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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Інформаційне та програмне забезпечення підсистеми обліку донорів»

Виконала:
студент (-ка) IV курсу, групи ІК-93
Полеха Наталя Миколаївна _____

Керівник:
Доцент кафедри ІСТ, к.т.н., доцент
Мамедова К. Ю. _____

Рецензент:
директор КБІС, к.т.н., доцент
Фінагенов О.Д. _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка Полеха Н.М. _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту
Полесі Наталі Миколаївні

1. Тема проєкту «Інформаційне та програмне забезпечення підсистеми обліку донорів», керівник проєкту Мамедова Катерина Юріївна доцент кафедри ІСТ, к.т.н., затверджені наказом по університету від «31» травня 2023 р. № 2101
2. Термін подання студентом проєкту 12.06.2023
3. Вихідні дані до проєкту технічна документація, існуючі реалізації
4. Зміст пояснювальної записки
   - аналіз предметної області
   - аналіз вимог до програмного забезпечення
   - структура системи обліку донорів
   - опис реалізації та тестування системи обліку донорів
5. Перелік графічного матеріалу
   - діаграма прецидентів
   - бізнес процеси
   - алгоритми основних процесів

- структура бази даних

7. Дата видачі завдання 01.02.2023

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	05.02.2023	
2.	Розроблення та узгодження технічного завдання	10.02.2023	
3.	Підготовка матеріалів першого розділу дипломного проєкту	17.03.2023	
4.	Налаштування інфраструктури проекту	23.04.2023	
5.	Проєктування бази даних	30.04.2023	
6.	Проектування системи	07.05.2023	
7.	Розробка програмного продукту	14.05.2023	
8.	Оптимізація програмного продукту	21.05.2023	
9.	Тестування веб-додатку	21.05.2023	
10.	Підготовка графічної частини дипломного проєкту	25.05.2023	
11.	Оформлення документації дипломного проєкту	01.06.2023	

Студент Наталя ПОЛЕХА

Керівник проєкту Катерина МАМЕДОВА

АНОТАЦІЯ

Полеха Н. М. Інформаційне та програмне забезпечення підсистеми обліку донорів.

Проєкт містить 67 с. тексту, 67 рисунків, 6 таблиць, посилання на 21 літературні джерела та 4 кресленика.

Об’єктом розробки є системи обліку та залучення донорів.

Мета розробки – підвищення ефективності процесу залучення донорів в критичних ситуаціях, автоматизація відправлення запрошення потенційному донору та оптимізація веденя обліку донорів.

У дипломному проєкті розроблено фрагменти системи обліку та залучення донорів у вигляді веб-застосунку. Веб-додаток являє собою клієнт-серверний застосунок, реалізований з використанням сучасних засобів для розробки веб-додатків. Інформаційна безпека сервісів реалізована за допомогою розподілу прав доступу та налаштування верифікації користувачів шляхом надсилання коду на електронну пошту. Система має відкритий API, тому може бути легко інтегрована в інші системи. Архітектура системи дозволяє додатку легко розширюватися, додаючи нові функціональні можливості.

За допомогою клієнтського інтерфейсу користувач з роллю адміністратора може надіслати запит на пошук та залучення оптимальних донорів в критичних ситуаціях. Внаслідок цього користувачам будуть відправлені листи на електронну пошту та на особисту сторінку. Медичні працівники в свою чергу зможуть за допомогою системи збільшити ефективність донорського обліку.

У даному дипломному проєкті розроблено: архітектуру системи, клієнтську та серверну частини застосунку, базу даних, алгоритм створення донора та алгоритм залучення донорів в критичних ситуаціях.

SUMMARY

Polekha N. M. Information and software support of the donor accounting subsystem.

The project contains 67 pages, 67 pictures, 6 tables, references to 21 literary sources and 4 drawings.

The object of development is the system of accounting and attraction of donors.

The purpose of the development is to increase the efficiency of attracting donors process in critical situations, to make the process of sending invitations to potential donors automatic and to optimize donor accounting.

The diploma project dedicated to creating an accounting system and the involvement of donors in critical situations. The web application is a client-server one implemented using modern tools for developing web applications. The information security of the services is implemented using the distribution of user access rights and user verification by sending a code to e-mail. The system has an open API, so it can be easily integrated into other systems. The architecture of the system allows the application to be easily extended by adding new functionality.

Using the client interface, user with the admin role can send a request to find and attract optimal donors in critical situations. As a result, letters will be sent to users' e-mail and personal page. Medical workers, will be able to use the system to increase the efficiency of donor accounting (adding a donor to the database, deleting, updating, creating a history of donations, etc.).

This diploma project developed: system architecture, client and server parts of the application, a database, an algorithm for creating a donor and an algorithm for attracting donors in critical situations.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3	А4	ІК93.150БАК.006ПЗ	Пояснювальна записка	67		
4	А3	ІК93.150БАК.006Д1	Система обліку та залучення	1		
5			донорів. Діаграма			
6			прецидентів			
7	А3	ІК93.150БАК.006Д2	Система обліку та залучення	1		
8			донорів. Бізнес процеси			
9	А3	ІК93.150БАК.006Д3	Система обліку та залучення	1		
10			донорів. Алгоритми основних			
11			процесів			
12	А3	ІК93.150БАК.006Д4	Система обліку та залучення	1		
13			донорів. Структура бази			
14			даних			
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						

Зм.	Ар	№ докум.	Підпис	Дат	ІК93.150БАК.006ПЗ		
Разроб		Полеха Н.М.			Інформаційне та програмне забезпечення підсистеми обліку донорів. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн		Мамедова К.Ю.				Т	
Затв.						КПІ ім. Ігоря Сікорського, ІК-93	

Пояснювальна записка
до дипломного проєкту
на тему: «Інформаційне та програмне забезпечення підсистеми обліку донорів»

Київ – 2023 року

ЗМІСТ

ВСТУП

У сучасному світі інформаційні технології і програмне забезпечення відіграють значну роль у багатьох галузях, включаючи медицину та донорську діяльність. Розвиток технологій надає можливості для покращення ефективності та якості процесу залучення та обліку донорів.

Традиційні методи обліку донорів, такі як паперові документи та ручне введення даних, можуть бути недостатньо ефективними та призводити до помилок та недоліків у системі. Саме тут відкриваються перспективи застосування сучасних інформаційних технологій та програмного забезпечення для підвищення ефективності процесу обліку донорів та пришвидшення пошуку оптимальних донорів для важливих переливань, операцій, тощо.

Метою даного дипломного дослідження є підвищення ефективності процесу залучення донорів в критичних ситуаціях, автоматизація відправлення запрошення потенційному донору та оптимізація веденя обліку донорів.

Сутність цієї теми полягає у вивченні та розробці системи, яка має на меті допомогти у зборі, зберіганні та обробці даних про донорів. Інформаційне забезпечення включає в себе базу даних, яка містить інформацію про донорів, таку як особисті дані, медичні дані, відомості про попередні донорські процедури, тощо.

Програмне забезпечення, в свою чергу, надає різний функціонал в залежності від ролі. Адміністратор має можливість відправити запит на пошук найбільш оптимальних донорів для певної процедури, асистент (працівник медичного закладу) має змогу додавати, видаляти та редагувати інформацію про донорів, їх медичні дані та історію донацій. Донор в свою чергу також має свою сторінку, де відображатимуться запрошення на донацію. Основою роботи є алгоритм пошуку оптимальних донорів, який використовує особисту та медичну інформацію користувача для залучення найбільш оптимальних донорів в залежності від безлічі показників таких як: група крові, місце проживання, кількість донацій, стан донора після кожної донорської процедури, тощо.

Основними перевагами системи є пришвидшення та оптимізація процесу залучення донорів в критичних ситуаціях, автоматичне відправлення запрошення потенційному донору та оптимізація обліку донорів.

Важливим аспектом інформаційного та програмного забезпечення підсистеми обліку донорів є забезпечення конфіденційності, безпеки та доступності даних. Оскільки дані, що використовуються в системі, містять особисту та медичну інформацію, вони потребують особливої уваги та заходів щодо їх захисту від несанкціонованого доступу та недоречного використання.

Дипломний проєкт складається зі вступу, чотирьох розділів, висновків до кожного із розділів та загальних висновків. У першому розділі проаналізовано предметну область та існуючі системи для забезпечення обліку та оптимізації донорських процесів, порівняно рішення, аргументовано вибір технологій для реалізації завдання. У другому розділі описаний аналіз вимог до програмного забезпечення та сформавано постановку задач системи. У третьому розділі описано процес проектування та архідектуру проєкту, а саме архітектуру клієнтської та серверної частин. Також у цьому розділі описано структуру бази даних та поясненно використання кожної з таблиць. В четвертому розділі описато процес реалізації та тестування додатку з наданням рисунків інтерфейсу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність системи обліку та залучення донорів

Актуальність системи базується на низці вагомих факторів, які вимагають уваги та ефективних рішень. Потреба в крові постійно зростає, в той час як кількість донорів обмежена. Це створює дисбаланс між попитом і пропозицією та підкреслює важливість системи обліку та залучення донорів.

Нагальна потреба в системі обліку полягає у забезпеченні ефективного планування, моніторингу та відстеженні донорської діяльності. Це дозволяє медичним організаціям прогнозувати попит на донорські матеріали, забезпечувати необхідні запаси та керувати ресурсами ефективно. Залучення нових донорів та збереження існуючих є необхідними кроками для підтримки сталого постачання крові.

Важливим фактором системи обліку та залучення донорів є безпека. Донорські дані, такі як медичні історії, результати тестів та особиста інформація, є конфіденційними та потребують надійного захисту. Ефективна система обліку гарантує, що дані зберігатимуться та оброблятимуться згідно з вимогами приватності та конфіденційності.

Додатковою перевагою системи обліку та залучення донорів є необхідність покращення комунікації та взаємодії між медичними організаціями, донорами та отримувачами донорських матеріалів. Ефективна система дозволяє здійснювати швидкий обмін інформацією про доступні запаси крові та потреби в них, спрощує процес запиту та розподілу донорських матеріалів між різними медичними установами.

Крім того, розвиток інформаційних технологій та програмного забезпечення надає можливість впровадження нових функцій та інструментів у систему обліку та залучення донорів. Наприклад, це можуть бути мобільні додатки або веб-платформи, які спрощують реєстрацію донорів, нагадують про можливість здати кров, надають інформацію про донорські заходи та події. Такі інновації сприяють залученню нових донорів та підтримці вже існуючих.

Таким чином, система обліку та залучення донорів є актуальною не лише з медичного, але й з соціального та технологічного поглядів. Вона сприяє покращенню доступу до донорських матеріалів, підвищенню ефективності управління донорською діяльністю, забезпечує конфіденційність даних та сприяє активній взаємодії між всіма сторонами, що займаються донорською діяльністю.

1.2 Характеристика та особливості предметної області

Предметна область обліку та залучення донорів включає в себе процеси, інструменти та методи, що стосуються збору, обробки, зберігання та управління інформацією про донорів та донорські матеріали. Ця область має на меті забезпечити ефективну роботу з донорами та підвищити продуктивність процесів їх залучення. Основні характеристики предметної області обліку та залучення донорів включають збір та обробку інформації про донора, залучення та утримання донорів, пошук оптимальних донорів в критичних ситуаціях, безпека та конфіденційність даних.

Особливостями процесу збору та обробки даних про донорів є надання можливості донору, а також медичному працівникові з правами асистента додавати, змінювати чи видаляти інформацію про донора та його історії донацій.

Пошук оптимальних донорів в критичних ситуаціях є основним завданням даного забезпечення. Суть цього процесу полягає в створенні надійного алгоритму, який на основі даних про користувачів буде підбирати найбільш оптимальних донорів для певної операції. Цей процес супроводжується автоматичним відправленням листів на почту донорам, тому важливо забезпечити належне управління дозволами та ролями задля запобігання надання доступу сторонім особам.

Важливим фактором для залучення донорів є відправлення запрошень на донацію. В даній системі це відбувається за допомогою електронної пошти та особистої сторінки користувача. Донор отримує повідомлення з детальною

інформацією про час та місце донації, а також про метеріал, необхідний для майбутньої операції чи переливання (кров, лімфа, тромбоцити, тощо).

Важливою особливістю предметної області є базпека даних та конфіденційність. Оскільки донорські дані є особистими і медичними, їх захист від несанкціонованого доступу і використання є надзвичайно важливим завданням. В системі обліку та залучення донорів повинні бути встановлені надійні механізми захисту даних, такі як шифрування, контроль доступу та належне управління ролями. Це дозволить забезпечити конфіденційність донорської інформації та запобігти можливих порушень приватності.

1.3 Аналіз відомих програмних рішень

Найбільш популярним рішенням обліку та залучення донорів в більшості клінік все ще залишається традиційний паперовий облік. Цей спосіб має безліч недоліків таких як неефективність ведення, відсутність оптимізації, що призводить до затримок в отриманні донорських матеріалів, тощо.

З розвитком технологій та програмного забезпечення почали з’являтися безліч електронних систем, мобільних та веб-додатків для забезпечення обліку та залучення донорів. Ці системи забезпечують зручність в реєстрації та спілкуванні з донорами, але можуть бути обмежені в можливостях пошуку оптимальних донорів у критичних ситуаціях.

Існує багато ефективних рішень які пов’язані з донорськими процесами:

- Електронні медичні системи, які включають модулі для обліку та управління донорською інформацією. Вони дозволяють збирати, зберігати та обробляти дані про донорів, додавати результати медичних тестів та контролювати процеси залучення донорів.
- Веб-платформи та додатки для залучення та управління донорами. Ці рішення надають можливість донорам реєструватися, записуватися на прийом, переглядати свої медичні дані та отримувати повідомлення про акції та події.

- Комп'ютерні системи розроблені для керування запасами донорських матеріалів.

- Аналітичні системи, що надають функції аналізу та звітності для підсистеми обліку донорів, дозволяють виконувати статистичний аналіз, генерувати звіти про кількість донорів та ефективність кампаній та.

Проаналізувавши всі відомі рішення, я провела їх порівняння з запропонованою системою. Основною перевагою запропонованої системи є можливість пошуку оптимальних донорів у критичних ситуаціях. Це дозволяє швидко та ефективно знайти найбільш підходящих донорів, що може вплинути на результати лікування пацієнтів. Система надає можливість аналізувати та порівнювати дані про донорів для пошуку найкращих відповідностей. Вона також може надати підтримку при організації та координації донорських процесів.

Висновки до розділу 1

В даному розділі було проведено аналіз предметної області обліку та залучення донорів. В першому пункті були описані причини актуальності даної теми в сучасному світі. Було з'ясовано, що інформаційне та програмне забезпечення обліку та залучення донорів зможе допомогти підвищити ефективність та пришвидшити донорські процеси. Також були описані характеристики та особливості предметної області такі як: збір та обробка інформації про донора, залучення та утримання донорів, пошук оптимальних донорів в критичних ситуаціях, безпека та конфіденційність даних. В останьому підрозділі було проаналізовано існуючі рішення та проведено коротке порівняння з запропонованою системою.

2 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Постановка задачі

Метою розробки є створення продукту, який допоможе підвищити ефективність обліку та залучення донорів. Проаналізувавши існуючі рішення, було вирішено зробити акцент на пошук оптимальних донорів в критичних ситуаціях.

Для досягнення мети необхідно вирішити такі задачі:

- Аналіз потреб

Перш за все, потрібно ретельно проаналізувати потреби та вимоги організацій, які займаються обліком та залученням донорів. Це включає вивчення їх поточних процесів, ідентифікацію проблемних сфер та визначення потреб у вдосконаленні, слід звернути увагу на особливості критичних ситуацій, їх терміновість та специфічні вимоги до донорів.

- Розробка функціоналу

На основі аналізу потреб потрібно розробити функціональні вимоги до продукту. Це включає визначення основних функцій, які мають бути в продукті, таких як система обліку, збір та аналіз даних про донорів, інструменти залучення та управління донорами, тощо.

- Розробка інтерфейсу користувача

Важливо створити зручний та інтуїтивно зрозумілий інтерфейс користувача, щоб легко використовувати продукт. Розробка дизайну, зручного навігаційного меню та інтерактивних елементів допоможе користувачам швидко орієнтуватися та використовувати продукт ефективно.

- Реалізація технічної інфраструктури

Для ефективної роботи продукту потрібна потужна технічна інфраструктура. Це включає вибір правильної технологічної платформи, налагодження серверів, налаштування бази даних та розробку зручних API для інтеграції з іншими системами.

- Розробка алгоритму пошуку

Необхідно створити алгоритм пошуку, який буде враховувати встановлені критерії відбору. Ці алгоритми повинні бути ефективними, швидкими та точними.

- Тестування та вдосконалення

Після розробки продукту необхідно провести його комплексне тестування для виявлення помилок та недоліків. На основі зворотного зв'язку від користувачів та результатів тестування потрібно вносити вдосконалення та виправлення.

Для більш детального аналізу кожної задачі було побудовано дерево задач (рис. 2.1).

Рисунок 2.1 – Дерево задач

2.2 Опис основного бізнес-процесу, який підлягає автоматизації

Процес пошуку та залучення донорів в критичних ситуаціях - це основний бізнес-процес, який підлягає автоматизації.

Опис цього процесу включає такі етапи:

- Створення та підтримка бази даних донорів, яка містить особисту (прізвище, ім’я, email, місце проживання) та медичну (група крові, кількість донацій, час останьої донації, ркакція організму на донорство) інформацію про донорів. Інформація може бути зібрана від потенційних донорів самостійно або за допомогою мединого працівника, який за згодою донора має змогу додати його до бази. Також інформація про донора може бути надана іншими медичними ресурсами з якими буде співпрацювати забезпечення, наприклад HELSI чи medcard24.

- Використання автоматизованих інструментів для пошуку потенційних донорів, що задовольняють критерії необхідних ресурсів. Це може включати використання бази даних, алгоритмів пошуку, фільтрації та зіставлення.

- Автоматизоване надсилання повідомлення потенційним донорам про потребу в ресурсах через різні комунікаційні канали, такі як електронна пошта, повідомлення у соціальних мережах, SMS тощо.

- Забезпечення зручного способу надання інформації про доступність та процес здавання донорського матеріалу. Запрошення на донацію буде приходити на електронну пошту, а також на сторінку користувача. На сторінці, відображатиметься детальна інформація про час та місце донації.

- Розробка функціоналу, який дозволить донорам зручно оновлювати свою інформацію. Це може включати використання онлайн-форм для реєстрації та можливість зберігання та оновлення даних.

Найкраще проілюструвати основний бізнес-процес можна за допомогою схеми з використанням BPMN нотації (рис. 2.2). Детальніше переглянути цей та інші бізнес процеси можна на кресленику ІК93.150БАК.006Д2.

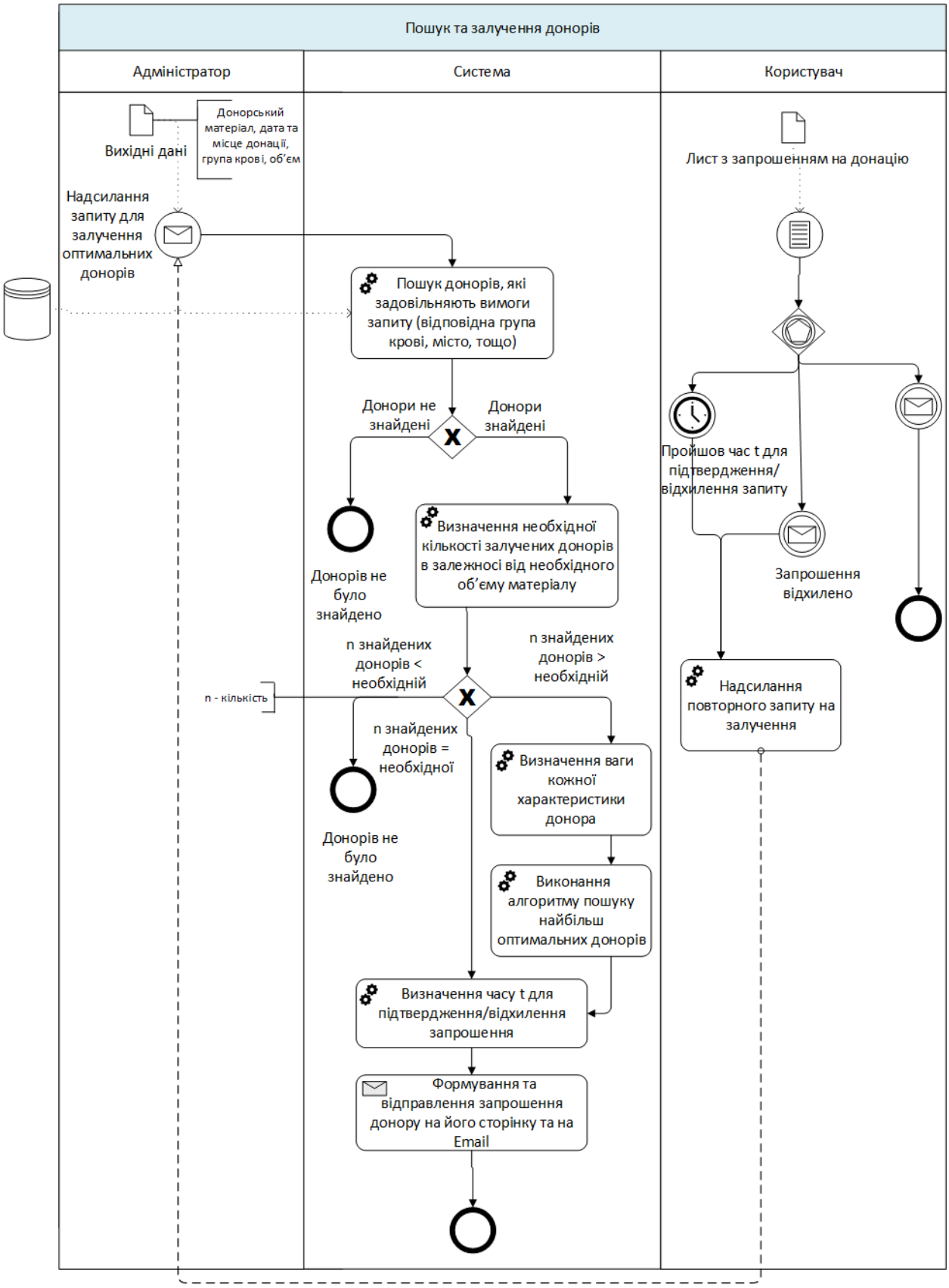


Рисунок 2.2 – Схема основного бізнес процесу

Коли виникає якась критична ситуація (термінова операція чи переливання) користувач з правами адміністратора може надіслати запит на залучення найбільш оптимальних донорів. Адміністратор заповнює форму, де необхідно ввести деталі донації такі як група крові, час збору матеріалу, центр донації та необхідний об’єм матеріалу. Після цього відбувається пошук відповідних донорів, якщо їх було знайдено більше ніж потрібно відбувається їх сортування в залежності від ваги, яка різна для кожного критерію. Після визначення групи оптимальних донорів, формується лист-запрошення, який відправляється на електронну пошту. Також цей лист відображатиметься на сторінці користувача. Прочитавши лист, користувач може ознайомитись з детальною інформацією про донацію. Також в залежності від часу збору матеріалу визначається час до якого донор повинен підтвердити чи відхилити запрошення. Якщо час проходить або донор відхиляє запрошення, надсилається ще один запит на залучення донора.

2.1.3 Опис структури інформаційної системи

Для кращого формування структури інформаційної системи, було вирішено збудувати діаграму прецидентів (кресленик ІК93.150БАК.006 Д1).

За допомогою діаграми можна виділити три основні ролі системи:

- Адміністратор;
- Асистент;
- Донор.

Роль адміністратора та асистента дозволяє вести облік, тобто додавати донорів, обновляти їх особисту та медичну інформацію, а також створювати в базі записи історії донацій. Адміністратор також має доступ до надсилання запиту на отримання оптимальних донорів в критичних ситуаціях та перегляду результатів цих запитів. Результатом запиту може бути знаходження донорів та надсилання їх запрошень або ж не знаходження оптимальних донорів. Роль донора користувач отримує зареєструвавшись на сайті, таким чином він

спричиняє створення запису про себе в базі. Також донор може змінювати свою особисту чи медичну інформацію. Так як донору приходять запрошення, він має змогу їх переглянути, прийняти чи відхилити. Таким чином, якщо запрошення не було прийнято до закінчення певного терміну, відбудеться повторний запит на залучення нових донорів.

В діаграмі прецидентів окрім донора, адміністратора та асистента можна побачити інші суб’єкти такі як:

- система, а саме запропонований застосунок, який відповідає за пошук та залучення донорів;
- застосунки медичного обліку, які співпрацюватимуть з системою та надаватимуть медичні дані донора за його згоди.

Висновки до розділу 2

У другому розділі було проведено аналіз вимог до програмного забезпечення системи обліку та залучення донорів. Для досягнення цієї мети було сформульовано такі задачі: аналіз потреб, розробка функціоналу, розробка інтерфейсу користувача, реалізація технічної інфраструктури, розробка алгоритму пошуку, тестування та вдосконалення.

Основний бізнес-процес, який підлягає автоматизації, полягає у пошуку та залученні донорів в критичних ситуаціях. Цей процес включає створення та підтримку бази даних донорів, використання автоматизованих інструментів для пошуку потенційних донорів, надсилання повідомлень про потребу в ресурсах, забезпечення зручного способу надання інформації про донацію, а також розробку функціоналу для зручного оновлення інформації донорами.

Для кращого формування структури інформаційної системи було побудовано діаграму прецидентів, в якій виділено три основні ролі: адміністратора, асистента та донора. Кожна роль має свої функціональні можливості, які дозволяють проводити облік донорів, виконувати пошук та надсилання запрошень, а також змінювати та оновлювати інформацію.

3 СТРУКТУРА СИСТЕМИ ОБЛІКУ ДОНОРІВ

3.1 Огляд засобів реалізації та обгрунтування їх вибору

Так як рішення є клієнт-серверним, тобто складається з двох самостійних частин, то і аналіз доступних мов і інструментів буде розділений на дві частини. Для збереження даних була використана база даних PostgreSQL, яка також буде описана в окремому розділі.

3.1.1 Огляд інструментів для розробки клієнтської частини

Оскільки платформа є web-орієнтованою, її клієнтська частина повинна запускатися через браузер. Найпоширенішою мовою для написання web-додатків є JavaScript [4], саме цю мову програмування я обрала для написання клієнтської частини.

Для ефективнішої побудови web-додатків використовуються бібліотеки та фреймворки JavaScript, такі як: React, VueJS та Angular. Кожен з них має свої переваги та особливості. У своєму застосунку я використовуватиму React [3]. Основні переваги React:

- Гнучкість

React є бібліотекою, що дає розробникам більшу гнучкість у виборі інших технологій та підходів. React може використовуватися як частина більшого стеку технологій, що робить його добрим варіантом для інтеграції з іншими бібліотеками.

- Продуктивність

React використовує віртуальний DOM та має ефективний механізм оновлення інтерфейсу, що сприяє високій продуктивності додатків. Віртуальний DOM дозволяє ефективно маніпулювати та оновлювати інтерфейс, що робить React швидким і масштабованим.

- Розширюваність

React має широку спільноту розробників, що призводить до великої кількості сторонніх бібліотек, компонентів та інструментів розширення. Це дозволяє розробникам швидше створювати новий функціонал, використовуючи наявні рішення.

- Компонентний підхід

Ключовою особливістю бібліотеки є поділ на компоненти, таким чином можна створити окремі частини (компоненти) і потім інтегрувати їх в інші частини проекту.

Для спрощення управління станом була використана бібліотека Redux, яка базується на концепції Flux. Для взаємодії React та Redux була використана бібліотека react-redux [12].

Для взаємодії серверної та клієнтської частини необхідним є здійснення http-запитів з web-додатку в середовище NodeJs. Для налаштування цієї взаємодії була використана бібліотека axios, яка має такі переваги:

- Promise-орієнтований API;
- Підтримка браузерів і Node.js;
- Зручна обробка запитів та відповідей;
- Перехоплення та обробка помилок;
- Вбудовані функції автоматичної серіалізації та розпакування даних.

Схема взаємодії частин проекту, а саме використання axios продемонстрована на рис. 3.1.

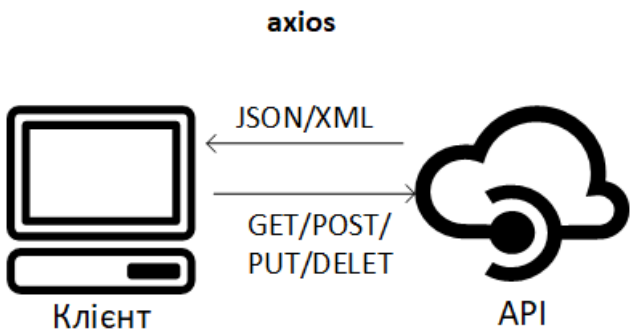


Рисунок 3.1 – Схема взаємодії серверної та клієнтської частин

Важливою частиною користувацького інтерфейсу є формування коректного http-запиту. Для отримання даних корисного навантаження необхідно надати користувачу можливість обрати чи ввести всі необхідні дані. Для цього необхідно створити зручну та зрозумілу форму та налаштувати об’єднання даних в один об’єкт. Цю функцію в проєкті здійснює бібліотека formic [10]. Вона надає зручні інструменти для створення, валідації та обробки форм, спрощуючи роботу зі станом форми та обробкою введених користувачем даних.

Користувацький інтерфейс повинен бути зручним та зрозумілим у використанні для будь-якого користувача, для цього необхідно налаштувати маршрутизацію та навігацію у web-додатку. Ефективним рішенням для цього завдання є бібліотека react-router-dom. Вона надає зручні інструменти для створення роутів, що дозволяють користувачам переміщатися між різними сторінками та компонентами додатка без перезавантаження сторінки.

3.1.2 Огляд інструментів для розробки серверної частини

Для розробки серверної частини, було обрано платформу NodeJs , яка використовує JavaScript для виконання коду. Основні переваги Node.js [7]:

- Node.js використовує події та асинхронний не блокуючий підхід (non-blocking I/O), що дозволяє обробляти багато запитів одночасно з низьким рівнем навантаження на сервер;

- Node.js має потужну систему модулів, що дозволяє розробникам використовувати готові модулі та бібліотеки;

- Node.js має високу швидкість виконання завдяки використанню рушія V8. Він може ефективно обробляти багато одночасних запитів, що робить його популярним для розробки високонавантажених додатків;

- Node.js має вбудовану підтримку різних мережевих протоколів, таких як HTTP, HTTPS, TCP, UDP і інші. Це дозволяє розробникам створювати

різноманітні типи серверних додатків, включаючи веб-сервери, API-сервери, чат-сервери та багато іншого.

Для оптимізації роботи з NodeJs, було використано NestJs - це прогресивний фреймворк для розробки серверних додатків на NodeJs, який використовує TypeScript [20] і пропонує елегантний та модульний підхід до будівництва додатків. TypeScript - це розширення мови програмування JavaScript, яке надає статичну типізацію та додаткові можливості для розробки великих, складних проектів. Він компілюється в звичайний JavaScript і може бути використаний для розробки клієнтських та серверних додатків, веб-інтерфейсів, мобільних додатків та багато іншого. Для кращого поясненя свого вибору, наведу основні переваги NestJs:

- Nest повністю підтримує TypeScript, що дозволяє використовувати сучасний синтаксис, типізацію та переваги TypeScript під час розробки.

- Nest пропонує модульну структуру, що дозволяє організовувати додаток на підставі функціональних модулів. Кожен модуль містить компоненти, провайдери і контролери, які пов'язані з певним функціоналом.

- Nest використовує декоратори для опису компонентів, провайдерів та контролерів. Декоратори забезпечують зрозумілу синтаксичну структуру та покращують читабельність коду. Провайдери – це основна концепція в Nest, яка дозволяє визначати та керувати залежностями в додатку. Можна використовувати провайдери для створення сервісів, репозиторіїв, конфігураційних класів та багато іншого.

- Nest пропонує різноманітні модулі, пакети та бібліотеки, які допомагають швидко розширювати функціональність додатка.

Однією з найважливіших функцій серверної частини є взаємодія з базою даних, в моєму випадку з PostgreSQL. Для реалізації цієї взаємодії була використана бібліотека Sequelize [16]. Sequelize - це об'єктно-реляційний мапер (ORM) для Node.js, який дозволяє розробникам легко взаємодіяти з базами даних, такими як MySQL, PostgreSQL, SQLite і іншими. Sequelize надає

високорівневий API для створення, запиту та модифікації даних у базі даних з використанням об'єктно-орієнтованого підходу. Основні особливості та переваги Sequelize:

- Sequelize дозволяє визначати моделі даних, які відповідають таблицям у базі даних. Таким чином, можна використовувати класи JavaScript для визначення схеми таблиць за допомогою яких Sequelize автоматично генерує SQL-запити для створення таблиць. Крім того, Sequelize підтримує міграції, що дозволяють контролювати версію схеми бази даних та виконувати зміни схеми без втрати даних;

- Sequelize надає потужний набір методів для створення складних запитів до бази даних. Бібліотека дає можливість використовувати ланцюжки методів, щоб групувати умови запиту, сортування, обмеження та інші параметри. Sequelize автоматично генерує SQL-запити на основі цих методів та виконує їх у базі даних;

- Sequelize дозволяє визначати взаємозв'язки між моделями даних, такі як один до одного, один до багатьох і багато до багатьох. Можна встановлювати асоціації між моделями та використовувати їх для виконання складних запитів, які пов'язані з декількома таблицями;

- Sequelize також надає засоби для валідації даних та захисту від SQL-ін'єкцій;

- Sequelize підтримує різні бази даних, включаючи MySQL, PostgreSQL, SQLite та MSSQL.

Надзвичайно важливим завдання застосунку, що працює з особистими та медичними даними є гарантія безпеки та конфіденційності. Саме тому передавання інформації між серверною та клієнтською частиною має бути безпечним. Для вирішення цієї задачі було використано відкритий стандарт (RFC 7519) JWT (JSON Web Token). Цей стандарт використовується для безпечного передавання інформації між двома сторонами у форматі JSON. JWT складається

з трьох основних частин: заголовка (header), набору клеймів (claims) та підпису (signature) (рис. 3.2).

Заголовок (Header) – частина, яка містить тип токену (який зазвичай є JWT) та алгоритм підпису, який використовується для генерації підпису токену. Заголовок кодується в форматі JSON та потім перекодовується в Base64url.

Клейми (Claims) – частина, яка містить корисну інформацію (payload) об'єкту.

Підпис (Signature) – це захищена частина JWT, яка гарантує цілісність даних та перевіряє автентичність. Підпис генерується з використанням секретного ключа або публічного/приватного ключових пар, які використовуються для шифрування.

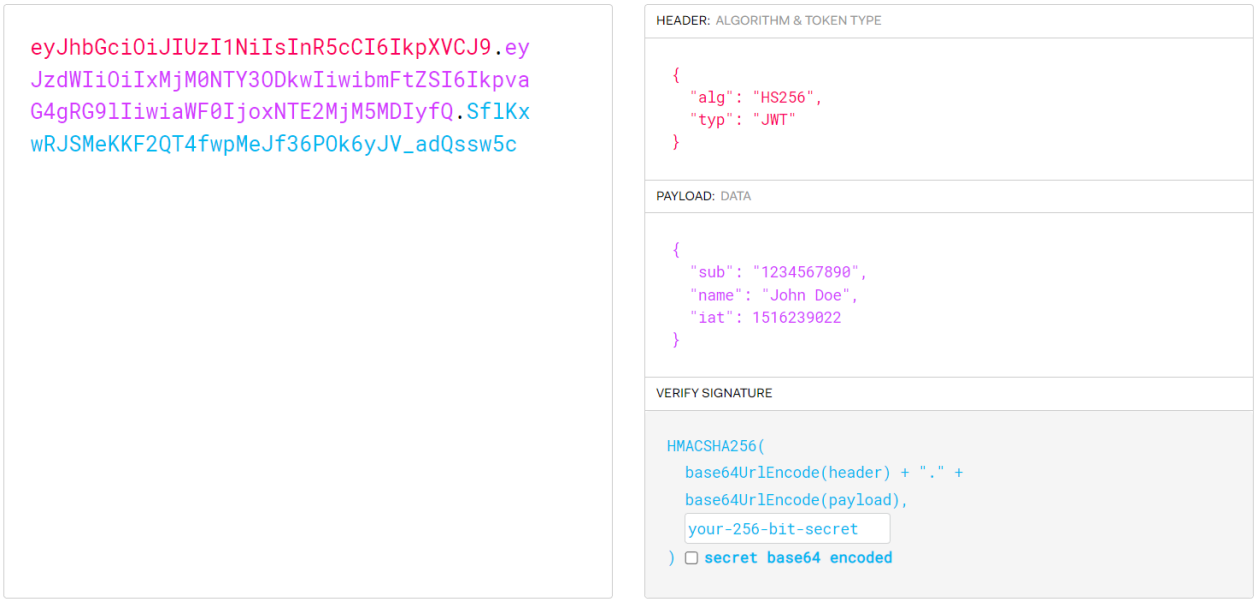


Рисунок 3.2– Структура JWT [1]

Дані, які зберігаються в базі також повинні зберігатися в належному форматі, що стосується паролів, то вони мають зберігатися в захешованому форматі. Для хешування паролів була використана бібліотека bcrypt. Вона забезпечує простий і зручний спосіб захищати паролі шляхом хешування і порівняння паролів захешованої версії. Хешування забезпечує безпеку паролів, оскільки хеш неможливо перетворити назад у вихідний пароль. Таким чином паролі з звичайному форматі взагалі не зберігаються в базі, що ускладнює процес

отримання доступу до сторінки користувача сторонньою особою.

Для залучення донорів, потрібно реалізувати надсилання листа з запрошенням на електронну пошту. Для цього, я використовувала бібліотеку mailgun-js [15]. Бібліотека не є безкоштовною, тому для першої версії додатку було використано стандартний безкоштовний домен та обмежену кількість електронних адресів для взаємодії (максимум 5) (рис. 3.3).

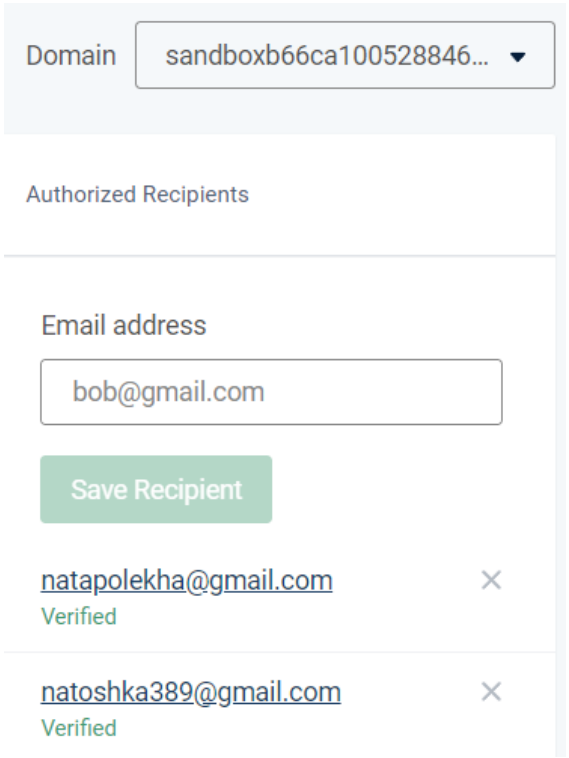


Рисунок 3.3 – Стандартний домен та електронні адреси для взаємодії

Для полегшення роботи з масивами та об’єктами була використана бібліотека lodash. Вона стала дуже популярною серед розробників завдяки своїм потужним та зручним функціям, які допомагають виконувати рутинні завдання швидко та ефективно.

Також слід відзначити Mapbox API [14]. Це набір інструментів і сервісів, розроблений компанією Mapbox, який надає розширені можливості для роботи з картами та географічними данними у веб-додатках. За допомогою Mapbox API можна використовувати різноманітні функції, такі як відображення карт,

навігація, пошук місць, маршрутизація та багато іншого. В запропонованому додатку Mapbox API використовується для заповнення адресу користувачем, це реалізує коректне введення адресу, координати якого надалі зможуть використовуватися для алгоритму пошуку на сервері. Саме тому бібліотека mapbox є частиною як серверної, так і клієнтської частин. Основний функціонал цієї бібліотеки, який реалізований в проєкті, зображений на рис. 3.4 та 3.5.

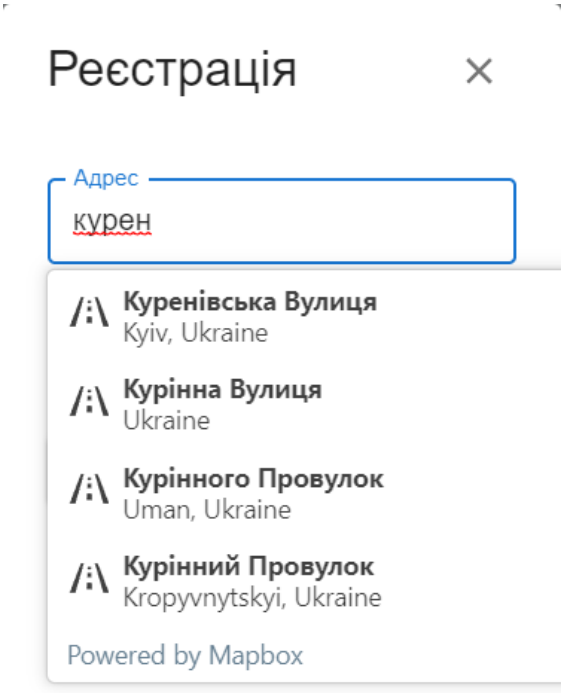


Рисунок 3.4 – Пошук адреси користувача на основі введеного тексту

```
{ type: 'Point', coordinates: [ 30.5241361, 50.4500336 ] }
{
  type: 'Point',
  coordinates: [ 30.440839, 50.474417 ],
  interpolated: true,
  omitted: true
}
{ type: 'Point', coordinates: [ 30.9128626, 50.067585 ] }
{ type: 'Point', coordinates: [ 30.44209, 50.475833 ] }
```

Рисунок 3.5 – Надання координат місцевості з залежності від адреса

3.1.2 Огляд інстументів для розробки бази даних

Для збереження інформації існують два основних типа баз даних – реляційні та нереляційні.

Реляційні бази даних (РБД): Це найпоширеніший тип баз даних, де дані організовані у вигляді таблиць зі структурованими стовпцями. Реляційні бази даних використовують мову SQL (Structured Query Language) для маніпулювання даними. Прикладами реляційних баз даних є MySQL, PostgreSQL, Oracle, Microsoft SQL Server і SQLite.

Нереляційні (NoSQL) бази даних: Ці бази даних використовують альтернативні моделі для організації та збереження даних. Вони не використовують традиційні таблиці зі стовпцями. Нереляційні бази даних часто використовуються для зберігання великих обсягів даних, таких як графи, ключ-значення, документи або часові ряди. Прикладами NoSQL баз даних є MongoDB, Cassandra, Redis, CouchDB та Neo4j.

Для свого застосунку я обрала реляційну базу даних PostgreSQL. Обравши реляційну базу даних, я звернула увагу на такі переваги цього типу БД:

- Структурованість

РБД зберігають дані у вигляді таблиць зі структурованими стовпцями, що дозволяє організувати дані в логічну структуру. Це полегшує розуміння та управління даними.

- Зв'язки між даними

РБД дозволяють встановлювати зв'язки між таблицями за допомогою зовнішніх ключів. Це дозволяє створювати складні структури даних, які відображають залежності та взаємозв'язки між різними елементами.

- Цілісність даних

РБД надають механізми для забезпечення цілісності даних, таких як обмеження цілісності, транзакції та перевірка правил. Це дозволяє забезпечити, що дані залишаються коректними та консистентними в усіх операціях з ними.

- Мова структурованих запитів (SQL)

РБД використовують мову SQL для маніпуляції та отримання даних. SQL є стандартом у сфері баз даних і має потужний набір операцій для фільтрації, сортування, групування, об'єднання та інших операцій з даними.

- Повторне використання та модульність

Реляційні бази даних дозволяють повторно використовувати дані та запити, що спрощує розробку та підтримку додатків. Вони також підтримують модульність, дозволяючи розбити дані на окремі таблиці та модулі, що сприяє організації коду.

- Безпека

Реляційні бази даних мають механізми безпеки, такі як рівні доступу, ролі користувачів та шифрування даних. Це дозволяє обмежувати доступ до даних та забезпечувати конфіденційність та цілісність інформації.

3.2 Архітектура проєкту

3.2.1 Структура бази даних

Структура бази даних складається з шести таблиць:

- Users – таблиця, яка містить інформацію про зареєстрованих користувачів. Там зберігається особиста та службова інформація. Для детальнішого опису моделі в таблиці 3.1 наведений опис її колонок. На рис. 3.6 зображена схема таблиці Users, за допомогою якої можна побачити тип колонки, ідентифікатор, обов’язковість та унікальність.

Таблиця 3.1 – Опис колонок таблиці Users

Назва	Тип даних	Додаткова інформація
id	integer	унікальний ідентифікатор (serial – для колонки автоматично генеруються унікальні числові значення)
email	varchar (255)	адреса електронної пошти
password	varchar (255)	пароль (зберігається в захешованому вигляді)
firstName	varchar (255)	ім’я користувача

lastName	varchar (255)	прізвище користувача
birthDate	bigInt	дата народження (дата зберігається в форматі timestamp)
mobileNumber	varchar (255)	номер телефону
role	enum	може містить одне з трьох значень: ADMIN, DONOR, ASSISTANT
verified	boolean	користувач стає верифікованим після підтвердження своєї пошти
address	varchar (255)	місце проживання користувача
city	varchar (255)	місто
emailToken	varchar (255)	Токен для верифікації користувача

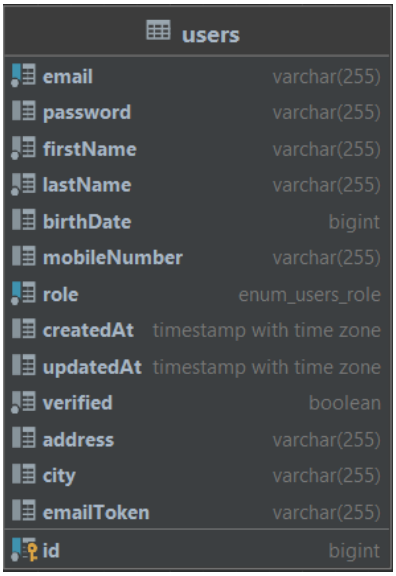


Рисунок 3.6 – Схема таблиці Users

- DonorInfo – таблиця в якій зберігається медична інформація донора. Детальну інформацію про колонки демонструє таблиця 3.2. Схематичне відображення можна переглянути на рис. 3.7.

Таблиця 3.2 – Опис колонок таблиці DonorInfo

Назва	Тип даних	Додаткова інформація

id	integer	унікальний ідентифікатор (serial)
bloodType	enum	група крові, може бути одним із запропонованих значень: 1+, 1-, 2+, 2-, 3+, 3-, 4+, 4-
donationsCount	integer	кількість донацій
lastDonationDate	bigInt	дата останньої донації (timestamp)
lastDonationType	enum	останній тип донації, може бути одним із значень: wholeBloodDonation, plasmaDonation, plateletDonation, erythrocyteDonation (донація цільної крові, плазми, тромбоцитів, еретроцитів)
lastDonationMethod	enum	останній метод донації, може бути одним із значень: general, plasmapheresis, plateletpheresis (загальний, плазмаферезис, тромбоферезис)
userId	integer	ідентифікатор донора, зовнішній ключ на таблицю Users

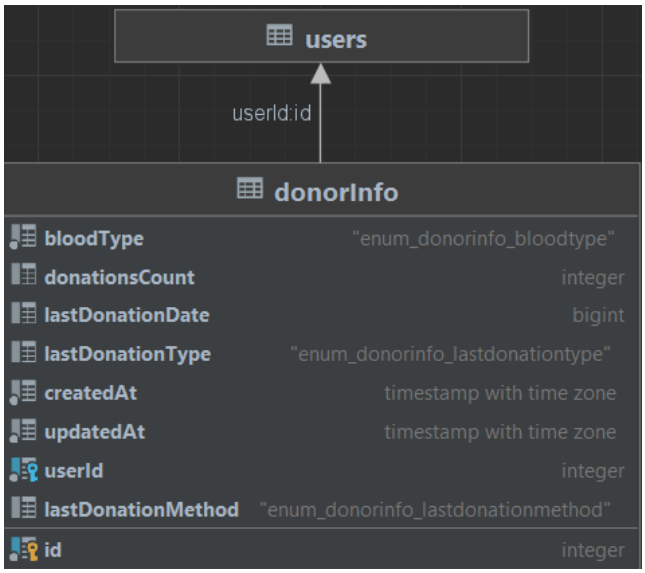


Рисунок 3.7 – Схема таблиці DonorInfo

- DonationPoints – таблиця, в якій зберігається інформація про центри донації. Колонки моделі описані в таблиці 3.3, схематичне відображення – на рис. 3.8.

Таблиця 3.3 – Опис колонок таблиці DonationPoints

Назва	Тип даних	Додаткова інформація
id	integer	унікальний ідентифікатор (serial)
address	varchar (255)	місце проживання користувача
city	varchar (255)	місто
containsCryopreservation	boolean	чи містить центр криоконсервацію (від цього залежить скільки можуть зберігатися донорські матеріали)

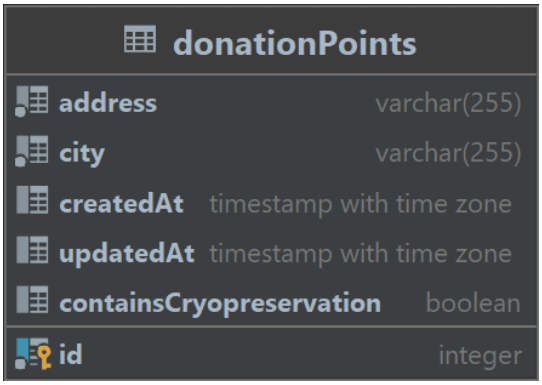


Рисунок 3.8 – Схема таблиці DonationPoints

- DonorHistories – таблиця, в якій зберігається історія донацій. Для детальнішого опису моделі в таблиці 3.4 наведений опис її колонок. На рис. 3.9 зображена схема таблиці DonorHistories.

Таблиця 3.4 – Опис колонок таблиці DonorHistories

Назва	Тип даних	Додаткова інформація
id	integer	унікальний ідентифікатор (serial)
donationDate	bigInt	дата донації (timestamp)

donationMethod	enum	метод донації, може бути одним із значень: general, plasmapheresis, plateletpheresis
donationType	enum	тип донації, може бути одним із значень: wholeBloodDonation, plasmaDonation, plateletDonation, erythrocyteDonation
userId	integer	ідентифікатор донора, зовнішній ключ на таблицю Users
donationPointId	integer	Ідентифікатор центра донації, зовнішній ключ на таблицю DonationPoints
healthStatus	enum	Стан здоров’я донора після донації, може бути одне із значень: normal, bad

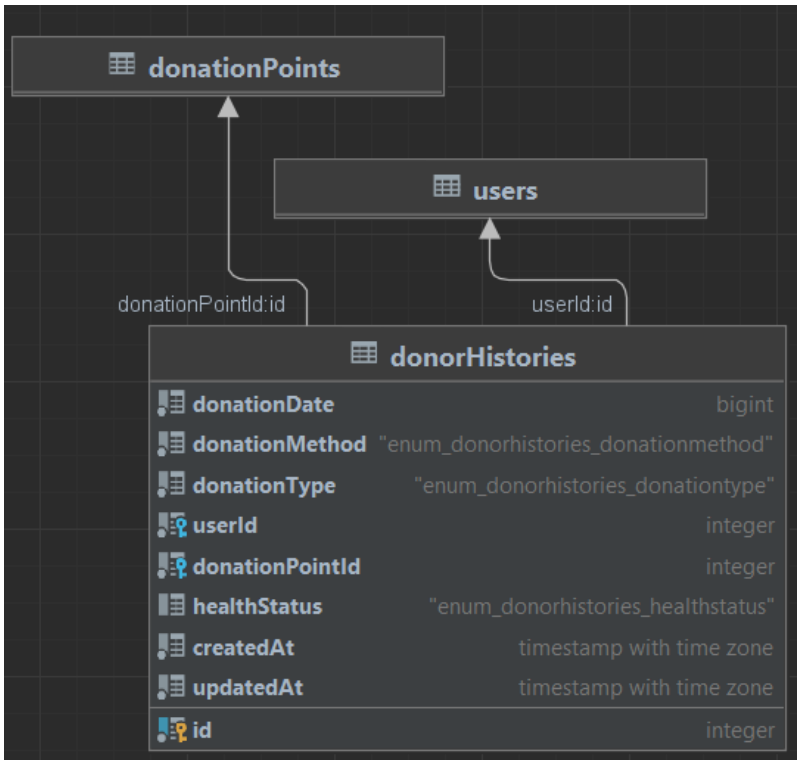


Рисунок 3.9 – Схема таблиці DonorHistories

- Requests – таблиця в якій зберігається інформація про запити адміністратора на отримання донорського матеріалу в критичних ситуаціях. Колонки описані в таблиці 3.5, а схематичне відображення – на рис. 3.10.

Таблиця 3.5 – Опис колонок таблиці Requests

Назва	Тип даних	Додаткова інформація
id	integer	унікальний ідентифікатор (serial)
receivingDate	bigInt	дата збору матеріалу (timestamp)
donationType	enum	тип донації, може бути одним із значень: wholeBloodDonation, plasmaDonation, plateletDonation, erythrocyteDonation
bloodAmount	real	об'єм матеріалу
donationPointId	integer	ідентифікатор центра донації, зовнішній ключ на таблицю DonationPoints
bloodType	enum	група крові, може бути одним із значень: 1+, 1-, 2+, 2-, 3+, 3-, 4+, 4-
senderId	integer	ідентифікатор користувача, зовнішній ключ на таблицю Users
result	enum	Результат виконання запиту, може бути одним з цих значен: error, found, notFound (помилка, знайдено, незнайдено)

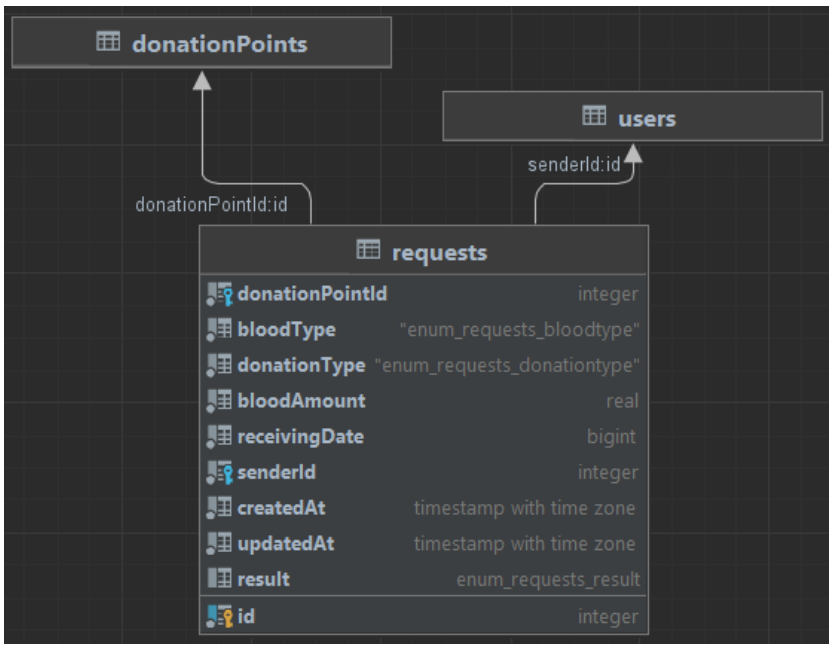


Рисунок 3.10 – Схема таблиці Requests

- Invites – таблиця, де зберігаються запрошення на донації. Детальну інформацію про колонки демонструє таблиця 3.6. Схематичне відображення можна переглянути на рис. 3.11.

Таблиця 3.6 – Опис колонок таблиці Invites

Назва	Тип даних	Додаткова інформація
id	integer	унікальний ідентифікатор (serial)
receiverId	integer	ідентифікатор отримувача запрошення, зовнішній ключ на таблицю Users
requestId	integer	ідентифікатор запиту, внаслідок якого було відправлене запрошення
acceptStatus	enum	статус запрошення, може бути одним із чотирьох значень: accepted, reviewed, notReviewed, deflected (прийняте, переглянуте, не переглянуте, відхилене)

timeToWaitBeforeApprove	integer	час очікування прийняття/відхилення запрошення донором

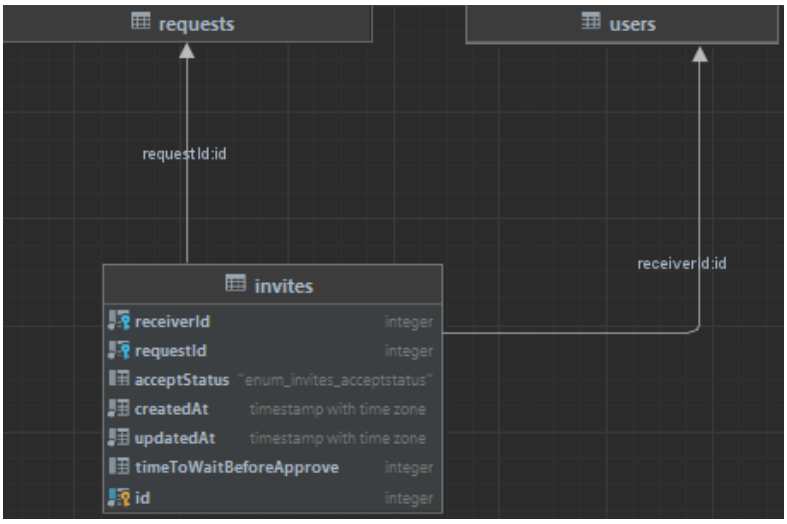


Рисунок 3.11 – Схема таблиці Invites

Детально переглянути зв’язки між таблицями можна в кресленику ІК93.150БАК.006Д4.

Для роботи з базою даних було використано IDE DataGrip яка має можливості для візуального проєктування баз даних, моделювання та експлуатації. DataGrip – це інтегроване середовище розробки для роботи з реляційними базами даних. Воно розроблено компанією JetBrains і надає зручні інструменти для підключення, керування та адміністрування різних реляційних систем у зручному та інтуїтивно зрозумілому інтерфейсі.

3.2.2 Архітектура серверної частини

Розглянемо архітектуру серверної частини застосунку. Серверна частина написана на NodeJs з використанням NestJs. Архітектура додатку на NestJS базується на принципах модульності та інверсії керування (Inversion of Control - IoC). NestJS надає ряд компонентів та концепцій, які допомагають організувати

додаток у логічні модулі та забезпечити ефективну структуру. Основні компоненти архітектури NestJS-додатку:

- Модулі (Modules)

Додаток на NestJS складається з модулів, які групують в собі функціональність. Кожен модуль містить контролери, провайдери, сервіси та інші компоненти. Модулі дозволяють організувати код у чисті логічні одиниці та декларувати залежності між компонентами. На рис. 3.12 наведений приклад модуля написаного на NestJs.

Рисунок 3.12 – Модуль для роботи з записами в таблиці DonorHistories

- Контролери (Controllers)

Контролери відповідають за обробку HTTP запитів та визначають точки входу в додаток. Вони приймають HTTP запити, викликають відповідні методи сервісів та повертають HTTP відповіді. Приклад контролера, який обробляє Get-запит на отримання всіх користувачів наведений на рис. 3.13.

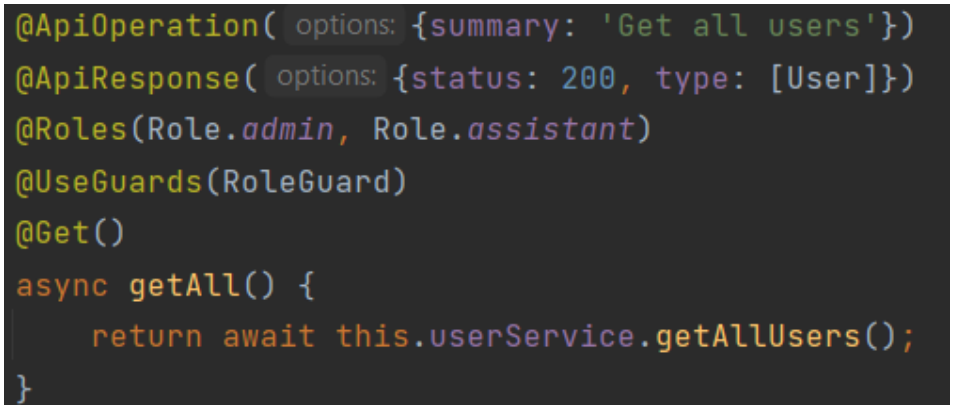


Рисунок 3.13 – Контролер, який обробляє запит на отримання всіх користувачів

- Сервіси (Services)

Сервіси містять бізнес-логіку додатку. Вони виконують конкретні операції, обробляють дані та взаємодіють з БД або іншими сервісами. Кожен сервіс є незалежним модулем, який може бути використаний в будь-якій частині додатку. Це сприяє повторному використанню коду та полегшує тестування. Сервіси можуть бути використані в контролерах або інших компонентах для виконання конкретних завдань. Сервіси можуть використовувати інші сервіси або провайдери, що забезпечує розширюваність та взаємодію між компонентами додатку. На рис. 3.14 відображений AuthService (сервіс, який використовується для авторизації користувачів та підключається до AuthModule).

```
@Injectable()
export class AuthService {
    constructor(
        private userService: UsersService,
        private jwtService: JwtService,
        private donorInfoService: DonorInfoService,
    ) {}

    public async login(@Body() userDto: LoginUserDto) {
        const user = await this.userService.getUserByEmail(userDto.email);

        if (!user.password) {
            return {message: userWithoutPasswordExistMessage, user};
        }

        const validUser = await AuthService.validateUser(user, userDto);
        return this.generateToken(validUser);
    }
```

Рисунок 3.14 – Сервіс, який використовується для авторизації користувачів

- Пайпи (Pipes)

Пайпи використовуються для валідації та обробки вхідних даних перед їх обробкою контролерами. Приклад використання пайпів наведений на рис. 3.15. Таким чином, перш ніж дані потраплять у контролер, вони будуть опрацьовані в пайпі (в нашому випадку пайп використовується для валідації).

```
@ApiOperation( options: {summary: 'Login user'})
@ApiResponse( options: {status: 200, type: User})
@UsePipes(ValidationPipe)
@Post( path: '/login')
async login(@Body() userDto: LoginUserDto) {
    return await this.authService.login(userDto);
}
```

Рисунок 3.15 – Приклад використання пайпів в проєкті

- Guard

Guard є компонентом, який використовується для захисту контролерів від несанкціонованого доступу. В даному застосунку важливо правильно налаштувати роботу з ролями та доступами, тому Guard використовується майже в кожному контролері. Приклад використання Guard наведений на рис. 3.16. Таким чином, доступ до контролера зможе отримати лише користувач з роллю адміністратора або асистента.

```
@ApiOperation( options: {summary: 'Get all users'})
@ApiResponse( options: {status: 200, type: [User]})
@Roles(Role.admin, Role.assistant)
@UseGuards(RoleGuard)
@Get()
async getAll() {
    return await this.userService.getAllUsers();
}
```

Рисунок 3.16 – Приклад використання Guard

В NestJs важливо розділити функціонал на окремі модулі, які будуть містити всі вище описані компоненти. Структура модуля авторизації наведена на рис. 3.17.

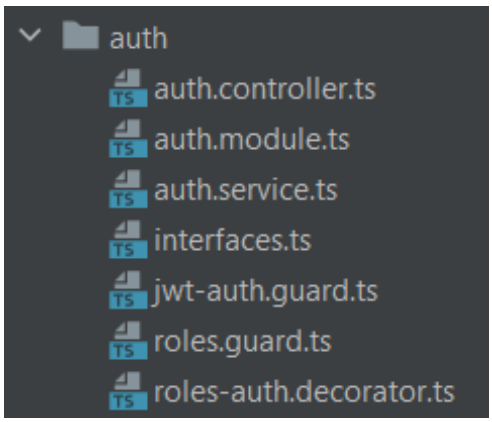


Рисунок 3.17 – Структура модуля авторизації

3.2.3 Архітектура клієнтської частини

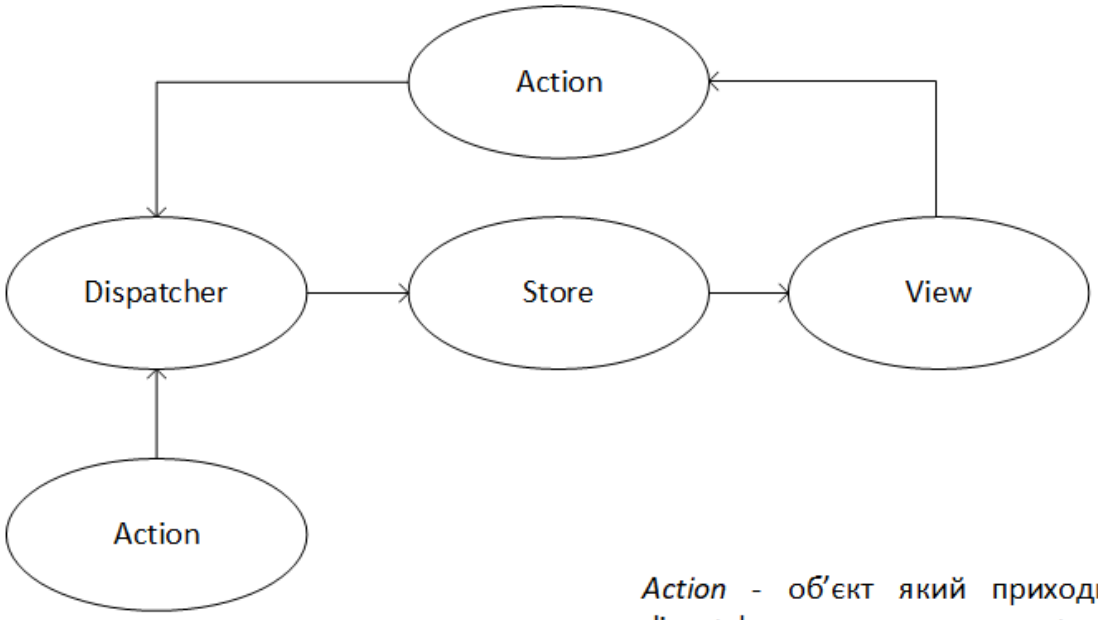
Клієнтська частина побудована за використання фреймворку React. React побудований навколо компонентної моделі. Кожен компонент відповідає за відображення певного елемента інтерфейсу користувача. Компоненти можуть бути класовими (з використанням класових компонентів) або функціональними (з використанням функціональних компонентів та хуків). Хуки є більш сучасним аналогом класових компонент, вони допомагають керувати локальним станом компоненти, виконувати побічні ефекти, такі як взаємодія з сервером, дозволяють створювати посилання на елементи DOM, кешувати значення, тощо.

Для навігації між сторінками або роутами використовується бібліотека маршрутизації, така як React Router [6], що дозволяє організувати роутинг у додатку та відображати відповідні компоненти для кожного шляху.

Для взаємодії з сервером і отримання або відправки даних можна використовується бібліотека Axios [5]. Ця бібліотека допомагає здійснювати асинхронні запити до сервера та обробляти отримані дані.

Для полегшення управління глобальним станом додатку використовуються така архітектурна концепція як Flux, а саме бібліотека Redux [11], яка базується на концепції Flux. Flux - це архітектурний шаблон, який вперше був представлений компанією Facebook. Він заснований на

односторонньому потоці даних і передбачає, що дані рухатимуться в одному напрямку від джерела дій до компонентів і назад (рис. 3.18).



Store - в основному містить стан та логіку програми. Він оновлюється у відповідь на action та сповіщає про це контролер.

View/Controller - компонент, який прослуховує події змін (actions), отримує дані зі сховища і повторно рендерить програму.

Action - об'єкт який приходить в dispatcher та має параметр type, що вказує на тип дії.
ActionCreator допоміжний метод, який створює Action та передає його диспетчеру.

Dispatcher - це центральний вузол, керує всіма потоками даних програми. Він діє як механізм для розподілу дій в додатку.

Рисунок 3.18 – Схема Flux архітектури

3.3 Проектування алгоритмів

3.3.1 Алгоритм пошуку та залучення донорів в критичних ситуаціях

Важливої частиною проектування системи є виділення процесів та відображення послідовності кроків їх виконання за допомогою алгоритмів.

Одним з найважливіших процесів системи є залучення потенційних донорів в критичних ситуаціях, алгоритм цього процесу продемонстрований в додатку В. Процес відбувається внаслідок відправки відповідного запиту адміністратором. Адміністратор заповнює форму і надає всі необхідні дані для виконання алгоритму, а саме:

- bloodAmount – кількість матеріалу необхідного для майбутньої операції чи переливання. Значення вимірюється в літрах і має тип real (десятковий дріб). На основі цього значення визначатиметься необхідна кількість залучених донорів;

- bloodType – група крові (+1, -1, +2, -2, +3, -3, +4, -4) ;

- donationPointId – унікальний ідентифікатор центра донації, посилається на таблицю DonationPoints, де міститься інформація про адресу центру;

- donationType – тип донації, може приймати одне із запропонованих значень: wholeBloodDonation, plasmaDonation, plateletDonation, erythrocyteDonation (донорство цільної крові, плазми, тромбоцитів чи еретроцитів) ;

- receivingDate – дата отримання матеріалу, тобто дата до якої необхідно зібрати матеріал.

Після збору всіх необхідних даних необхідно провести їх валідацію, а саме:

- donationPointId має бути коректним ідентифікатором, що існує в базі;

- receivingDate повинен бути не пізніше раніше ніж за два дня від дня подання запиту та не пізніше ніж за 90 днів.

Якщо дані невалідні, запит закінчується помилкою та створюється запис в таблиці Requests, де колонка result буде дорівнювати error.

Якщо валідація даних пройшла успішно, відбувається запит до БД на отримання верифікованих користувачів з роллю DONOR з відповідною групою крові та з адресою, яка відповідає адресі розташування центру донації в запиті. Наступним кроком відбувається фільтрація донорів по даті можливої наступної здачі, наприклад донор який здавав плазму 02.06.23 не зможе повторити донацію до 02.07.23. Також відбувається фільтрація по наявності запрошення в потенційного донора на випадок якщо донор вже був залучений або відхилив запрошення. Таким чином, після виконання всіх фільтрацій, ми отримуємо оптимальних донорів.

Якщо їх кількість менша потрібної, то запит закінчується та до таблиці Requests додається запис з колонкою result, яка дорівнюватиме notFound (не знайдено).

Якщо ж кількість знайдених донорів дорівнює оптимальній, відбувається розсилання запрошень донорам на пошту. Також створюються записи в таблиці Invites для відображення запрошень на особистій сторінці користувача та запис в таблиці Requests з колонкою result, яка буде дорівнювати found (знайдено).

Якщо ж кількість знайдених оптимальних донорів більша ніж необхідна відбувається їх сортування в залежності від ваги, яка різна для кожного критерію. Критерії, які впливають на оптимальність:

- Кількість часу для здачі матеріалу

Чим більше часу для потенційного донора здати матеріал, тим більша імовірність здачі. Наприклад, якщо донор буде мати змогу здати кров лише за день до отримання матеріалу, вірогідність того, що він піде на донацію зменшується. На рис. 3.19 видно, що проміжок часу на донацію у першого донора більший, саме тому він отримуватиме більший ваговий коефіцієнт.

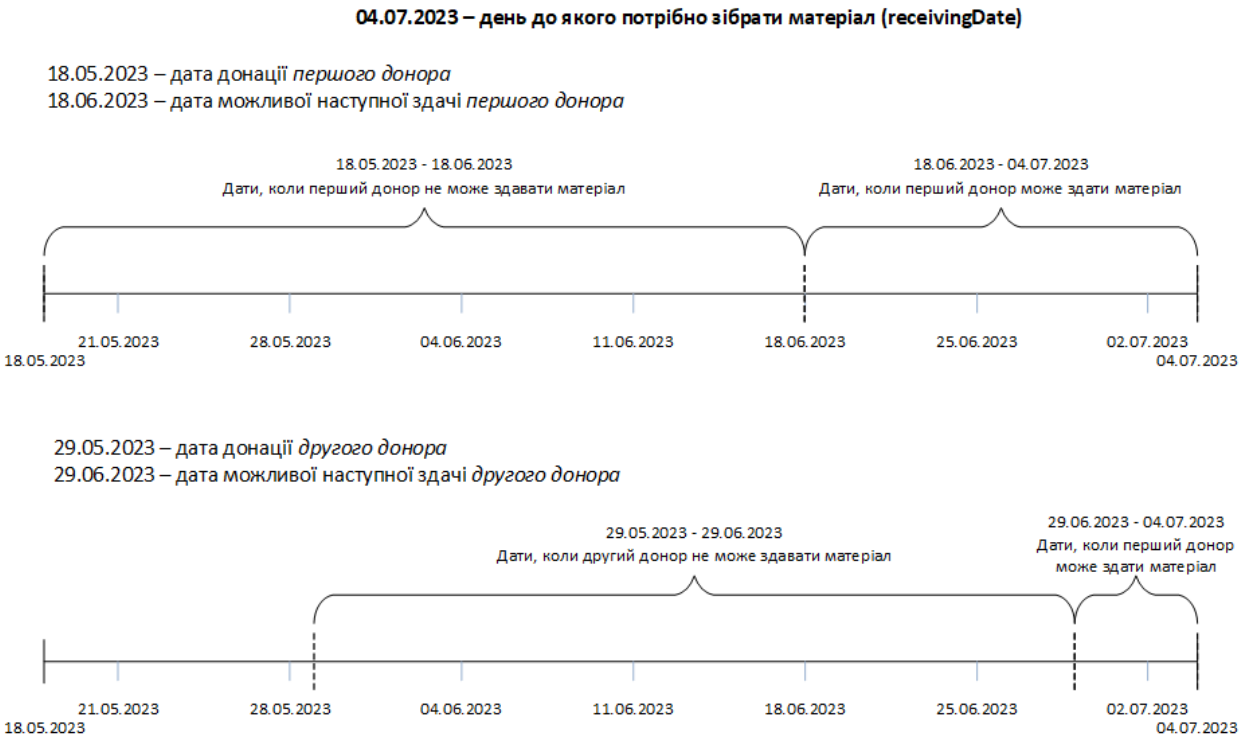


Рисунок 3.19 – Приклад оцінки часу на здачу матеріалу

Важливо зазначити, що алгоритм враховує кількість часу для збереження матеріалу. Центр донації може містити або не містити кріоконсервацію, це впливає на час збереження донорського матеріалу. На рис. 3.20 відображаються константи з значеннями часу (в мілісекундах) для збереження того чи іншого матеріалу в залежності від наявності кріоконсервації.

```
export const bloodStorageTimeWithoutCryopreservation = {
    [DonationType.wholeBloodDonation]: 2764800000, // 32 days
    [DonationType.erythrocyteDonation]: 2764800000, // 32 days
    [DonationType.plasmaDonation]: 259200000, // 3 days
    [DonationType.plateletDonation]: 432000000, // 5 days
}

export const bloodStorageTimeWithCryopreservation = {
    [DonationType.wholeBloodDonation]: 126144000000, // 1460 days
    [DonationType.erythrocyteDonation]: 126144000000, // 1460 days
    [DonationType.plasmaDonation]: 93312000000, // 1080 days
    [DonationType.plateletDonation]: 62208000000, // 720 days
}
```

Рисунок 3.20 – Константи для збереження часу зберігання донорського матеріалу

Таким чином, час збереження буде впливати на можливий час здачі донора, наприклад на рис. 3.21 видно, що так як плазма в центрі донації без кріоконсервації може зберігатися лише три дні, час здачі для обох донорів зменшується до трьох днів (01.07 – 04.07).

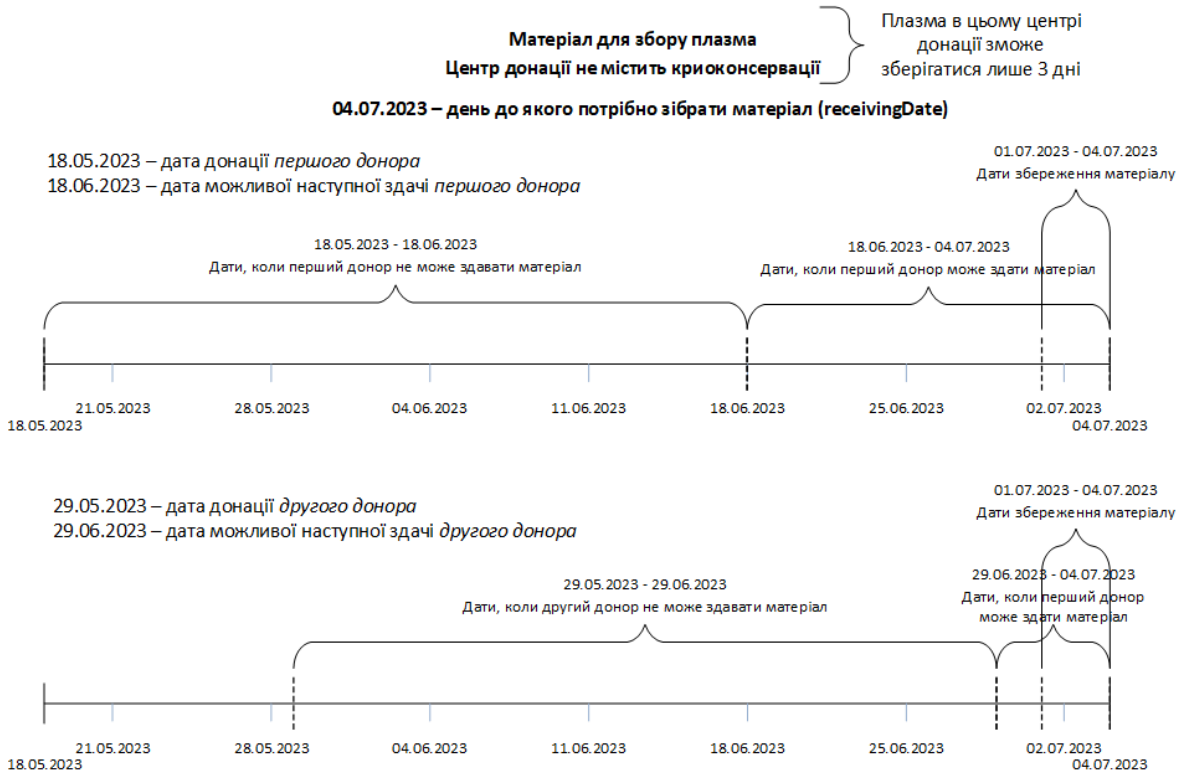


Рисунок 3.21 – Приклад оцінки часу на здачу матеріалу з врахуванням часу збереження матеріалу

- Адреса потенційного донора

Значення адреси донора також впливає на імовірність того чи прийде донор на донацію. Чим його місце проживання дальше від адреси центра донації, тим меншу вагу він отримує.

- Кількість донацій

Чим більша кількість донацій потенційного донора, тим більша імовірність того, що він відвідає центр донації.

- Стан здоров’я після донацій

Відповідно якщо стан здоров’я після донації поганий, вага донора буде меншою.

- Кількість відхилених запрошень

Відповідно чим більша кількість відхилених запрошень, тим менша вага донора.

На основі всіх факторів відбувається підрахунок ваги кожного донора. Після додавання коефіцієнтів донори з найбільшою вагою і стають найбільш

оптимальними. Їм на пошту приходить повідомлення з запрошенням. Також створюються записи в таблиці Invites для відображення запрошень на особистій сторінці користувача та запис в таблиці Requests з колонкою result, яка буде дорівнювати found (знайдено).

3.3.2 Алгоритм створення та верифікації донора

Донор може бути створений двома способами. Перший спосіб це реєстрація користувача на сайті, внаслідок чого відбувається створення запису в таблиці Users з колонкою role, яка дорівнює DONOR. Алгоритм цього процесу знаходиться в додатку В. Другий варіант – це створення користувача асистентом чи адміністратором. В обох випадках важливою частиною є верифікація електронної пошти донора для забезпечення конфіденційності та запобігання отримання доступу до особистих даних сторонніми особами.

Розглянемо перший випадок створення донора – реєстрацію. Для початку донор вводить в форму дані про себе такі як: ім’я, прізвище, електронна пошта, дата народження та номер телефону. Для реєстрації донор також вводить надійний пароль (мінімум вісім символів). Після цього відправляється запит на сервер з усіма заповненими даними та відбувається їх валідація (перевірка коректності). Якщо валідація пройшла успішно, створюється неверифікований донор. Наступним кроком формується код верифікації, це повинні бути вісім випадкових символів. Код верифікації відправляється на пошту донору та зберігається в базі в колонці emailToken в таблиці Users. Після відправки коду верифікації донором відбувається його перевірка, він повинен дорівнювати значенню колонки emailToken цього донора. Якщо код верифікації не правильний, донор залишається неверифікованим. В нього не буде доступу до будь яких дій на своїй особистій сторінці, також він не зможе бути залученим на донацію в критичній ситуації. Відповідно, якщо код правильний – відбувається верифікація користувача.

Висновки до розділу 3:

В даному розділі було оглянуто засоби реалізації додатку та обгрунтовано їх вибір. Окремо було оглянуто інструменти для розробки клієнтської, серверної частин та бази даних. Для розробки клієнтської частини проєкту було обрано JavaScript, з використанням бібліотеки React. Для керування станом була використана бібліотека Redux, а для взаємодії з сервером – бібліотека Axios. Навігацію та маршрутизацію в додатку забезпечує бібліотека react-router-dom. Для серверної частини була обрана платформа Node.js разом з Nest.js і Sequelize, що дозволяє розробляти потужні та ефективні серверні додатки з високою швидкістю виконання та підтримкою асинхронного підходу. Використання TypeScript у Nest.js дозволяє забезпечити статичну типізацію та покращити розробку складних проектів. Бібліотека Sequelize додає зручні інструменти для роботи з базами даних. В якості бази даних була обрана реляційна БД PostgreSQL.

Також було розглянуто архітектуру проєкту та описано структуру бази даних з детальним поясненням до кожної таблиці.

Для кращої демонстрації основних процесів та відображення кожного кроку було збудовано алгоритми (кресленик ІК93.150БАК.006 ДЗ) та наведено детальний опис кожного етапу.

4 ОПИС РЕАЛІЗАЦІЇ ТА ТЕСТУВАННЯ СИСТЕМИ ОБЛІКУ ДОНОРІВ

4.1 Інструкція користувача

4.1.1 Реєстрація донора

Реєстрація донора є першим способом додавання донора в систему. Для реалізації цього процесу, необхідно розробити клієнтську та серверну частину. Клієнтська частина включає в себе форму, яку необхідно заповнити для реєстрації користувача (рис. 4.1).

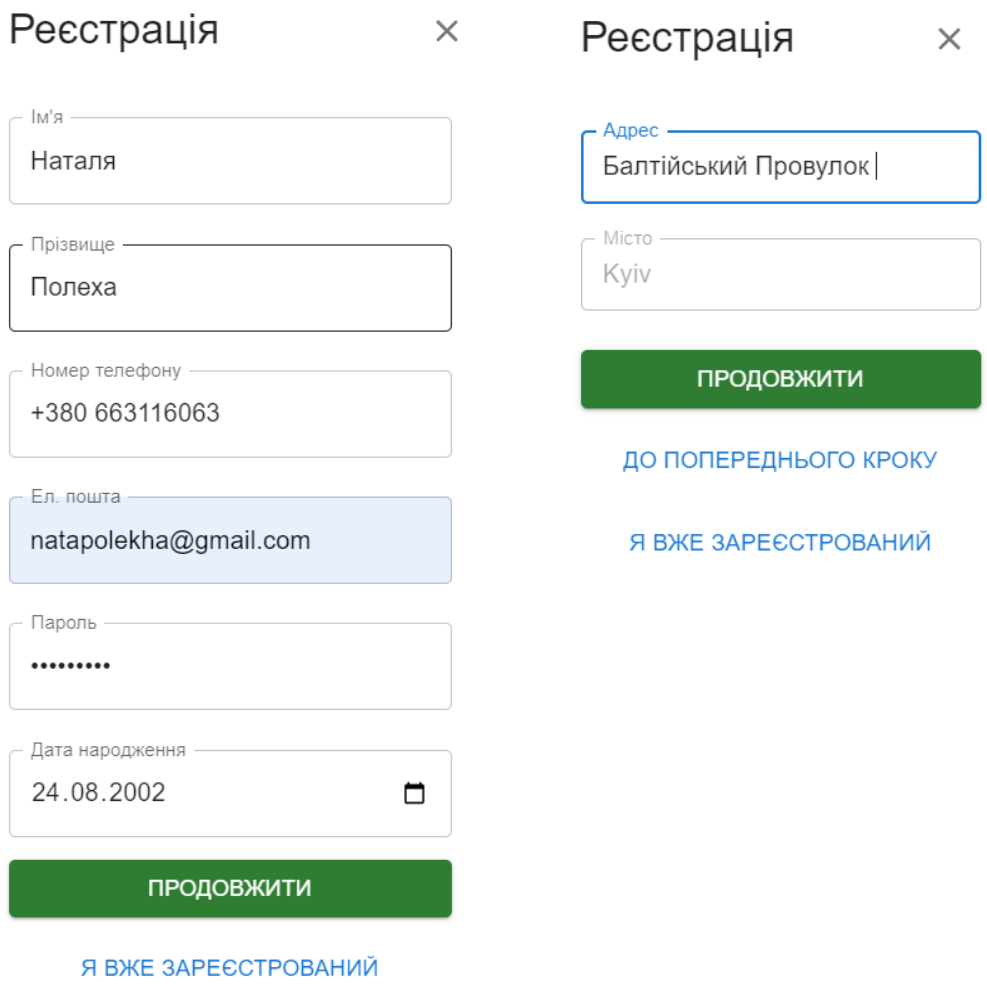


Рисунок 4.1 – Форма реєстрації донора

Важливою частиною розробки клієнтського інтерфейсу є зручність у використанні, саме тому для заповнення різних типів даних повинні бути розроблені відповідні поля. Наприклад поле для заповнення дати повинне давати

користувачу можливість обрати рік, місяць та день за допомогою зручного календаря (рис.4.2).

Рисунок 4.2 – Поле для зручного вибору дати

Важливою також є розробка поля для вибору адреси проживання користувача. Окрім того, що поле повинне бути зручним у використанні, воно повинно забезпечити ввід коректних даних (в нашому випадку існуючої адреси). Для розробки цього поля була використана бібліотека mapbox, яка надає доступ до бази безкоштовних картографічних даних. На рис. 4.3 зображено поле для заповнення адреси, яке підлаштовується під введений користувачем текст.

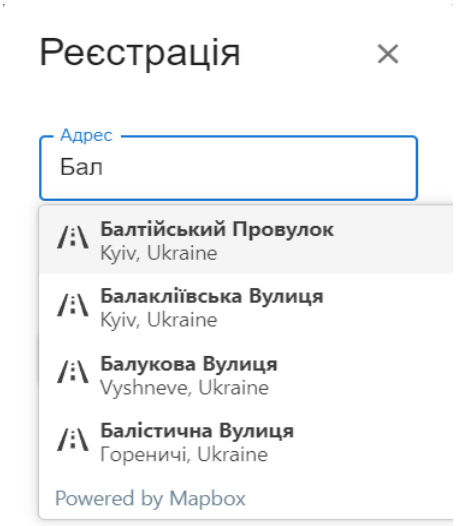


Рисунок 4.3 – Поле для зручного вибору адреси

Наступним важливим фактором розробки форми є валідація даних (перевірка коректності). Для цього була використана бібліотека Jup, яка дозволяє описувати схеми валідації, де можуть бути присутні такі особливості поля як: обов'язковість, максимальна/мінімальна кількість символів, валідний адрес електронної пошти, тощо. На рис. 4.4 видно, що якщо обов'язкові поля не будуть заповненими, форма не дозволить користувачу відправити запит на реєстрацію.

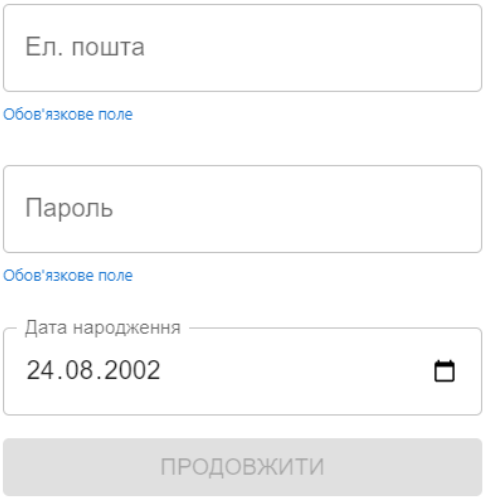


Рисунок 4.4 – Валідація форми

Після коректного заповнення форми запит на реєстрацію користувача відправляється на сервер (рис. 4.5).

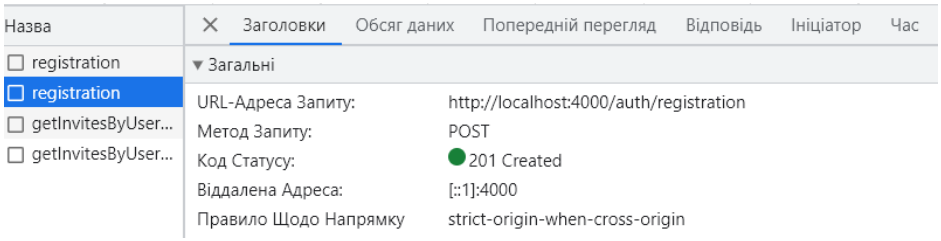


Рисунок 4.5 – Відправка запиту на реєстрацію

Розробка серверної частини, також передбачає валідацію запиту, адже він може бути відправленим не тільки з клієнтської частини, але й з API-платформ таких як Postman чи SoapUI. Валідація в NestJs здійснюється за допомогою Пайпів (Pipes), які були описані вище. Наступним кроком відбувається перевірка існування донора з введеною адресою електронної пошти, якщо такий існує,

падає помилка з кодом 400. В іншому випадку створюється неверифікований донор, тобто колонка verified в таблиці Users буде дорівнювати false. Важливо відмітити, що пароль зберігається в базі в захешованому вигляді. Запит на реєстрацію користувача повертає токен, де зберігається найважливіша інформація про користувача така як ідентифікаційний номер (id), роль, статус верифікації та адрес електронної пошти.

Важливою частиною забезпечення конфіденційності є верифікація користувача. Для цього потрібно реалізувати можливість надсилання листів на електронну пошту. В моєму випадку це здійснюється за допомогою бібліотеки mailgun. На сервері генерується код верифікації з восьми випадкових символів та надсилається на пошту (рис. 4.6).

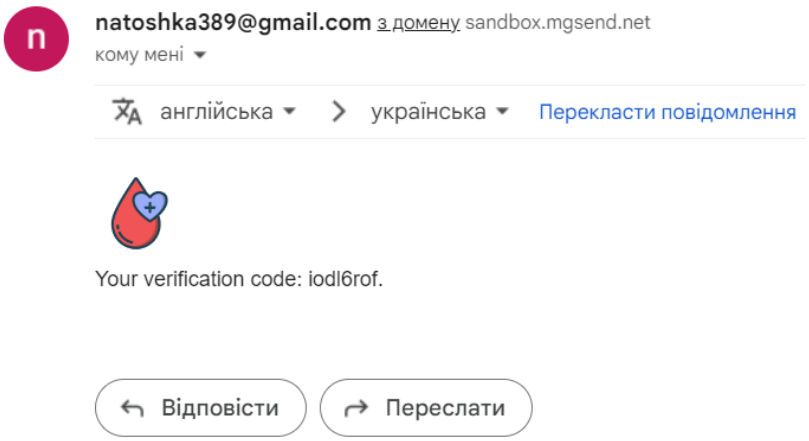


Рисунок 4.6 – Лист з кодом верифікації

За допомогою форми для верифікації (рис. 4.7) йде запит на верифікацію користувача.

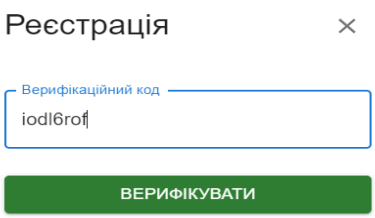
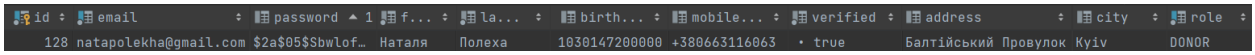


Рисунок 4.7 – Форма верифікації користувача

Лише після верифікації користувач отримає доступ до функціоналу в додатку. Приклад створеного верифікованого донора в базі наверений на рис. 4.8.



id	email	password	f...	la...	birth...	mobile...	verified	address	city	role
128	natapolekha@gmail.com	$2a$05$Sbwlof…	Наталя	Полеха	1030147200000	+380663116063	true	Балтійський Провулок	Kyiv	DONOR

Рисунок 4.8 – Приклад верифікованого донора в базі

4.1.2 Авторизація донора

Авторизація є простішим процесом, адже не потребує валідації особистих даних донора. Запит на авторизацію користувача приймає адресу електронної пошти донора та його пароль. Ці дані заповнюються за допомогою форми наведеної на рис. 4.9.

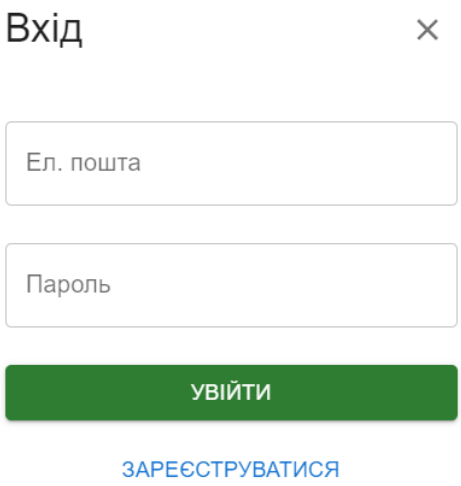


Рисунок 4.9 – Форма авторизації

На сервері пароль з запиту хешується за допомогою бібліотеки bcrypt та порівнюється з хешом пароля відповідного користувача в базі. Якщо захешовані паролі дорівнюють одне одному, користувач отримує доступ до своєї сторінки.

4.1.3 Додавання донора асистентом чи адміном

Другий спосіб створення донора є додавання його в базу асистентом. Асистент може додати особисті та медичні дані донора виключно за згодою самого донора. Верифікація здійснюється таким же чином як і у випадку реєстрації, лише окрім коду верифікації на пошту приходить і пароль

користувача, який так само генерується з восьми випадкових чисел та зберігається в базі у захешованому вигляді.

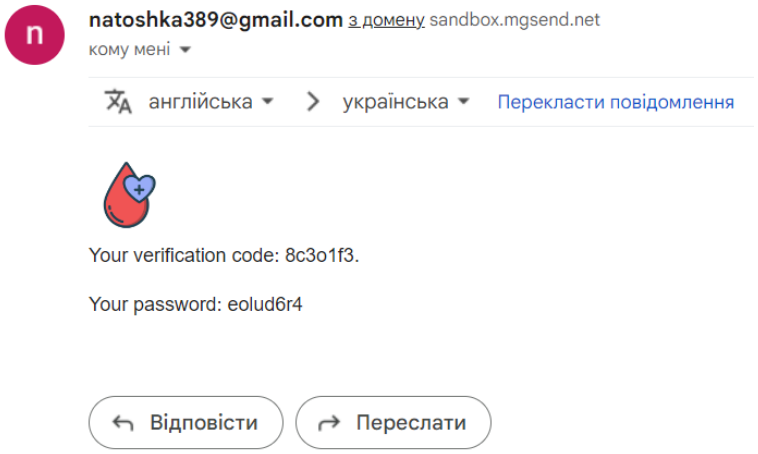


Рисунок 4.10 – Лист з кодом верифікації та паролем користувача

Лише після верифікації асистент зможе заповнити медичну інформацію донора, а саме загальну інформацію про донації таку як група крові, кількість донацій, дата останньої донації, тип та метод останньої донації. Форма для заповнення цієї інформації наведена на рис. 4.11.

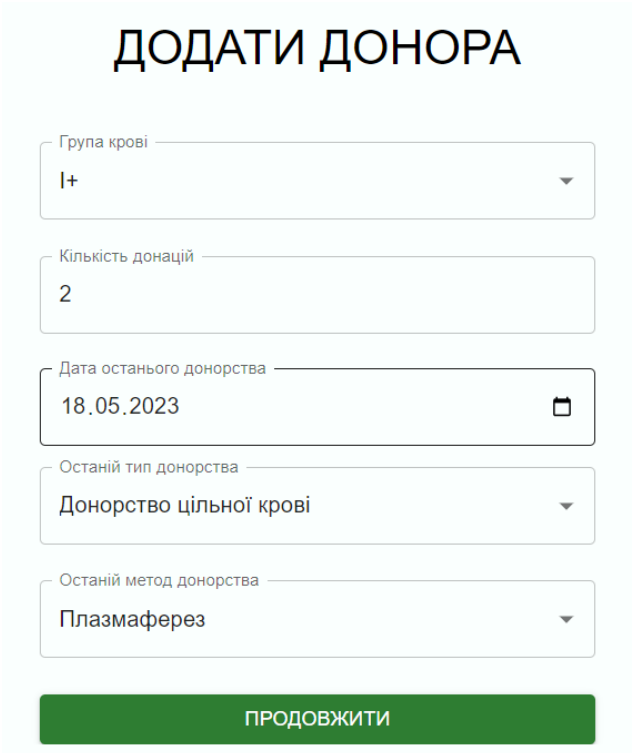


Рисунок 4.11 – Фома для заповнення загальної донорської інформації

4.1.4 Перегляд усіх донорів в базі

Доступ до перегляду всіх донорів в базі мають користувачі з роллю ADMIN та ASSISTANT. Для забезпечення належної перевірки ролей та доступів був використаний компонент архітектури NestJs – Guard, який використовується для забезпечення автентифікації, авторизації та контролю доступу до різних ресурсів додатку. Для отримання інформації про донорів відправляється Get-запит на сервер (рис. 4.12).

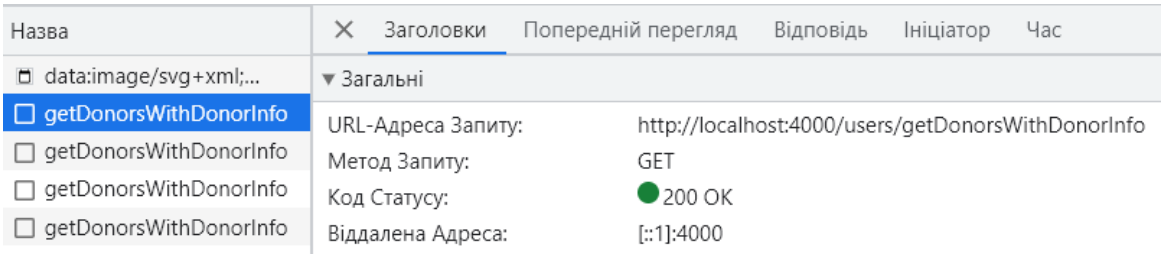


Рисунок 4.12 – Відправка запиту на отримання всіх донорів

Результатом запиту є масив об’єктів з особистою та медичною інформацією про донорів (рис. 4.13).

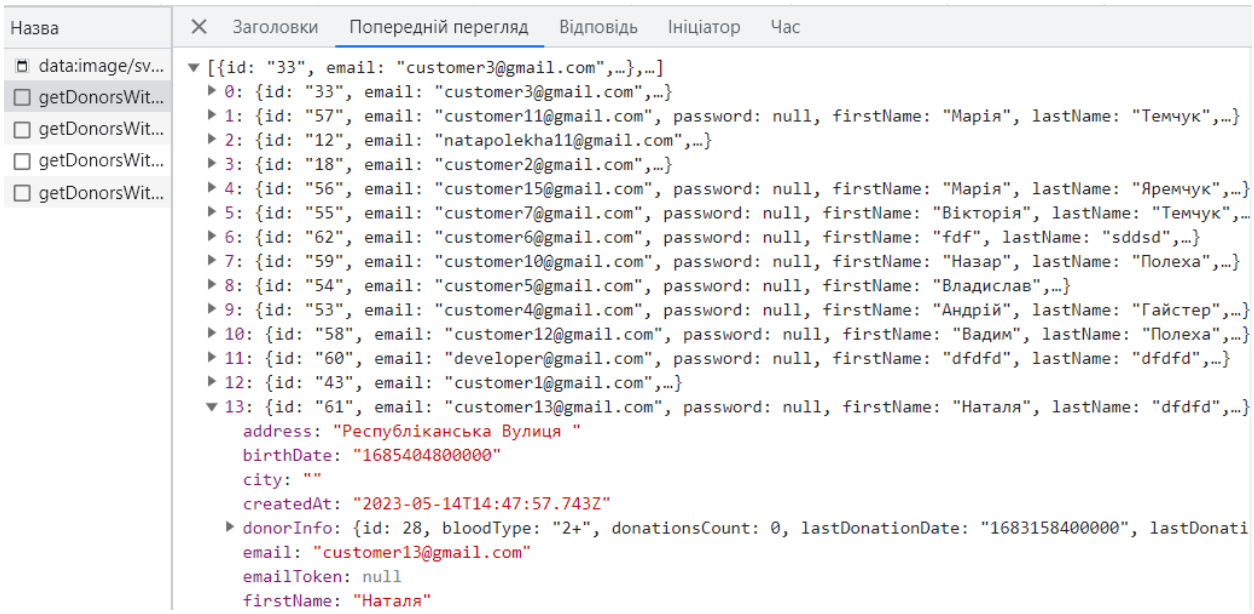


Рисунок 4.13 – Результат запиту на отримання всіх донорів

На сайті ця інформація для зручності відображається у вигляді таблиці (рис. 4.14).

Id	Ел. пошта	ПІБ	Дата народження	Верифікований	Група крові	Кількість донацій	Дата останього донорства	Останій тип донорства	Останій метод донорства	Місто	Редагувати
33	customer3@gmail.com	Назар Мельник	2023-05-16	☐	II-	3	2023-05-01	Донорство еритроцитів	Звичайна здача крові	Kyiv	РЕДАГУВАТИ
57	customer11@gmail.com	Марія Темчук	2023-05-17	☐	I+	0	2001-01-01	-	-	Bricquebec	РЕДАГУВАТИ
12	natapolekha11@gmail.com	Маряна Полеха	2023-05-17	☑	II-	2	2023-05-01	Донорство плазми	Тромбоцитаферез		РЕДАГУВАТИ
18	customer2@gmail.com	Вадим Атаманюк	2001-01-01	☑	IV+	5	2023-04-11	Донорство цільної крові	Звичайна здача крові	Kharkiv	РЕДАГУВАТИ
56	customer15@gmail.com	Марія Яремчук	2023-05-24	☑	II-	0	2023-05-23	Донорство тромбоцитів	Звичайна здача крові	Lamboing	РЕДАГУВАТИ
55	customer7@gmail.com	Вікторія Темчук	2023-05-24	☑	II-	0	2023-05-23	Донорство тромбоцитів	Звичайна здача крові	Lamboing	РЕДАГУВАТИ
62	customer6@gmail.com	fdf sddsd	2023-06-02	☑	I-	0	2023-05-02	-	-	Слобода	РЕДАГУВАТИ

Рисунок 4.14 – Відображення донорів в додатку

4.1.5 Редагування інформації про донора

Редагувати інформацію про донора може адміністратор та асистент. Донор також може змінити особисті та медичні дані на своїй сторінці. В меню донор може зайти в профіль, де буде форма для обновлення особистих та медичних даних (рис. 4.15).

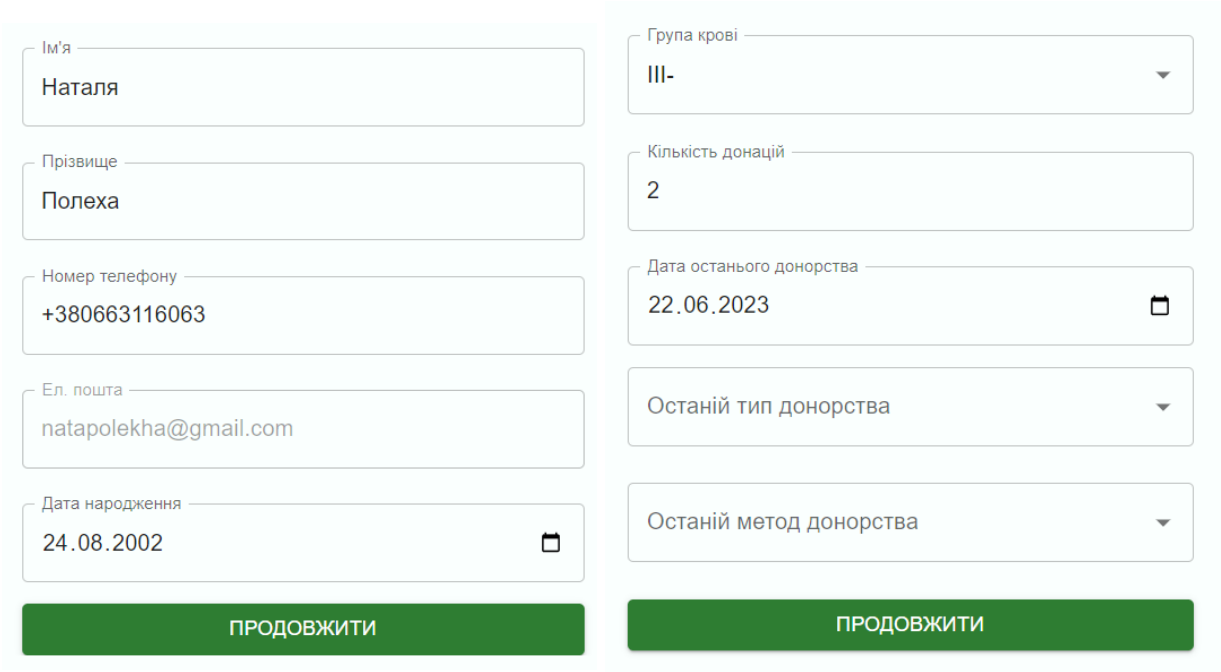


Рисунок 4.15 – Форма редагування донорської інформації

4.1.6 Створення історії донацій

Створення історії про донацію відбувається безпосередньо після здачі матеріалу донором. В реєстратурі клініки асистент додає донора в базу, якщо його ще немає в системі та створює історію донації. Історії донацій зберігаються в таблиці DonorHistories та посилаються на таблицю Users. Таким чином, у донора може бути декілька історій донацій, тому що кожна історія еквівалентна кожній здачі матеріалу. Основними полями, які необхідно заповнити для створення історії є дата донорства, ідентифікатор донора, ідентифікатор пункту здачі крові, метод та тип донорства, а також стан донора після донації. Форма для створення історії про донацію наведена на рис. 4.16.

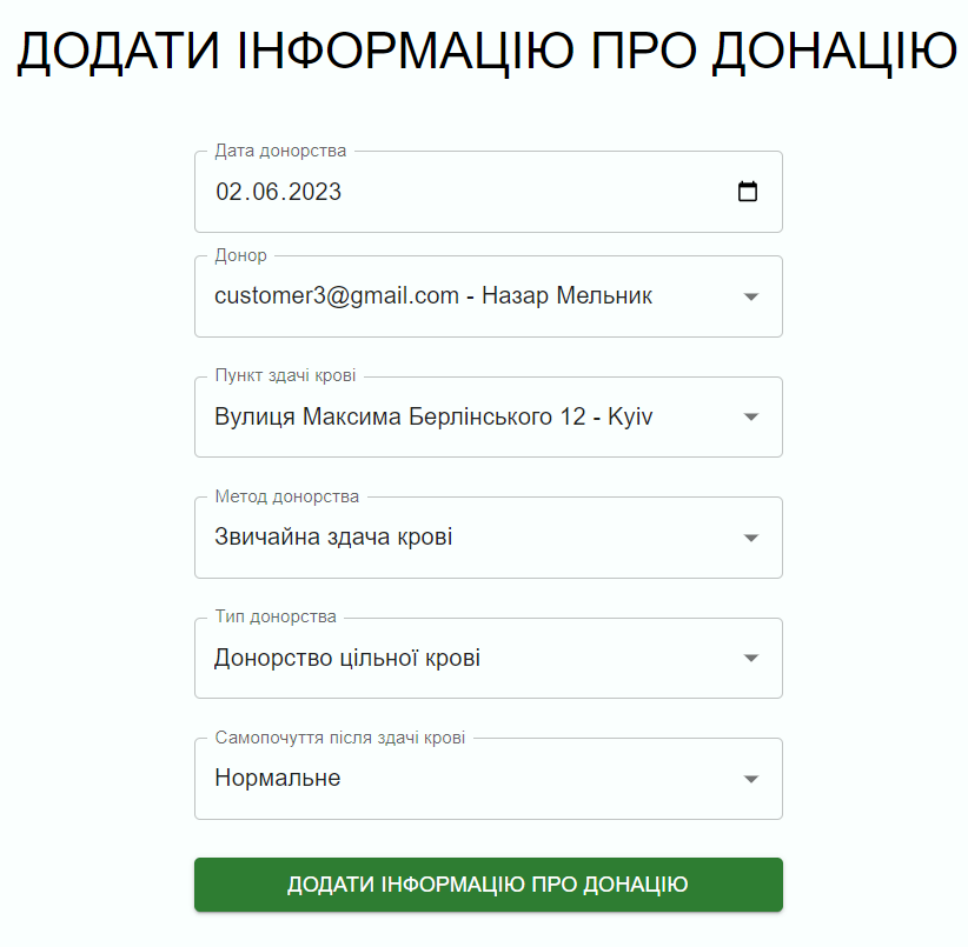


Рисунок 4.16 – Форма створення історії донації

Для відображення полів з вибором донора та центра донації (рис. 4.17) на базу відправляються запити на отримання цих записів.

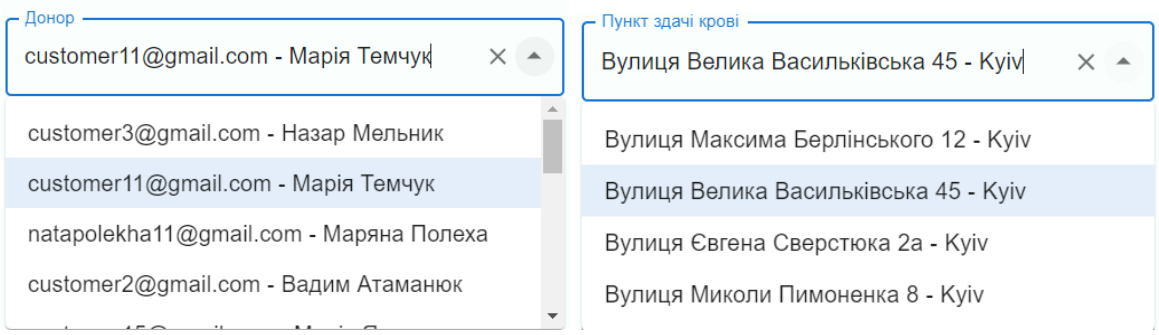


Рисунок 4.17 – Поля для вибору донора та центра донації

4.2 Опис реалізації та тестування функціоналу пошуку та залучення донорів в критичних ситуаціях

Доступ до методу пошуку та залучення донорів в критичних ситуаціях має лише користувач з роллю ADMIN.

Для наглядної демонстрації виконання процесу пошуку та залучення донорів в критичних ситуаціях, буде розглянутий приклад роботи алгоритму з використанням тестових даних.

Для виконання запиту адміністратор повинен спочатку заповнити форму, де йому необхідно буде ввести такі дані як дата до якої слід здати кров, ідентифікатор центру донації, необхідна група крові, необхідний тип донорства та кількість матеріалу. Форма наведена на рис. 4.18.

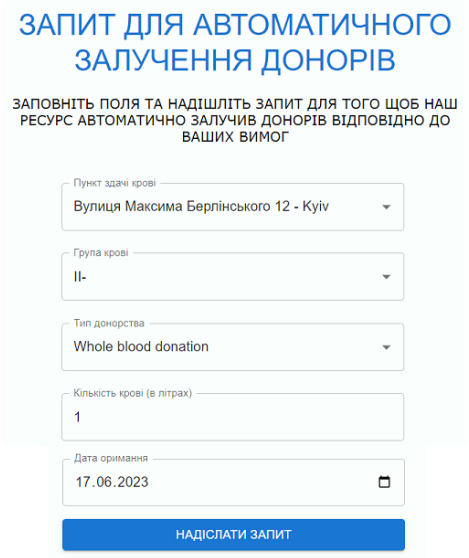


Рисунок 4.18 – Форма запиту для залучення донорів

Після надсилання запиту, необхідно реалізувати валідацію даних на сервері, якщо всі дані коректні відбувається процес пошуку та залучення оптимальних донорів.

Першим кроком була реалізована перевірка існування центру донації в базі, код наведений на рис. 4.19. Якщо центру не існує, створюється запис в таблиці Requests, де колонка result буде дорівнювати error (рис. 4.20).

```
// 1.1) validate donation point
const donationPoint = await this.donationPointsService.getOneDonationPoint( filter: {
    where: {id: involveDonorsDto.donationPoint, city: {[Op.not]: null}},
});
// create request with error result and throw error if donation point is not founded
if (!donationPoint) {
    await this.createRequest(involveDonorsDto, RequestResult.error, donationPoint.id, senderId);
    throw new HttpException( response: 'ID003: There are no donation points with these id', HttpStatus.BAD_REQUEST);
}
```

Рисунок 4.19 – Код перевірки існування центру донації

id	donationPointId	bloodType	donationType	bloodAmount	receivingDate	senderId	result
12	3	2-	wholeBloodDonation	1	1693785600000	31	error

Рисунок 4.20 – Запис в таблиці Requests про помилкове завершення запиту

Наступним кроком відбувається перевірка кінцевої дати збору донорського матеріалу. Дата повинна бути не пізніше ніж за 90 та не раніше ніж за 2 дні після подання запиту. Код перевірки дати збору матеріалу наведений на рис. 4.21. Якщо дата некоректна, створюється запис в таблиці Requests, де колонка result буде дорівнювати error.

```
// 1.2) validate receivingDate from InvolveDonorsDto
if (involveDonorsDto.receivingDate < dateNow + ONE_DAY_INTERVAL) {
    await this.createRequest(involveDonorsDto, RequestResult.error, donationPoint.id, senderId);
    throw new HttpException(
        response: 'ID001: The receiving date should be at least a day before tomorrow', HttpStatus.BAD_REQUEST);
}
if (involveDonorsDto.receivingDate > dateNow + NINETY_DAY_INTERVAL) {
    await this.createRequest(involveDonorsDto, RequestResult.error, donationPoint.id, senderId);
    throw new HttpException(
        response: 'ID002: The receiving date should be not more than after 90 days', HttpStatus.BAD_REQUEST);
}
```

Рисунок 4.21 – Код перевірки дати збору матеріалу

Далі необхідно реалізувати розрахунок кількості залучених донорів в залежності від кількості донорського матеріалу. Середня кількість матеріалу отриманого при донації дорівнює 450мл = 0,45л. Розрахунок необхідної кількості донорів наведений на рис. 4.22.

```
// 1.3) get required donors count
const requiredDonorsCount = this.getRequiredDonorsCount(involveDonorsDto.bloodAmount);

private getRequiredDonorsCount = (bloodAmount: number): number => {
    return Math.ceil( x: bloodAmount / averageBloodAmountOfDonation);
}

export const averageBloodAmountOfDonation = 0.45;
```

Рисунок 4.22 – Код для розрахунки необхідно ї кількості залучених донорів

Далі відбувається пошук верифікованих донорів за такими критеріями як група крові та місто проживання (повинно відповідати місту розташування центру донації). Код пошуку наведений на рис. 4.23.

```
// 1.4) find donors with needed blood type
// 1.5) find donors with the same city as donation point city
const users = await this.usersService.getDonorsWithDonorInfo(
    userWhere: {city: donationPoint.city, verified: true},
    donorInfoWhere: {bloodType: involveDonorsDto.bloodType}
);

// create request with notFound result and throw error if donors not founded
if (!users?.length) {
    await this.createRequest(involveDonorsDto, RequestResult.notFound, donationPoint.id, senderId);
    throw new HttpException( response: 'ID004: Unfortunately, donors are not found', HttpStatus.NOT_FOUND);
}
```

Рисунок 4.23 – Код первинного пошуку оптимальних донорів

Ці критерії є обов’язковими, тому якщо запит не повертає масив донорів, створюється запис в таблиці Requests, де колонка result буде дорівнювати notFound (рис. 4.24).

id	donationPointId	bloodType	donationType	bloodAmount	receivingDate	senderId	result
24	7	2-	plateletDonation	0.4	1689206400000	31	notFound

Рисунок 4.24 – Запис в таблиці Requests про завершення запиту без знаходження оптимальних донорів

В результаті виконання коду наведеного на рис. 4.23 формується SQL-запит (рис. 4.25).

```
SELECT *
FROM "users" AS "User"
    INNER JOIN "donorInfo"
        ON "User"."id" = "donorInfo"."userId" AND "donorInfo"."bloodType" = '2-'
    LEFT OUTER JOIN "donorHistories" AS "history"
        ON "User"."id" = "history"."userId"
    LEFT OUTER JOIN "invites"
        ON "User"."id" = "invites"."receiverId"
WHERE "User"."role" = 'DONOR' AND "User"."city" = 'Kyiv' AND "User"."verified" = true;
```

Рисунок 4.25 – Приклад SQL-запиту для пошуку оптимальних донорів

Враховуючи вміст бази запит поверне 23 записи наведених на рис. 4.26.

23 rows — XML

	"User".id	donorInfo.id	email		"firstName"	"lastName"	"birthDate"	role	verified
1	151	52	customer29@gmail.com	$2…	Олександра	Дудченко	1685577600000	DONOR	true
2	137	38	customer153@gmail.com	$2…	Вікторія	Яремчук	1685577600000	DONOR	true
3	136	37	customer143@gmail.com	$2…	Марія	Темчук	1685577600000	DONOR	true
4	149	50	customer27@gmail.com	$2…	Євген	Лісовенко	1685577600000	DONOR	true
5	142	43	customer20@gmail.com	$2…	Вадим	Петренко	1685577600000	DONOR	true
6	12	12	natapolekha11@gmail.com	$2…	Маряна	Полеха	1684281600000	DONOR	true
7	146	47	customer24@gmail.com	$2…	Катерина	Феленюк	1685577600000	DONOR	true
8	139	40	customer17@gmail.com	$2…	Василь	Терехов	1685577600000	DONOR	true
9	143	44	customer21@gmail.com	$2…	Назар	Кузняр	1685577600000	DONOR	true
10	145	46	customer23@gmail.com	$2…	Василина	Шпотюк	1685577600000	DONOR	true
11	140	41	customer18@gmail.com	$2…	Іван	Мельник	1685577600000	DONOR	true
12	153	54	customer31@gmail.com	$2…	Дмитро	Вихватнюк	1685577600000	DONOR	true
13	56	23	customer15@gmail.com	$2…	Марія	Яремчук	1684886400000	DONOR	true
14	152	53	customer30@gmail.com	$2…	Анастасія	Полюлях	1685577600000	DONOR	true
15	155	56	customer33@gmail.com	$2…	Павло	Поночовний	1685577600000	DONOR	true
16	156	57	customer34@gmail.com	$2…	Максим	Головко	1685577600000	DONOR	true
17	141	42	customer19@gmail.com	$2…	Владислав	Патока	1685577600000	DONOR	true
18	138	39	customer16@gmail.com	$2…	Валиль	Шевчено	1685577600000	DONOR	true
19	148	49	customer26@gmail.com	$2…	Соломія	Дуденко	1685577600000	DONOR	true
20	147	48	customer25@gmail.com	$2…	Софія	Філіжанко	1685577600000	DONOR	true
21	154	55	customer32@gmail.com	$2…	Петро	Чорновіл	1685577600000	DONOR	true
22	150	51	customer28@gmail.com	$2…	Євгенія	Сосновська	1685577600000	DONOR	true
23	144	45	customer22@gmail.com	$2…	Олександер	Значков	1685577600000	DONOR	true

Рисунок 4.26 – Виконання запиту первинного пошуку оптимальних донорів

Наступним кроком відбувається фільтрація знайдених донорів по існуванню запрошення на випадок, якщо донор уже був залучений. Код фільтрації наведений на рис. 4.27.

```
// 1.6) filter users by invites
const donorsWithoutInvites = users.filter((user: User) =>
    user?.invites.find((invite: Invite) => (invite.requestId !== involveDonorsDto?.requestId))
);
```

Рисунок 4.27 – Код фільтрації донорів по наявності запрошення

Далі відбувається фільтрація донорів по даті можливої наступної здачі, наприклад донор який здавав кров 02.06.23 не зможе повторити донацію до 02.08.23. В файлі з костанками зберігаються значення інтервалів очікування перед наступною здачею. Детальніше про інтервали між донайціями можна дізнатися з таблиці, де цифри – це кількість днів, після яких можна продовжити здавати кров. Таблиця зображена на рис. 4.28 взята з офіційного сайту Центру громадського здоров'я МОЗ України. Код фільтрації наведений на рис. 4.29.

Назва процедури	Здавання крові	Плазмаферез	Тромбоцитоферез
Здавання крові	60	30	30
Плазмаферез апаратний (доза: 500–800 мл)	14	14	14
Плазмаферез ручний одинарний	7	7	7
Плазмаферез ручний подвійний	14	14	14
Тромбоцитоферез	14	14	14

Рисунок 4.28 – Інтервали між донайціями [2]

```
// 1.7) filter by valid donation interval
const validDonors = this.getDonorsWithValidDonationInterval(
    donorsWithoutInvites.filter(donor => donor?.donorInfo),
    involveDonorsDto.receivingDate
);

private getDonorsWithValidDonationInterval(
    donors: User[],
    receivingDate: number,
): User[] {
    return donors.filter((donor: User) => {
        const lastDonationDate = +donor.donorInfo.lastDonationDate;

        // if the donor has never donated blood, it is no need to check donation interval
        if (!lastDonationDate) {
            return true;
        }

        // accessible submit start date
        const accessibleSubmitStartDate = this.getAccessibleSubmitStartDate(lastDonationDate, donor);

        return accessibleSubmitStartDate < receivingDate - ONE_DAY_INTERVAL;
    });
}
```

Рисунок 4.29 – Код фільтрації донорів по даті можливої наступної здачі

Після фільтрації донорів кількість оптимальних донорів зменшилась з 23 до 17 (рис. 4.30).

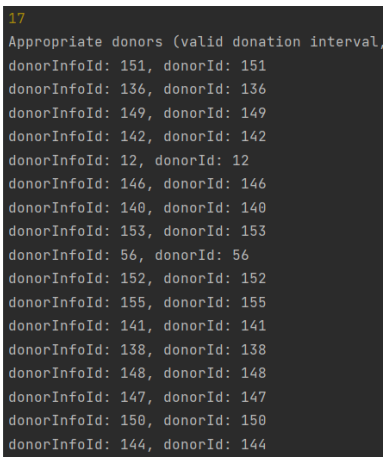


Рисунок 4.30 – результат фільтрації донорів

Якщо кількість донорів після фільтрації менша необхідної, створюється запис в таблиці Requests, де колонка result буде дорівнювати notFound. Якщо кількість донорів дорівнює необхідній, відбувається залучення донорів, яке буде описане нижче та створюється запис в таблиці Requests, де колонка result дорівнюватиме found.

Якщо ж кількість відфільтрованих донорів більша за необхідну відбувається оцінка донорів за певними критеріями. Таким чином, записи з більшою вагою і будуть більш оптимальними.

Перший критерій оцінки – кількість часу для здачі матеріал, принцип його впливу був описаний у попередньому розділі. Загалом чим більше часу для донора здати кров, тим більша вірогідність здачі.

Для кращої демонстрації розрахунку цього критерію, процес буде описаний нижче на прикладі одного донора.

Запит який присилає адміністратор передбачає збір матеріалу до 17.06.23. Дата надсилання запиту – 07.06.23. Тобто у донора є максимум 10 днів для здачі матеріалу. Слід зазначити, що деякі матеріали такі як плазма можуть зберігатися без кріоконсервації лише три дні, тому важливо враховувати час збереження матеріалу в центрі донації. Це може вплинути на максимальний можливий час здачі матеріалу донорами. В нашому випадку центр донації містить кріоконсервацію, тому час зберігання матеріалу більше 1000 днів. Таким чином,

у випадку тестового перебігу алгоритму, значення часу збереження матеріалу в центрі донації не буде впливати на максимально можливий час донації. Тобто після визначення часу збереження, можливий час здачі матеріалу залишається 10 днів. Наступні значення визначаються особисто для кожного донора. До уваги слід взяти значення останньої здачі матеріалу донором, так як це може зменшити можливий час здачі. В нашому випадку донор з ідентифікатором 146, останній раз здавав кров 12.04.23, тому наступна його донація може бути не раніше ніж 12.06.23, таким чином кількість можливого часу здачі зменшується з десяти до п’яти днів. Для збільшення амплітуди залежності критеріїв, було вирішено домножити всі значення на 1000. Тобто вага описаного донора дорівнюватиме 5 * 1000 = 5000. Значення ваги всіх донорів з прикладу наведені на рис. 4.31.

```
Donor submit interval weight mapping:
userId: 56 - 9000
userId: 138 - 9000
userId: 139 - 9000
userId: 140 - 9000
userId: 141 - 9000
userId: 142 - 9000
userId: 143 - 5000
userId: 144 - 5000
userId: 145 - 3000
userId: 146 - 5000
userId: 149 - 8000
userId: 150 - 9000
userId: 151 - 5000
userId: 152 - 9000
userId: 153 - 5000
userId: 154 - 9000
userId: 156 - 1000
```

Рисунок 4.31 – Вага донорів в залежності від можливого часу здачі

Наступним критерієм оцінки є відстань від центру донації до місця проживання донора, адже чим менша ця відстань, тим більша вірогідність здачі. Для визначення відстані, необхідно знати координати адреси центру донації та адреси місця проживання донора. Визначити критерії на основі адреси допоможе mapbox API, робота з яким була описана вище. На рис. 4.32 наведено програмний код отримання координат переданої адреси.

```
private async getAddressCoordinates(address: string, city: string): Promise<{longitude: number, latitude: number}> {
    const addressWithCity = `${address} ${city}`
    const addressInfo = await this.httpService.get(
        url: `${mapboxPlacesRequest(addressWithCity)}${mapBoxTokenQueryParam}`
    ).toPromise();

    const longitude = Array.isArray(addressInfo?.data?.features[0]?.geometry?.coordinates) ?
        addressInfo?.data?.features[0]?.geometry?.coordinates[0] : null;
    const latitude = Array.isArray(addressInfo?.data?.features[0]?.geometry?.coordinates) ?
        addressInfo?.data?.features[0]?.geometry?.coordinates[1] : null;

    return {longitude, latitude};
}
```

Рисунок 4.32 – Код визначення координат адреси за допомогою mapbox API

Відстань місця проживання донорів від центру донації наведена на рис. 4.33.

```
Distances from donor to donation center:
userId: 56, distance 0.8133703962000883
userId: 138, distance 0.26738245903695823
userId: 139, distance 0.10862069851520123
userId: 140, distance 0.0640089901469969
userId: 141, distance 1.3622641748120268
userId: 142, distance 0.09627178372022802
userId: 143, distance 1.3622641748120268
userId: 144, distance 0.10862069851520123
userId: 145, distance 1.365707392861284
userId: 146, distance 0.18809291733528946
userId: 149, distance 0.24471756702168776
userId: 150, distance 0.03364740594800226
userId: 151, distance 0.03364740594800226
userId: 152, distance 0.03364740594800226
userId: 153, distance 0.03364740594800226
userId: 154, distance 0.03364740594800226
userId: 156, distance 0.03364740594800226
```

Рисунок 4.32 – Відстань місця проживання донорів він центра донації

Для кращої демонстрації впливу критерію розглянем відповідність на прикладі одного з донорів. Наприклад відстань від місця проживання донора з ідентифікатором 56 до центру донації дорівнює 0,813. Середня відстань для всіх донорів дорівнює 0,364. Коефіцієнт залежності відстані, який є константним значенням дорівнює 150. Процес розрахунку коефіцієнту в коді зображений на рис. 4.33.

```
Object.entries(distanceFromDonationCenterMapping).forEach((mapping : [string, unknown] ) => {
    donorWeightMapping[mapping[0]] -= +((Number(mapping[1]) * coefficientForAverageDistance) / averageDistance)
        .toFixed(digitsAfterTheDecimalPointNumber);
}, {});
```

Рисунок 4.33 – Код визначення ваги донорів в залежності від відстані

Вага донора з ідентифікатором 56 дорівнює (0,813 * 150) / 0,364, що приблизно дорівнює 335. Попередня вага цього донора, яка була 9000 зменшується на 335 і буде дорівнювати 8665. Вага всіх донорів в залежності від критерію відстані та часу можливої здачі наведена на рис. 4.34.

```
Donor submit interval and distance weight mapping:
userId: 56 - 8664.82
userId: 138 - 8889.815
userId: 139 - 8955.239
userId: 140 - 8973.623
userId: 141 - 8438.627
userId: 142 - 8960.328
userId: 143 - 4438.627
userId: 144 - 4955.239
userId: 145 - 2437.208
userId: 146 - 4922.489
userId: 149 - 7899.155
userId: 150 - 8986.134
userId: 151 - 4986.134
userId: 152 - 8986.134
userId: 153 - 4986.134
userId: 154 - 8986.134
userId: 156 - 986.134
```

Рисунок 4.34 – Вага всіх донорів в залежності від критерію відстані та часу можливої здачі

Третім критерієм від якого залежить вага донорів є кількість донацій, принцип його визначення є еквівалентним розрахунку ваги в залежності від відстані. Значення кількості донацій для кожного донора наведено на рис. 4.35, а їх вага після додавання ваги в залежності від кількості донацій наведена на рис. 4.36.

```
Donors donations count:
userId: 151 - 1
userId: 149 - 3
userId: 142 - 1
userId: 146 - 1
userId: 139 - 1
userId: 143 - 1
userId: 145 - 1
userId: 140 - 1
userId: 153 - 2
userId: 56 - 1
userId: 152 - 2
userId: 156 - 5
userId: 141 - 1
userId: 138 - 1
userId: 154 - 2
userId: 150 - 3
userId: 144 - 1
```

Рисунок 4.35 – Кількість донацій кожного донора

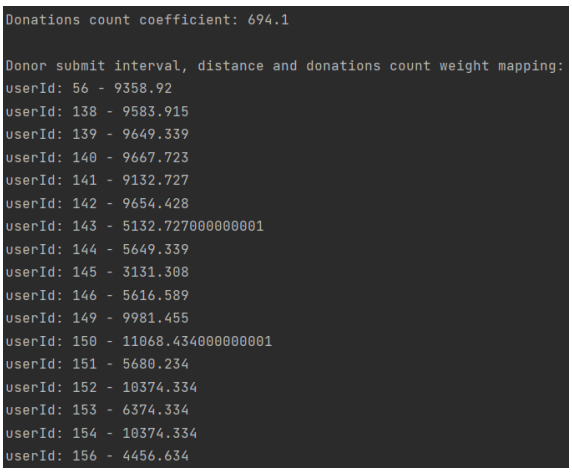


Рисунок 4.36 – Вага всіх донорів в залежності від критерію відстані, часу можливої здачі та кількості здач

Наступними критеріями є кількість відхилених запрошень, що зменшує вагу донора та критерій стану здоров’я донора після донації. Після підрахунку ваги всіх критеріїв отримуємо наступний результат (рис. 4.37).

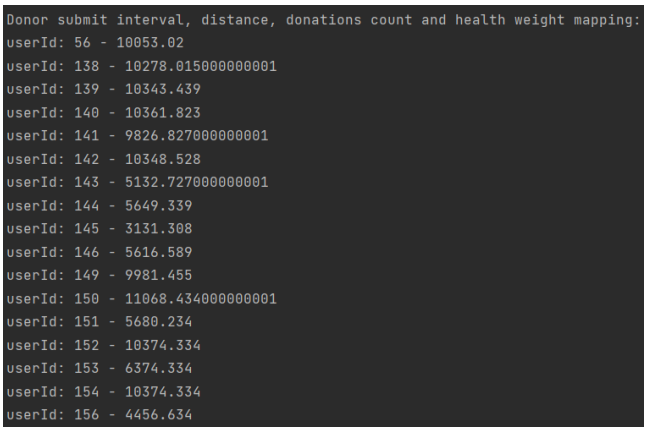


Рисунок 4.37 – Вага всіх донорів в залежності від критерію відстані, часу можливої здачі, кількості здач, кількості відхилених запрошень та стану здоров’я після донацій

Далі відбувається сортування всіх донорів від найбільшої ваги до найменшої і обирається необхідна кількість донорів з найбільшою вагою. В нашому випадку залученими будуть наступні три донора (рис. 4.38).

```
Donors to invite:
donorInfoId: 51, donorId: 150
donorInfoId: 55, donorId: 154
donorInfoId: 53, donorId: 152
```

Рисунок 4.38 – Ідентифікатори найоптимальніших донорів

Для залучення донорів необхідно налаштувати надсилання листів на електронну пошту донора. Опис бібліотеки, яка буде використана та її особливості висвітлені у попередньому розділі. Для відправки листа необхідно реалізувати генерацію повідомлення, шаблон якого наведений на рис. 4.39. Приклад надісланого листа наведений на рис. 4.40.

```
const html = `
    <p>
        ${header}<br/>
        Добрий день, ${user.firstName}.<br/>
        Просимо Вас стати донором крові в центрі донації за адресою ${donationPoint.address} (${donationPoint.city}).<br/>
        Перейдіть за посиланням <a href="${webDomen}">${webDomen}</a> та увійдіть на свій профіль.<br/>
        Підтвердьте або відхиліть запрошення протягом ${stringifiedTimeToWaitBeforeApprove}.<br/>
        На сайті Ви також зможете детальніше ознайомитись з типом донації.<br/>
        Донацію необхідно здійснити до ${receivingDate}.<br/>
    </p>
`;
```

Рисунок 4.39 – Шаблон запрошення

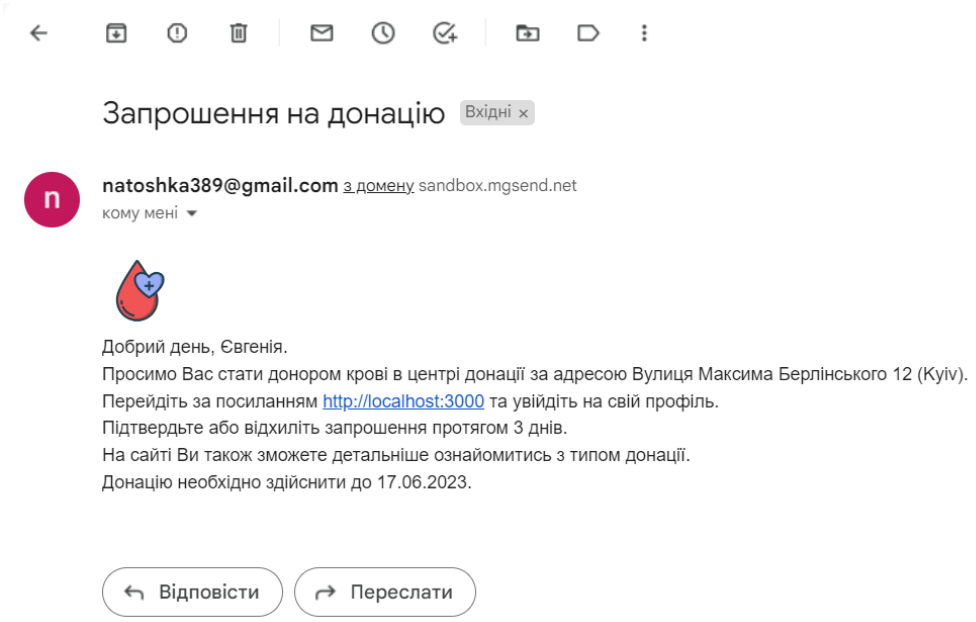


Рисунок 4.40 – Приклад листа з запрошенням на донацію

Для відображення запрошень на сторінці користувача необхідно створити в таблиці Invites записи про запрошення. Також необхідно реалізувати інтерфейс для перегляду запрошень. Якщо донору приходить запрошення на донацію, на його сторінці відображатиметься іконка сповіщення (рис. 4.41), при натисканні на яку буде відображатися запрошення (рис. 4.42).

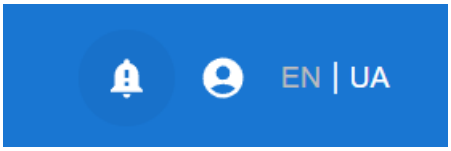


Рисунок 4.41 – Сповіщення користувача

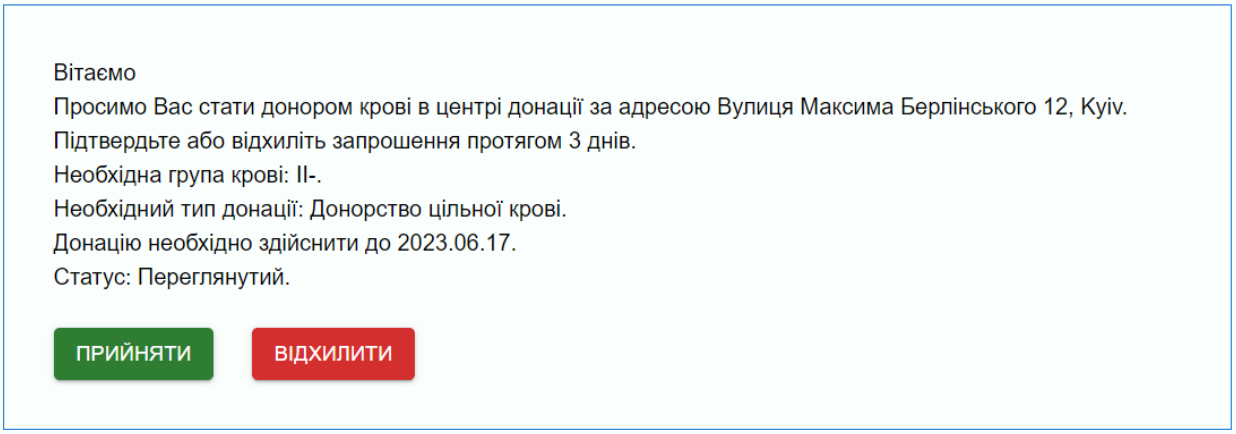


Рисунок 4.42 – Запрошення на донацію на особистій сторінці донора

Протягом трьох днів донор повинен прийняти запрошення, в іншому випадку воно автоматично буде відхилене.

Статус запрошення може бути чотирьох видів:

- notReviewed – не переглянуте;

- reviewed – переглянуте, запрошення набуває цього статусу якщо донор зайшов на сторінку запрошень, проте не натиснув ні прийняти, ні відхилити;

- accepted – прийняте

- deflected – відхилене

Для детальнішого перегляду кожного запиту та його результатів було розроблено API документацію, яку можна переглянути запустивши серверну частину та перейшовши за наступним посиланням –

http://localhost:4000/api/docs/. API документація була розроблена за допомогою Swagger, що робить її зручною для розуміння. Загальний вигляд деяких запитів зображений на рис. 4.43.

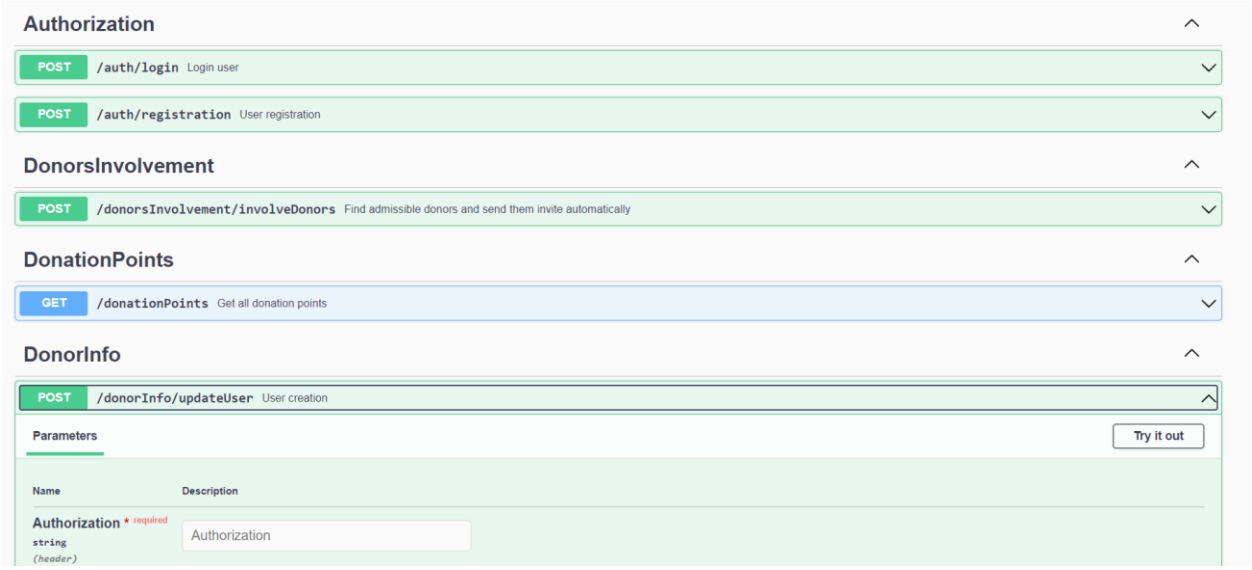


Рисунок 4.43 – Частина розробленої API документації додатку

4.3 Рекомендації щодо подальшого вдосконалення

Рекомендації для вдосконалення системи обліку та залучення донорів і покращення її ефективності:

- Звітність

Розробити систему звітів, яка надасть детальну інформацію про донорські внески, розподіл коштів та використання ресурсів. Звіти допоможуть зрозуміти ефективність системи обліку донорів і забезпечити прозорість перед донорами та зацікавленими сторонами.

- Зручність

Покращення користувацького інтерфейсу та забезпечення зручного онлайн-доступу до інформації про донорські внески буде сприяти залученню донорів.

- Розвиток взаємодії з донорами

Створити зручні канали комунікації з донорами, окрім електронної пошти такі як соціальні мережі або чат-боти. Це дозволить збирати зворотний зв'язок, вирішувати питання та побажання донорів, а також підтримувати зв'язок з ними.

- Аналітика

Використання інструментів аналізу даних для виявлення тенденцій, розуміння поведінки донорів та визначення ефективних стратегій залучення та утримання донорів. Аналітичні дані допоможуть приймати кращі рішення та планувати майбутні дії.

Висновки до розділу 4:

В даному розділу було описано процес реалізації основного функціоналу з наданням великої кількості скриншотів інтерфейсу. Також було проведено тестування основних функцій додатку з наданням прикладів результатів запитів. Детально було оглянуто функціонал обліку, пошуку та залучення донорів в систему, а саме такі процеси як реєстрація та авторизація донора, додання донора в систему адміністратором чи асистентом, перегляд усіх донорів, редагування інформації про донорів та створення історій донацій. Також було детально оглянуто процес пошуку та залучення донорів в критичних ситуаціях та наведено приклад роботи методу.

ВИСНОВКИ

В даній дипломній роботі було проведено ретельний аналіз предметної області, пов'язаної з обліком та залученням донорів. Також були проаналізовані відомі програмні рішення в даній області.

На основі аналізу вимог до програмного забезпечення було сформульовано постановку задачі і описано основний бізнес-процес, який підлягає автоматизації. Було розроблено діаграму прецидентів для кращого відображення структури інформаційної системи.

У розділі проектування системи обліку та залучення донорів був проведений огляд засобів реалізації, вибір інструментів для розробки клієнтської та серверної частини, а також бази даних. Була розроблена архітектура проекту, включаючи проєктування таблиць бази даних. Також були проведені моделювання алгоритмів для пошуку та залучення донорів в критичних ситуаціях, а також для створення та верифікації донорів.

В розділі реалізації та тестування системи було описано розробку функціоналу обліку донорів, включаючи реєстрацію, авторизацію, додавання та редагування інформації про донорів та створення історії донацій. Також було реалізовано функціонал пошуку та залучення донорів в критичних ситуаціях. У завершальному розділі були надані рекомендації щодо подальшого вдосконалення системи обліку та залучення донорів.

Таким чином, розроблена система інформаційного та програмного забезпечення є ефективним інструментом для здійснення обліку та залучення донорів, що сприятиме покращенню роботи центрів донації та медичних закладів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. JWT.io - веб-сайт інструментарію для роботи з JWT (JSON Web Tokens). URL: https://jwt.io/ (дата звернення: 08.06.2023).
2. Центр громадського здоров'я України - Розділ "Донорство крові та компонентів" URL: https://phc.org.ua/promociya-zdorovya/donorstvo-krovi-ta-ii-komponentiv/dlya-donoriv (дата звернення: 08.06.2023).
3. Офіційна документація по бібліотеці JavaScript – React URL: https://uk.reactjs.org/ (дата звернення: 08.06.2023).
4. Опис мови програмування JavaScript URL: https://uk.wikipedia.org/wiki/JavaScript (дата звернення: 08.06.2023).
5. Документація модуля Axios URL: https://axios-http.com/ (дата звернення: 08.06.2023).
6. Документація бібліотеки React Router URL: https://reactrouter.com/ (дата звернення: 08.06.2023).
7. Mike Cantelon, Marc Harter, T.J. Holowaychuk "Node.js in Action"
8. Shelley Powers "Learning Node.js"
9. Основні парадигми програмування URL: https://studfile.net/preview/5994723/ (дата звернення: 08.06.2023).
10. Formik open source form library URL: https://formik.org/ (дата звернення: 08.06.2023).
11. Redux documentation URL: https://redux.js.org/ (дата звернення: 08.06.2023).
12. React redux documentation URL: https://react-redux.js.org/ (дата звернення: 08.06.2023).
13. Maps and location for developers URL: https://www.mapbox.com/ (дата звернення: 08.06.2023).
14. Mapbox Geocoding API URL: https://docs.mapbox.com/api/search/geocoding/ (дата звернення: 08.06.2023).

15. MailGun documentation URL: https://www.mailgun.com/ (дата звернення: 08.06.2023).
16. Sequelize documentation URL: https://sequelize.org/ (дата звернення: 08.06.2023).
17. NestLS documentation URL: https://nestjs.com/ (дата звернення: 08.06.2023).
18. Best Way to Structure Your Directory/Code (NestJS) URL: https://medium.com/the-crowdlinker-chronicle/best-way-to-structure-your-directory-code-nestjs-a06c7a641401 (дата звернення: 08.06.2023).
19. NestJS Folder Structure Best Practices URL: https://climbtheladder.com/10-nestjs-folder-structure-best-practices/ (дата звернення: 08.06.2023).
20. Typescript documentation URL: https://www.typescriptlang.org/ (дата звернення: 08.06.2023).
21. Learn TypeScript URL: https://www.freecodecamp.org/news/learn-