunteerapp.model.Post;

import java.util.List;
import java.util.Optional;

@Component
public interface ParticipantRepository extends JpaRepository<Participant, Long> {
   Optional<Participant> findByEmail(String email);

   Page<Participant> findAllByLocationIn(Pageable pageable, List<Location> locations);

}
```

PostRepository.java

```
package ua.isvyry.volunteerapp.repository;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.model.Location;
import ua.isvyry.volunteerapp.model.Post;

import java.util.List;

@Component
public interface PostRepository extends JpaRepository<Post, Long> {
   boolean existsByCategory_Id(Long categoryId);

   Page<Post> findAllByLocationIn(Pageable pageable, List<Location> locations);
}
```

RoleRepository.java

```
package ua.isvyry.volunteerapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.model.Role;

@Component
public interface RoleRepository extends JpaRepository<Role, Long> {
}
```

AuthenticationService.java

```
package ua.isvyry.volunteerapp.service;

import lombok.RequiredArgsConstructor;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.dto.request.AuthenticationRequest;
import ua.isvyry.volunteerapp.dto.response.AuthenticationResponse;
import ua.isvyry.volunteerapp.model.Participant;
import ua.isvyry.volunteerapp.security.VolunteerAppUserDetailsService;

import static ua.isvyry.volunteerapp.service.PasswordService.hashPassword;

@Service
@RequiredArgsConstructor
public class AuthenticationService {

    private final ParticipantService participantService;
    private final VolunteerAppUserDetailsService userDetailsService;
    private final JwtTokenUtil jwtTokenUtil;

    @Transactional
    public AuthenticationResponse authenticate(AuthenticationRequest request) {
        Participant participant = participantService.findByEmail(request.getEmail());
        String password = participant.getPassword();
        if (!hashPassword(password, participant.getSalt())
                .equals(hashPassword(request.getPassword(), participant.getSalt()))) {
            throw new BadCredentialsException("");
        }
        String token =
jwtTokenUtil.generateToken(userDetailsService.loadUserByUsername(request.getEmail()));
        return AuthenticationResponse.builder()
                .token(token)
                .build();
    }

}
```

CategoryManagementService.java

```
package ua.isvyry.volunteerapp.service;

import lombok.RequiredArgsConstructor;
import org.springframework.core.convert.ConversionService;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.dto.request.CategoryCreateRequest;
import ua.isvyry.volunteerapp.dto.request.CategoryUpdateRequest;
import ua.isvyry.volunteerapp.dto.response.CategoryCreateResponse;
```

```
import ua.isvyry.volunteerapp.dto.response.CategoryResponse;
import ua.isvyry.volunteerapp.dto.response.CategoryUpdateResponse;
import ua.isvyry.volunteerapp.exception.CategoryDeleteException;
import ua.isvyry.volunteerapp.model.Category;

import javax.persistence.EntityExistsException;

import java.util.List;
import java.util.stream.Collectors;

import static ua.isvyry.volunteerapp.constants.Constants.ENTITY_ALREADY_EXISTS_MESSAGE_WITH_NAME;

@Service
@RequiredArgsConstructor
public class CategoryManagementService {
   private final CategoryService categoryService;
   private final ConversionService conversionService;
   private final PostService postService;

   @Transactional
   public CategoryCreateResponse createCategory(CategoryCreateRequest request) {
      Category category = conversionService.convert(request, Category.class);
      return conversionService.convert(categoryService.create(category), CategoryCreateResponse.class);
   }

   @Transactional(readOnly = true)
   public List<CategoryResponse> getAllCategories(Integer pageNumber, Integer pageSize) {
      Page<Category> categoriesRange = categoryService.findAll(PageRequest.of(pageNumber, pageSize));
      return categoriesRange.stream()
            .map(elem -> conversionService.convert(elem, CategoryResponse.class))
            .collect(Collectors.toList());
   }

   @Transactional(readOnly = true)
   public CategoryResponse getById(Long id) {
      return conversionService.convert(categoryService.findById(id), CategoryResponse.class);
   }

   public CategoryUpdateResponse updateCategory(Long id, CategoryUpdateRequest request) {
      Category categoryToUpdate = categoryService.findById(id);
      if (categoryService.existsByName(request.getCategoryName())) {
         throw new EntityExistsException(String.format(ENTITY_ALREADY_EXISTS_MESSAGE_WITH_NAME,
               Category.class.getSimpleName(), request.getCategoryName(), id));
```

```
        }
        categoryToUpdate.setCategoryName(request.getCategoryName());
        return conversionService.convert(categoryService.update(categoryToUpdate),
CategoryUpdateResponse.class);
    }

    @Transactional(readOnly = true)
    public void deleteCategoryById(Long id) {
        if (postService.existsByCategoryId(id)) {
            throw new CategoryDeleteException(String.format("Could not delete Category with
id: %s, " +
                    "because it has dependent posts", id));
        }
        categoryService.delete(id);
    }
}
```

CategoryService.java

```
package ua.isvyry.volunteerapp.service;

import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.model.Category;
import ua.isvyry.volunteerapp.repository.CategoryRepository;

import javax.persistence.EntityNotFoundException;

import static
ua.isvyry.volunteerapp.constants.Constants.ENTITY_NOT_FOUND_MESSAGE_ID;

@Service
@RequiredArgsConstructor
public class CategoryService {

    private final CategoryRepository categoryRepository;

    @Transactional
    public Category create(Category category) {
        return categoryRepository.save(category);
    }

    @Transactional
    public Category update(Category category) {
        return categoryRepository.save(category);
    }

    @Transactional(readOnly = true)
```

```
    public Page<Category> findAll(Pageable pageable) {
        return categoryRepository.findAll(pageable);
    }
    @Transactional(readOnly = true)
    public Category findById(Long id) {
        return categoryRepository.findById(id)
                .orElseThrow(() -> new
EntityNotFoundException(String.format(ENTITY_NOT_FOUND_MESSAGE_ID,
                        Category.class.getSimpleName(), id)));
    }

    @Transactional(readOnly = true)
    public boolean existsByName(String name) {
        return categoryRepository.existsByCategoryName(name);
    }

    @Transactional
    public void delete(Long id) {
        categoryRepository.deleteById(id);
    }
}
```

JwtTokenUtil.java

```
package ua.isvyry.volunteerapp.service;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;

import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;
import java.security.Key;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Service
public class JwtTokenUtil {

    public static final long JWT_TOKEN_VALIDITY = 40 * 60;   // 40min

    private final Key key;

    public JwtTokenUtil(@Value("${jwt.secret}") String secret) {
        byte[] apiKeySecretBytes = DatatypeConverter.parseBase64Binary(secret);
        this.key = new SecretKeySpec(apiKeySecretBytes,
SignatureAlgorithm.HS512.getJcaName());
```

```
    }

    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return generateTokenWithClaims(claims, userDetails.getUsername());
    }

    public String getUsernameFromToken(String token) {
        return getClaimFromToken(token, Claims::getSubject);
    }

    public <T> T getClaimFromToken(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = getAllClaimsFromToken(token);
        return claimsResolver.apply(claims);
    }

    public Date getExpirationDateFromToken(String token) {
        return getClaimFromToken(token, Claims::getExpiration);
    }

    private Claims getAllClaimsFromToken(String token) {
        return
Jwts.parserBuilder().setSigningKey(key).build().parseClaimsJws(token).getBody();
    }

    private Boolean isTokenExpired(String token) {
        final Date expiration = getExpirationDateFromToken(token);
        return expiration.before(new Date());
    }

    private String generateTokenWithClaims(Map<String, Object> claims, String subject) {
        return Jwts.builder()
                .setClaims(claims)
                .setSubject(subject)
                .setIssuedAt(new Date(System.currentTimeMillis()))
                .setExpiration(new                Date(System.currentTimeMillis()                +
JWT_TOKEN_VALIDITY * 1000))
                .signWith(key).compact();
    }

    public Boolean validateToken(String token) {
        return (Jwts.parserBuilder()
                .setSigningKey(key)
                .build().isSigned(token) && !isTokenExpired(token));
    }

}
```

LocationService.java

```
package ua.isvyry.volunteerapp.service;

import lombok.RequiredArgsConstructor;
```

```
import lombok.extern.slf4j.Slf4j;
import org.locationtech.jts.geom.Coordinate;
import org.locationtech.jts.geom.GeometryFactory;
import org.locationtech.jts.geom.Point;
import org.locationtech.jts.geom.PrecisionModel;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.model.Location;
import ua.isvyry.volunteerapp.repository.LocationRepository;

import java.util.List;

@Slf4j
@Service
@RequiredArgsConstructor
public class LocationService {

   private final LocationRepository locationRepository;

   private final GeometryFactory factory = new GeometryFactory(new PrecisionModel(),
4326);

   @Transactional(readOnly = true)
   public Page<Location> findAll(Pageable pageable) {
      return locationRepository.findAll(pageable);
   }

   public List<Location> findAround(double lat, double lon, double distanceM){
      log.info("Looking for a locations around ({},{}) withing {} meters", lat, lon, distanceM);
      Point p = factory.createPoint(new Coordinate(lon, lat));
      return locationRepository.findNearWithinDistance(p, distanceM);
   }
}
```

NewsManagementService.java

```
package ua.isvyry.volunteerapp.service;

import lombok.RequiredArgsConstructor;
import org.springframework.core.convert.ConversionService;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.dto.request.CategoryCreateRequest;
import ua.isvyry.volunteerapp.dto.request.CategoryUpdateRequest;
import ua.isvyry.volunteerapp.dto.request.NewsCreateRequest;
import ua.isvyry.volunteerapp.dto.request.NewsUpdateRequest;
import ua.isvyry.volunteerapp.dto.response.*;
import ua.isvyry.volunteerapp.model.News;
```

```
import java.util.Date;
import java.util.List;
import java.util.stream.Collectors;


@Service
@RequiredArgsConstructor
public class NewsManagementService {

    private final NewsService newsService;
    private final ConversionService conversionService;

    @Transactional
    public NewsCreateResponse createNews(NewsCreateRequest request) {
        News news = conversionService.convert(request, News.class);
        return conversionService.convert(newsService.create(news),
NewsCreateResponse.class);
    }

    @Transactional(readOnly = true)
    public List<NewsResponse> getAllNews(Integer pageNumber, Integer pageSize) {
        Page<News> newsRange = newsService.findAll(PageRequest.of(pageNumber,
pageSize));
        return newsRange.stream()
                .map(elem -> conversionService.convert(elem, NewsResponse.class))
                .collect(Collectors.toList());
    }

    @Transactional(readOnly = true)
    public NewsResponse getById(Long id) {
        return conversionService.convert(newsService.findById(id), NewsResponse.class);
    }

    @Transactional
    public NewsResponse updateNews(Long id, NewsUpdateRequest request) {
        News newsToUpdate = newsService.findById(id);
        newsToUpdate.setText(request.getText());
        newsToUpdate.setDetails(request.getDetails());
        newsToUpdate.setModifiedDate(new Date());
        return conversionService.convert(newsService.update(newsToUpdate),
NewsResponse.class);
    }

    @Transactional(readOnly = true)
    public void deleteNewsById(Long id) {
        newsService.delete(newsService.findById(id));
    }
}
```

NewsService.java

```
package ua.isvyry.volunteerapp.service;

import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.model.News;
import ua.isvyry.volunteerapp.repository.NewsRepository;

import javax.persistence.EntityNotFoundException;

import static ua.isvyry.volunteerapp.constants.Constants.ENTITY_NOT_FOUND_MESSAGE_ID;

@Service
@RequiredArgsConstructor
public class NewsService {
    private final NewsRepository newsRepository;

    @Transactional
    public News create(News news) {
        return newsRepository.save(news);
    }

    @Transactional
    public News update(News news) {
        return newsRepository.save(news);
    }

    @Transactional(readOnly = true)
    public Page<News> findAll(Pageable pageable) {
        return newsRepository.findAll(pageable);
    }

    @Transactional(readOnly = true)
    public News findById(Long id) {
        return newsRepository.findById(id)
                .orElseThrow(() -> new EntityNotFoundException(String.format(ENTITY_NOT_FOUND_MESSAGE_ID,
                        News.class.getSimpleName(), id)));
    }

    @Transactional
    public void delete(News news) {
        newsRepository.delete(news);
    }
}
```

ParticipantManagementService.java

```
package ua.isvyry.volunteerapp.service;
```

```
import lombok.RequiredArgsConstructor;
import org.locationtech.jts.geom.Point;
import org.springframework.core.convert.ConversionService;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.dto.request.ParticipantCreateRequest;
import ua.isvyry.volunteerapp.dto.response.CategoryResponse;
import ua.isvyry.volunteerapp.dto.response.ParticipantCreateResponse;
import ua.isvyry.volunteerapp.dto.response.ParticipantResponse;
import ua.isvyry.volunteerapp.model.Category;
import ua.isvyry.volunteerapp.model.Location;
import ua.isvyry.volunteerapp.model.Participant;
import ua.isvyry.volunteerapp.model.Post;

import java.util.List;
import java.util.stream.Collectors;

@Service
@RequiredArgsConstructor
public class ParticipantManagementService {
   private final ConversionService conversionService;
   private final ParticipantService participantService;
   private final LocationService locationService;

   @Transactional
   public ParticipantCreateResponse createParticipant(ParticipantCreateRequest request) {
      Participant participant = conversionService.convert(request, Participant.class);
      String salt = PasswordService.generateSalt();
      participant.setSalt(salt);
      participant.setPassword(PasswordService.hashPassword(request.getPassword(), salt));
      return conversionService.convert(participantService.create(participant),
ParticipantCreateResponse.class);
   }

   @Transactional(readOnly = true)
   public List<ParticipantResponse> getAllParticipants(Integer pageNumber, Integer
pageSize) {
      Page<Participant> participantsRange =
participantService.findAll(PageRequest.of(pageNumber, pageSize));
      return participantsRange.stream()
            .map(elem -> conversionService.convert(elem, ParticipantResponse.class))
            .collect(Collectors.toList());
   }

   @Transactional(readOnly = true)
   public List<ParticipantResponse> findAllInRadiusRange(Pageable pageable, Long id,
double radius) {
      Participant currentUser = participantService.findById(id);
```

```
        Point currentUserPoint = currentUser.getLocation().getGeom();
        List<Location> locationsInRange = locationService
                .findAround(currentUserPoint.getY(), currentUserPoint.getX(), radius);
        Page<Participant> participantsRange =
participantService.findAllByLocationIn(pageable, locationsInRange);
        return participantsRange.stream()
                .map(elem -> conversionService.convert(elem, ParticipantResponse.class))
                .collect(Collectors.toList());
    }

    @Transactional(readOnly = true)
    public ParticipantResponse getById(Long id) {
        return conversionService.convert(participantService.findById(id),
ParticipantResponse.class);
    }

    public ParticipantResponse subscribe(Participant currentUser, Long subscribeToUserId) {
        Participant subscribeToUser = participantService.findById(subscribeToUserId);
        currentUser.getSubscriptions().add(subscribeToUser);
        return conversionService.convert(participantService.create(currentUser),
ParticipantResponse.class);
    }
}
```

ParticipantService.java

```
package ua.isvyry.volunteerapp.service;

import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.model.Location;
import ua.isvyry.volunteerapp.model.Participant;
import ua.isvyry.volunteerapp.model.Post;
import ua.isvyry.volunteerapp.repository.ParticipantRepository;

import javax.persistence.EntityNotFoundException;

import java.util.List;

import static
ua.isvyry.volunteerapp.constants.Constants.ENTITY_NOT_FOUND_MESSAGE_EMAIL;
import static
ua.isvyry.volunteerapp.constants.Constants.ENTITY_NOT_FOUND_MESSAGE_ID;

@Service
@RequiredArgsConstructor
public class ParticipantService {
    private final ParticipantRepository participantRepository;
```

```
    @Transactional(readOnly = true)
    public Participant findById(Long id) {
        return participantRepository.findById(id)
                .orElseThrow(() -> new
EntityNotFoundException(String.format(ENTITY_NOT_FOUND_MESSAGE_ID,
                        Participant.class.getSimpleName(), id)));
    }

    @Transactional(readOnly = true)
    public Participant findByEmail(String email) {
        return participantRepository.findByEmail(email)
                .orElseThrow(() -> new
UsernameNotFoundException(String.format(ENTITY_NOT_FOUND_MESSAGE_EMAIL
,
                        Participant.class.getSimpleName(), email)));
    }

    public Page<Participant> findAllByLocationIn(Pageable pageable, List<Location>
locations) {
        return participantRepository.findAllByLocationIn(pageable, locations);
    }

    @Transactional(readOnly = true)
    public Page<Participant> findAll(Pageable pageable) {
        return participantRepository.findAll(pageable);
    }

    @Transactional
    public Participant create(Participant participant) {
        return participantRepository.save(participant);
    }

    @Transactional
    public Participant update(Participant participant) {
        return participantRepository.save(participant);
    }
}
```

PasswordService.java

```
package ua.isvyry.volunteerapp.service;

import com.google.common.hash.Hashing;
import org.apache.commons.lang3.RandomStringUtils;
import org.springframework.stereotype.Service;

import java.nio.charset.StandardCharsets;

@Service
public class PasswordService {
```

```
    public static String generateSalt() {
        return RandomStringUtils.random(15,
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ-1234567890");
    }

    public static String hashPassword(String password, String salt) {
        return Hashing.sha256().hashString(password.concat(salt),
StandardCharsets.UTF_8).toString();
    }

}
```

PostManagementService.java

```
package ua.isvyry.volunteerapp.service;

import lombok.RequiredArgsConstructor;
import org.locationtech.jts.geom.Point;
import org.springframework.core.convert.ConversionService;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.dto.request.PostCreateRequest;
import ua.isvyry.volunteerapp.dto.response.PostCreateResponse;
import ua.isvyry.volunteerapp.model.Location;
import ua.isvyry.volunteerapp.model.Participant;
import ua.isvyry.volunteerapp.model.Post;

import java.util.List;

@Service
@RequiredArgsConstructor
public class PostManagementService {
    private final PostService postService;
    private final ConversionService conversionService;
    private final ParticipantService participantService;
    private final CategoryService categoryService;

    private final LocationService locationService;

    public PostCreateResponse createPost(PostCreateRequest postCreateRequest) {
        var postToCreate = conversionService.convert(postCreateRequest, Post.class);
        var author = participantService.findById(postCreateRequest.getAuthorId());
        var category = categoryService.findById(postCreateRequest.getCategoryId());
        postToCreate.setAuthor(author);
        postToCreate.setCategory(category);
        return conversionService.convert(postService.createPost(postToCreate),
PostCreateResponse.class);
    }

    @Transactional(readOnly = true)
```

```
    public Page<Post> findAllInRadiusRange(Pageable pageable, Participant currentUser,
double radius) {
        Point currentUserPoint = currentUser.getLocation().getGeom();
        List<Location> locationsInRange = locationService
                .findAround(currentUserPoint.getY(), currentUserPoint.getX(), radius);
        return postService.findAllByLocationIn(pageable, locationsInRange);
    }
}
```

PostService.java

```
package ua.isvyry.volunteerapp.service;

import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ua.isvyry.volunteerapp.model.Location;
import ua.isvyry.volunteerapp.model.Post;
import ua.isvyry.volunteerapp.repository.PostRepository;

import java.util.List;

@Service
@RequiredArgsConstructor
public class PostService {

    private final PostRepository postRepository;

    @Transactional
    public Post createPost(Post post) {
        return postRepository.save(post);
    }

    public boolean existsByCategoryId(Long id) {
        return postRepository.existsByCategory_Id(id);
    }

    public Page<Post> findAll(Pageable pageable) {
        return postRepository.findAll(pageable);
    }

    public Page<Post> findAllByLocationIn(Pageable pageable, List<Location> locations) {
        return postRepository.findAllByLocationIn(pageable, locations);
    }
}
```

CategoryCreateRequestToCategoryConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
```

```
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.request.CategoryCreateRequest;
import ua.isvyry.volunteerapp.model.Category;

@Component
public class CategoryCreateRequestToCategoryConverter implements
Converter<CategoryCreateRequest, Category> {
   @Override
   public Category convert(CategoryCreateRequest source) {
      return Category.builder()
            .categoryName(source.getCategoryName())
            .build();
   }
}
```

CategoryToCategoryCreateResponseConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.response.CategoryCreateResponse;
import ua.isvyry.volunteerapp.model.Category;

@Component
public class CategoryToCategoryCreateResponseConverter implements
Converter<Category, CategoryCreateResponse> {
   @Override
   public CategoryCreateResponse convert(Category source) {
      return CategoryCreateResponse.builder()
            .categoryId(source.getId())
            .categoryName(source.getCategoryName())
            .build();
   }
}
```

CategoryToCategoryResponseConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.response.CategoryResponse;
import ua.isvyry.volunteerapp.model.Category;
import ua.isvyry.volunteerapp.model.Post;

import java.util.stream.Collectors;

@Component
public class CategoryToCategoryResponseConverter implements Converter<Category,
CategoryResponse> {
   @Override
```

```
    public CategoryResponse convert(Category source) {
        return CategoryResponse.builder()
                .categoryId(source.getId())
                .categoryName(source.getCategoryName())
                .postIds(source.getPosts().stream().map(Post::getId).collect(Collectors.toList()))
                .build();
    }
}
```

CategoryToCategoryUpdateResponseConverter.java

```
package ua.isvyry.volunteerapp.converter;


import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.response.CategoryUpdateResponse;
import ua.isvyry.volunteerapp.model.Category;

@Component
public class CategoryToCategoryUpdateResponseConverter implements
Converter<Category, CategoryUpdateResponse> {
    @Override
    public CategoryUpdateResponse convert(Category source) {
        return CategoryUpdateResponse.builder()
                .categoryId(source.getId())
                .categoryName(source.getCategoryName())
                .build();
    }
}
```

LocationCreateRequestToLocationConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.request.LocationCreateRequest;
import ua.isvyry.volunteerapp.model.Location;

@Component
public class LocationCreateRequestToLocationConverter implements
Converter<LocationCreateRequest, Location> {
    @Override
    public Location convert(LocationCreateRequest source) {
        return Location.builder()
                .country(source.getCountry())
                .city(source.getCity())
                .state(source.getState())
                .build();
    }
}
```

NewsCreateRequestToNewsConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.request.NewsCreateRequest;
import ua.isvyry.volunteerapp.model.News;

import java.util.Date;

@Component
public class NewsCreateRequestToNewsConverter implements
Converter<NewsCreateRequest, News> {
   @Override
   public News convert(NewsCreateRequest source) {
      return News.builder()
            .text(source.getText())
            .details(source.getDetails())
            .build();
   }
}
```

NewsToNewsCreateResponseConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.response.NewsCreateResponse;
import ua.isvyry.volunteerapp.model.News;

@Component
public class NewsToNewsCreateResponseConverter implements Converter<News,
NewsCreateResponse> {
   @Override
   public NewsCreateResponse convert(News source) {
      return NewsCreateResponse.builder()
            .id(source.getId())
            .createdDate(source.getCreatedDate())
            .text(source.getText())
            .details(source.getDetails())
            .build();
   }
}
```

NewsToNewsResponseConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
```

```
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.response.NewsResponse;
import ua.isvyry.volunteerapp.model.News;

@Component
public class NewsToNewsResponseConverter implements Converter<News,
NewsResponse> {
   @Override
   public NewsResponse convert(News source) {
      return NewsResponse.builder()
            .id(source.getId())
            .createdDate(source.getCreatedDate())
            .modifiedDate(source.getModifiedDate())
            .text(source.getText())
            .details(source.getDetails())
            .build();
   }
}
```

ParticipantCreateRequestToParticipantConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.request.ParticipantCreateRequest;
import ua.isvyry.volunteerapp.model.Participant;
import ua.isvyry.volunteerapp.model.ParticipantType;

@Component
public class ParticipantCreateRequestToParticipantConverter implements
Converter<ParticipantCreateRequest, Participant> {
   @Override
   public Participant convert(ParticipantCreateRequest source) {
      return Participant.builder()
            .name(source.getName())
            .description(source.getDescription())
            .email(source.getEmail())
            .participantType(ParticipantType.determineType(source.getParticipantType()))
            .build();
   }
}
```

ParticipantToParticipantCreateResponseConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.response.ParticipantCreateResponse;
import ua.isvyry.volunteerapp.model.Participant;
```

```
@Component
public class ParticipantToParticipantCreateResponseConverter implements
Converter<Participant, ParticipantCreateResponse> {
   @Override
   public ParticipantCreateResponse convert(Participant source) {
      return ParticipantCreateResponse.builder()
            .userId(source.getId())
            .participantType(source.getParticipantType().getName())
            .name(source.getName())
            .email(source.getEmail())
            .description(source.getDescription())
            .build();
   }
}
```

ParticipantToParticipantResponseConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.response.ParticipantResponse;
import ua.isvyry.volunteerapp.model.Participant;
import ua.isvyry.volunteerapp.model.Post;

import java.util.stream.Collectors;

@Component
public class ParticipantToParticipantResponseConverter implements Converter<Participant,
ParticipantResponse> {

   @Override
   public ParticipantResponse convert(Participant source) {
      return ParticipantResponse.builder()
            .userId(source.getId())
            .participantType(source.getParticipantType().getName())
            .name(source.getName())
            .email(source.getEmail())
            .description(source.getDescription())

.subscribedPostIds(source.getSubscribedPosts().stream().map(Post::getId).collect(Collectors
.toList()))
            .subscriptionIds(source.getSubscriptions().stream().map(Participant::getId)
                  .collect(Collectors.toList()))

.subscriberIds(source.getSubscribers().stream().map(Participant::getId).collect(Collectors.to
List()))
            .build();
   }
}
```

PostCreateRequestToPostConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.request.PostCreateRequest;
import ua.isvyry.volunteerapp.model.Post;

@Component
public class PostCreateRequestToPostConverter implements Converter<PostCreateRequest,
Post> {
   @Override
   public Post convert(PostCreateRequest source) {
      return Post.builder()
            .about(source.getAbout())
            .problem(source.getProblem())
            .solution(source.getSolution())
            .build();
   }
}
```

PostToPostCreateResponseConverter.java

```
package ua.isvyry.volunteerapp.converter;

import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;
import ua.isvyry.volunteerapp.dto.response.PostCreateResponse;
import ua.isvyry.volunteerapp.model.Post;

@Component
public class PostToPostCreateResponseConverter implements Converter<Post,
PostCreateResponse> {
   @Override
   public PostCreateResponse convert(Post source) {
      return PostCreateResponse.builder()
            .postId(source.getId())
            .solution(source.getSolution())
            .about(source.getAbout())
            .problem(source.getProblem())
            .categoryId(source.getCategory().getId())
            .authorId(source.getAuthor().getId())
            .build();
   }
}
```

AuthenticationController.java

```
package ua.isvyry.volunteerapp.controller;

import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import ua.isvyry.volunteerapp.dto.request.AuthenticationRequest;
import ua.isvyry.volunteerapp.dto.response.AuthenticationResponse;
import ua.isvyry.volunteerapp.service.AuthenticationService;

@RestController
@RequestMapping("/api/v1")
@RequiredArgsConstructor
public class AuthenticationController {

   private final AuthenticationService authenticationService;

   @PostMapping("/authenticate")
   public AuthenticationResponse authenticate(AuthenticationRequest request) {
      return authenticationService.authenticate(request);
   }
}
```

CategoryController.java

```
package ua.isvyry.volunteerapp.controller;

import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;
import ua.isvyry.volunteerapp.dto.request.CategoryCreateRequest;
import ua.isvyry.volunteerapp.dto.response.CategoryCreateResponse;
import ua.isvyry.volunteerapp.dto.response.CategoryResponse;
import ua.isvyry.volunteerapp.dto.response.CategoryUpdateResponse;
import ua.isvyry.volunteerapp.dto.request.CategoryUpdateRequest;
import ua.isvyry.volunteerapp.service.CategoryManagementService;

import java.util.List;

@RestController
@RequiredArgsConstructor
@RequestMapping("/api/v1/categories")
public class CategoryController {
   private final CategoryManagementService categoryManagementService;

   @PostMapping("/create")
   public CategoryCreateResponse createCategory(@RequestBody CategoryCreateRequest
request) {
      return categoryManagementService.createCategory(request);
   }

   @GetMapping
   public List<CategoryResponse> getAllCategories(@RequestParam(defaultValue = "0")
Integer pageNumber,
                                                  @RequestParam(defaultValue = "20") Integer pageSize) {
```

```
        return categoryManagementService.getAllCategories(pageNumber, pageSize);
    }

    @GetMapping("/{id}")
    public CategoryResponse getCategoryById(@PathVariable Long id) {
        return categoryManagementService.getById(id);
    }

    @PutMapping("/{id}")
    public CategoryUpdateResponse updateCategory(@PathVariable Long id, @RequestBody
CategoryUpdateRequest updateRequest) {
        return categoryManagementService.updateCategory(id, updateRequest);
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteCategory(@PathVariable Long id) {
        categoryManagementService.deleteCategoryById(id);
    }
}
```

DefaultControllerAdvice.java

```
package ua.isvyry.volunteerapp.controller;

import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import ua.isvyry.volunteerapp.dto.response.ErrorResponse;
import ua.isvyry.volunteerapp.exception.CategoryDeleteException;

import javax.persistence.EntityNotFoundException;
import javax.servlet.http.HttpServletRequest;

import java.time.LocalDateTime;

import static org.springframework.http.HttpStatus.CONFLICT;
import static org.springframework.http.HttpStatus.NOT_FOUND;
import static org.springframework.http.HttpStatus.BAD_REQUEST;

@RestControllerAdvice
public class DefaultControllerAdvice {

    @ExceptionHandler(EntityNotFoundException.class)
    @ResponseStatus(NOT_FOUND)
    public ErrorResponse entityNotFoundException(HttpServletRequest request,
EntityNotFoundException exception) {
        return ErrorResponse.builder()
                .timestamp(LocalDateTime.now())
                .status(NOT_FOUND.value())
```

```
                .error(exception.getMessage())
                .path(request.getRequestURI())
                .build();
    }

    @ExceptionHandler(UsernameNotFoundException.class)
    @ResponseStatus(NOT_FOUND)
    public ErrorResponse usernameNotFoundException(HttpServletRequest request,
UsernameNotFoundException exception) {
        return ErrorResponse.builder()
                .timestamp(LocalDateTime.now())
                .status(NOT_FOUND.value())
                .error(exception.getMessage())
                .path(request.getRequestURI())
                .build();
    }

    @ExceptionHandler(BadCredentialsException.class)
    @ResponseStatus(BAD_REQUEST)
    public ErrorResponse badCredentialsException(HttpServletRequest request,
BadCredentialsException exception) {
        return ErrorResponse.builder()
                .timestamp(LocalDateTime.now())
                .status(BAD_REQUEST.value())
                .error(exception.getMessage())
                .path(request.getRequestURI())
                .build();
    }

    @ExceptionHandler(CategoryDeleteException.class)
    @ResponseStatus(CONFLICT)
    public ErrorResponse categoryDeleteException(HttpServletRequest request,
CategoryDeleteException exception) {
        return ErrorResponse.builder()
                .timestamp(LocalDateTime.now())
                .status(CONFLICT.value())
                .error(exception.getMessage())
                .path(request.getRequestURI())
                .build();
    }
}
```

NewsController.java

```
package ua.isvyry.volunteerapp.controller;

import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;
import ua.isvyry.volunteerapp.dto.request.NewsCreateRequest;
import ua.isvyry.volunteerapp.dto.request.NewsUpdateRequest;
import ua.isvyry.volunteerapp.dto.response.*;
```

```
import ua.isvyry.volunteerapp.service.NewsManagementService;

import java.util.List;

@RestController
@RequiredArgsConstructor
@RequestMapping("/api/v1/news")
public class NewsController {

   private final NewsManagementService newsManagementService;

   @PostMapping("/create")
   public NewsCreateResponse createNews(@RequestBody NewsCreateRequest
createRequest) {
      return newsManagementService.createNews(createRequest);
   }

   @GetMapping
   public List<NewsResponse> getAllCategories(@RequestParam(defaultValue = "0")
Integer pageNumber,
                                              @RequestParam(defaultValue = "20") Integer pageSize) {
      return newsManagementService.getAllNews(pageNumber, pageSize);
   }

   @GetMapping("/{id}")
   public NewsResponse getCategoryById(@PathVariable Long id) {
      return newsManagementService.getById(id);
   }

   @PutMapping("/{id}")
   public NewsResponse updateCategory(@PathVariable Long id, @RequestBody
NewsUpdateRequest updateRequest) {
      return newsManagementService.updateNews(id, updateRequest);
   }

   @DeleteMapping("/{id}")
   @ResponseStatus(HttpStatus.NO_CONTENT)
   public void deleteCategory(@PathVariable Long id) {
      newsManagementService.deleteNewsById(id);
   }
}
```

ParticipantController.java

```
package ua.isvyry.volunteerapp.controller;

import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.PageRequest;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.context.request.RequestContextHolder;
import org.springframework.web.context.request.ServletRequestAttributes;
```

```
import ua.isvyry.volunteerapp.constants.Constants;
import ua.isvyry.volunteerapp.dto.request.ParticipantCreateRequest;
import ua.isvyry.volunteerapp.dto.response.ParticipantCreateResponse;
import ua.isvyry.volunteerapp.dto.response.ParticipantResponse;
import ua.isvyry.volunteerapp.security.VolunteerAppUser;
import ua.isvyry.volunteerapp.security.VolunteerAppUserDetailsService;
import ua.isvyry.volunteerapp.service.JwtTokenUtil;
import ua.isvyry.volunteerapp.service.ParticipantManagementService;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/v1/participants")
@RequiredArgsConstructor
public class ParticipantController {

    private final ParticipantManagementService participantManagementService;
    private final VolunteerAppUserDetailsService userDetailsService;
    private final JwtTokenUtil jwtTokenUtil;

    @ResponseStatus(HttpStatus.CREATED)
    @PostMapping("/create")
    public ParticipantCreateResponse createParticipant(@RequestBody @Valid
ParticipantCreateRequest request) {
        return participantManagementService.createParticipant(request);
    }

    @GetMapping
    public List<ParticipantResponse> getAllParticipants(@RequestParam(defaultValue = "0")
Integer pageNumber,
                                                        @RequestParam(defaultValue = "20") Integer pageSize) {
        return participantManagementService.getAllParticipants(pageNumber, pageSize);
    }

    @GetMapping("/{id}")
    public ParticipantResponse getParticipantById(@PathVariable Long id) {
        return participantManagementService.getById(id);
    }

    @PostMapping("/{id}/subscribe")
    public ParticipantResponse subscribe(@PathVariable Long id) {
        HttpServletRequest request = getCurrentHttpRequest().get();
        String username = jwtTokenUtil.getUsernameFromToken(
                request.getHeader(Constants.AUTHORIZATION_HEADER).replace("Bearer ",
""));
        return participantManagementService.subscribe(((VolunteerAppUser)
userDetailsService
                .loadUserByUsername(username)).getUser(), id);
    }
```

```
    @GetMapping("/{id}/getNearest")
    public List<ParticipantResponse> getNearest(@PathVariable Long id,
                                    @RequestParam(defaultValue = "0") Integer pageNumber,
                                    @RequestParam(defaultValue = "20") Integer pageSize,
                                    @RequestParam(defaultValue = "1000") Integer radius) {
        return
participantManagementService.findAllInRadiusRange(PageRequest.of(pageNumber,
pageSize), id, radius);
    }

    private Optional<HttpServletRequest> getCurrentHttpRequest() {
        return Optional.ofNullable(RequestContextHolder.getRequestAttributes())
                .filter(ServletRequestAttributes.class::isInstance)
                .map(ServletRequestAttributes.class::cast)
                .map(ServletRequestAttributes::getRequest);
    }
}
```

PostController.java

```
package ua.isvyry.volunteerapp.controller;

import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.PageRequest;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;
import ua.isvyry.volunteerapp.dto.request.PostCreateRequest;
import ua.isvyry.volunteerapp.dto.response.ParticipantResponse;
import ua.isvyry.volunteerapp.dto.response.PostCreateResponse;
import ua.isvyry.volunteerapp.service.PostManagementService;

import java.util.List;

@RestController
@RequiredArgsConstructor
@RequestMapping("/api/v1//posts")
public class PostController {
    private final PostManagementService postManagementService;

    @PostMapping("/create")
    @ResponseStatus(HttpStatus.CREATED)
    public PostCreateResponse createPost(@RequestBody PostCreateRequest
postCreateRequest) {
        return postManagementService.createPost(postCreateRequest);
    }

    @GetMapping("/{id}/getNearest")
    public List<PostCreateResponse> getNearest(@PathVariable Long id,
                                    @RequestParam(defaultValue = "0") Integer pageNumber,
                                    @RequestParam(defaultValue = "20") Integer pageSize,
                                    @RequestParam(defaultValue = "1000") Integer radius) {
```

```
        return postManagementService.findAllInRadiusRange(PageRequest.of(pageNumber,
pageSize), id, radius);
    }
}
```

ДОДАТОК Б

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Наукового керівника *Бердника Михайла Геннадійовича, доктора технічних наук, доцента, професора кафедри програмного забезпечення комп'ютерних систем*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

на магістерську роботу

студента *Свириденка Ігоря Євгеновича*
(прізвище, ім'я, по батькові)

курсу II групи *121м-21-1*
спеціальності *121 Інженерія програмного забезпечення*
освітньої програми *Інженерія програмного забезпечення*
на тему *Розробка та дослідження ефективності впровадження серверної частини волонтерської платформи засобами Java, Spring Boot, PostgreSQL, OpenAPI 3.0.*

Актуальність теми *Тема магістерської роботи є актуальної, бо наш час волонтерство розповсюджено і є однією з найважливіших сфер діяльності в країни, та оптимізація взаємодії волонтерів є однією з найважливіших задач.*

Мета досліджень *Метою роботи розробка та оптимізація універсальної серверної частини волонтерської платформи.*

Коротка характеристика розділів роботи *Перший розділ був присвячений опису предметної галузі, також було визначено актуальність розробки та розглянуті існуючі аналоги. У другому розділі розглядаються математичні методи, застосовані в розробці. Водночас, третій розділ пояснює схему роботи розробленого застосунку, проілюстровано діаграму бази даних та способи взаємодії з програмним інтерфейсом.*

Практичне значення роботи *Практичне значення роботи полягає в розробці та отпимізації серверної частини волонтерської платформи, що полегшить взаємодію між волонтерами, та допоможе будувати їм плідну співпрацю.*

Зауваження та недоліки

Висновки та оцінка *Кваліфікаційна робота заслуговує оцінки «відмінно», виконавець заслуговує на присвоєння відповідної кваліфікації.*

Науковий керівник *Бердник Михайло Геннадійович, професор, каф. ПЗКС*
(прізвище, ім'я, по батькові, посада, місце роботи)

«_19__»__грудня_____2022 р. _____
(підпис)

ДОДАТОК В

РЕЦЕНЗІЯ
на кваліфікаційну роботу

студента *Свириденка Ігоря Євгеновича*
(прізвище, ім'я, по батькові)

курсу ІІ групи *121м-21-1*
кафедри програмного забезпечення комп'ютерних систем
спеціальності *121 Інженерія програмного забезпечення*

освітньої програми *Інженерія програмного забезпечення*

Тема роботи *Розробка та дослідження ефективності впровадження серверної частини волонтерської платформи засобами Java, Spring Boot, PostgreSQL, OpenAPI 3.0.*

Стисла характеристика розділів роботи *Перший розділ складається з опису предметної галузі, завдання, аналізу існуючих систем-аналогів та описанння програмного продукту.*
У другому розділі студент дає роз'яснення математичним методам та формулам, що будуть використані в розробці продукту.
У третьому розділі розглядається розробка система, описується структура бази даних, архітектура програми та інтерфейс користувача.

Пропозиції, внесені студентом, рівень їх наукового обґрунтування
Пропозиції внесені студентом, рівень їх наукового обґрунтування в роботі, яка рецензується і написана сучасною науковою обґрунтовані і випливають із суті зробленої роботи.

Практичне значення роботи *Практична цінність полягає в розробці та оптимізації серверної частини волонтерської платформи, що полегшить зв'язок між волонтерами, та допоможе їм будувати ефективну взаємодію.*

Якість оформлення роботи *робота виконана у відповідності до вимог оформлення кваліфікаційних робіт і відповідає поставленій задачі.*

Недоліки в роботі

Загальний висновок *отримані результати є закінченою науково- дослідною роботою і викликають науковий інтерес і демонструють здатність Свириденка Ігоря Євгеновича до самостійного аналізу і проведення наукової роботи та вміння розробки комп'ютерних програм. Кваліфікаційна робота заслуговує оцінки "відмінно", а Свириденка І.Є. – присвоєння відповідної кваліфікації*

(підготовленість студента до самостійної роботи як спеціаліста)

Оцінка магістерської роботи *"відмінно",*

Рецензент *Шедловський І.А., к.т.н., доцент, доцент каф. ІТКІ*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

«_19__» грудня__2022 р.__ _____

(підпис)

ДОДАТОК Г

Перелік файлів на диску

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Свириденко.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Свириденко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Свириденко.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Свириденко.ppt	Презентація кваліфікаційної роботи