МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна технологія підтримки та інтеграції біженців у новому середовищі»
здобувачки групи ІН.м-22 Малюк Алли Сергіївни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Алла Малюк
(підпис)

Керівник,
в.о. завідувача кафедри,
кандидат технічних наук, доцент Ігор ШЕЛЕХОВ _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо- професійної програми «Інформатика»
здобувача групи ІН.м-22 Малюк Алли Сергіївни

1. Тема роботи: «Інформаційна технологія тестування системи електронної комерції»
затверджую наказом по СумДУ від «01» грудня 2023 року № 0475-VI
2. Термін здачі здобувачем кваліфікаційної роботи до 16 грудня 2023 року
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для розробки веб додатків. 3) Розробка інформаційної моделі. 4) Вибір методології, мови програмування та засоби програмної реалізації 5) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «06» листопада 2023 р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються для розробки веб додатків*		
3	*Розробка інформаційної моделі*		
4	*Вибір методології, мови програмування та засоби програмної реалізації*		
5	*Аналіз результатів*		
6	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 60 стр., 32 рис., 1 додаток, 2 табл., 30 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі розробки моделей та інформаційної технології проєктування веб-орієнтованих систем допомоги біженцям.

Об'єкт дослідження — процес проєктування систем інформаційної допомоги військовим біженцям

Мета роботи — Розробка інформаційних моделей, методів та засобів інформаційної технології проєктування веб-додатку призначеного для допомоги біженцям.

Методи дослідження — методи інформаційного аналізу і синтезу сучасних веб-додатків.

Результати — Проаналізовано можливі методів розробки веб-додатку, разом з найсучаснішими технологіями та інструментами для створення. Розроблено інформаційну модель веб-орієнтованої системи для допомоги біженцям. В результаті створено веб-додаток який допоможе краще зрозуміти потреби біженців та зробити внесок в полегшення їхньої адаптації та інтеграції в нове середовище.

ВЕБ-ДОДАТОК, ДОПОМОГА БІЖЕНЦЯМ, МЕТОДОЛОГІЇ ПРОГРАМУВАННЯ, JAVA, SPRING FRAMEWORK, JAVASCRIPT, REACT.

ЗМІСТ

ВСТУП

Актуальність В наш час, в умовах глобальних конфліктів та лих питання міграції та підтримки біженців стають більш актуальними та невідкладними, ніж будь-коли. Події пов’язані з війною в Україні призвели до масової міграції військових біженців, які шукають безпеку та притулок в інших містах або навіть країнах. У цьому контексті стає критичним пошук та розробка інструментів та ресурсів, які б полегшили процес адаптації цих людей у нове середовище.

Ця магістерська робота присвячена створенню та дослідженню веб-додатку для допомоги військовим біженцям з окупованих або розташованих в межах активних бойових дій міст для їх безпеки. Цей додаток призначений для допомоги в різних аспектах, починаючи з інформаційної до економічної допомоги.

Об’єктом дослідження даної роботи є процес розробки та створення додатку для допомоги військовим біженцям, який буде надавати цим людям необхідні їм дані та допоможе значно полегшити можливість знаходження нової домівки.

Предметом дослідження цієї роботи є аналіз можливих методів розробки даного веб-додатку, разом з найсучаснішими технологіями та інструментами для створення. Крім цього буде вибрано найбільш підходящий алгоритм для допомоги біженцям. Також буде проведено аналіз питань та викликів які стоять перед цими людьми та способи якими додаток зможе їм допомогти. Окрім цього буде проведено аналіз існуючих рішень пов’язаних із підтримкою та інтеграцією біженців.

Гіпотеза полягає в тому, що врахування особливостей допомоги біженців і підвищення ефективності послуг, що їм надаються, можна досягти шляхом адаптації існуючих та розробці оригінальних функціональних елементів відповідної веб-орієнтованої системи в ході її проектування та програмної реалізації.

Наукова новизна.

На відміну від існуючих аналогів запропонована інформаційна технологія, що складається з адаптованих з урахуванням особливостей предметної галузі і постановки задачі інформаційних моделей, методів та засобів інформаційного аналізу і синтезу веб-орієнтованої системи допомоги біженцям, дозволяє реалізувати ефективні механізми їх інтеграції в нове середовище та соціальної адаптації.

Структура роботи. Кваліфікаційна робота складається з складається зі вступу, інформаційно-аналітичного огляду, постановки задачі дослідження, аналітичної частини щодо вибору методів та інструментів для рішення поставленої задачі, практичної частини з деталізованим описом особливостей програмної реалізації засобів запропонованої інформаційної технології, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень та публікацій

Питання допомоги біженцям на жаль стоїть дуже гостро в останні роки в усьому світі, через це було створено велика кількість досліджень відносно цієї теми з ціллю допомоги таким людям. В цьому пункті було оглянуто та зроблено висновки відносно цих потреб.

Біженець – це людина яка через деякі події (воєнні чи соціальні) втратила покинула країну в якій вона проживала. Конвенція ООН називає біженцями людей які внаслідок обґрунтованих страхів щодо релігійних, соціальних чи політичних поглядів знаходиться поза межами своєї країни та не має можливості чи бажання користуватися її захистом. Це визначення використовується більшістю країнами, зокрема і Україною.[1]

Положення, що відносяться до статусу біженців містяться в конвенції про захист цивільного населення на час війни, угоді щодо скасування віз та декларації ООН щодо територіального притулку та закріплено у низці міжнародних документів.

На даний момент у світі існує багато причин з яких люди стають біженцями, найбільш поширеними з них є війна та переслідування. Найбільша кількість воєнних біженців виходять з Сирії (приблизно 6.5 мільйонів), України (приблизно 5.7 мільйонів) та Афганістану з Венесуелою (приблизно 5.5 мільйонів). Ці біженці намагаються знайти прихисток у найближчих до них країнах. Українські біженці зазвичай знаходять прихисток у країнах Європи рис. 1.1. [2]

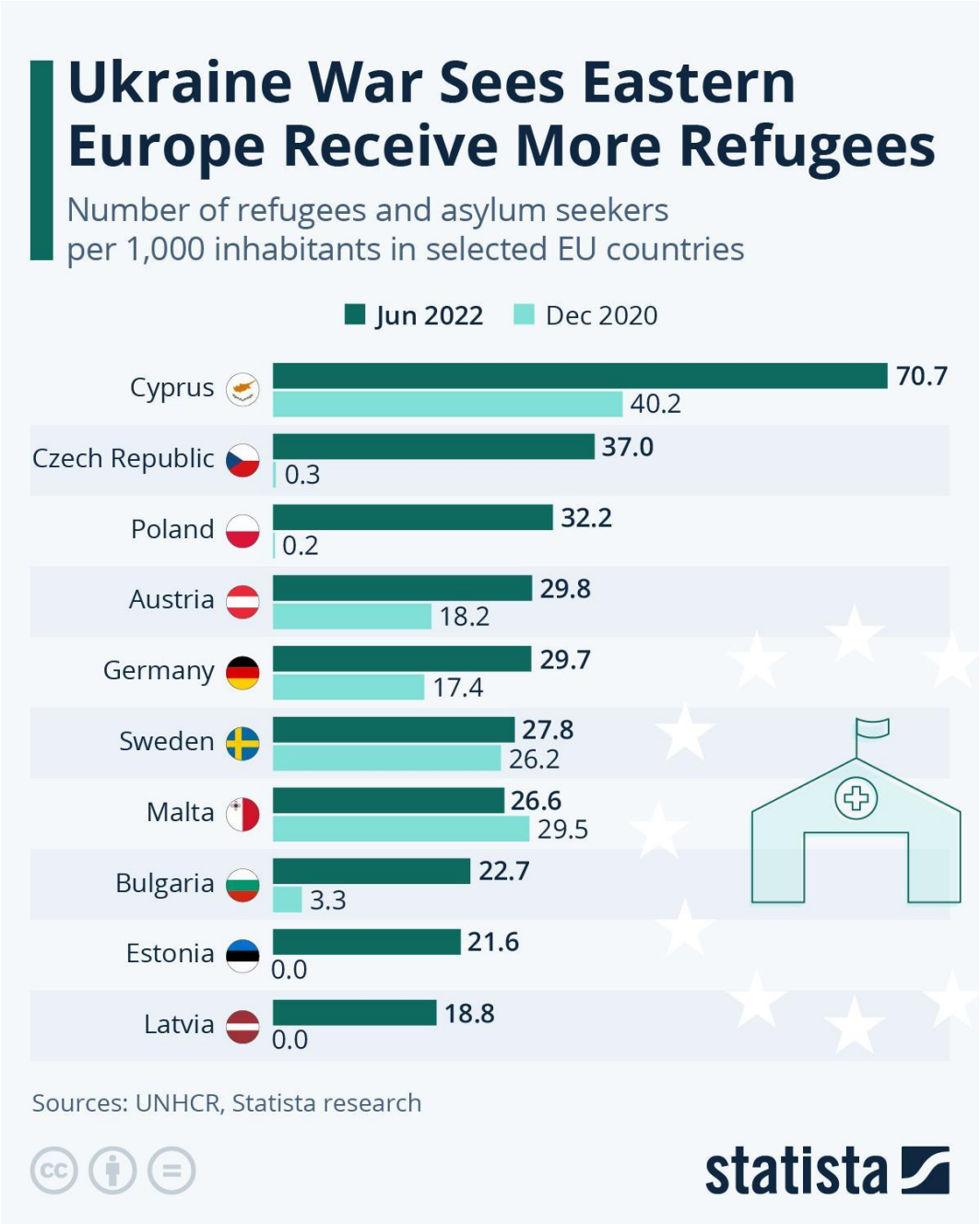


Рисунок 1.1 – Кількість українських біженців в країнах Європи

Основною проблемою пов’язаною з біженцями є проблеми переслідування та можливості цих біженців для їх дислокації. Крім цього найбільш переслідуємими групами є жінки та діти, які подорожують самітньо. Ця група людей найбільш підвержена ризику нападу та дискримінації відносно їх статі та проблем. Нажаль з цими проблемами стикаються не лише в інших країнах, та навіть в містах всередині України.[3]

Найбільш складною проблемою з якою стикаються біженці є можливість швидко покинути область бойових дій, це пов’язано з багатьма причинами – психологічними (людина боїться невизначеності), грошовими та навіть банальною відсутністю можливості швидко знайти необхідні ресурси чи людей які їм можуть допомогти.

Перше з чим необхідно ознайомитися для допомоги з біженцями це психологічні проблеми з якими вони стикаються, адже навіть після отримання статусу біженця та переїзду до іншого місця проживання, більшість з них стикається з великою кількість психологічних проблем які потребують професійного втручання та догляду.[4]

Найперше з чим стикаються біженці є тривога та депресія пов’язані з невідомістю та нерозумінням що робити далі, окрім цього зазвичай ці люди стикаються з обсесивно-компульсивним розладом, розладом вживання психоактивних речовин та багатьма іншими проблемами.[5]

Як висновок – перша допомога (окрім фізичної) яку мають отримати біженці це професійний огляд спеціалістів які можуть допомогти їм заспокоїтися та уникнути паніки. Зазвичай для цього наймають спеціалістів, проте на момент їх переїзду цим зазвичай займаються волонтери.

Окрім цього великій кількості людей потребується допомога з переїздом до іншого міста чи навіть країни. Україна намагається забезпечити цих людей допомогою, проте існує велика кількість волонтерів, які за власною ініціативою бажають допомогти цим людям. Ці волонтери мають зазвичай знайти якусь організацію яка зможе керувати цими людьми та казати що їм робити. Проте цей процес може бути занадто довгим та незрозумілим для звичайної людини, тому існує велика необхідність у створенні ресурсу в якому така людина може описати яку допомогу вона має можливість надати, задля спрощення і автоматизації цього процесу. [6]

На даний момент в Україні діють декілька ресурсів які можуть надати допомогу таким людям, ці ресурси будуть оглянуті в наступному пункті, проте зазвичай ці ресурси не мають можливості людині швидко знайти потрібні їх знання не зважаючи на її технологічні навички. Багато хто з цих людей, зазвичай люди літнього віку чи люді не дуже ознайомлені з технологіями, не можуть розібратися в цих ресурсах, що створює гостру потребу в ресурсі який швидко і без зайвих кроків зможе допомогти цим людям знайти необхідний прихисток.

Окрім цього, найбільш складною частиною є допомога людям в отриманні статусу біженців, для чого потребуються спеціалісти які зможуть детально пояснити та допомогти людині що і як їй потрібно зробити. На жаль цей процес неможливо автоматизувати, проте існує хоча б можливість надання людині необхідних контактів або онлайн ресурсів які можуть полегшити цю задачу.[7]

Наприкінці потрібно не забувати, що ці люди можуть мати фізичні проблеми зі здоров’ям, які можуть потребувати спеціального транспортування або лікування, що робить необхідним наявність лікарів під час, або після їх транспортування. На жаль цей процес також неможливо повністю автоматизувати, адже він потребує присутність цих людей, проте веб ресурс може допомогти лікарям чи волонтерам зазначити що вони можуть з цим допомогти та полегшити можливість контакту з ними.

Проаналізувавши ці дані – зрозуміло, що існує гостра необхідність у створенні додатку, який допоможе як біженцям у знаходженні притулку та допомозі їм з матеріальної та психологічної точки зору, так і людям-волонтерам, які мають можливість допомогти людям з їх потребами та спростити ці процеси.[8]

1.2 Огляд ресурсів аналогів

На даний момент існує декілька ресурсів які можуть забезпечити допомогу українським біженцям як в межах країнах, так і за кордоном.

Найбільш відомими ресурсами для допомоги біженцям в переїзді та допомозі в межах країни є –

- «Ukraine now»
- «prykhystok.gov.ua»

Крім цього існують ресурси для допомоги виїзду за кордон –

- «Разом для України»
- «Допомога RO»

Перед розробкою власного додатку необхідно проаналізувати ці ресурси, визначити їх плюси та мінуси, та зробити висновки відносно того що має бути в веб-додатку, а чого потрібно уникнути.

«Ukraine now» – веб-ресурс, який допомагає біженцям у питанні їх переїзду в межах країни, та містить можливість отримати допомогу в деяких інших випадках.

Цей ресурс має велику кількість можливостей для людей які потребують допомогу. На ньому існує велика кількість розділів, серед них найбільш необхідні :

- Секція з можливістю допомагати біженцям речами або грошима
- Інформаційна секція з новинами та важливою інформацією
- Секція для волонтерів в якій можливо вказати яку допомогу ти можеш надати та можливість їх реєстрації
- Секція з історіями біженців
- Секція для біженців з переліком можливої допомоги

Після аналізу цього ресурсу, мною було виділено такі плюси:

- Велика кількість функціоналу – сайт надає можливість як і біженцям знайти допомогу, так і волонтерам які хочуть допомогти.
- Існує окремі секції сайту яка містить останні новини з якою біженці та громадяни країни можуть ознайомитися.
- Існує секція з історіями біженців та людей країни з якими можуть ознайомитися інші люди задля розуміння усієї проблеми.
- Можливість змінювати мову

Та мінуси:

- Доволі важкий для розуміння інтерфейс, людина яка потребує допомоги буде доволі довго шукати потрібну інформацію, адже на сайті усі секції ідуть одна після іншої, а кнопки для допомоги ніяк не виділені і їх дуже важко знайти.
- За замовчування ввімкнена англійська мова, що може дуже сильно заважати людям, які її не знають у пошуках потрібної інформації. Окрім цього, якщо людина не дуже ознайомлена з технологіями(як наприклад літні люди) вона може не знайти де її змінити.
- Перевантаження головної сторінки – занадто багато інформації міститься на головній сторінці, що вкупі з відсутністю структурованості може плутати користувача.

Наступним ресурсом який буде проаналізовано є «prykhystok.gov.ua» – сайт з оголошеннями на якому можливо подати, або знайти житло в Україні.

Плюсами цього ресурсу є:

- Максимально простий інтерфейс, на головній сторінці є чітко виділені кнопки для знаходження чи запропонування житла, є окрема статистика по регіонам.
- Сайт за замовчуванням відкривається українська мова, що може допомогти людям які не розуміють англійську.
- Можливість зміни мови

Мінусами:

- Вузька спеціалізація, з усіх проблем з якими мають діло біженці, цей сайт надає лише можливість знайти прихисток.
- Відсутність допомоги при пересуванні людей до місця проживання, якщо людина знаходиться в іншому місту – вона не зможе за допомогою цього сайту туди швидко доїхати.

Після цього було проаналізовано ресурси які можуть допомогти з переїздом за кордон. Першим з цих ресурсів є «Разом для України».

Цей ресурс містить інформацію щодо потрібних документів та кроків для переїзду, можливість отримати юридичну та освітню підтримку, та допомогу з працевлаштуванням.

Плюси цього ресурсу:

- Максимально доступний інтерфейс (при переході на головну сторінку сайту тебе запитують яку мову ти бажаєш використовувати, всі необхідні елементи одразу на головній сторінці)
- Можливість отримати онлайн консультацію де тобі підкажуть усе необхідне.

Мінуси:

- Більша частина допомоги – онлайн консультації, не має можливості допомогти людині фізично виїхати, мета ресурсу ознайомити людину з інформацією.

Останнім ресурсом є «Допомога RO».

Цей веб-додаток допомагає українцям емігрувати до Румунії, містить як необхідну інформацію для переїзду, так і можливість отримати фізичну допомогу(лікарі, речі, гроші, працевлаштування).

Плюсами цього ресурсу є :

- Можливість отримати консультацію, після якої ти отримаєш детальну інструкцію, що робити далі.
- Можливість ознайомитися з усіма деталями проживання в Румунії, такими як освіта, правові статуси та інше.

Мінусами:

- Сайт спрямований лише на переїзд до Румунії, окрім цього не має можливості зареєструватися волонтерам.
- Складний для розуміння та перевантажений інтерфейс в якому складно розібратися.

Після аналізу усіх плюсів та мінусів разом із вмістом усіх веб ресурсів, було зроблено такі висновки:

1. Необхідно зробити максимально зрозумілий для всіх людей інтерфейс в якому можливо одразу знайти найголовніші пункти.
2. Необхідно зробити так, щоб сайт містив українську мову одразу при переході на нього, адже багато хто з біженців можливо не знає англійську.
3. Необхідно зробити можливість як і біженцям знайти необхідну їм допомогу, так і волонтерам надати свої контакти.

4. Потрібно уникати перевантаженості сайту, та чітко структурувати сайт, щоб біженці з повільним інтернетом могли уникнути необхідності завантажувати непотрібні фрагменти, та зпростит користування сайту.

1.3 Постановка задачі

Після аналізу ресурсів аналогів, та необхідності біженців та волонтерів, були поставлені такі задачі.

1) Створення функціональних вимог та визначення функціоналу який може надавати веб-додаток. Для цього необхідно визначитися з тим, що потребують волонтери та біженці використовуючи аналізи з попереднього пункту.
2) Дослідити та обрати необхідні технологій, разом з методами розробки. Це включає в себе аналіз мови програмування, фреймворків та бібліотек які можуть спростити виконання цього завдання.
3) Створити структурну схему роботи веб-додатку, для цього використати діаграми які будуть відображати процеси всередині цього додатку, разом з поясненнями.
4) Спроектувати юзер інтерфейс додатку, через який людина буде взаємодіяти з цим додатком. Для цього необхідно використати розроблену раніше структуру роботи додатку.
5) Розробити фактичну реалізацію додатку, разом із усіма необхідними елементами, такими як бази даних, реалізація алгоритмів роботи, тощо.
6) Розробити подальше покращення програми, а саме – додавання нового функціоналу за потребою, виправлення недоліків, тестування та інше.

2 АНАЛІТИЧНА ЧАСТИНА

2.1 Створення функціональних вимог

Перед створенням додатку, необхідно визначитися із тим, які користувачі будуть ним користуватися, та які потреби вони мають.

Спочатку було визначено користувачі веб-додатку – а саме біженець та волонтер. (таблиця 2.1)

Таблиця 2.1. Користувачі web-додатку

Назва	Опис
Біженець	Людина яка потребує допомоги з проблемами пов’язаними з переїздом, консультацією з психологом, лікарем, тощо.
Волонтер	Людина яка може допомогти біженцю з його проблемами.

Після цього, необхідно визначитися з потребами які мають ці користувачі.

Як можна зрозуміти, ці користувачі функціонально відрізняються тим, що волонтер має надавати інформацію відносно допомоги яку він може запропонувати, а біженець навпаки, отримувати цю інформацію.

Отже волонтери повинні мати можливість надавати повну інформацію щодо себе та послуг які вони можуть надавати, а біженці мати можливість швидко фільтрувати цю інформацію з метою отримання потрібної їм інформації чи контактів.

2.2 Створення структурної схеми роботи

Після визначення функціональних вимог, необхідно розробити структуру роботи додатку. Для цього спочатку необхідно визначити яка логіка буде всередині додатку, а потім відобразити ці процеси за допомогою діаграм

Для зображення цієї логіки більш за все підходять IDEF0 та Use Case діаграми, які мають можливість показати загальну логіку додатку, та її конкретну реалізацію

Спершу мною було створено с діаграму, яка відображає повну структуру додатку, показуючи потреби користувачів і яка дозволяє зрозуміти ризики та необхідні дані. (рис 2.1)

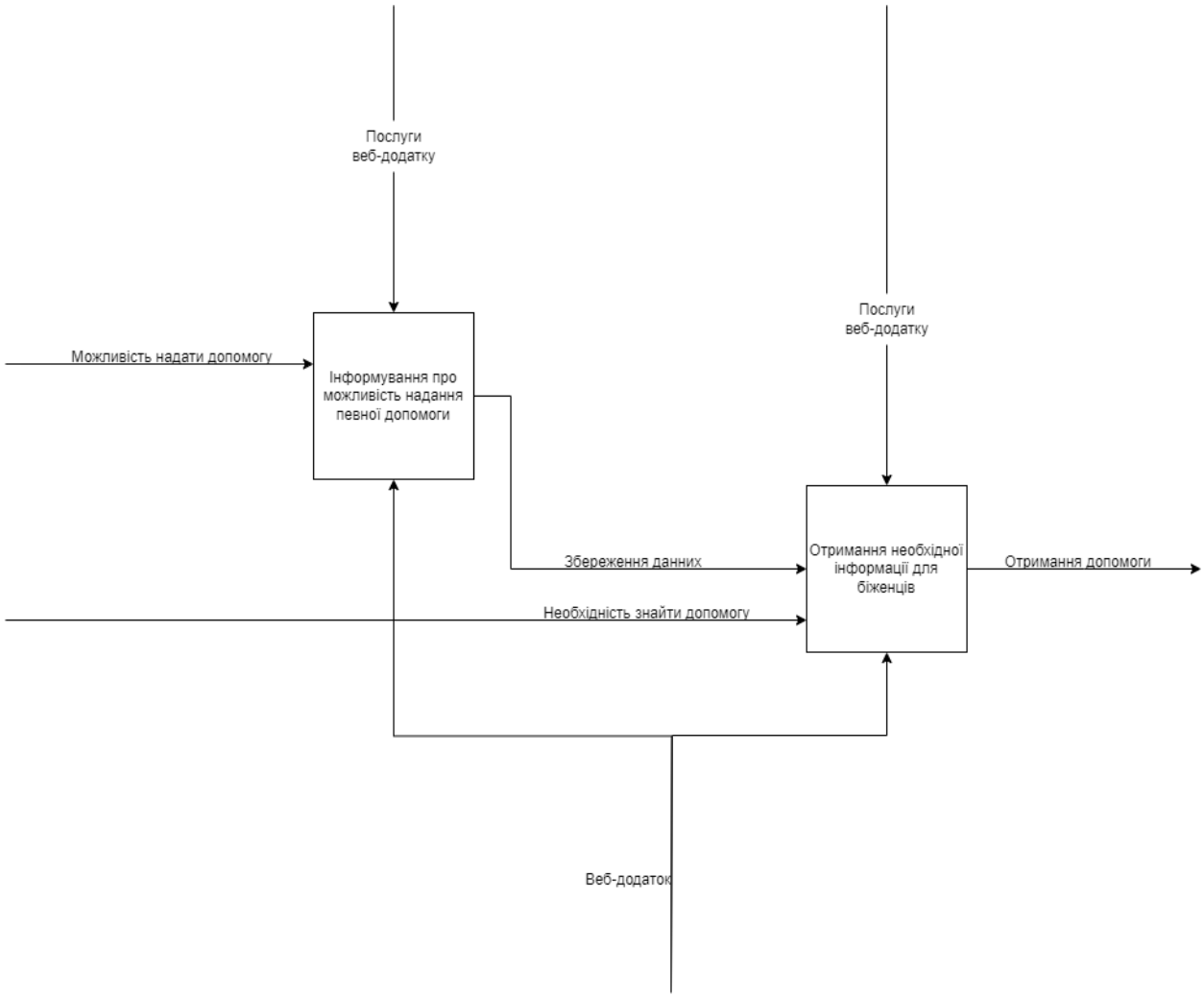


Рисунок 2.1 - IDEF0 діаграма

Після створення IDEF0 необхідно створити Use Case діаграму, адже IDEF0 не має повної інформації щодо логіки, а для розробки веб-додатку необхідно мати повне розуміння роботи програми.

Use Case діаграми містять більш детальну реалізацію логіки, що дозволить зрозуміти як має працювати додаток, та спростити його програмну реалізацію. Для їх створення необхідно показати процеси та їх взаємодію між собою та акторами(сутностями які беруть участь у цих процесах).

Для створення Use Case діаграм, спочатку необхідно визначитися з усіма процесами необхідними для користування додатком. Для цього було виділено такі процеси:

- Реєстрація та авторизація волонтерів. Вона необхідна для того, щоб адміністрація сайту мала можливість перевірити цих людей перед тим як вони отримають можливість надавати свої послуги. Реєстрації для біженців не передбачено, адже все що їм потрібно – отримати необхідні контакти волонтерів і якомога швидше, а отже реєстрація та авторизація може значно сповільнити цей процесс.
- Надання волонтером інформації щодо своїх послуг, разом із контактами.
- Отримання біженцем необхідної для нього інформації за допомогою фільтрації введених волонтером даних.

Після цього необхідно визначитися з акторами які будуть взаємодіяти з веб-додатком. Акторами які взаємодіють з веб додатком разом з їх описом наведено в таблиці далі(таблиця 2.2).

Таблиця 2.2. Актори в Use Case діаграмах

Назва	**Опис**
Біженець	Людина яка потребує допомоги з проблемами пов’язаними з переїздом, консультацією з психологом, лікарем, тощо.
Волонтер	Людина яка може допомогти біженцю з його проблемами.
База даних	База даних в якій зберігаються усі потрібні для функціонування системи дані.
Система	Програмна реалізація алгоритмів, які виконують необхідний процесс

Для даних процесів були створені відповідні Use Case діаграми:

- Діаграма реєстрації (рис 2.2), яка описує процес реєстрації волонтерів в системі, разом з валідацією введених даних та їх збереженням.

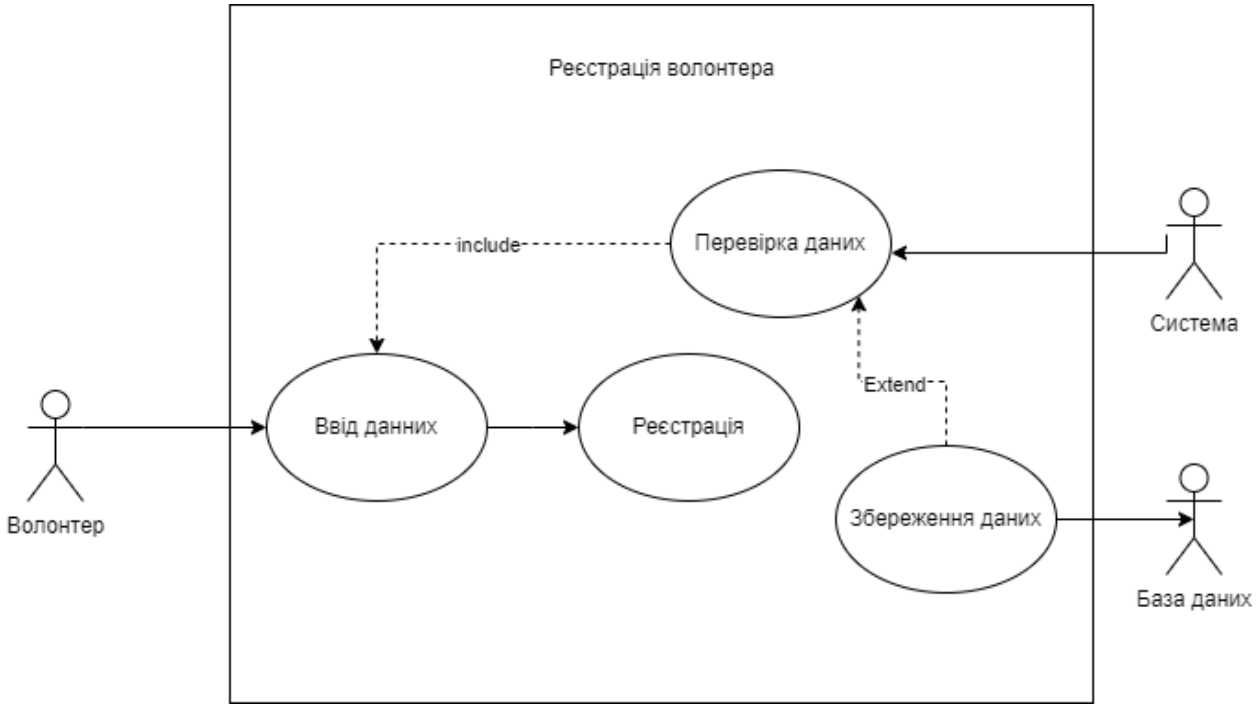


Рисунок 2.2 - Use Case діаграма опису процесу реєстрації

- Діаграма авторизації(рис 2.3), яка описує процес авторизації волонтера в системі, що містить у собі перевірку введених даних у порівнянні з указаними при реєстрації.

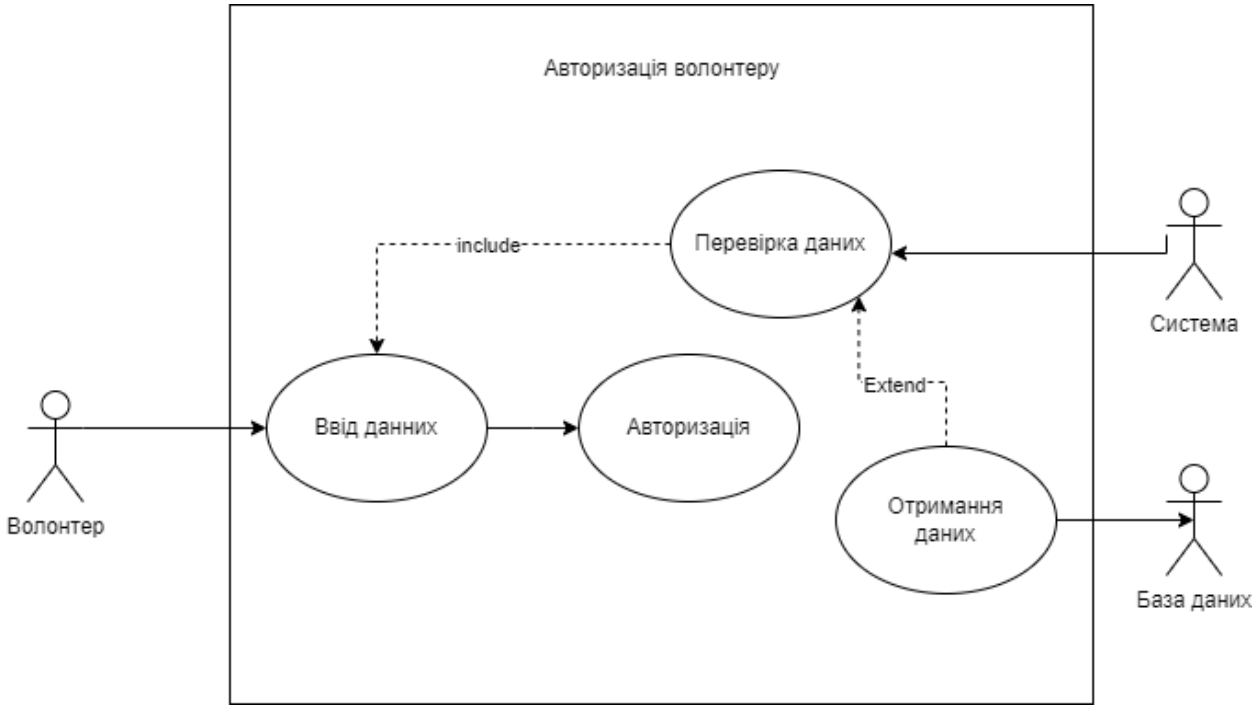


Рисунок 2.3 - Use Case діаграма опису процесу авторизації

- Діаграма надання волонтером інформації щодо своїх послуг(рис. 2.4)

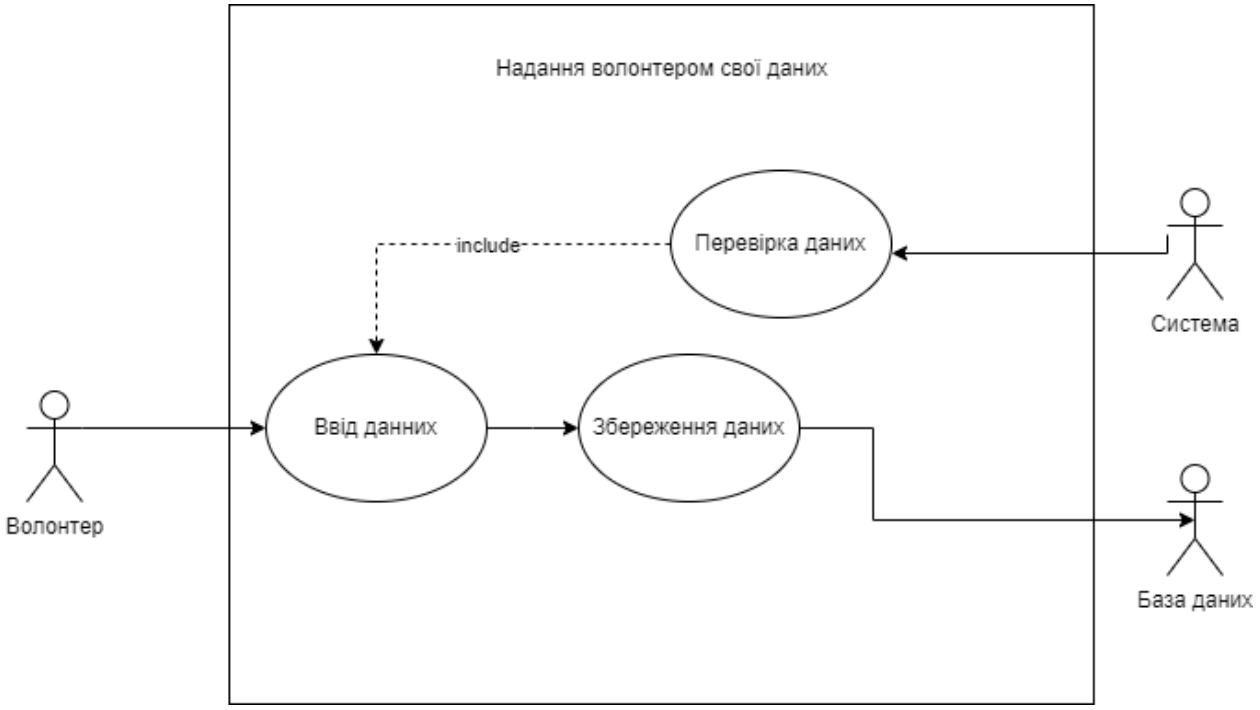


Рисунок 2.4 - Use Case діаграма опису процесу надання волонтером свої даних

- Діаграма отримання біженцем необхідної для нього інформації (рис. 2.5)

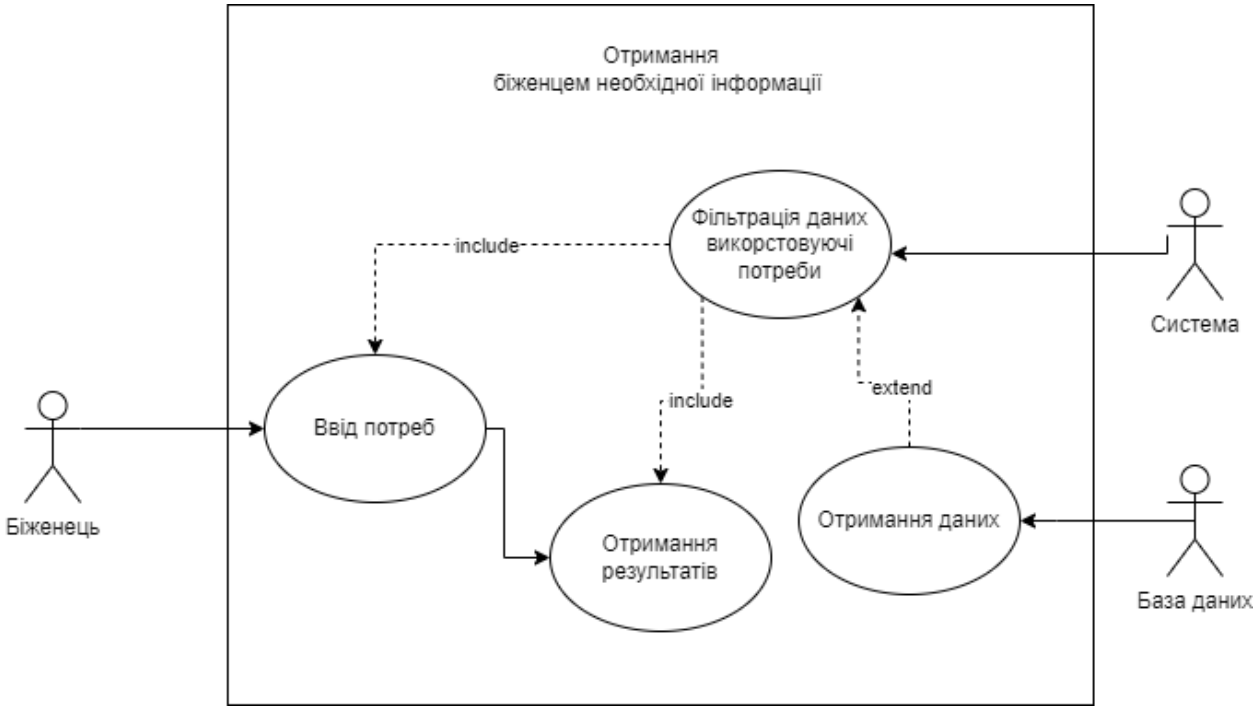


Рисунок 2.5 - Use Case діаграма опису процесу отримання біженцем необхідної для нього інформації

Створені діаграми надали можливість пришвидшити та полегшити розробку та розуміння необхідного веб-додатку, разом із чим покращити розуміння необхідних бізнес процесів поза додатком, які неможливо програмно реалізувати, проте вони необхідні.

Одним із таких процесів є перевірка волонтерів під час реєстрації, в веб-додатку вона буде виконана автоматично, проте при реальному використанні необхідно знайти відповідальних за це людей, які перед активацією аккаунту будуть спілкуватися з ними та вирішувати, чи активувати їх аккаунт чи ні.

Окрім цього такі волонтери як лікарі чи психологи, мають також окремими людьми виділятися задля того, щоб при необхідності, приставляти їх до людей які можуть транспортувати біженців, чи зробити для них окремий центр в якому вони зможуть приймати людей, які цього потребують.

Більшість інформації яку будуть надавати волонтери буде відображено біженцям після того як вони виберуть тип допомоги який їх необхідно і дадуть відповідь на питання, які зможуть допомогти вибрати необхідну для них людину. Оскільки людина може мати обмежений доступ до інтернету, обов’язковими буде надання як свого емейлу, так і телефону, щоб людина могла швидко подзвонити до волонтера.

Окрім цього з цих діаграм можливо виділити такі ризики: якщо для певної групи людей буде підходити лише один волонтер, то існує ймовірність того, що йому будуть дзвонити чи писати багато людей одночасно, що може душе сповільнити швидкість отримання необхідної допомоги. На жаль програмно вирішити цю проблему неможливо, єдиним що може допомогти може бути надати можливість волонтеру отримати допомогу когось з адміністрації сайту, хто буде допомагати йому на цей час. Для цього необхідно вказати контакти адміністрації на сайті.

2.3 Аналіз методологій програмування

Після створення структури роботи веб-додатку необхідно ознайомитися та вибрати найбільш підходящу методологію програмування для розробки даного веб-додатку. На жаль для інтерфейсу у веб-додатках можливе використання лише JavaScript який є об’єктно орієнтованою, скриптовою мовою програмування, тому вибір і ознайомлення буде виконано лише для серверної частини додатку.

Спочатку було виділено найбільш популярні методології, а саме об’єктно-орієнтоване програмування , функціональна та декларативна, з якими буде ознайомлено далі. Методологія програмування – це сукупність рішень та методів, які будуть застосовані під час розробки продукту, що об’єднує однаковий характер, та які можуть забезпечити найкращу швидкість та ефективність розробки.

Першим буде розглянуто **об’єктно орієнтоване програмування** (**ООП**) – це методологія яка застосовує концепт розбиття задачі на окремі сутності(об’єкти) та логіку їх взаємодії між собою(інтерфейси). Об’єкти зазвичай створюються за допомогою класів, які містять набір даних, та методи з якими є можливість взаємодіяти. Плюсом цієї методології є можливість чітко структурувати код, що дозволяє спростити розуміння частини логіки та її в цілому. Проте через це існує також і проблема пов’язана з тим, що навіть в найбільш простих задачах необхідно мати цю саму логіку, що може ускладнити логіку в цілому.

Ця методологія є однією з найпоширеніших на цей час і дозволить чітко структурувати код проекту, що дозволить в майбутньому розширювати веб-додаток без якихось проблем, не зважаючи на складність його логіки.

Після цього було розглянуто **функціональне програмування** – це концепція програмування, яка передбачає собою поділ проблеми на окремі функції, які відповідають за певну логіку яку необхідно виконати. На відміну від об’єктно орієнтованого програмування, ця парадигма не обмежена використанням класів, що може значно розширити та полегшити працю з даними, адже ти не прив’язаний до певних класів. Разом усі ці функції утворюють програму, яка і вирішує дану проблему.

Плюсами цієї методології є те, що розробка коду значно спрощується, не має потреби в визначенні класів та їх логіки взаємодії між собою, що несе у собі загальне полегшення логіки. Головним мінусом є погане розширення, адже чим більша логіка, тим складніше її зрозуміти, адже на відміну від ООП, у тебе немає можливості чітко все структурувати і відрізати можливість доступу до якихось функцій або якихось конкретних об’єктів, що може призвести до помилок і збільшенню часу розробки.

І останнім було ознайомлено з **декларативною методологією** програмування, адже саме вона є основою HTML сторінок, які разом із JavaScript і будуть становити інтерфейс додатку. Ця парадигма розуміє під собою описання результату який необхідно отримати, а не кроки його виконання(на відміну від ООП). Така мова використовується в HTML сторінках, де описується лише те, що на сторінці має бути, а не кроки цього відображення.

Ця парадигма є відмінною від емпіричної, яка навпаки потребує опису цього алгоритму, що значно спрощує написання коду.

Ознайомившись із цими методологіями, було зроблено висновок, що краще за все буде використати мову в основі якої є об’єктно орієнтоване програмування, адже саме воно може допомогти структурувати код, що дозволяє розширювати його в майбутньому.

2.4 Аналіз мов програмування та необхідних бібліотек

Для розробки серверної частини необхідно обрати мову програмування та відповідний фреймворк, щоб спростити розробку веб-додатку. Окрім цього необхідно виконати аналіз бібліотек які знадобляться при створенні продукту.

Спочатку було проаналізовано дві мови програмування які використовують(або можуть використовувати) об’єктно орієнтоване програмування. Для цього було обрано дві найбільш популярні мови які використовуються для розробки серверної частини веб-додатків, а саме Python та Java . [9]

Python – це високорівнева мова програмування, яка є популярною в багатьох галузях, зокрема в веб-розробці. Мова Python була розроблена і вперше випущена в 1980 році Гвідо ван Россумом. Ця мова є динамічно строго типізованою, та об’єктно орієнтованою. Особливістю Python є його синтаксис – блоки коду виділяються за допомогою відступів, що дозволяє збільшити його зрозумілість, хоча, на ділі, це не дуже сильно впливає на розуміння коду.

Python є популярним у веб-розробці через велику кількість фактів, головними з яких є:

- **Веб фреймворки** – Python є однією з найпопулярніших мов для написання фреймворків, що спрощує написання коду та допомагає швидше і якісніше виконати поставлену задачу. Найбільш популярними фреймворками для веб-розробки є Django, Flask та Pyramid. Використання цих фреймворків допоможе спростити розробку веб-застосунку, та дозволить зробити більш якісніший продукт. Ці фреймворки зазвичай мають у собі встроєні інструменти для керування базами даних, адміністрування, авторизації, тощо.

- **Універсальність** – Python є універсальною мовою програмування, яку використовують в багатьох галузях, що може спростити розробку задач які напряму не зв’язані з серверною частиною.
- **Інтеграція з API** – Python має велику кількість ресурсів, які можуть допомогти при роботі з API інших сервісів та ресурсів, що може допомогти при інтеграції.
- **Масштабуємість** – Одним з найбільших плюсів за використання цієї мови є те, що вона дозволяє швидко розробляти як малі проекти, так і дуже великі, адже ця мова гарно масштабується. Ця особливість дозволяє дуже пришвидшити розробку при використанні мікро сервісної архітектури, де за кожну задачу відповідає окремий малий сервіс.
- **Документація та спільнота** – однією з найбільш важливих речей при роботі з мовою програмування є її документація. Якщо ця документація відсутня або погано структурована чи подана, швидкість знаходження потрібної інформації дуже сповільнюється, що впливає на загальний час розробки. Python має дуже гарну документацію, яка справляється з цими проблемами. Окрім цього, якщо не вдається щось знайти, існує велика кількість спільнот, в яких люди будуть мати змогу допомогти чи відповісти на якісь важливі питання.[10]

На даний момент часу, Python залишається однією із найпопулярніших мов взагалі, та при розробці веб-додатків. Ця мова є доволі простою та в той же час дуже швидкою, що може пришвидшити роботу веб-додатку.[11]

Після аналізу Python, було виконано аналогічний аналіз для Java. Java є строго типізованою, та об’єктно орієнтованою мовою програмування, яка вперше з’явилася в 1995 році. Основними особливостями цієї мови окрім цього є збірник сміття, чутливість до реєстру та JVM, яка дозволяє використовувати цю мову на різних платформах. [12]

На відміну від Python, Java не є широко розповсюдженою, та є помітно повільнішою у менших проектах, проте є все одно розповсюдженою при розробці веб-застосунків, причинами цього є:

- **Фреймворки** – аналогічно з Python, на Java написано декілька фреймворків які використовуються у веб-розробці, а саме - Spring, JavaServer Faces (JSF) та Struts. Нажаль інші альтернативи є значно гіршими, та з трьох найбільш популярних, найбільш повний та зручний функціонал надає Spring, що дуже сильно звужує можливості вибору фреймворку.
- **Бібліотеки та збирачі проектів** – Java має дуже зручну можливість для організації проектів, а саме – збирачі проектів. Одними з найбільш популярними є Maven, Groove та Ant. Ці збирачі дозволяють швидко налаштувати проект, зберігаючи інформацію у файлі конфігурації, що дозволяє швидко його змінювати при необхідності. Однією з найбільш важливих особливостей цього файлу є можливість підключати бібліотеки, які спрощують процес розробки та мають широкий спектр застосунку.
- **Крос-платформеність** – найбільшою особливістю Java є можливість використовувати розроблений код на різних платформах без його зміни. Це забезпечує його гнучкість та спрощує розробку та використання.

- **Масштабування –** незважаючи на те, що Java має дуже повільну швидкість в маленьких проектах, її потенціал розкривається в більш великих проектах, що дозволяє розробляти такі проекти без затримок у часі. Окрім цього синтаксис Java дозволяє створювати масштабуючі проекти, які буде легше зрозуміти та розширити після цього.

- **Безпека** - Java забезпечує високий рівень безпеки веб-застосунків завдяки безлічі вбудованих механізмів захисту. Це важливо для додатків, які обробляють чутливі дані та фінансові операції.

- **Велика спільнота та багата екосистема** - Java має велику спільноту розробників та багату екосистему бібліотек, які полегшують створення веб-додатків. Це включає бібліотеки для роботи з базами даних, створення інтерфейсів і взаємодії із зовнішніми сервісами.

- **Java Virtual Machine (JVM)** - Java програми виконуються на JVM, що дозволяє абстрагуватися від апаратної платформи. Це забезпечує портабельність коду та спрощує оновлення та підтримку додатків.[13]

Зробивши аналіз двох мов програмування, я побачила що відносно мого проекту обидві мови мають як переваги так і недоліки. Мною було обрано мову Java через її масштабованість, та крос-платформеність. Після цього було проведено аналіз необхідних фреймворків та бібліотек.

Для серверної частини разом з Java було використано фреймворк Spring разом з бібліотекою Lombok. Для фронтенд частини буде використано JavaScript разом з фреймворком React та бібліотекою MaterialUI.[14]

Наразі комбінація Spring та React є дуже популярною і дозволяє швидко створювати потрібні веб-додатки без затрат часу. Обидва ці фреймворки містять усі необхідні інструменти для розробки.

Spring - це популярний фреймворк для Java, який спрощує розробку веб-застосунків і забезпечує безліч корисних функцій. Ось чому Spring став першим вибором для багатьох Java-розробників:

- Інверсія керування залежностями та впровадження (IoC/DI): Spring застосовує концепції IoC та DI, що робить код більш модульним та керованим. Це забезпечує легкість внесення змін та тестування. Ці концепції позначають те, що робота з методами виконується за допомогою інтерфейсів, якими в свій час керує сам фреймворк.
- Spring Boot: Spring Boot спрощує створення та налаштування веб-додатків, надаючи вбудовані сервери програм та автоматичну конфігурацію. Він дозволяє швидко розвернути необхідний додаток не витрачаючи зайвий час на його початкову конфігурацію та підключення необхідних залежностей.
- Безпека: Spring Security забезпечує потужний набір інструментів для забезпечення безпеки програми. В комбінацію з Java це дозволяє досягти високого рівня безпеки.[15]

React - це JavaScript-бібліотека для створення інтерфейсів користувача, розроблена Facebook. Вона стала однією з найпопулярніших бібліотек для розробки фронтенду, і ось чому:

- Компонентний підхід: React будує інтерфейси навколо компонентів, що робить код більш зрозумілим та перевикористовуваним. Компоненти – це невеликі, незалежні блоки, які легко підтримувати. Такий підхід спрощує розробку через те, що розробка з використанням HTML тегів є незрозумілою та складною, напроти ж компоненти є більш інтуїтивно зрозумілим підходом.

- Віртуальний DOM: React використовує віртуальний DOM для ефективного оновлення інтерфейсу користувача. Це дозволяє збільшити продуктивність програми. Окрім цього додаток з використанням React є односторінковим, що також покращує його швидкість
- Екосистема: Існує безліч бібліотек та інструментів, пов'язаних з React, таких як Redux для керування станом та React Router для навігації. Велика кількість бібліотек може спростити та пришвидшити розробку веб-додатку, окрім цього спроститься необхідність оновлювати код, адже цим буде займатися розробник цієї бібліотеки.
- Співтовариство: React має активну спільноту розробників, яка постійно створює нові інструменти та ресурси для навчання. Ці спільноти можуть допомогти у вирішенні можливих проблем.[16]

Окрім цього необхідно виділити дві бібліотеки, а саме Lombok та MaterialUI. Це дві найголовніші бібліотеки які зможуть значно пришвидшити розробку коду.

Lombok – це бібліотека на Java, яка зменшує кількість коду потрібного для коректної роботи програми. Основою цієї бібліотеки є анотації, які дозволяють замінити зазвичай необхідні речі, такі як конструктори, гетери сетери та багато іншого. Ця бібліотека автоматично генерує потрібні шматки кода після їх компіляції де вони потім запускаються.[17]

MaterialUI – це бібліотека яка буде використана у парі з React. Вона має у собі шаблони багатьох необхідних для веб-додатку елементів(наприклад кнопок, форм, тощо). Ця бібліотека дозволить скоротити час для створення цих елементів. Перевагою цієї бібліотеки є її гнучкість, усі ці шаблонні елементи можуть бути змінені залежно від потреб.

Spring та React надають надійний та потужний стек технологій для веб-розробки, який дозволяє створювати сучасні та високопродуктивні програми.

Їхня комбінація відкриває нові можливості для розробників, дозволяючи їм легко керувати бізнес-логікою та створювати інтерактивні інтерфейси, що робить їх ідеальним поєднанням для розробки сучасних веб-додатків.[18]

У випадку проекту даної роботи ці дві технології дозволять побудувати якісний та гнучкий додаток, який зможе бути масштабованим до більших розмірів у майбутньому. Окрім цього при необхідності буде можливість додавати роботу з персональними даними, адже ці інструменти дозволяють безпечно з ними працювати.

3 ПРАКТИЧА ЧАСТИНА

3.1 Реалізація бази даних

Перед створенням програмної реалізація необхідно визначитися з архітектурою бази даних, та серверної частини. Ці дані необхідні нам при розробці і для цього було створено діаграму бази даних та розібрано Rest архітектуру.

Для початку було створено діаграму з описом бази даних в якій описала потрібні таблиці разом з полями. Оскільки веб-додаток є демонстрацією продукту яку можливо доповнювати, мною було створено дві таблиці в яких можливо зберігати дані щодо персональної інформації волонтеру та опису їх об’яв(рисунок 3.1).

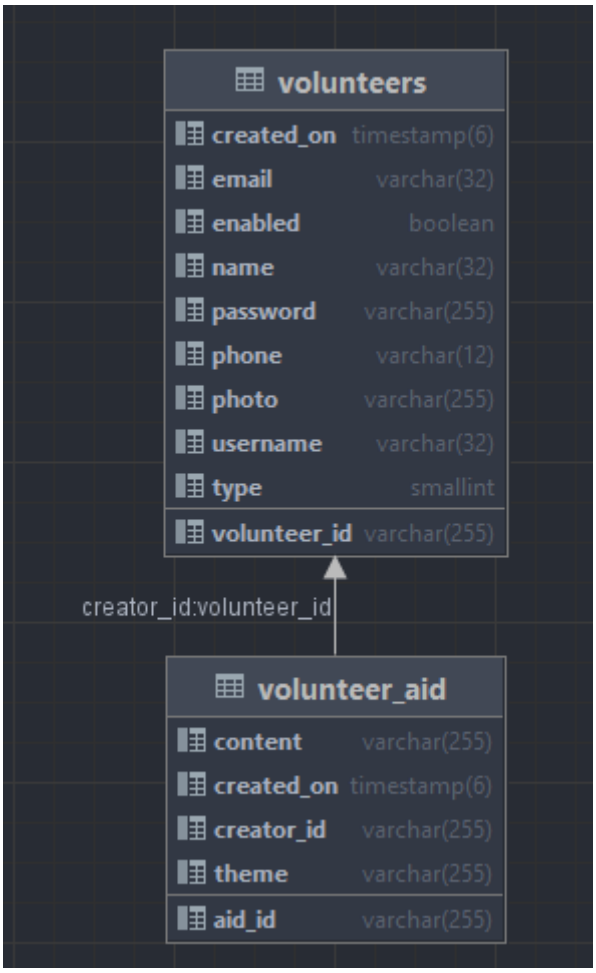


Рисунок 3.1- Архітектура бази даних

Як видно зі схеми, першою таблицею є volunteers в якій зберігаються ключові дані щодо волонтеру, а саме його імейл, ім’я та прізвище, пароль, телефон, фото та тип. Ця таблиця буде використовуватися при реєстрації волонтеру, та для отримання його контактної інформації

Другою таблицею є volunteer_aid в яку зберігається інформація щодо об’яви волонтера. В цій таблиці є поля з контентом об’яви, його темою та айді волонтеру який його створив.

Обидві таблиці пов’язані зв’язком один до багатьох (у одного волонтера може бути декілька об’яв) та для доступу до волонтерів з об’яв використовується поле creator_id. Ця структура є приблизною і допоможе при створенні цих баз даних. Мною було обрано PostgreSql як базу даних, адже в ній є можливість зберігати структуровані дані, та за допомогою індексації швидко робити сортування для виводу даних.

Після цього було виконано аналіз Rest архітектури, яку було обрано для серверної частини додатку. Ця архітектура є найпопулярнішою на даний момент і є найбільш розвиненою відносно більш старих архітектур. Вона була обрана мною через її гнучкість, простоту та масштабуємість, що ідеально підходить для необхідного веб-додатку.

REST (Representational State Transfer) – це структурний стиль для проектування веб-додатків. Він описує, як ресурси (як правило, представлені у вигляді URL) повинні бути доступні та взаємодіяти один з одним. Основними перевагами цього стилю є його простота, масштабуємість, стійкість до збоїв та підтримка кешування.

Основними концепціями REST архітектури є:

- **Ресурси** - сутність, до якої можна звертатися по URL.
- **HTTP Методи**- REST використовує стандартні методи HTTP для виконання операцій над ресурсами. Найчастіше використовувані методи - це GET, POST, PUT і DELETE.

- **Подання** - Ресурси можуть мати різні уявлення, наприклад, JSON, XML або HTML.
- **Стан** - RESTful взаємодія є безстановою, що означає, кожен запит від клієнта до серверу повинен містити всю необхідну інформацію до виконання запиту.
- **Уніфікація інтерфейсу** - REST використовує єдиний та зручний інтерфейс, що спрощує взаємодію та розуміння архітектури.

3.2 Програмна реалізація та тестування серверної частини

Перед початком створення коду для серверної частини було обрано необхідний шаблон проектування, а саме – MVC(Model View Controller).

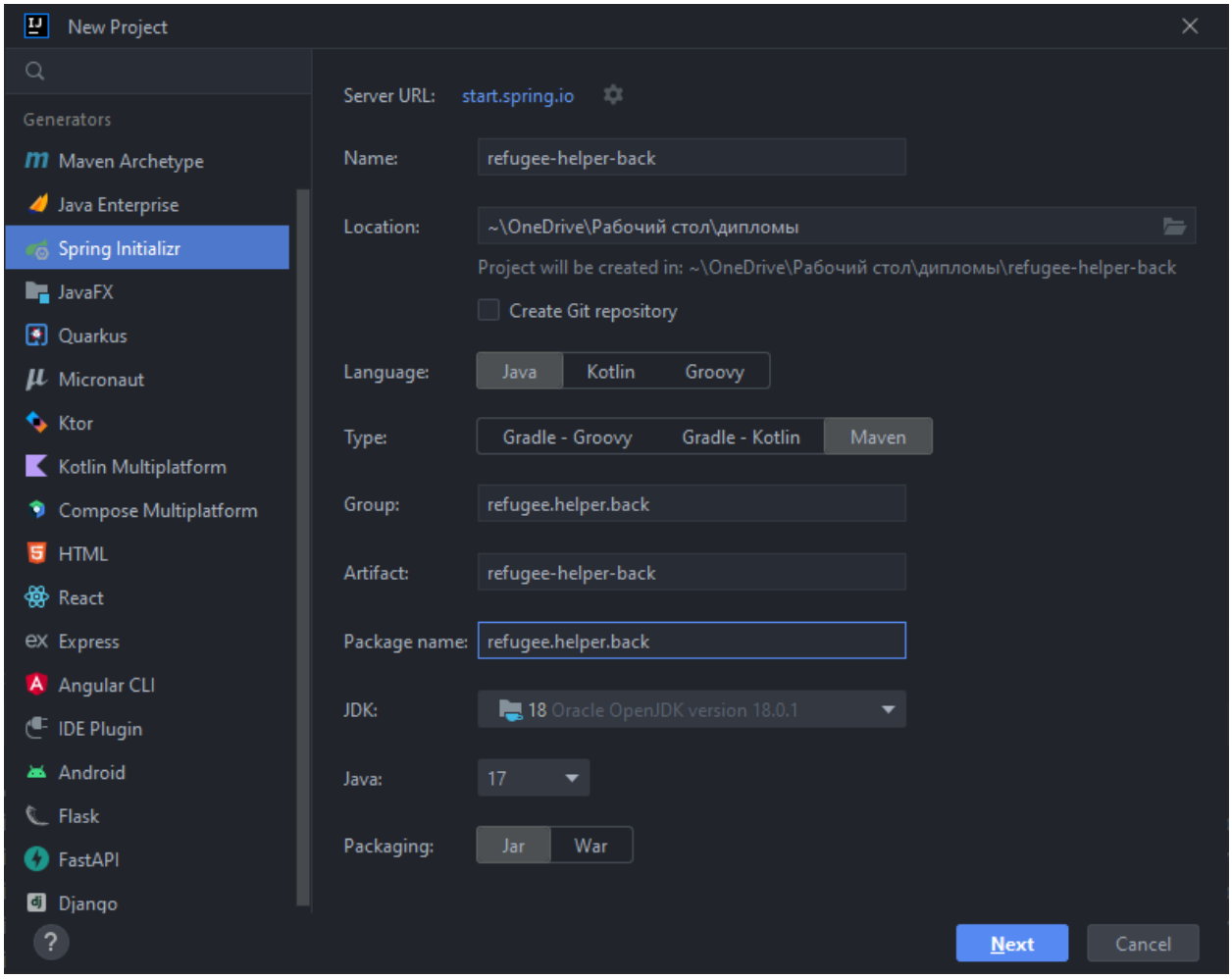
Цей шаблон дозволяє чітко структурувати код серверної частини розділивши його на три основні частини-

- Model – моделі з якими працює додаток, вони містять інформацію щодо даних які мають оброблюватися серверною частиною разом з їх структурою.
- View – необхідні дані для юзеру, в даному веб-додатку цим буде виступати JavaScript з React, які будуть виводити потрібні дані.
- Controller – логічна частина додатку, яка буде отримувати дані та оперувати ними, передаючи їх до View.

Після того, яку було вирішено усі архітектурні питання, було створено відповідний проект. Для його створення було використано IntelijIdea , яка має

підтримку створення Spring проектів(рисунок 3.2). Альтернативно є можливість створення проекту на відповідному сайті.

Рисунок 3.2 – створення проекту



Ця можливість в IntelijIdea лише створює пустий проект, який потрібно все одно конфігурувати. Для цього в Maven проектах використовується файл pom.xml в який було додано усю потрібну конфігурацію, разом з необхідними бібліотеками та плагінами(рисунки 3.3, 3.4 та 3.5 відповідно).

```
<groupId>refugee.helper</groupId>
<artifactId>refugee-helper-back</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<properties>
  <org.mapstruct.version>1.5.3.Final</org.mapstruct.version>
  <org.projectlombok.version>1.18.24</org.projectlombok.version>
  <org.commons-io.version>2.11.0</org.commons-io.version>
  <maven-compiler.plugin.version>3.8.1</maven-compiler.plugin.version>
  <org.springdoc.version>1.7.0</org.springdoc.version>
  <springboot.version>3.0.5</springboot.version>
  <java.version>17</java.version>
  <maven.compiler.source>${java.version}</maven.compiler.source>
  <maven.compiler.target>${java.version}</maven.compiler.target>
  <org.maven-compiler-plugin.version>3.8.1</org.maven-compiler-plugin.version>
  <spring-boot-maven-plugin.version>2.7.4</spring-boot-maven-plugin.version>
  <springdoc-openapi-starter-webmvc-ui.version>2.1.0</springdoc-openapi-starter-webmvc-ui.version>
</properties>

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-dependencies</artifactId>
      <version>${springboot.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Рисунок 3.3 – Конфігурування проекту

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-configuration-processor</artifactId>
    <optional>true</optional>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>${springdoc-openapi-starter-webmvc-ui.version}</version>
  </dependency>
```

Рисунок 3.4 – Налаштування потрібних бібліотек

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <version>${spring-boot-maven-plugin.version}</version>
      <executions>
        <execution>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </exclude>
        </excludes>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Рисунок 3.5 – Налаштування потрібних плагінів

Налаштування проекту дає нам змогу перейти до розробки потрібного коду. Спочатку було відокремлено три основні частини проекту – частина яка відповідає за роботу з базами даних - database, частина яка оброблює дані - service та частина яка отримує та відображає потрібні дані – web. Окремо було створено спеціальний пакет в якому зберігаються файлі конфігурації додатку, бд та інше - config(рисунок 3.6). Ця структура дозволяє чітко відокремити усю логіку відносно обраного шаблону проектування(MVC) та дозволить в майбутньому розширювати веб-додаток при необхідності, адже кожна частина буде працювати не спираючись на реалізацію інших класів.

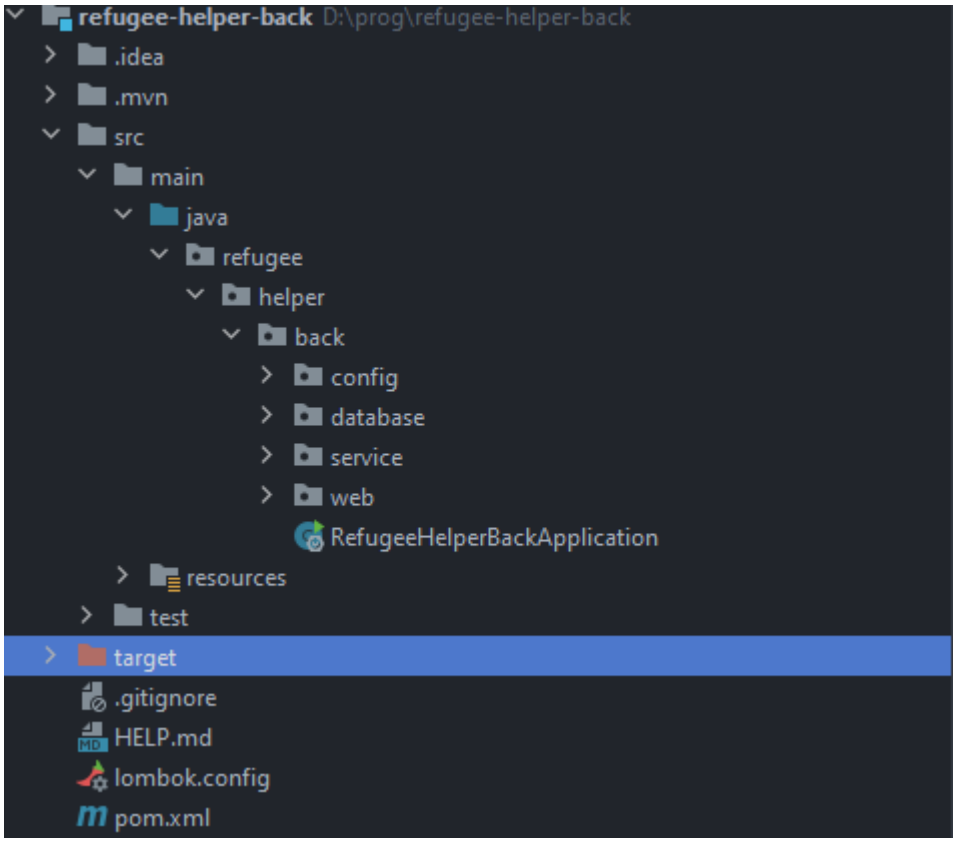


Рисунок 3.6 – Повна структура проекту

Реалізації логіки проекту потребує насамперед можливість роботи з базами даних, тому я локально створила пусту базу даних, сутності які відображають структуру таблиць та репозиторії, які відповідають за роботу з базою даних.

Для створення і роботою з базами даних я використала спеціальну бібліотеку - spring-boot-starter-data-jpa основною перевагою якої є те, що вона дозволяє програмно змінювати структуру бази даних лише змінивши сутність. Окрім цього вона дозволяє автоматично генерувати запити до бази даних та повертає дані в потрібній мені структурі. Для її роботи необхідно в файлі конфігурації лише вказати дані авторизації до бази даних(назва бази, назва юзеру та його пароль). Зазвичай такі дані зберігаються у змінних середи, що забезпечує більш безпечніший підхід, адже вони не будуть знаходитися в коді, до якого можуть отримати доступ зловмисники, проте оскільки цей додаток знаходиться локально на моєму комп’ютері, я зберегла їх в коді.

Приклади реалізації сутностей та репозиторіїв можливо побачити на рисунках 3.7 та 3.8 відповідно.

```java
@Data
@Table(name = "volunteers")
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class VolunteerEntity {
  @Id
  @GeneratedValue(generator = "uuid")
  @GenericGenerator(name = "uuid", strategy = "uuid2")
  @Column(name = "volunteer_id")
  private String id;

  @Column(name = "email", length = 32, unique = true)
  private String email;

  @Column(name = "name", length = 32)
  private String name;

  @Column(name = "username", length = 32)
  private String surname;

  @Column(name = "phone", length = 12)
  private String phone;
```

Рисунок 3.7- Реалізація сутності волонтеру

```java
2 usages
@Repository
public interface VolunteerRepository extends JpaRepository<VolunteerEntity, String> {
  2 usages
  Optional<VolunteerEntity> getUserEntityByEmail(String email);
}
```

Рисунок 3.8- Реалізація репозиторію для сутності волонтера

Мною було створено дві сутності та два репозиторія, для роботи з таблицями volunteer та volunteers_aid відповідно. Після чого була отримана можливість перейти до створення контролерів. Перед цим було створено декілька файлів конфігурації в яких є файли конфігурації безпеки(в яких контролерах потрібна авторизація, а в яких ні), сконфігурований свагер який був використаний для перевірки та налаштування бін для шифрування пароля.

Перед створенням контролерів необхідно визначитися зі шляхом до них та їх описом. Ці контролери потім будуть доповнені спеціальними ендопінтами в яких буде використана логіка з сервіс частини.

Для роботи з інтерфейсом було створено три контролера –

- RefugeeController – контролер усі ендпоінти якого починаються з /a/rest/refugee, та який містить логіку відносну до біженців.
- VolunteerAccountController – контролер усі ендпоінти якого починаються з /a/rest/volunteer/account, та який містить логіку відносну до акаунтів волонтерів.
- VolunteerAidController – контролер усі ендпоінти якого починаються з /a/rest/volunteer/aid, та який містить логіку відносну до акаунтів волонтерів.

З цих контролерів усі ендпоінти VolunteerAidController та ендпоінт з отриманням інформації юзера захищені за допомогою авторизації з використанням токену. Перед кожним запитом перевіряється чи присутній він і чи є валідним. Якщо так то повертається потрібна інформація. Більш детальніше про цей токен буде розібрано далі у роботі.

Для роботи з даними в контролерах використовуються спеціально структуровані моделі – DTO, в яких указана необхідна структура запиту та правила валідації. Після чого ці данні зазвичай перетворюють до моделей які використовуються у серверах і відповідно до них передаються. Для цього використовуються так звані «мапери», це спеціальний код, який може перетворити одну структуру даних до іншої. В даному проекті було використано бібліотеку Mapstruct, яка дозволяє автоматизувати цей процесс. Приклади реалізації контролерів та DTO наведено на рисунках 3.9 та 3.10 відповідно.

```java
@AllArgsConstructor
@RestController
@RequestMapping("/a/rest/refugee")
public class RefugeeController {
    2 usages
    private final AidService aidService;

    @GetMapping("/aid")
    public ResponseEntity<List<VolunteerAidResponseDto>> getFilteredVolunteerAid(
            @RequestParam(required = false) String city,
            @RequestParam(required = false) Type type,
            @RequestParam(defaultValue = "0") int page,
            @RequestParam(defaultValue = "5") int limit) {
        return ResponseEntity.ok(
                WebLayerMapper.I.volunteerAidModelListToResponseDto(
                        aidService.getFilteredVolunteerAid(
                                city.equals("null") ? null : city, type, page, limit)));
    }

    @GetMapping("/aid/{aidId}")
    public ResponseEntity<VolunteerAidResponseDto> getAidById(
            @PathVariable(value = "aidId") String aidId) {
        return ResponseEntity.ok(
                WebLayerMapper.I.volunteerAidModelToResponseDto(aidService.getVolunteerAidById(aidId)));
    }
}
```

Рисунок 3.9 – Контроллер RefugeeController

```java
@NoArgsConstructor
@AllArgsConstructor
@Data
@Builder
public class VolunteerAidRequestDto {
    @NotEmpty(message = "Тема не може бути порожньою")
    private String theme;

    @NotEmpty(message = "Контент не може бути порожнім")
    private String content;

    @NotEmpty(message = "Город в якому ви можете надати свої послуги не може бути порожнім")
    private String city;
}
```

Рисунок 3.10 – Приклад реалізації DTO

Реалізація контролерів дозволяє перейти до створення сервісів – головного «ланцюжка» який буде поєднувати контролери разом із базою даних та обробляти дані.

Найголовнішими сервісами які потрібно окремо розглянути це JwtService та UserService. Ці два сервіси відповідають за безпеку додатку під час реєстрації та авторизації.

UserService – це сервіс який відповідає за обробку даних юзерів веб-додатку, в ньому створена можливість отримувати дані юзера, створювати(реєструвати його), перевіряти пароль та інше. В ньому використовується спеціальна модель юзера, яка наслідується від системної моделі, що дозволяє віддавати його до контексту, для аутентифікації. Окрім цього в цьому сервісі використовується створена раніше конфігурація з біном хешування паролю, задля безпеки користувача в базі даних зберігається лише його хешована версія, що дозволить уникнути небезпеки у разі доступу до баз даних зловмисником. Сам сервіс лише кешує цей пароль при реєстрації, та потім звіряє введені дані при авторизації з цим паролем.

Другим важливим сервісом є JwtService – сервіс в якому реалізована робота з Jwt токеном – спеціальним токеном який передається з фронт частини додатку і використовується для аутентифікації юзера. Цей токен являє собою захешовану строку яка містить емейл юзера. Окрім цього це токен має час його дії, після чого його вже не можливо використовувати для авторизації.

Окрім цих сервісів було створено окремі сервіси для роботи з картинками(збереження та отримання) та сервіс для роботи з об’явами волонтерів. Усі ці сервіси використовують окремі моделі, які при необхідності роботи з базами даних перетворюються на сутності та навпаки. Приклад реалізації сервісу та його моделі наведено на рисунках 3.11 та 3.12 відповідно.

Аналогічно з конфігурацією бази даних, для досягнення більшої безпеки, строку яка використовується для хешування краще зберігати окремо.

```
public class UserServiceImpl implements UserService {
  1 usage
  private final PictureService pictureService;
  2 usages
  private final VolunteerRepository volunteerRepository;
  2 usages
  private final PasswordEncoder passwordEncoder;

  2 usages
  @Override
  public SecurityUserModel getUserByEmail(String email) {
    return ServiceLayerMapper.I.volunteerEntityToSecurityUserModel(
        volunteerRepository
            .getUserEntityByEmail(email)
            .orElseThrow(
                () ->
                    new IncorrectCredentialsException(
                        Map.of( k1: "email", String.format("Юзер з емейлом '%s', не знайдений", email)),
                        "Incorrect email")));
  }

  1 usage
  @Override
  public VolunteerModel registerUser(VolunteerModel volunteerModel, MultipartFile picture) {
    volunteerModel.setPhoto(pictureService.savePicture(picture, volunteerModel.getEmail()));
    volunteerModel.setPassword(passwordEncoder.encode(volunteerModel.getPassword()));
    volunteerModel.setCreatedOn(LocalDateTime.now());
    volunteerModel.setEnabled(false);

    return ServiceLayerMapper.I.volunteerEntityToModel(
        Optional.of(
                volunteerRepository.save(
                    ServiceLayerMapper.I.volunteerModelToEntity(volunteerModel)))
            .orElseThrow(() -> new RuntimeException("Помилка при збереженні юзера")));
  }
```

Рисунок 3.11 – Приклад вигляду сервісу

```
@NoArgsConstructor
@AllArgsConstructor
@Data
@Builder
public class AidModel {
  private String id;
  private VolunteerModel creator;
  private String theme;
  private String city;
  private String content;
  private LocalDateTime createdOn;
}
```

Рисунок 3.12 – приклад вигляду моделі

Результатом розробки стала серверна частина яка може отримувати, обробляти та повертати потрібні данні. Ця частина буде використана разом з інтерфейсом і дозволить юзеру отримувати те, що йому потрібно.

Загальну структуру проекту наведено на рисунку 3.13. Така структура дозволяє в майбутньому розширювати цей проект, що в свою чергу спростить покращення якості даного продукту.

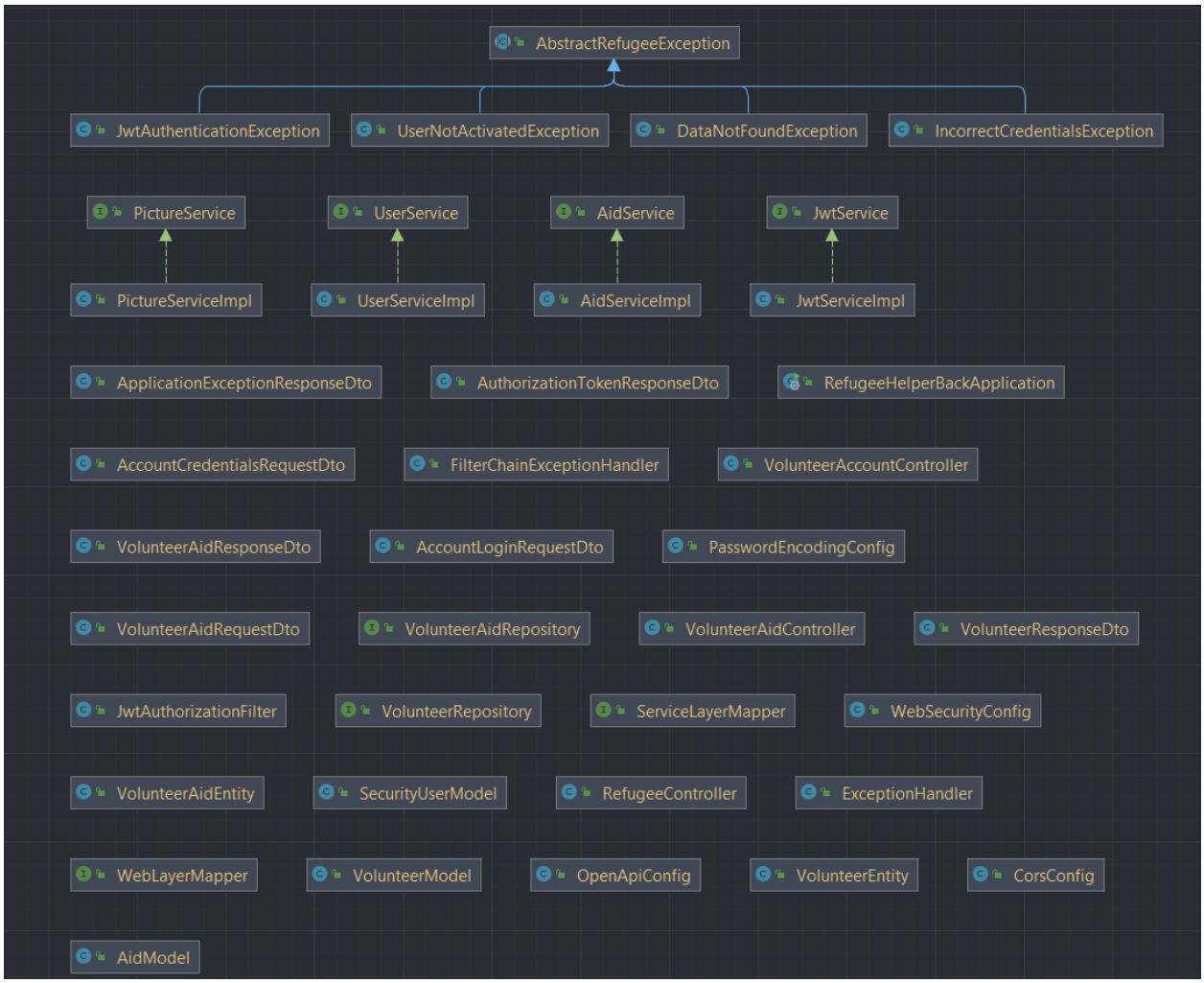


Рисунок 3.13 – Загальна структура проекту

3.3 Програмна реалізація та тестування клієнтської частини

Після створення серверної частини мною було перейдено до розробки графічного інтерфейсу. Для початку його розробки необхідно також створити пустий проект. Для цього необхідно виконати команду npx create-react-app my-app в командній строчці, де замість my-app підставити назву проекту. Для розробки коду було використано Visual Code.

Після створення проекту його аналогічно потрібно конфігурувати. Для цього в package.json необхідно додати бібліотеки з їх версіями(рисунок 3.14), та виконати команду npm install, після чого усі ці бібліотеки підтягнуться до проекту. Після чого можливо перейти до розробки програмної реалізації.

Рисунок 3.14 – Файл package.json

Перше з чим потрібно розпочати роботу – це конфігурування «роутів» - шляху до посилань, перейшовши за яким юзеру отримає потрібну сторінку. Для цього необхідно у файлі App.js задати цю структуру(рисунок 3.15).

```
export default function App() {
  return (
    <div>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<MainPage />} />
          <Route
            path="/volunteer/register"
            element={<VolunteerRegisterPage />}
          />
          <Route path="/volunteer/login" element={<VolunteerLoginPage />} />
          <Route path="/volunteer/main" element={<VolunteerMainPage />} />
          <Route
            path="/volunteer/create/aid"
            element={<VolunteerCreateAidPage />}
          />
          <Route
            path="/refugee/help/filter"
            element={<RefugeeFilteredHelpPage />}
          />
          <Route path="/refugee/help" element={<RefugeeHelpPage />} />
          <Route path="/refugee/aid/:aidId" element={<RefugeeAidPage />} />
          <Route path="*" element={<NoMatch />} />
        </Route>
      </Routes>
    </div>
  );
```

Рисунок 3.15 – Структура файлу App.js

Після чого необхідно створити ці самі сторінки. Для цього спочатку необхідно задати структуру сторінок. Найбільш популярним підходом до цього є розділ сторінки на 3 частини – хедер, футер там сам контент сторінки.

В хедері зазвичай відображається найпотрібніша інформація яку юзеру потрібно бачити завжди, наприклад, кнопку створення об’яв в нашому проекті. Футер використовується для відображення менш потрібної інформації – мапа сайту, контактна інформація, копірайти, тощо. І в самому контенті сторінки відображається потрібна юзеру інформація. Цю структуру було задано у файлі Layout.jsx(рисунок 3.16).

Рисунок 3.16 – Структура сторінки

Після задання цієї структури мною було створено зовнішній вигляд хедеру, в якому було розташовано назву сайту, кнопку для допомоги біженцям, та кнопки виходу з акаунту, яку видно лише якщо волонтер авторизується.

Далі було розроблено 8 окремих сторінок які будуть відображати потрібний юзеру контент –

- MainPage
- RefugeeAidPage
- RefugeeFilteredHelpPage
- RefugeeHelpPage
- VolunteerCreateAidPage
- VolunteerLoginPage
- VolunteerMainPage
- VolunteerRegisterPage

Реалізація цих сторінок має зазвичай подібну структуру, де в jsx файлі задається структура сторінки, а в css файлі – його зовнішній вигляд(Рисунки 3.17 та 3.18 відповідно).

```css
.refugee-aid-main-page {
  min-height: calc(100vh - 111px);
  background-image: url("/public/images/background.avif");
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: cover;
  padding-top: 30px;
  box-sizing: border-box;
}

.aid-card-page {
  padding-bottom: 15px;
  height: 600px !important;
}
```

Рисунок 3.17 - Приклад реалізації css файлу

```jsx
return (
  <div className="refugee-aid-main-page">
    <Container
      className="refugee-filtered-help-main-page-body"
      component="main"
      maxWidth="sm"
    >
      <Card
        className="aid-card-page"
        variant="outlined"
        sx={{ marginLeft: "20px" }}
      >
        <CardContent sx={{ height: "100%" }}>
          <img
            src={aidData.creator.photo}
            alt="img"
            style={{ height: "250px" }}
          />
          <Typography
            sx={{ fontSize: 16 }}
            color="text.secondary"
            gutterBottom
          >
            Волонтер: {aidData.creator.name + " " + aidData.creator.surname}
          </Typography>
          <div style={{ height: "100px" }}>
            <Typography variant="p" component="div" sx={{ fontSize: 16 }}>
              Місто допомоги: {aidData.city}
            </Typography>
            <Typography
              variant="p"
              component="div"
              sx={{
                marginTop: "10px",
                wordWrap: "break-word",
                overflow: "hidden",
                fontSize: "18px",
              }}
            >
              Телефон: {aidData.creator.phone}
            </Typography>
```

Рисунок 3.18 – Приклад реалізації jsx файлу

Ці сторінки були заповнені відповідним контентом. MainPage(рисунок 3.19) була головною сторінкою на яку юзер переходив з початку, з метою найбільшою зрозумілості, на ній відображено лише посилання для біженців, при натискання на яке вони переходили на RefugeeHelpPage(рисунок 3.20). Ця сторінка відображає стислу форму в якій можливо вибрати тип потрібної допомоги та місто в якій вона потребується.

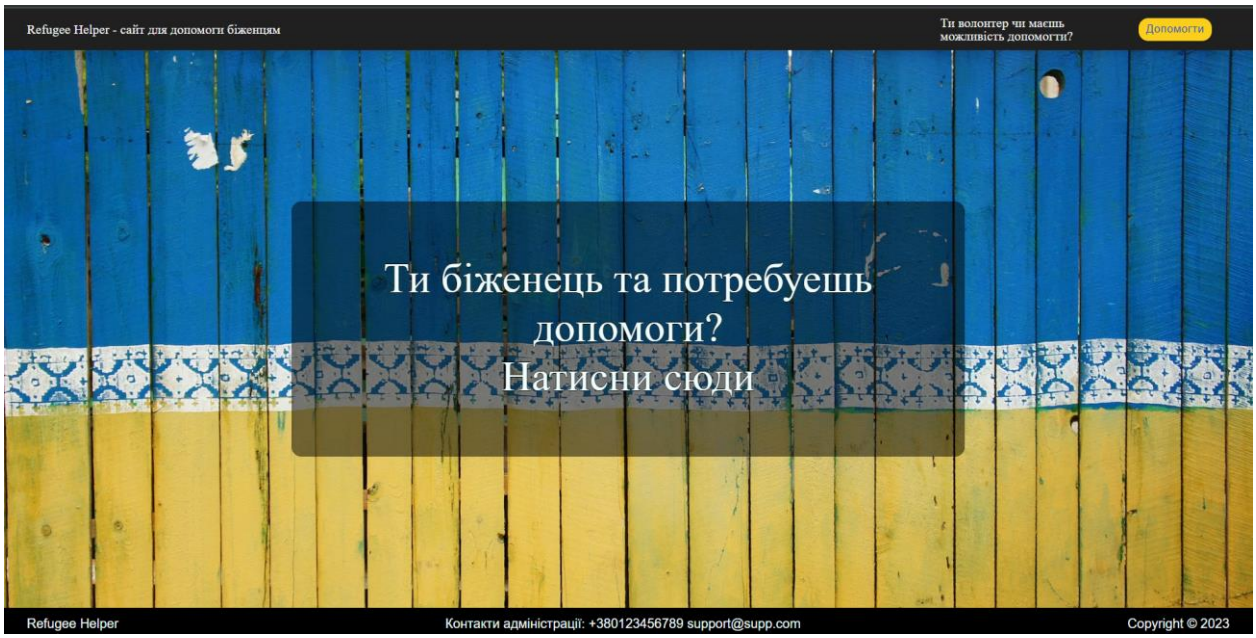


Рисунок 3.19 – MainPage

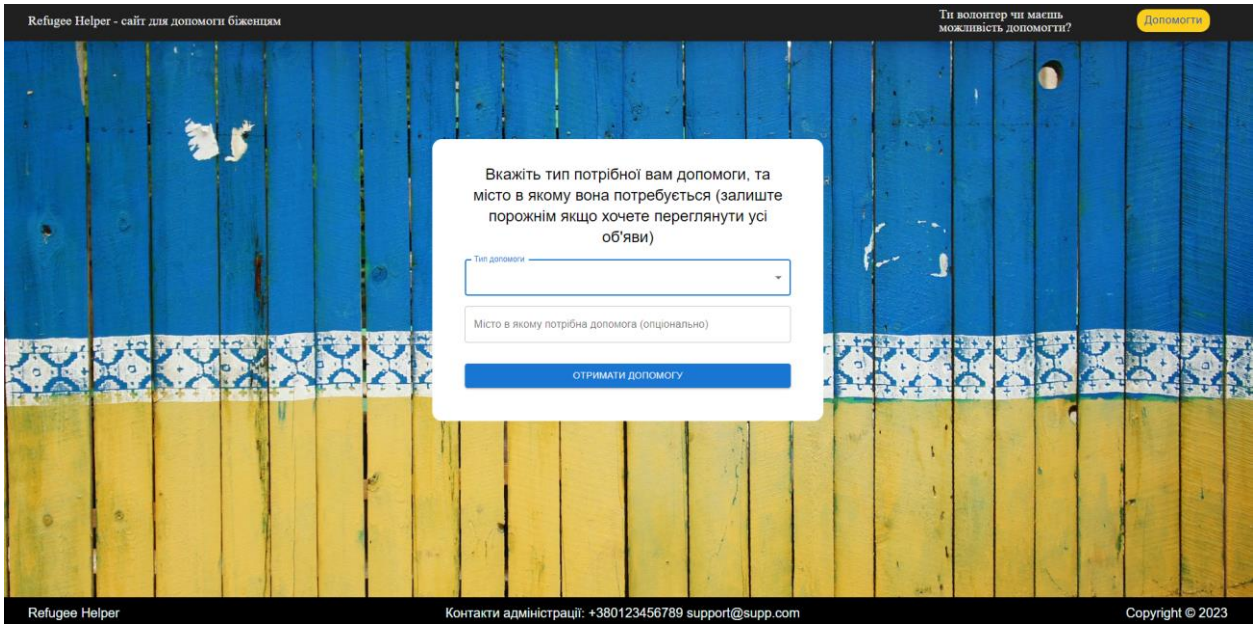


Рисунок 3.20 – RefugeeHelpPage

Введені дані валідуються та в разі коректних даних біженця перекидає на RefugeeFilteredHelpPage(рисунок 3.21) – сторінку на якій відображаються усі створені об’яви за його запитом. На цій сторінці об’яви розділяються на декілька сторінок між якими біженець має можливість переключатися. Після вибору потрібної йому об’яви, він може відкрити її в окремій сторінці – RefugeeAidPage(рисунок 3.22) , де повністю відобразяться повні дані щодо цієї об’яви.

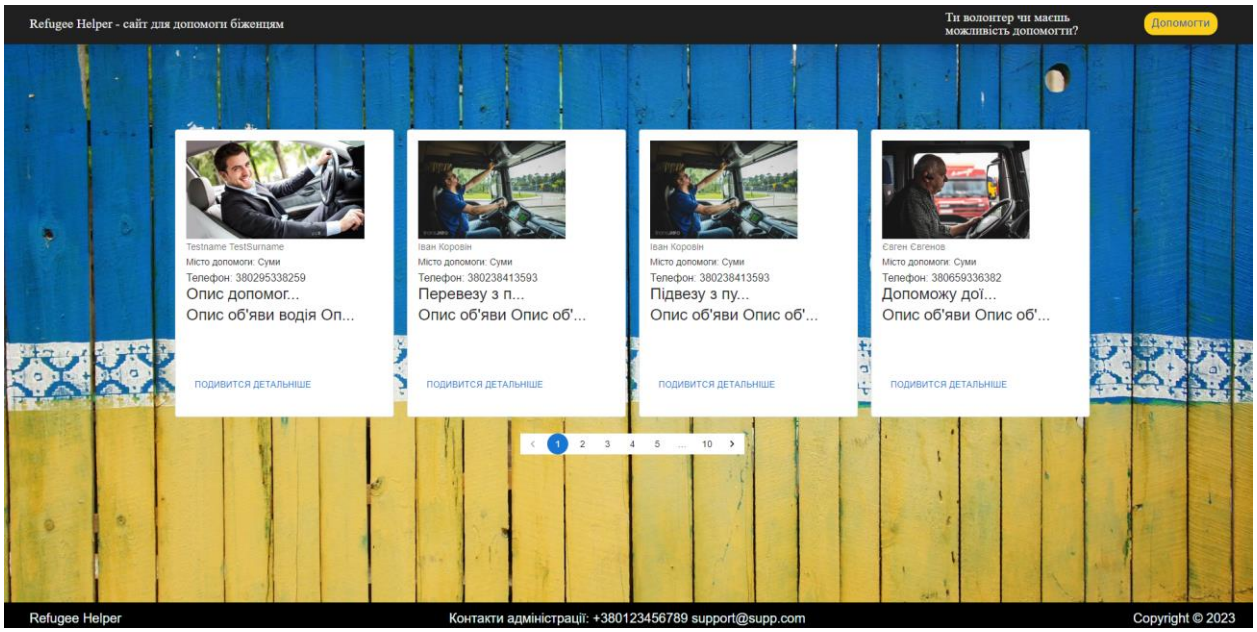


Рисунок 3.21 – RefugeeFilteredHelpPage

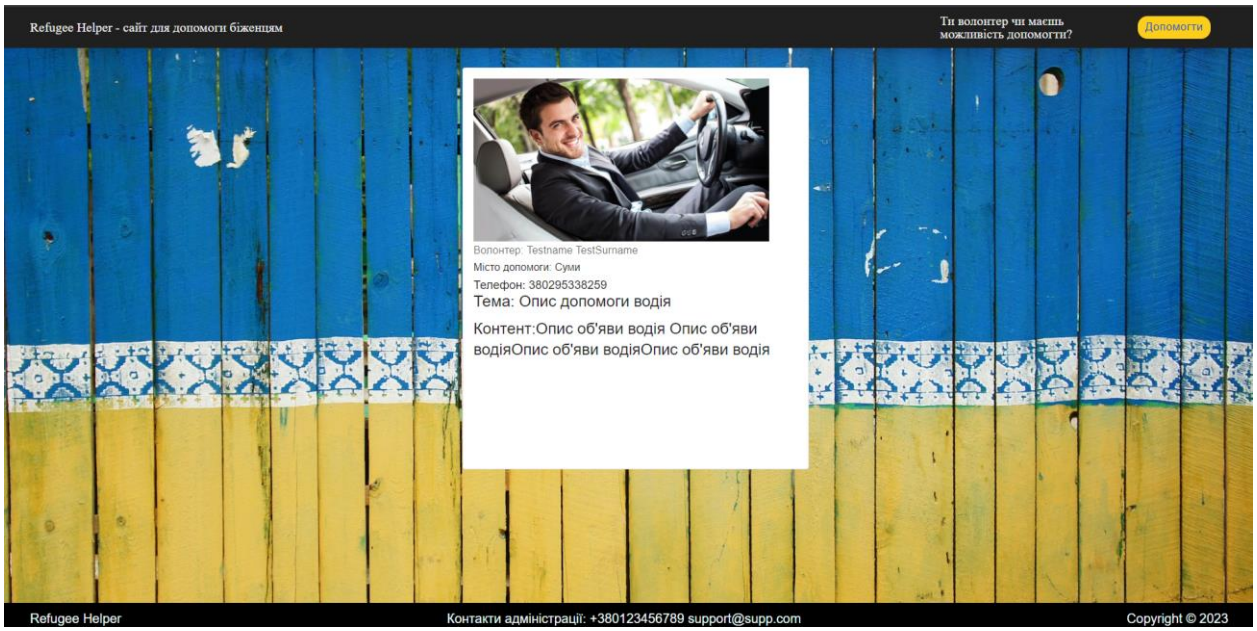


Рисунок 3.22 – RefugeeAidPage

Окрім цього при натисканні на кнопку допомоги зверху, юзер перенаправляється до VolunteerLoginPage(рисунок 3.22) – сторінці де волонтер може зайти до свого акаунту, окрім цього в разі відсутності акаунту волонтер має можливість перейти до VolunteerRegisterPage(рисунок 3.23) де він може ввести свої дані та зареєструвати аккаунт.

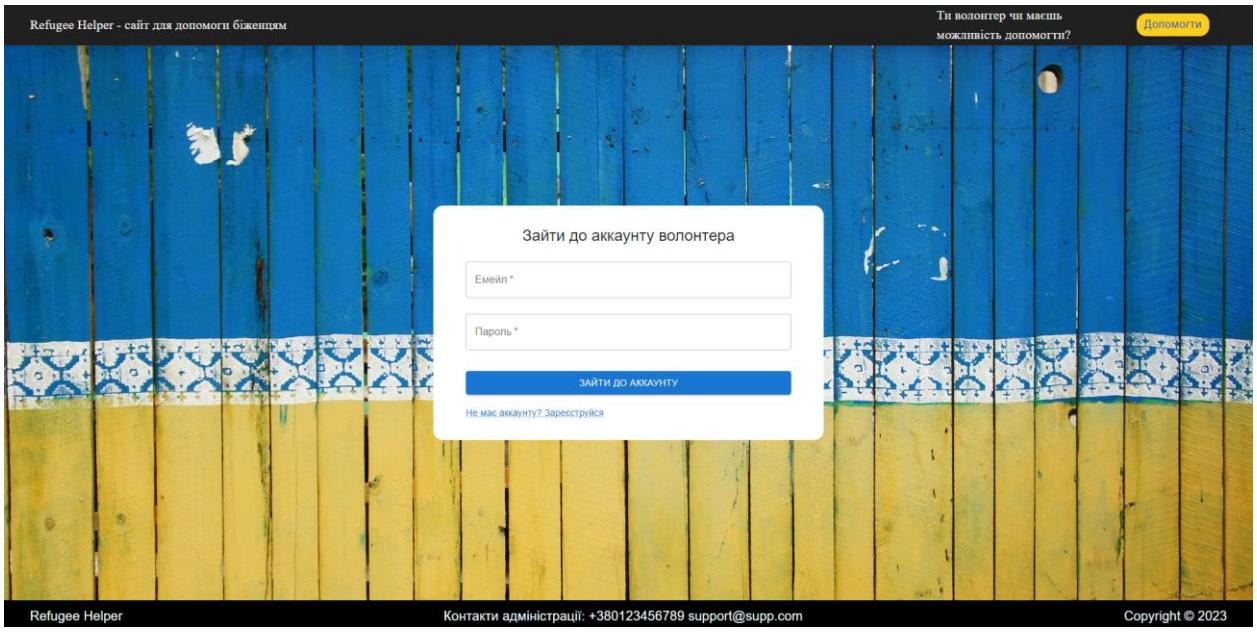


Рисунок 3.22 – VolunteerLoginPage

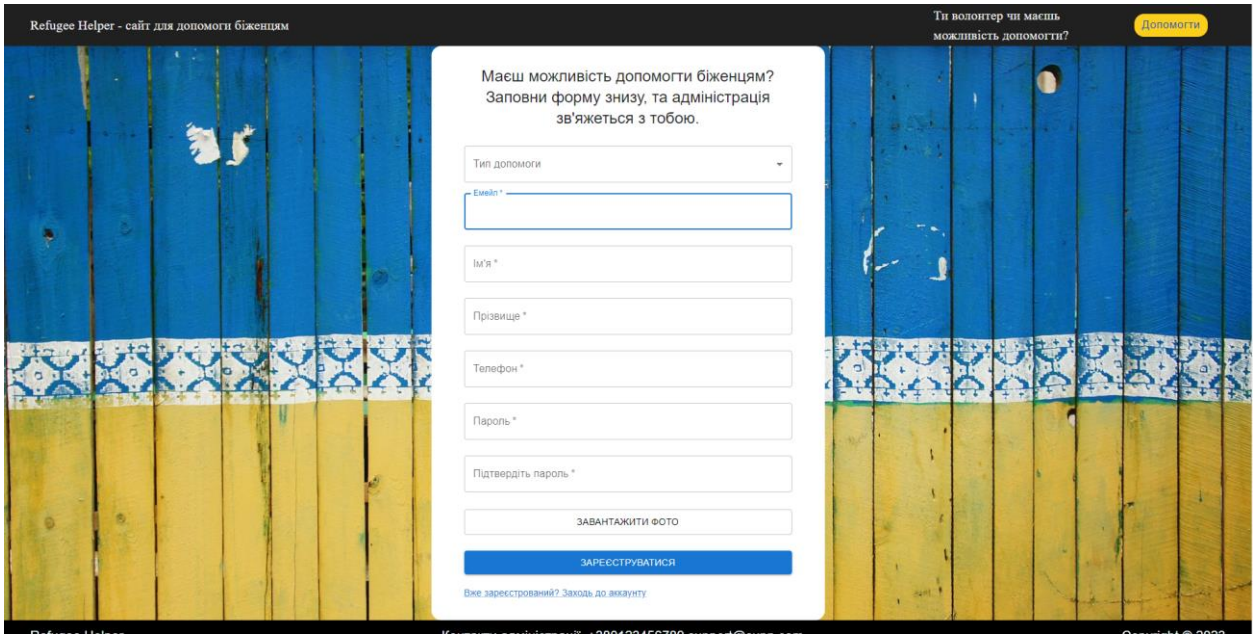


Рисунок 3.22 – VolunteerRegisterPage

Після успішної авторизації волонтер перенаправляється до VolunteerMainPage(рисунок 3.24) де відображається інформація щодо його профіля, разом з іншою важливою інформацією, де він має можливість перенаправитись до VolunteerCreateAidPage(рисунок 3.25), де він зможе створити нову об’яву.

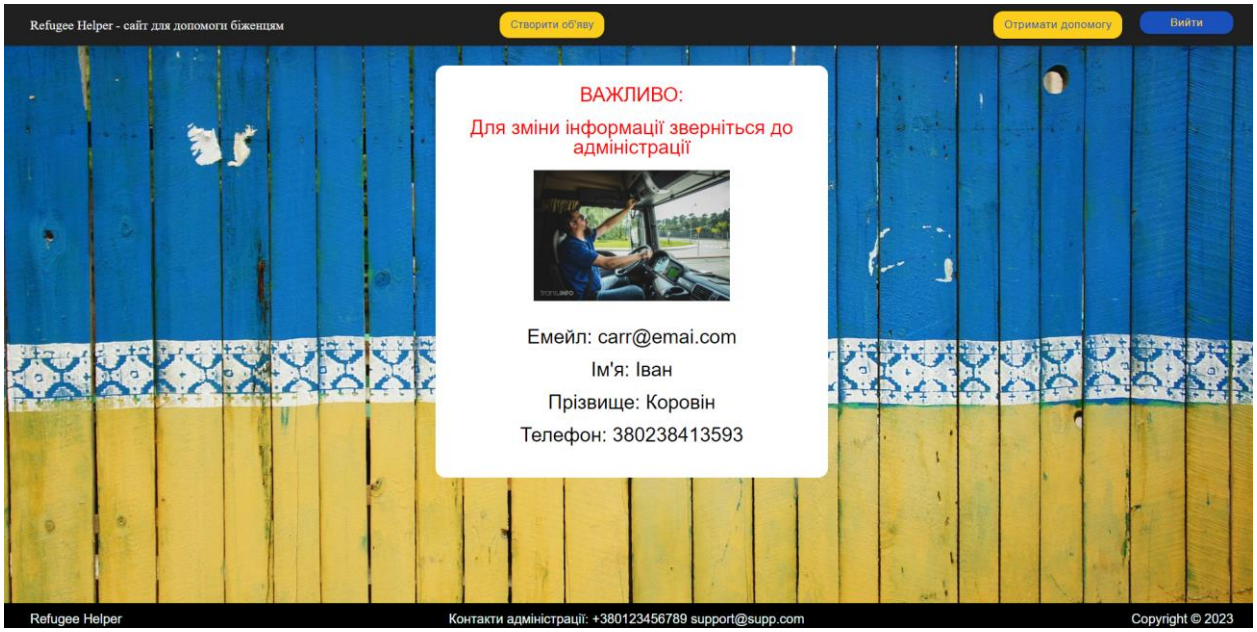


Рисунок 3.22 – VolunteerMainPage

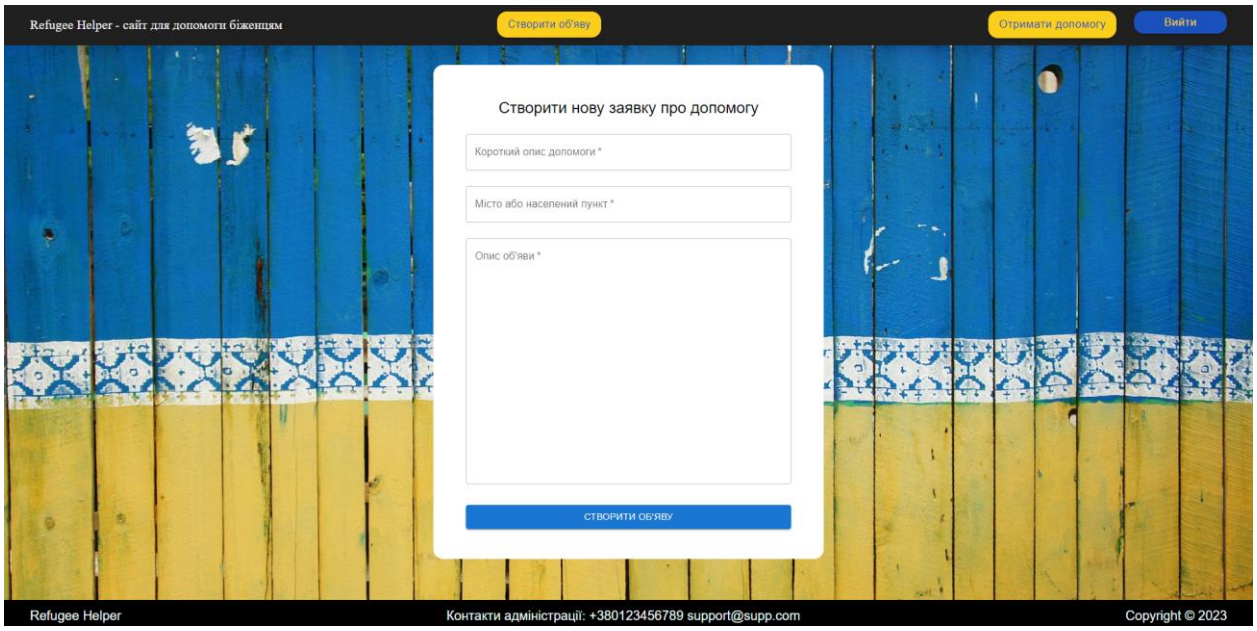


Рисунок 3.22 – VolunteerCreateAidPage

Усі ці сторінки для отримання потрібних даних відправляють запити на серверну частину, з якої вони отримують потрібні дані та відображають їх(рисунок 3.23).

```
const createNewAid = () => {
  fetch("http://localhost:8080/a/rest/volunteer/aid", {
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
      Authorization: `Bearer ${token}`,
    },
    method: "POST",
    body: JSON.stringify({
      theme: data.theme,
      content: data.content,
      city: data.city,
    }),
  })
    .then((res) => res.json())
    .then((res) => {
      if (res.errors) {
        for (const [key, value] of Object.entries(res.errors)) {
          setError(`${key}`, value, value);
        }
        return;
      }
      toast("Об'ява створена");
      handleVolunteerMainRedirect();
    });
};
```

Рисунок 3.23 – Приклад запиту до серверної частини

Для полегшення розробки було використано дві бібліотеки - MaterialUI Tostify, які дозволили зменшити час потрібний для розробки зовнішнього вигляду сторінок, та вікон з помилками.

Результатом розробки став повноцінний інтерфейс, який дає можливість юзеру взаємодіяти з серверною частиною, та отримувати потрібні йому дані.

ВИСНОВКИ

У ході виконання роботи, мною було створено функціональні вимоги та визначено потрібний функціонал для біженців та волонтерів. Окрім цього було оцінено проблеми біженців та їх можливі вирішення, разом з оцінкою вже існуючих сайтів. Після цього було досліджено необхідні технології разом з методами розробки та створена структура роботи веб-додатку. Після чого було розроблено серверну частину разом з його інтерфейсом.

Отриманий продукт наразі має деякі недоліки, які потрібно вирішувати після того як він буде доступний для користувачів. Першим з них є активація акаунту, на даний момент вона можлива лише адміністрацією через базу даних, для покращення цього процесу необхідно розробити алгоритм відносно потреб та можливостей користувачів. Окрім цього можлива інтеграція інших мов, якщо цей продукт буде використовуватися за межами України.

Основними плюсами цього додатку є його простота – інтерфейс є мінімалістичним і все що потребує біженець винесено на головну сторінку. Окрім цього алгоритм роботи біженця з сайтом максимально спрощений – людина не має реєструватися, та може знайти потрібну їй інформацію в декілька кліків.

Після дослідження актуальності було виявлено, що на даний час такий додаток є актуальним і в ньому існує потреба. Подальша розробка цього додатку має зосередитися на потребах біженців та додаванні нових типів допомоги .

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Flanagan, D. (2006). "JavaScript: The Definitive Guide." O'Reilly Media.
2. McFarland, D. (2017). "HTML5: The Missing Manual." O'Reilly Media.
3. Freeman, E., & Robson, E. (2014). "Head First HTML and CSS." O'Reilly Media.
4. Duckett, J. (2015). "JavaScript and JQuery: The Web Technologies Series." Wiley.
5. Cooper, A., Reimann, R., & Cronin, D. (2014). "About Face: The Essentials of Interaction Design." Wiley.
6. Rubin, J., & Chisnell, D. (2008). "Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests." Wiley.
7. Goodwin, P. (1999). "Contextual Design: Defining Customer-Centered Systems." Morgan Kaufmann.
8. Betancourt, T. S., et al. (2017). "Mental Health in Refugee Children: A Preventive Intervention." Preventive Medicine Reports, 8, 90–93.
9. Gollmann, D. (2011). "Computer Security." Wiley.
10. Schneier, B. (2015). "Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World." W. W. Norton & Company.
11. Friedman, B., & Nissenbaum, H. (1996). "Bias in Computer Systems." ACM Transactions on Information Systems (TOIS), 14(3), 330–347.
12. Introna, L. D., & Wood, D. (2004). "Pondering the Imponderable: The Role of Ethical Theories in Information Technology." Ethics and Information Technology, 6(1), 27–40.
13. Freeman, E., & Robson, E. (2016). "Head First JavaScript Programming." O'Reilly Media.
14. Snyder, C. (2003). "Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces." Morgan Kaufmann.

15. Hynes, M., Shear, M. K., Nestadt, G., & Marmar, C. (1998). "The Validity of the PTSD Checklist as a Measure of Symptomatic Change in Combat-Related PTSD." Behaviour Research and Therapy, 36(11), 1081–1088.
16. Schneier, B. (2019). "Click Here to Kill Everybody: Security and Survival in a Hyper-connected World." W. W. Norton & Company.
17. Johnson, D. G. (2017). "Computer Ethics." Wiley.
18. Floridi, L. (2016). "The Ethics of Information." Oxford University Press.
19. Joshua Bloch. (2008). "Effective Java"
20. Brian Goetz. (2006). "Java Concurrency in Practice"
21. Herbert Schildt. (2018). "Java: The Complete Reference"
22. Craig Walls. (2019). "Spring in Action"
23. Iuliana Cosmina. (2017). "Pro Spring 5: An In-Depth Guide to the Spring Framework"
24. John Carnell, Illary Huaylupo Sánchez, and Matt Rasband. (2017). "Spring Microservices in Action"
25. Marijn Haverbeke. (2018). "Eloquent JavaScript: A Modern Introduction to Programming"
26. Kyle Simpson. (2015-2018). "You Don't Know JS" (book series)
27. Douglas Crockford. (2008)."JavaScript: The Good Parts"
28. Stoyan Stefanov. (2016)."React Up and Running: Building Web Applications"
29. Alex Banks, Eve Porcello. (2017). "Learning React: Functional Web Development with React and Redux"
30. Michele Bertoli. (2017). "React Design Patterns and Best Practices"

ДОДАТОК

```java
@AllArgsConstructor
@RestController
@RequestMapping("/a/rest/refugee")
public class RefugeeController {
    2 usages
    private final AidService aidService;

    @GetMapping("/aid")
    public ResponseEntity<List<VolunteerAidResponseDto>> getFilteredVolunteerAid(
            @RequestParam(required = false) String city,
            @RequestParam(required = false) Type type,
            @RequestParam(defaultValue = "0") int page,
            @RequestParam(defaultValue = "5") int limit) {
        return ResponseEntity.ok(
                WebLayerMapper.I.volunteerAidModelListToResponseDto(
                        aidService.getFilteredVolunteerAid(
                                city.equals("null") ? null : city, type, page, limit)));
    }

    @GetMapping("/aid/{aidId}")
    public ResponseEntity<VolunteerAidResponseDto> getAidById(
            @PathVariable(value = "aidId") String aidId) {
        return ResponseEntity.ok(
                WebLayerMapper.I.volunteerAidModelToResponseDto(aidService.getVolunteerAidById(aidId)));
    }
}
```

```java
@AllArgsConstructor
@RestController
@RequestMapping("/a/rest/volunteer/aid")
public class VolunteerAidController {
    1 usage
    private final AidService aidService;

    @PostMapping
    @Operation(security = @SecurityRequirement(name = "Authorization"))
    public ResponseEntity<VolunteerAidResponseDto> createVolunteerAid(
            @RequestBody @Valid VolunteerAidRequestDto requestDto) {
        return ResponseEntity.ok(
                WebLayerMapper.I.volunteerAidModelToResponseDto(
                        aidService.createVolunteerAid(
                                WebLayerMapper.I.volunteerAidRequestDtoToModel(requestDto))));
    }
}
```

```java
@Service
@AllArgsConstructor
public class UserServiceImpl implements UserService {
    1 usage
    private final PictureService pictureService;
    2 usages
    private final VolunteerRepository volunteerRepository;
    2 usages
    private final PasswordEncoder passwordEncoder;

    2 usages
    @Override
    public SecurityUserModel getUserByEmail(String email) {
        return ServiceLayerMapper.I.volunteerEntityToSecurityUserModel(
                volunteerRepository
                        .getUserEntityByEmail(email)
                        .orElseThrow(
                                () ->
                                        new IncorrectCredentialsException(
                                                Map.of( k1: "email", String.format("Юзер з емейлом '%s', не знайдений", email)),
                                                "Incorrect email")));
    }

    1 usage
    @Override
    public VolunteerModel registerUser(VolunteerModel volunteerModel, MultipartFile picture) {
        volunteerModel.setPhoto(pictureService.savePicture(picture, volunteerModel.getEmail()));
        volunteerModel.setPassword(passwordEncoder.encode(volunteerModel.getPassword()));
        volunteerModel.setCreatedOn(LocalDateTime.now());
        volunteerModel.setEnabled(false);

        return ServiceLayerMapper.I.volunteerEntityToModel(
                Optional.of(
                        volunteerRepository.save(
                                ServiceLayerMapper.I.volunteerModelToEntity(volunteerModel)))
                        .orElseThrow(() -> new RuntimeException("Помилка при збереженні юзера")));
    }
```

```java
@Service
public class JwtServiceImpl implements JwtService {
    2 usages
    private final byte[] secretKey =
        "3D6FAEBD94B4FF9CA5C00376DD752F47".getBytes(StandardCharsets.UTF_8);
    2 usages
    private final ConfigurableJWTProcessor<SimpleSecurityContext> jwtProcessor =
        new DefaultJWTProcessor<>();

    @PostConstruct
    public void init() {
        JWKSource<SimpleSecurityContext> jweKeySource = new ImmutableSecret<>(secretKey);
        JWEKeySelector<SimpleSecurityContext> jweKeySelector =
            new JWEDecryptionKeySelector<>(
                JWEAlgorithm.DIR, EncryptionMethod.A128CBC_HS256, jweKeySource);
        jwtProcessor.setJWEKeySelector(jweKeySelector);
    }

    1 usage
    @Override
    public String createJwt(String email) {
        try {
            Date now = new Date();
            Date expDate = new Date(now.getTime() + 1000 * 60 * 60 * 24 * 10); // expires in 10 days

            JWTClaimsSet claims =
                new JWTClaimsSet.Builder()
                    .claim( name: "email", email)
                    .expirationTime(expDate)
                    .notBeforeTime(now)
                    .build();

            Payload payload = new Payload(claims.toJSONObject());

            JWEHeader header = new JWEHeader(JWEAlgorithm.DIR, EncryptionMethod.A128CBC_HS256);

            DirectEncrypter encrypter = new DirectEncrypter(secretKey);

            JWEObject jweObject = new JWEObject(header, payload);
```

```
useEffect(() => {
  if (!token) {
    toast("Спочатку необхідно зайти до аккаунту", {
      progressClassName: "red-progress",
    });
    handleLoginRedirect();
  }
  fetchUserInfo();
}, [token]);

const fetchUserInfo = () => {
  fetch("http://localhost:8080/a/rest/volunteer/account", {
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
      Authorization: `Bearer ${token}`,
    },
    method: "GET",
  })
    .then((res) => res.json())
    .then((res) => {
      if (res.errors) {
        for (const [key, value] of Object.entries(res.errors)) {
          if (key === "email") {
            localStorage.removeItem("token");
            toast("Спочатку необхідно дочекатися активації акаунту", {
              progressClassName: "red-progress",
            });
            handleMainRedirect();
          }
        }
      }
      setUserInfo(() => ({
        email: res.email,
        name: res.name,
        surname: res.surname,
        phone: res.phone,
        photo: res.photo,
      }));
    });
};
```

```
return (
  <div className="refugee-help-main-page">
    <Container
      className="refugee-help-main-page-body"
      component="main"
      maxWidth="sm"
      onSubmit={handleSubmit}
    >
      <Box component="form" onSubmit={handleSubmit} noValidate sx={{ mt: 1 }}>
        <Typography
          component="h1"
          variant="h5"
          style={{ textAlign: "center" }}
        >
          Вкажіть тип потрібної вам допомоги, та місто в якому вона
          потребується (залиште порожнім якщо хочете переглянути усі об'яви)
        </Typography>
        <FormControl fullWidth sx={{ mt: "18px", mb: "0" }}>
          <InputLabel id="data">Тип допомоги</InputLabel>
          <Select
            labelId="type"
            value={data.type}
            label="Тип допомоги"
            name="type"
            onChange={handleChange}
          >
            <MenuItem value={"CARRIER"}>Перевізник</MenuItem>
            <MenuItem value={"LANDLORD"}>Домовласник</MenuItem>
            <MenuItem value={"MEDIC"}>Лікар</MenuItem>
            <MenuItem value={"PSYCHOLOGIST"}>Психолог</MenuItem>
            <MenuItem value={"LAWYER"}>Юрист</MenuItem>
          </Select>
        </FormControl>
        <TextField
          margin="normal"
          onChange={handleChange}
          fullWidth
          id="city"
          label="Місто в якому потрібна допомога (опціонально)"
          name="city"
          autoFocus
        />
```