Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра інформаційних систем та технологій
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня магістра

Студента Максютенко Леоніда Едуардовича
(ПІБ)

академічної групи 123м-22-1
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)

на тему «Обґрунтування структури та розробка комп'ютерної системи із детальною проробкою інтерфейсу користувача для донорського центру "Biopharma"»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Бешта Д.О.			
розділів:				
теоретичний розділ	доц. Бешта Д.О.			
синтез системи	доц. Бешта Д.О.			
розроблення програмного забезпечення	ас. Панферова Я.В.			
Рецензент				
Нормоконтролер	доц. Шедловська Я.І.			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)

"06" вересня 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра
(бакалавра, спеціаліста, магістра)

студенту Максютенко Л.Е. **академічної групи** 123м-22-1
(прізвище та ініціали) (шифр)

спеціальності 123 «Комп'ютерна інженерія»

за освітньо-професійною програмою 123 «Комп'ютерна інженерія»
(офіційна назва)

на тему «Обґрунтування структури та розробка комп'ютерної системи із детальною проробкою інтерфейсу користувача для донорського центру "Biopharma"»,

затверджену наказом ректора НТУ «Дніпровська політехніка» від 09.10.2023р. № 1227-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	10.10.2023
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	25.10.2023
Синтез системи	Розробка комп'ютерної системи	15.11.2023
Розроблення програмного забезпечення	Розробка програмного забезпечення	29.11.2023
Експериментальний розділ	Проведення і обробка результатів експериментів	06.12.2023

Завдання видано _____ доц. Бешта Д.О.
(підпис керівника) (прізвище, ініціали)

Дата видачі 06 вересня 2023 р.

Дата подання до екзаменаційної комісії 13.12.2023

Прийнято до виконання _____ Максютенко Л.Е
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 124 с., 71 рис. , 16 табл., 1 дод., 19 джерел.

ДОНОРСЬКИЙ ЦЕНТР, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ВЕБ-ЗАСТОСУНОК, PYTHON, DJANGO, POSTGRESQL

Об’єкт розробки: комп’ютерна система донорського центру «Biopharma».

Мета роботи: обґрунтування та розробка комп’ютерної системи донорського центру «Biopharma» із детальною проробкою інтерфейсу користувача та відповідної бази даних.

У пояснювальній записці здійснено всебічний аналіз проблематики, пов'язаної з функціонуванням комп'ютерної системи донорського центру, в якому були виявлені недоліки. Ці відомості стали основою для формулювання цілей та завдань дослідження.

У теоретичному розділі роботи було проведено аналіз актуальних тенденцій у розробці програмного забезпечення, включаючи вибір мов програмування, розгляд веб-фреймворків, ретельний аналіз та обґрунтування використання реляційних СКБД, а також визначення вимог до КС та ПЗ.

У розділі «Синтез системи» було проведено розробку структурної та функціональної схеми комп'ютерної системи донорського центру, які наглядно відображають всі компоненти центру та їх взаємозв'язки.

У розділі «Розроблення програмного забезпечення» надано докладний опис розробленого веб-застосунку, включаючи його основні характеристики функціоналу.

В експериментальному розділі проведено тестування розробленого веб-додатку з метою оцінки його ефективності, коректності функціонування інтерфейсу та БД.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення;

МОЗ – міністерство охорони здоров’я;

NHS – national health service;

ЗПСШ – захворювання, що передаються статевим шляхом;

BLOB – binary large object;

БД – бази даних;

СКБД – система керування базами даних.

ORM – Object-Relational Mapping;

DTL – Django Template Language;

GPL – General Public License ;

ВСТУП

У сучасному світі галузь медицини визначається своєю надзвичайною динамікою та високим рівнем технологічного розвитку, що постійно вдосконалюється. Донорські центри, які спеціалізуються на зборі та обробці донорської крові та її компонентів, відіграють ключову роль у гарантуванні медичним установам стабільного та постійного доступу до необхідних медичних ресурсів.

Один із прикладів такого центру – це донорський центр під назвою «Biopharma», спеціалізований на зборі та обробці донорської плазми крові. Діяльність цього центру охоплює не лише фізичний збір плазми крові, але й систематизацію та управління інформацією, пов'язаною з донорами та результатами аналізів, що становить важливу складову процесу надання високоякісних медичних послуг.

Актуальність даного дослідження полягає у тому, що оптимізація процесів управління інформацією про донорів та результатами аналізів крові може значно покращити роботу донорського центру та підвищити ефективність його функціонування. Вдосконалення системи обліку донорів сприятиме більш точному веденню медичної інформації, що, в свою чергу, сприятиме покращенню процедур збору крові та забезпеченню важливих медичних ресурсів. Оптимізація процесів аналізу крові та швидкий доступ до цих результатів можуть значно полегшити проведення донорство та забезпечити швидку реакцію у випадках надзвичайних потреб. Такі заходи допоможуть покращити загальну роботу центру, сприяючи забезпеченню належного та ефективного донорства крові та її компонентів.

Об’єкт дослідження – діяльність донорського центру «Biopharma».

Предмет дослідження – процес обліку донорів та їх результатів аналізів крові.

Методи дослідження. Для досягнення поставленої мети в даному дослідженні використовувалися методи, спрямовані на розробку веб-застосунків. Серед них були використані методи програмування та розробки програмного забезпечення з використанням мов програмування, таких як Python, і використання фреймворку Django для створення веб-додатків. Крім того, використовувалися методи проектування баз даних, таких як PostgreSQL, для зберігання та управління інформацією про донорів.

Ідея дослідження: створення веб-застосунку з використанням БД для керуванням і контролю доступу до інформації, та використання мережі для обміну інформації між сервером та користувачами.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Стисла характеристика галузі охорони здоров’я та об’єкта впровадження

Система охорони здоров’я була і зараз є однією з важних частин інфраструктури держави, яка забезпечує підтримку і покращення стану здоров’я населення. Данна система складається з законодавчих, наукових, організаційних та медичних об’єктів.

МОЗ має повноваження з управління та нагляду за цією системою. Воно займається розробкою стратегій у галузі охорони здоров'я, контролює якість медичних послуг, що надаються, і забезпечує доступ населення до ліків і вакцин.

Досить важливою частиною галузі охорони здоров'я є донорські центри, які забезпечують різними біологічними матеріалами для різних медичних процедур. У не кризовий час, щоб забезпечити кров'ю та її компонентами населення, необхідний 1% від усього населення, це приблизно тридцять постійних донорів на тисячу осіб. А під час повномасштабного вторгнення росії на території України, необхідність крові та її елементів зросла ще на 60%. Адже їх потребуватимуть не тільки звичайні пацієнти, а й військові та мирні, які постраждали під час бойових дій [1].

Одним з підприємств на території України, які збирають кров та її компоненти є Biopharma. Дане підприємство було створено в 1895 році, тоді вона спеціалізувалось на розробці та виготовлення ліків. Відносно нещодавно, в 2010х роках підприємство почало відкривати донорські центри зборів крові та її компонентів. На стан 2023 року, донорські центри цього підприємства збирають приблизно двісті тисяч літрів плазми у рік. Донори які зробили донацію, отримують результати крові на загальні відхилення, та на наявність ЗПСШ [2]. Один з таких центрів розташовано за адресою, м.Дніпро, просп. Богдана Хмельницкого, 17.

Підприємство «Biopharma» активно займається різноманітними діяльностями, спрямованими на розвиток у даній сфері, а саме:

– Збір та зберігання компонентів крові. Компанія активно шукає різні способи отримати компоненти крові, одним з таких способів було відкриття власних донорських центрів крові, на території всієї України.
– Виробництво фармацевтичної продукції. Biopharma спеціалізується на розробці ліків, які націлені на відновлення організму після інфекційних захворювань і також після травм, пов'язаних з крововтратою.
– Оптова і роздроблена торгівля. До повномасштабного вторгнення росії, компанія була націлена на продаж в Європейські країни, але зараз компанія активно націлена на підтримку ринку на територіях України.

Підприємство «Biopharma» виявляє великий інтерес у власних донорських центрах, розташованих на території України, які виконують важливу функцію у забезпеченні основною сировиною для виробництва фармацевтичних препаратів. Зазначено десять таких центрів, які становлять ключовий елемент ланцюга постачання компанії.

Центри цілеспрямовано здійснюють збір плазми, яка є основним джерелом сировини, однак також проводять збір цільної крові та тромбоцитів, розширюючи спектр отримуваних матеріалів. Кожна донорська одиниця крові піддається обов'язковій лабораторній перевірці, здійснюваній відповідно до встановлених стандартів якості.

Отримана сировина, яка відповідає високим стандартам якості, підлягає негайному зберіганню у спеціально обладнаних морозильних камерах. Температурний режим цього зберігання забезпечується утриманням температури на рівні -20 градусів за Цельсієм, що є необхідним заходом для забезпечення збереження властивостей та якості зібраної сировини. Такий комплексний підхід до управління донорськими центрами підкреслює

вагомість їхньої ролі в процесі виробництва високоякісних медичних препаратів.

1.2 Аналіз наявної проблеми донорського центру

Процес передачі цільної крові або її компонентів включає в себе кілька послідовних етапів, починаючи з реєстрації донора. Для того щоб донор міг здійснити донацію, йому необхідно зареєструватися в черзі. У поточному стані донорського центру це досягається шляхом звертання до центру та вказання часу та особистої інформації. Цей процес змушує працівників відволікатися від своєї роботи для відповіді на надходженні дзвінки.

Після реєстрації донор відвідує донорський центр, де заповнює анкету та чекає, доки його викличуть. У цьому місці проводиться перевірка даних, після чого відбувається взяття крові для попереднього аналізу. Наступним етапом є звернення до терапевта, де проводиться медичний огляд для визначення можливості передачі крові або її компонентів.

Після цього відвідувач направляється в приміщення для здачі зразків крові та її компонентів з метою подальшого відправлення їх у незалежну лабораторію. У лабораторії проводяться аналіз крові на загальні показники та на захворювання, що передаються шляхом статевого контакту. Результати цих аналізів повертаються до донорського центру, який може передати цю інформацію донору у фізичній формі.

Протягом роботи донорського центру виникла суттєва проблема, пов'язана із збільшеним навантаженням та затримками через постійні відвідування донорів, які мають намір отримати інформацію щодо результатів своєї крові. Ці візити стали регулярністю, оскільки донорам відсутні можливості отримати інформацію віддалено. На додачу, утворюються довгі черги, що ускладнюють роботу персоналу центру та призводять до додаткових непорозумінь серед інших відвідувачів. Працівники, які приділяють додатковий час відвідувачам, які прибули за результатами, втрачають зможу

відстежувати загальний стан донорів, які в той час проходять процедуру донації.

Слід відзначити, що донори також витрачають значний час на досягнення центру, що може викликати незадоволення та дискомфорт, що, в свою чергу, може здати на відступ донорів від подальших спроб здати кров чи її компоненти.

Дані обставини значно впливають на продуктивність функціонування донорського центру. З метою вирішення цих проблем підприємству рекомендується впровадити систему зберігання та передачі інформації донорам через спеціальні програмні рішення. Це сприятиме зменшенню навантаження на персонал, розширенню доступу донорів до своїх результатів, а також сприяє зменшенню часових витрат як для донорів, так і для працівників центру.

1.3 Завдання та мета роботи

Мета роботи – обґрунтування та розробка комп'ютерної системи донорського центру «Biopharma» із детальною проробкою інтерфейсу користувача та відповідної бази даних.

Для вирішення поставленої мети в роботі вирішуються наступні завдання:

- аналіз та вибір СКБД відповідно до поставленої мети;
- розробка структури бази даних для підприємства;
- вибір мови програмування для розробки програмного забезпечення;
- аналіз вимог обладнання для розроблюваного програмного забезпечення;
- розробка програмного забезпечення з інтерфейсом користувача.
- аналіз та вибір тестування програмного забезпечення;
- розробка тестових наборів для валідації програмного забезпечення.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Аналіз існуючих рішень

2.1.1 Аналіз веб-сторінки

Прикладом існуючого рішення онлайн платформи для донорського центру є веб-сайт від National Health Service (NHS) Blood and Transplant, Give Blood. Ця платформа розроблена для інформування та спрощення реєстрації і отримання результатів крові для донорів, на території Об’єднаного Королівства. Даний сайт має такий функціонал:

1. створення облікового запису донора, швидка та зручна форма реєстрації на сайті;
2. інформаційні матеріали, загальна інформація яка може знадобитися донору, та цікаві факти;
3. пошук місць для здачі крові або її компонентів, пошук місць виконується інтерактивним пошуковим полем або завдяки мапі;
4. реєстрація в черзі на здачу крові та її компонентів;
5. звіти результатів крові та статистика виконаних процедур донором, дозволяє швидко отримати інформацію щодо їх стану здоров’я та надає загальну інформацію скільки було здано крові та її компонентів, та надає інформацію о даті коли донор може знову здати кров або її компоненти.

За допомогою цього веб-сайту, донори можуть приєднатися до донорства без прив'язки до конкретного донорського центру, оскільки він підтримується Національною Служби Здоров’я Сполученого Королівства [16].

2.1.2 Аналіз мобільного застосунку

Іншим прикладом вирішення є мобільний застосунок від Американського червоного хреста «Blood Donor», що дозволяє донорам швидко та зручно створити обліковий запис та отримувати інформацію щодо

їх результатів у телефоні. Нижче зазначено перелік функціоналу мобільного застосунку:

– пошук пунктів переливання крові і центрів здачі крові;
– резервування місця в пунктах і центрах;
– перевірка результатів фізичного огляду;
– повідомлення про нехватку крові та нагадування про заплановані зустрічі;
– відслідковування про загальну кількість зданої крові та відтворення історії стану здоров’я донора у вигляді таблиць;
– накопичення значків за пожертвування [15].

2.2 Визначення напрямку рішення поставлених завдань

З урахуванням вище зазначених проблем, можливо розглянути впровадження веб-платформи, через яку донори зможуть отримувати доступ до результатів крові та інших важливих повідомлень через свій особистий обліковий запис. Також необхідно забезпечити онлайн запис на процедуру донації, що полегшить процес вставання в чергу для донорів і зменшить навантаження на персонал.

Вибір мови програмування має суттєвий вплив на розробку комп'ютерної системи з використанням веб-фреймворків. Він визначає можливості взаємодії з апаратним забезпеченням, швидкістю розробки та іншими аспектами. Для донорського центру, вибір мови програмування є важливим фактором, який вплине на успішність реалізації проекту. JavaScript та Python мови програмування, які часто обирають для розробки програмного забезпечення з використанням веб-фреймворків та мають велику підтримку спільнотами.

Використання веб-фреймворку має різноманітні переваги, які структурують та оптимізують процес розробки продукту. Перевагами слід вказати швидкість розробки, наявність рівня безпеки, можливості простого масштабування.

Наявність у веб-фреймворках рівня безпеки мінімізує загрози та небезпеки, пов'язані з уразливими місцями, що могли б з'явитися під час розробки.

Масштабованість фреймворків дає змогу легко адаптувати функціонал веб-сайту під свої потреби. Також веб-фреймворк дає змогу через готові рішення типових завдань, уникати рутинних завдань.

Для веб-фреймворку одним з ефективних способів зберігати та обробляти дані є використання реляційних СКБД.

Реляційні СКБД - це система управління даними, що представлені в організованій структурі у вигляді таблиць, де кожен стовпець – це атрибут, а рядок – це запис. Зв'язки між таблицями організовані в зовнішніх ключах, які пов'язують дані за допомогою посилань на стовпці інших таблиць [3].

Використання реляційної СКБД для веб-фреймворку має такі переваги:

– Структурованість. Завдяки структуруванню даних за допомогою таблиць, дані мають чіткі відносини між різними частинами, що полегшує організацію та управління інформацією;
– Нормалізація. Реляційні СКБД дають змогу використовувати принципи нормалізації даних, що забезпечить можливість уникати дублювання інформації та аномалій під час змін;
– Підтримка ORM. Більшість веб-фреймворків мають Object-Relational Mapping (ORM), він дає розробникам працювати безпосередньо з об'єктами в коді програми, а не виконувати SQL запити. Це спрощує розробку програмного забезпечення завдяки використанню об'єктно-орієнтованого підходу.

2.3 Аналіз мов програмування

Веб-розробка в сучасному світі посіла важливу роль, являючи собою ключовий аспект сучасних обчислень. Це стало можливим завдяки розробці веб-сайтів і веб-додатків, які служать різним цілям. У контексті веб-розробки вибір мови програмування є ключовим фактором, що визначає успішність

кінцевого результату. Кожна з мов програмування має свої унікальні сильні та слабкі сторони, які можуть значно покращити ефективність, масштабованість і зручність веб-проектів.

JavaScript – це мова сценаріїв та програмування, яка забезпечує можливість реалізації складних функцій на веб-сторінках. Її активне використання дає змогу впроваджувати інтерактивність і динамічну функціональність у веб-сторінки, надаючи користувачеві можливість взаємодії з контентом у режимі реального часу, минаючи перезавантаження всієї сторінки. Серед багатьох додатків JavaScript виділяється розробка одно сторінкових додатків (SPA), що являють собою веб-застосунки, які оновлюють вміст сторінки динамічно в процесі взаємодії користувача, не вимагаючи повного перезавантаження [6].

Проте, для використання JavaScript у серверній частині, необхідне середовище виконання Node.js. Це середовище виконання дозволяє виконувати код JavaScript на сервері, що значно розширює сферу застосування мови за межами клієнтської веб-розробки.

З використанням Node.js розробники можуть створювати ефективні серверні додатки та API, а також виконувати різноманітні завдання на стороні сервера. Така уніфікація мови програмування як на клієнтському, так і на серверному боці, що забезпечується JavaScript і Node.js, сприяє спрощенню розроблення та обслуговування веб-додатків, надаючи універсальний інструментарій для створення сучасних веб-технологій.

Переваги JavaScript:

- відкритий вихідний код;
- підтримка класів, інтерфейсів і модулів;
- універсальність використання на клієнті та сервері;
- можливість створення великих додатків за допомогою середовища виконання;
- взаємодія з даними на рівні користувача;
- негайний зворотний зв'язок і реакція на дії користувачів.

Недоліки JavaScript:

– обмежені можливості роботи з файловою системою;

– відсутність багатопроцесорної многопоточності;

– слабка продуктивність під час складних обчислень.

Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Останнім часом ця мова програмування набуває значної популярності в галузі веб-розробки. Вона вирізняється своєю простотою, легкою читабельністю та універсальністю, що сприяє її широкому застосуванню в різних аспектах веб-розробки, як-от обробка веб-сторінок, аналіз даних і машинне навчання.

Однією з ключових переваг Python є його простота використання. Текст програми, написаний на Python, вирізняється читабельністю і легкістю написання, що робить його кращим вибором як для новачків, так і для досвідчених розробників. Крім того, Python є універсальною мовою програмування, маючи великий набір бібліотек і платформ для розв'язання різноманітних завдань у сфері веб-розробки. Поступово Python завойовує популярність у таких сферах, як наука про дані та машинне навчання, роблячи його невід'ємною навичкою для сучасних веб-розробників.

Переваги Python:

– простий синтаксис;

– великий набір бібліотек та інструментів;

– багатозадачність і многопоточність;

– гнучкість;

– кросплатформеність.

Недоліки Python:

– залежність від інтерпретатора;

– високі вимоги до ресурсів пристрою;

– відсутність суворої типізації даних.

У таблиці 2.1 наведено головні відмінності вище згаданих мов програмування [4].

Таблиця 2.1 - Відмінності мов програмування Python і JavaScript

Мова програмування	JavaScript	Python
Типи даних	Незмінні	Змінні та незмінні
Кодування	UTF-16	ASCII
Числові типи даних	Присутні числа з плаваючою комою	Цілочислові, з плаваючою комою, комплексні
Синтаксис	Фігурні дужки	Відступи
Модель успадкування	Спадкування на основі прототипів	Успадкування на основі класів
Області застосування	Мобільні та веб-додатки	Аналіз даних, машинне навчання, математичні операції, веб-додатки

2.4 Аналіз веб-фреймворків

Express.js – це веб-фреймворк, побудований на основі Node.js, що забезпечує великий набір інструментів для розробки веб-застосунків. Він застосовується у створенні як одно сторінкових, так і багатосторінкових веб-застосунків. Express.js є абстракцією поверх Node.js, надаючи зручні засоби управління серверами та визначення маршрутів.

Express.js було сформовано з метою полегшення розроблення API та веб-застосунків, забезпечуючи явне скорочення часу кодування при збереженні спроможності ефективного створення веб-застосунків. Перевага полягає також у тому, що Express.js використовує мову JavaScript, яка відома своєю простотою, забезпечуючи доступність для розробників без попереднього досвіду в мовах програмування. Express.js спрощує входження нових розробників у сферу веб-розробки.

Express.js було концептуалізовано з урахуванням кількох фундаментальних цілей:

– значне зменшення обсягу коду, що зменшує витрати часу на кодування, водночас забезпечуючи розробку ефективних веб-застосунків;
– розробка мовою програмування JavaScript, яка є простою в освоєнні;
– використання асинхронних можливостей Node.js, підвищуючи ефективність обробки паралельних операцій.

Express.js демонструє значні можливості, що сприяють ефективній розробці веб-додатків на серверній частині. Зокрема, інтеграція функціональності Node.js з Express.js сприяє помітному заощадженню часу під час проектування серверної складової застосунку. Крім цього, використання проміжного програмного забезпечення в Express.js являє собою засоби обробки запитів і відповідей, що збагачує функціональність застосунку.

Додатково, Express.js демонструє високу ефективність у сфері маршрутизації, забезпечуючи ефективне управління тим, як кінцеві точки URL програми взаємодіють із запитами від клієнтів. Ця функціональність, своєю чергою, забезпечує оптимальну навігацію всередині програми.

Механізми шаблонізації, вбудовані в Express.js, надають можливості для створення динамічного контенту на веб-сторінках. Цей підхід здійснюється шляхом генерації HTML-шаблонів на стороні сервера, що сприяє ефективнішій розробці та обслуговуванню динамічних веб-сторінок.

Крім того, Express.js забезпечує зручності в процесі налагодження, ідентифікуючи місце розташування помилок у додатку з високою точністю. Цей аспект спрощує завдання розробників, сприяючи ефективнішому виявленню та усуненню проблемних моментів у процесі розробки.

Переваги Express.js:

– незалежність;
– можливість використання Middleware;
– використання однієї мови для фронтенд і бекенд розробки;

– динамічно відображає HTML-сторінки на основі передавання аргументів у шаблони.

Недоліки Express.js:

– складність інтерпретації помилок;
– обмеження в структурованості коду;
– складність в керуванні зворотних викликів [5].

Adonis.js – це веб-фреймворк, заснований на Node.js, що значною мірою спирається на концепції з Laravel. Подібно до Laravel, Adonis.js прагне надати аналогічний підхід до розроблення, включно зі структурою впровадження залежностей (Dependency Injection) і провайдерів (Providers), що забезпечує чистий і впорядкований код. Цей фреймворк покликаний забезпечити подібну структуру впровадження залежностей і провайдерів, красивий код та інтуїтивно зрозумілий дизайн. Його мета – задовільнити розробників веб-додатків, надаючи послідовне та виразне API.

Базуючись на патерні MVC, подібному до Laravel, цей фреймворк фокусується на ключових аспектах розробки масштабованих, стабільних і ефективних веб-додатків. Він охоплює такі аспекти, як:

– зручність для розробника;
– узгоджене API;
– швидкість і продуктивність;
– підтримка функцій розсилки, аутентифікації, перевірки даних, Redis тощо.

Переваги Adonis.js:

– структура папок забезпечує порядок і легкість підтримки коду;
– вбудований валідатор спрощує перевірку даних, що вводяться користувачем;
– ORM Lucid забезпечує чудову підтримку реляційних СКБД;

– Inversion of Control і провайдери служб полегшують управління залежностями;
– вбудовані інструменти забезпечують безпеку від поширених веб-атак;
– простота тестування дає змогу розробникам створювати модульні тести для веб-застосунків.

Недоліки Adonis.js:

– невелике і менш розвинута спільнота і підтримка через новизну платформи, що може ускладнити отримання допомоги в разі проблем;
– незавершеність документації, що обмежує розуміння і використання деяких можливостей;
– відсутність широкого спектра плагінів, зумовлена меншою популярністю та молодістю платформи.
– обмежений досвід фактичного використання через відносну меншу популярність, що знижує впевненість у його продуктивності на великих проектах [7].

Django – це веб-фреймворк, який надається користувачам у безоплатному доступі та з відкритим вихідним кодом, призначений для прискорення процесу розробки веб-застосунків, використовуючи мову програмування Python. Цей фреймворк стає кращим вибором при створенні веб-застосунків, які потребують міжсайтового скриптингу та обробки великої кількості користувачів або складних функцій, як-от підключення API або автентифікація користувачів.

Однією з головних переваг Django є великий набір функціональних можливостей. Понад 10 000 пакетів Django забезпечують практично повне покриття потреб веб-застосунків. Ці пакети включають API, системи управління контентом, аутентифікацію користувачів, перевірку форм і захист CAPTCHA [8].

Переваги Django:

– управління базами даних схожим на Python за допомогою ORM;
– створення динамічних сторінок за допомогою шаблонів мови DTL;
– захист від SQL-ін'єкцій і автоматичний захист від підробки міжсайтових запитів (CSRF) через API-інтерфейси.

Недоліки Django:

– складність в освоєнні;
– може бути надлишковим для невеликих проєктів через велику кількість коду і вимог до обчислювальних потужностей сервера;
– має обмеження свободи у виборі архітектури проекту і володіє власним унікальним підходом до організації файлової структури.

Flask – це веб-фреймворк, заснований на мові програмування Python, призначений для розроблення веб-застосунків із фокусом на простоті та гнучкості. Найчастіше асоційований з веб-розробкою, також його застосовують для створення простих веб-інтерфейсів, API та інструментів візуалізації даних, надаючи вченим і аналітикам можливість представлення даних та аналітичних матеріалів у зручній та інтерактивній формі. Flask спроектований з урахуванням простоти і масштабованості ядра додатка, оперує принципом мінімальності, не включаючи в себе абстрактні рівні для підтримки баз даних. Замість цього, Flask пропонує механізм розширень, які надають можливість інтегрувати необхідні функціональності в додаток [9][10].

Переваги Flask:

– простота розробки;
– повний контроль над розробкою;
– гнучкість і модульність;
– підтримка тестування;
– механізм розширення.

Недоліки Flask:

– почергова обробка запитів;
– обмеженість стандартного функціоналу;
– мала кількість спільнот і документацій;

2.5 Аналіз реляційних СКБД

MySQL – це система керування базами даних SQL, яка являє собою платформу з широким набором функцій і переваг. Вона розроблена і підтримується корпорацією Oracle, забезпечуючи вільний доступ до свого вихідного коду. Відкритий доступ до вихідного коду надає користувачам привілей вивчення та внесення змін до програмного забезпечення відповідно до індивідуальних вимог. Можливість такого вільного доступу, що забезпечується ліцензією GPL, надає можливість використання цієї системи усім зацікавленим особам, водночас пропонуючи альтернативу комерційній ліцензії для використання у сфері комерційних проектів.

Широкий спектр додаткового програмного забезпечення, розроблений у тісній взаємодії з кінцевими користувачами, забезпечує не лише створення та масштабування баз даних, а й забезпечує високу ефективність під час опрацювання запитів і гарантує безпеку даних у мережевих оточеннях [11].

Переваги MySQL:

– відкритий вихідний код;
– підтримка спільноти;
– висока продуктивність;
– масштабованість;
– гнучкість і різноманітність інструментів.

Недоліки MySQL:

– обмежена підтримка деяких функцій щодо інших СКБД;

– зниження продуктивності під час роботи з великими обсягами даних;
– необхідність ліцензії для комерційних проектів.

PostgreSQL – це реляційна система керування базами даних з відкритим вихідним кодом, що забезпечує підтримку як реляційних SQL, так і нереляційних JSON запитів. Його застосовують як основне сховище даних для безлічі додатків, включно з веб, мобільними та аналітичними. PostgreSQL пропонує широкий набір можливостей і гнучкість у роботі з різними типами даних.

Ця СКБД також має підтримку розширених типів даних і оптимізацією продуктивності, що можна порівняти з функціональністю комерційних аналогів, включно з Oracle і SQL Server. PostgreSQL має великий інструментарій, що робить її гарним вибором для широкого спектра додатків і завдань у корпоративному та розробницькому контекстах [12].

Переваги PostgreSQL:
– відкритий вихідний код;
– можливості розширення;
– обробка складних типів даних;
– підтримка JSON.

Недоліки PostgreSQL:
– низька швидкість читання;
– менша підтримка суспільством [13].

У таблиці 2.2 наведено відмінності між MySQL і PostgreSQL [14].

Таблиця 2.2 – відмінності між реляційними СКБД

Реляційні СКБД	MySQL	PostgreSQL
Технології БД	Реляційна система керування	Об’єктно-реляційна система керування
Підтримка функцій	Перегляди, тригери і процедури	матеріалізовані представлення, тригери «INSTEAD OF» і процедури збереження кількома мовами
Типи даних	Числові, символьні, дата та час та просторові	Числові, символьні, дата й час, просторові, геометричні, нумеровані, мережеві адреси, масиви, JSON, XML, hstore і композитні
Сумісність ACID	Тільки з кластерними механізмами зберігання InnoDB і NDB	Завжди сумісний
Індексі	B-tree та R-tree	індекси виразів, часткові індекси і хеш-індекси разом із деревами.
Продуктивність	Високочастотні операції читання	Високочастотні операції запису

2.6 Вимоги до системи

2.6.1 Вимоги до обладнання

2.6.1.1 Вимоги до маршрутизатора

Вимоги до характеристик маршрутизатора включають наявність мінімум двох модульних інтерфейсів NIM і форм-фактора 1 юніт. Необхідний порт управління з пропускною спроможністю 1 гігабіт на секунду (1GE), а також підтримка Power over Ethernet (PoE) до 530 ват. Слід забезпечити наявність 24 комутованих портів Ethernet LAN, здатних використовувати PoE. Вбудовані слоти для інтегрованих сервісних карт і розширених сервісних модулів також є важливою вимогою.

Функціональність пристрою має бути доповнена наявністю вбудованих портів WAN (1 гігабіт Ethernet / SFP, 1 гігабіт Ethernet, 1 SFP) і USB-портів типу A. Безпека і продуктивність мережі вимагають наявності брандмауера з підтримкою Virtual Routing and Forwarding (VRF) і Network Address Translation (NAT), а також механізмів запобігання вторгнень. Важливим аспектом також є можливість розширення оперативної пам'яті та пам'яті зберігання до 16 гігабайт.

Нарешті, маршрутизатор повинен забезпечувати пропускну здатність даних до 300 мегабіт на секунду для забезпечення необхідної продуктивності мережевих операцій.

2.6.1.2 Вимоги до комутатора

Вимоги до комутатора включають форм-фактор 1 юніт і наявність механізмів запобігання вторгнень. Важливо мати два інтерфейси вихідної ланки (SFP або 1000BASE-T) і доступну потужність Power over Ethernet (PoE) до 123 ват. Кількість портів Ethernet має становити 24 порти Ethernet 10/100.

Пропускна здатність комутатора має бути не менше 16 гігабіт на секунду, оперативна пам'ять - 128 мегабайт, а флеш-пам'ять - 64 мегабайти. Ці параметри істотні для забезпечення ефективної роботи мережі, забезпечуючи необхідну продуктивність і функціональність комутаційного обладнання.

2.6.1.3 Вимоги до сервера

Необхідний сервер, здатний вміститися у форм-фактор 2U, з певними характеристиками. Необхідна наявність одного слота для центрального процесора на материнській платі, а також чотирьох слотів під модулі оперативної пам'яті DDR4 UDIMM, або DDR5-5200. Потрібна наявність не менше чотирьох слотів SATA з підтримкою контролера SAS/SATA і можливістю конфігурації RAID 0/1/5/10. Процесор повинен мати не менше ніж шість ядер і тактову частоту від 2.0 ГГц. Охолодження процесора має здійснюватися кулером із потужністю не більше 120 Вт, призначеним для корпусу 2U. Загальний об'єм оперативної пам'яті має становити не менше 64 ГБ, що відповідає стандарту DDR4 з частотою 3200 МГц, або вищого стандарту. Також необхідна наявність SSD для операційної системи та програмного забезпечення сервера, а також двох жорстких дисків ємністю 2 ТБ кожен. Мережева карта повинна забезпечувати один LAN-роз'єм і підтримувати швидкість передавання даних не менше 1 Гбіт/с. Очікується наявність блока живлення потужністю від 400 Вт.

2.6.1.4 Вимоги до персонального комп'ютера

Для забезпечення робочого оточення персоналу потрібен моноблок з екраном, що підтримує роздільну здатність 1920 на 1080 пікселів і використовує технологію IPS. Процесор повинен належати сімейству Intel Core i5 з шістьма ядрами, дванадцятьма потоками і базовою тактовою частотою від 2.0 ГГц. Для візуальних потреб пристрою слід використовувати інтегровану відеокарту. Оперативна пам'ять, що відповідає стандарту DDR4 з тактовою частотою 2666 МГц, повинна мати об'єм не менше 8 ГБ. Для швидкого завантаження операційної системи та програмного забезпечення рекомендується використання SSD з мінімальною ємністю 250 ГБ.

Мережевий інтерфейс, представлений вбудованим RJ-45, повинен підтримувати швидкість передачі даних не менше 100 Мбіт/с, забезпечуючи стабільне підключення до мережі.

2.6.1.5 Вимоги до переферійних пристроїв

Периферійний пристрій, призначений для аналізу крові, має володіти певними характеристиками, щоб забезпечити точні та ефективні гематологічні дослідження.

Необхідна висока продуктивність цього пристрою для проведення від 10 аналізів на годину, що охоплюють 20 різних гематологічних параметрів. Цей пристрій має бути гнучким у роботі зі зразками різного об'єму – від 50 мікролітрів цільної крові до 20 мікролітрів у режимі попереднього розведення.

Важливо, щоб цей пристрій використовував різноманітні методи аналізу, як-от DC метод для WBC, RBC/PLT і безціанідний метод для визначення гемоглобіну (HGB). Керування пристроєм має бути зручним завдяки наявності сенсорного екрана, а також можливості підключення зовнішнього принтера для виведення результатів досліджень.

Для забезпечення ефективного обміну даними, цей пристрій повинен мати два мережевих порти, що дають змогу взаємодіяти з іншими пристроями та системами. Крім того, важливо забезпечити підтримку спеціалізованих реагентів і контрольних матеріалів, а також мати системи внутрішнього і зовнішнього контролю якості для гарантованої точності одержуваних результатів. Такі характеристики та функціональні можливості периферійного пристрою дадуть змогу забезпечити високу точність і достовірність результатів під час проведення гематологічних досліджень.

2.6.2 Вимоги до ПЗ пристроїв

2.6.2.1 Вимоги до ПЗ комп'ютера для персоналу

Для забезпечення оптимальної працездатності рекомендується встановити останні версії програмного забезпечення на моноблок. Операційна система Windows 10, антивірус Eset Node 32, вбудований веб-браузер і Google Chrome, архіватор 7Zip мають бути встановлені в їхніх останніх доступних версіях. Винятком є Microsoft Office 2016, оскільки бажано встановити дану версію офісного пакета.

2.6.3 Вимоги до БД

База даних для донорського центру має відповідати принципам реляційної структури та ефективної організації даних. Її основна функція полягає в систематизації і зберіганні інформації про користувачів, донорів, записи на прийом, співробітників, процедури донації, та результати аналізів крові.

Однією з ключових характеристик цієї бази даних є необхідність забезпечити реагування на запити на запис, зміну та зчитування інформації протягом не більше ніж 7 секунд. Крім того, система має автоматично створювати резервні копії бази даних кожні 7 днів для забезпечення безпеки і можливості відновлення даних у разі потреби.

Незважаючи на обробку даних, безпека та довгострокове зберігання інформації є критичними аспектами цієї бази даних. Вона повинна забезпечувати збереження інформації не менш як протягом 5 років і бути готовою до віддзеркалення даних для забезпечення стійкості та надійності зберігання.

2.6.4 Вимоги до ПЗ, яке розроблюється

2.6.4.1 Вимоги до функціональних характеристик

ПЗ має забезпечувати можливість реєстрації нових користувачів та аутентифікації вже зареєстрованих. Крім того, необхідна реалізація інструментів управління записами про донорство, що включають в себе функції створення, зміни, перегляду та видалення записів як з боку користувачів, які внесли дані, так і з боку персоналу донорського центру. ПЗ має також надавати можливість створення, зміни та перегляду інформації, що міститься в картці донора для персоналу.

Додатково, ПЗ повинно мати можливість автоматизації процесу отримання даних з аналізатора крові та їхнього подальшого запису в базу даних у разі успішного завершення аналізу. Також необхідно передбачити створення і перегляд записів про процедури донорства для персоналу, а також

забезпечення доступу до остаточних результатів аналізів крові як для донорів, так і для персоналу донорського центру.

Крім того, ПЗ має надавати можливість проведення пошуку за різними списками записів, що містять інформацію про донації, донорів, результати аналізів крові та інші пов'язані дані, які доступні персоналу. Прикінцевими функціями мають бути можливість перегляду профілю користувача та безпечний вихід із застосунку, що забезпечує конфіденційність персональної інформації та безпеку даних.

2.6.4.2 Вимоги до інформаційної та програмної сумісності

ПЗ має бути сумісним із системними вимогами персональних комп'ютерів, що використовуються персоналом. Також ПЗ має забезпечувати сумісність із сучасними операційними системами сімейства Windows і популярними дистрибутивами операційних систем Linux. Воно повинно коректно функціонувати і бути сумісним з провідними веб-браузерами, такими як Opera, Firefox, Chrome, Microsoft Edge, в їхніх останніх версіях.

Також ПЗ не повинно викликати конфліктів з іншими програмами, забезпечуючи плавну роботу без втручання у функціональність інших додатків.

2.6.4.3 Вимоги до надійності

ПЗ, розроблене для управління донорським центром, зобов'язане забезпечувати безперервну та безперебійну роботу системи протягом 24 годин на добу, 7 днів на тиждень.

ПЗ також повинне мати механізми, що дозволяють оперативно ідентифікувати та усувати несправності, а також відновлювати працездатність програми після можливої відмови протягом однієї години. Цей процес включає в себе виявлення проблеми, проведення необхідних дій для її усунення та швидке відновлення функціональності ПЗ з метою мінімізації часу простою та забезпечення безперервної роботи донорського центру.

Також ПЗ має бути спроектовано для запуску тільки на офіційних інтерпретаторах.

2.7 Висновки

У результаті проведених аналізів, було обрано оптимальні інструменти та технології для розробки програмного забезпечення. Використання мови програмування Python зумовлене її зручністю, багатим функціоналом, безліччю бібліотек і кросплатформеністю, що робить її чудовим вибором для розроблення різних застосунків.

Вибір фреймворка Django обґрунтовано його зручністю в управлінні базами даних, потужним ORM, можливістю створення динамічних сторінок із використанням шаблонів DTL, а також вбудованими механізмами захисту від потенційних атак, таких як SQL-ін'єкції та CSRF.

Віддача переваги СКБД PostgreSQL пояснюється його відкритим вихідним кодом, гнучкістю розширення функціонала, можливістю опрацювання складних даних і вбудованою підтримкою JSON.

Сукупність обраних інструментів – Python, Django та PostgreSQL – забезпечує надійне, гнучке та безпечне для розробників програмне забезпечення, з високим ступенем зручності роботи з базами даних, оброблення різноманітних даних та захисту від потенційних вразливостей, що є ключовим для успішного розроблення сучасних застосунків.

3 СИНТЕЗ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Розробка схеми структури

На минулих етапах функціонування донорського центру зберігання інформації про пацієнтів і результати аналізів підтримувалися у формі документів на папері та в локальній файловій системі. Звіти про результати аналізів, що видавалися аналізатором, були представлені в друкованій формі, які надалі вручну дублювалися в електронному вигляді. Процес запису на донацію здійснювався за допомогою телефонного дзвінка або звернення особисто до персоналу.

З урахуванням модернізації в технологічному та програмному аспекті функціонування донорського центру, було розроблено нову схему структури, що значно оптимізує процеси збору, зберігання та обробки інформації.

Дані, отримані від аналізатора крові, передаються через інтерфейс Ethernet на сервер. Ці дані миттєво і записуються в БД, що виключає ризик помилок, пов'язаних з ручним введенням інформації.

Крім того, для зручності донорів було впроваджено систему онлайн-запису, яка дає змогу потенційним донорам реєструватися через інтернет. Цей процес здійснюється шляхом доступу до спеціальної веб-платформи, де вони можуть заповнити свої дані та обрати зручний для них час і дату для донації. Отримані дані з веб-платформи негайно інтегруються в БД, забезпечуючи актуальність інформації про записаних донорів і спрощуючи процес управління розкладом прийому.

Розроблена структурна схема системи донорського центру зображена на рисунку 3.1.

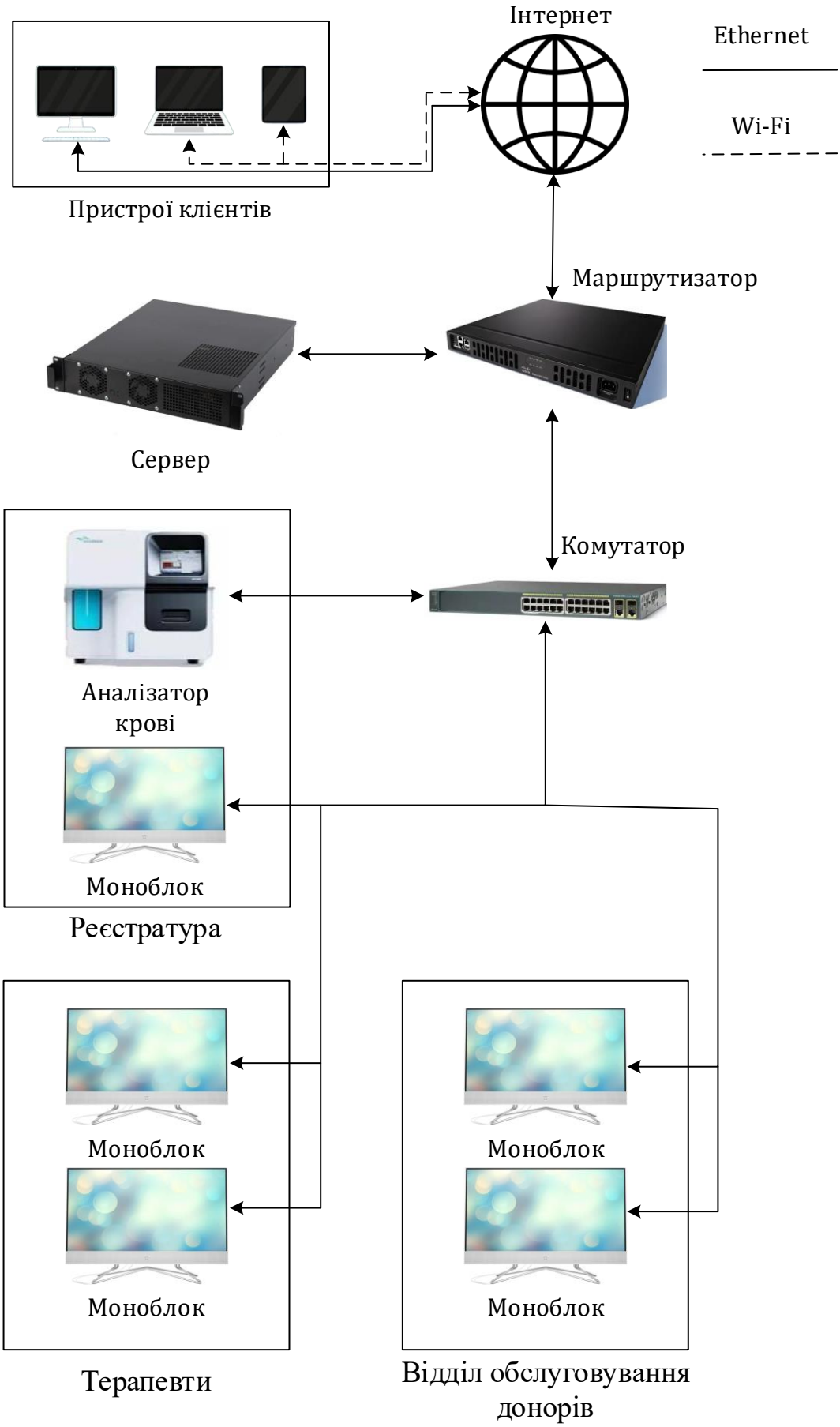


Рисунок 3.1 – Розроблена структурна схема донорського центру

3.2 Розробка функціональної схеми

Для забезпечення ефективного функціонування донорського центру було розроблено функціональну схему, яка описує основні етапи та взаємодію між учасниками процесу.

Донор має можливість отримати доступ до реєстрації, авторизації та запису на донацію, заповнивши відповідну форму на веб-сайті. Він також може запросити результати аналізу своєї крові після процедури донації через веб-сайт.

У працівника є можливість отримати інформацію о записах на донацію, картах донорів, загальних результатів крові донорів, повних результатів крові донорів та донації.

Працівник здійснює збір зразків крові для подальшого аналізу та проведення процедури донорства плазми. Після цього він відправляє інформацію про донацію або запит на аналіз на сервер для запису в базу даних. У додаток до цього, співробітник має можливість створити карту донора, вносити зміни до даних донора, видаляти записи на донації.

Аналізатор проводить аналіз зразків крові або плазми та створює файл із результатами аналізу, наприклад, у форматі JSON. Потім відправляє ці результати на сервер через HTTPS-протокол для збереження в базі даних.

Сервер приймає і зберігає інформацію про реєстрацію донорів, запити на аналіз, результати аналізів та інформацію про донацію крові або плазми за безпечним HTTPS-протоколом. Отримані дані зберігаються в базі даних для подальшого доступу та управління. Крім того, сервер надає доступ донорам до результатів їхніх аналізів через веб-сайт за запитом також за допомогою HTTPS-протоколу.

Розроблена функціональна схема зображена на рисунку 3.2.

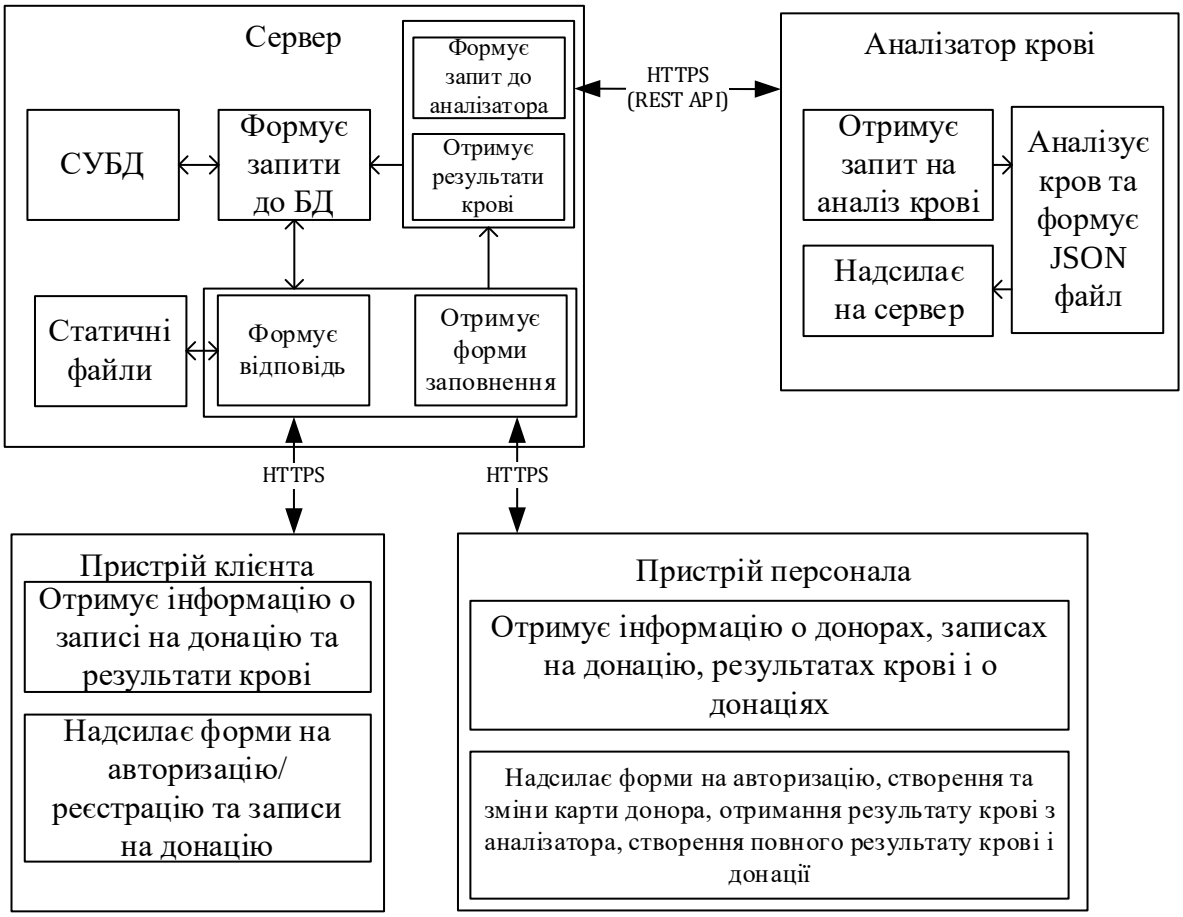


Рисунок 3.2 – Розроблена функціональна схема донорського центру

3.3 Вибір елементної бази комп'ютерної системи донорського центру

Донорський центр обладнаний аналізатором крові Sysmex XP-300, який надавав результати дослідження в друкованому форматі. Однак, крім цієї функції, прилад також обладнаний інтерфейсом Ethernet, призначеним для передачі даних в електронному форматі. Технічні характеристики периферійного пристрою відображено в таблиці 3.1. [17]

Таблиця 3.1 –характеристики аналізатора крові Sysmex XP-300

Параметри	Значення
Технології	WBC, RBC/PLT: DC метод (кондуктометрія); HGB: безціанідний метод визначення гемоглобіну
Продуктивність	до 60 зразків на 1 годину

Продовження таблиці 3.1

Вимірювані параметри	WBC, RBC, HGB, HCT, MCV, MCH, MCHC, PLT, LYM% (W-SCR), MXD% (W-MCR), NEUT% (W-LCR), LYM# (W-SCC), MXD# (W-MCC), NEUT# (W-LCC), RDW-SD, RDW-CV, PDW, MPV, P-LCR, PCT
Інтерфейси	LCD; Сенсорний дисплей; термопринтер; сканер штрих-кодів; два мережевих LAN порти.
Живлення	120/230/240 В змінного струму ±10%; 50/60 Гц

Встановлені маршрутизатори, комутатори та моноблоки, що використовуються працівниками, відповідають мінімальним вимогам, встановленим для їхньої функціональності та ефективної роботи. Тому, з точки зору оптимального використання наявних ресурсів, доцільним видається збереження поточної конфігурації цих пристроїв.

Таблиця 3.2 – характеристики маршрутизатора Cisco Router ISR 4331/K9

Параметри	Значення
Кількість модулів мережевого інтерфейсу	2
Максимальна кількість комутованих портів Ethernet LAN	24
Тип сервісного модуля EtherSwitch	1 одинарний
Підтримка PoE (потужність) без підсилення PoE	250 Вт

Продовження таблиці 3.2

Запобігання вторгненню	так
Форм-фактор	1 юніт
Тип живлення	AC
Порт керування	1 GE
ОЗУ	4 ГБ
Пам’ять	4 ГБ
Вбудовані порти	1 GE / SFP, 1 GE, 1 SFP
Продуктивність	100 Мбіт/с з можливістю підвищення до 300 Мбіт/с

Таблиця 3.3 – характеристики комутатора Cisco WS-C2960-24LC-L

Параметри	Значення
Форм-фактор	1 юніт
Запобігання вторгненню	Так
Інтерфейси вихідного зв’язку	2 (SFP або 1000BASE-T)
Доступна потужність PoE	123 Вт
Порти Ethernet	24 порти Ethernet 10/100
Пропускна здатність пересилання	16 Гбіт/с
Оперативна пам'ять	128 Мб
Флеш-пам'ять	64 МБ

Таблиця 3.4 – характеристики моноблоку HP All-in-One 24-df0057

Параметри	Значення
Дисплей	1920 x 1080
Дисплей, тип матриці	IPS
Процесор, модель	Intel Core i5-10400T
Процесор, тактова частота – turbo, ГГц	2,0 – 3,6

Продовження таблиці 3.4

Процесор, к-сть ядер/потоків	6 ядер/12 потоків
Відеокарта, тип	Інтегрована
Відеокарта	Intel UHD Graphics 630
Відеокарта, об'єм пам'яті	Виділено з ОП
Оперативна пам'ять	DDR4 - 2666 МГц
Оперативна пам'ять, об'єм	8 ГБ
Оперативна пам'ять, к-ть слотів	2
Вбудований накопичувач (об'єм), ГБ	256 NVMe SSD
Бездротові інтерфейси, Wi-Fi	802.11 a/b/g/n/ac
Мережеве підключення, LAN RJ-45, шт	1
Мережеве підключення LAN RJ-45, Мбіт/с	10/100/1000 Ethernet
Порт USB 2.0 Type-A/ Type-B/ mini USB/ micro USB	2 x USB 2.0 Type-A
Порт USB 3.0/3.1/3.2 Gen 1 Type-A/Type-B/micro USB	2 x USB 3.0/3.1 Gen 1 Type-A

Для забезпечення ефективного зберігання інформації та забезпечення її безпеки в робочому середовищі, часто потрібне використання виділеного сервера. Раніше згадувалося, що інформація зберігалася на пристроях персоналу у вигляді файлової системи. Тому для структурованого зберігання інформації, слід використовувати сервер, який відповідає вимогам до сервера.

В таблицях 3.5 та 3.6 наведені сервери, що відповідають поставленим вимогам [18][19].

Таблиця 3.5 – характеристики серверу ARTLINE Business R19v26

Параметри	Значення
Форм-фактор	2 юніта
Процесор	Intel Core i7-12700
Процесор, тактова частота – turbo, ГГц	2.1 – 4.9
Процесор, к-сть ядер/потоків	12/20
Відеокарта	Інтегрована
Відеокарта, інтегрована	Intel UHD Graphics 770
Відеокарта, об'єм пам'яті	Виділено з ОП
Оперативна пам'ять	DDR4 – 3200МГц
Оперативна пам'ять, об'єм	64 ГБ
Оперативна пам'ять, к-ть слотів	4
Вбудований накопичувач, об'єм, ГБ	256 (NVMe SSD), 2000 (HDD), 2000 (HDD)
Слоти SATA, к-ть слотів	6
Контролери SAS/SATA	Вбудований у чіпсет
RAID	0/1/5/10
Мережеве підключення LAN RJ-45, шт	1
Мережеве підключення, LAN RJ-45, Мбіт/с	10/100/1000/2500 Ethernet
Блок живлення	450 Вт
Ціна	44464

Таблиця 3.6 – характеристики серверу ARTLINE Business R36v22

Параметри	Значення
Форм-фактор	2 юніта
Процесор	AMD Ryzen 9 7900
Процесор, тактова частота – turbo, ГГц	3.7 – 5.4
Процесор, к-сть ядер/потоків	12/24
Відеокарта	Інтегрована
Відеокарта, інтегрована	AMD Radeon Graphics
Відеокарта, об'єм пам'яті	Виділено з ОП
Оперативна пам'ять	DDR5-5200МГц
Оперативна пам'ять, об'єм	64
Оперативна пам'ять, к-ть слотів	4
Вбудований накопичувач, об'єм, ГБ	2 x 500 ГБ SSD 2 x 2 ТБ HDD
Слоти SATA, к-ть слотів	8
Гаряча заміна накопичувачів	Так
Контролери SAS/SATA	Вбудований у чіпсет
RAID	0/1/10
Мережеве підключення, LAN RJ-45, шт	2
Мережеве підключення, LAN RJ-45, Мбіт/с	10/100/1000/10000
Кількість блоків живлення	2
Блок живлення	550 Вт
Ціна	89655

Сервер ARTLINE Business R19v26 з процесором Intel Core i7-12700, 64 ГБ оперативної пам'яті, NVMe SSD і HDD забезпечує високу продуктивність, широке сховище і швидке передавання даних. Його більш доступна ціна в

44464 грн. робить його кращим вибором для ефективного і безпечного зберігання інформації в робочому оточенні.

3.4 Висновки

У розділі синтезу системи для донорського центру було розроблено структурну схему, яка мала на меті відображення складових та взаємозв'язків всіх елементів системи. Також була створена функціональна схема, що визначала функції та способи взаємодії окремих компонентів центру. Під час аналізу донорського центру було приділено особливу увагу відповідності технічних засобів вимогам, враховуючи їхні функціональні можливості. Це дало можливість створити сприятливе середовище для проведення процедур збору та наступної обробки результатів аналізів крові, сприяючи ефективності та точності центру у виконанні його функцій.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ДОНОРСЬКОГО ЦЕНТРА

4.1 Призначення й сфера застосування програми

Програма призначена для оптимізації управління донорським центром шляхом забезпечення зручного доступу та обробки інформації про донорів, результатів аналізів крові та донацій плазми. Основною метою є полегшення роботи персоналу, спрощення ведення бази даних та забезпечення швидкого та зручного доступу до важливої інформації для ефективного управління та моніторингу донорської діяльності центру. Крім цього, програма надає можливість донорам отримувати обмежений функціонал для перегляду власної інформації та результатів своїх донацій крові, а також здійснювати запис на наступні донації, сприяючи зручності та ефективності участі у процесі донорства.

4.2 Організація та функції програми

4.2.1 Опис функціонування програми

Проект базується на фреймворку Django, що забезпечує зручну та ефективну розробку веб-застосунків. Використання Django дозволяє швидко розробляти веб-сайт для оптимізації роботи донорського центру. Програма взаємодіє з БД, забезпечуючи можливість створення облікових записів, карток донорів, реєстрацію на донацію, а також управління результатами аналізів крові та іншою інформацією.

Для забезпечення взаємодії з базою даних у проекті використовується ORM. Це дає змогу використовувати високорівневі абстракції бази даних та спрощує роботу з ними. СКБД PostgreSQL обрана для забезпечення надійності, ефективності та масштабованості у зберіганні даних для даного проекту.

4.2.2 Опис організації вхідних і вихідних даних

СКБД PostgreSQL використовується для зберігання вхідних даних, які отримуються та обробляються через фреймворк Django. Ці вхідні дані можуть включати інформацію, яка подається або вводиться через веб-сайт.

За допомогою Django, дані, збережені в PostgreSQL, використовуються для створення вихідних даних, які подаються користувачам веб-сайту або програми. Ці вихідні дані можуть включати інформацію, яка відображається на веб-сторінках.

4.3 Опис розробленої програми

4.3.1 Загальні відомості

4.3.1.1 Найменування програми

DonorPlus – це веб-застосунок, розроблений для оптимізації роботи донорського центру. Ця платформа призначена для внутрішнього використання працівниками центру, а також для клієнтів.

4.3.1.2 Функціональне призначення

Функціональне призначення веб-застосунку призначене для донорського центру Biopharma з метою управління інформацією про донорів. Основна мета полягає в забезпеченні ефективного управління даними про донації крові та плазми, а також забезпеченні можливості перегляду результатів аналізів крові.

Цей веб-застосунок надає можливість здійснювати широкий спектр функцій, зокрема:

а) реєстрація та авторизація:

- дозволяє користувачам зареєструватися та увійти до системи для доступу до функціоналу веб-застосунку.

б) запис інформації:

- надає можливість внесення нових даних про донорів, у тому числі особистої інформації та даних про донації.

в) збереження даних:

– забезпечує збереження інформації про донорів та їх донації крові та плазми в безпечному і доступному форматі.

г) оновлення даних:

– дозволяє оновлювати існуючі дані про донорів та записи на донації.

ґ) контроль доступу до даних:

– забезпечує контроль за рівнем доступу до різних типів інформації, гарантуючи конфіденційність та безпеку особистих даних донорів.

4.3.1.3 Логічна структура

Логічна структура починається з БД, яка визначає основні сутності та зв'язки між ними, створюючи фундамент для зберігання та організації інформації. Ця структура спрямована на систематизацію даних, управління зв'язками та ефективний доступ до них, що визначає функціональність будь-якої інформаційної системи. У даному випадку, розроблено веб-застосунок з використанням СКБД PostgreSQL версії 16.1, фреймворку Django версії 4.2.7 та мови програмування Python версії 3.10.

База даних складається з таких таблиць:

– користувачі (auth_user) – зберігає дані про користувачів системи;
– групи користувачів (auth_group) – містить інформацію про групи користувачів системи;
– донори (queue_app_donor) – містить інформацію про донорів плазми;
– працівники (queue_app_worker) – зберігає дані про працівників;
– донації (queue_app_donation) – містить дані про процедури забору плазми від донорів;
– результати повних аналізів крові (queue_app_bloodresult) – зберігає результати аналізів крові, проведених у донорів терапевтом;
– загальні результати аналізів крові (queue_app_bloodcommonresult) - містить загальні результати аналізів крові;

– записи на прийом (queue_app_appointment) – зберігає дані про записи користувачів на прийом.

Опис таблиць БД зображено в таблицях 4.1 – 4.8.

Таблиця 4.1 – таблиця користувачі

Назва поля	Призначення поля	Тип	Довжина	Ключ
id	Унікальний ідентифікатор користувача	integer	–	PK
password	Пароль користувача	character	–	–
last_login	Дата останнього входу користувача	timestamp	–	–
is_superuser	Чи є користувач суперкористувачем	boolean	–	–
username	Ім'я користувача	character	–	–
first_name	Ім'я користувача	character	–	–
last_name	Прізвище користувача	character	–	–
email	Адреса електронної пошти користувача	character	–	–
is_staff	Чи є користувач співробітником	boolean	–	–
is_active	Чи є користувач активним	boolean	–	–
date_joined	Дата реєстрації користувача	timestamp	–	–
group_id	Ідентифікатор групи, до якої належить користувач	integer	–	FK

Таблиця 4.2 – таблиця групи користувачів

Назва поля	Призначення поля	Тип	Довжина	Ключ
id	Унікальний ідентифікатор групи	integer	–	PK
name	Назва групи	character	–	–

Таблиця 4.3 – таблиця донори

Назва поля	Призначення поля	Тип	Довжина	Ключ
id	Унікальний ідентифікатор донора	bigint	–	PK
birthday_date	Дата народження донора	date	–	–
gender	Стать донора (за паспортом)	boolean	–	–
weight	Вага донора	integer	–	–
height	Зріст донора	integer	–	–
phone_number	Номер телефону донора	character	–	–
passport_series	Серія паспорта донора	character	–	–
passport_number	Номер паспорта донора	character	–	–
taxpayer_id	Ідентифікаційний код платника податків донора	character	–	–
blood_group	Група крові донора	integer	–	–
rhesus_factor	Резус–фактор донора	boolean	–	–
address	Адреса донора	character	–	–
user_id	Ідентифікатор користувача, до якого належить донор	integer	–	FK

Таблиця 4.4 – таблиця працівники

Назва поля	Призначення поля	Тип	Довжина	Ключ
id	Унікальний ідентифікатор працівника	bigint	–	PK
first_name	Ім'я працівника	character	–	–
last_name	Прізвище працівника	character	–	–
patronymic	По батькові працівника	character	–	–
birthday_date	Дата народження працівника	date	–	–

Продовження таблиці 4.4

taxpayer_id	Ідентифікаційний номер платника податків працівника	character	–	–
passport_series	Серія паспорта працівника	character	–	–
passport_number	Номер паспорта працівника	character	–	–

Таблиця 4.5 – таблиця донації

Назва поля	Призначення поля	Тип	Довжина	Ключ
id	Унікальний ідентифікатор донації	bigint	–	PK
device_number	Номер пристрою	integer	–	–
donation_date	Дата донації	date	–	–
procedure_time	Тривалість процедури	integer	–	–
successful_procedure	Чи була процедура успішною	boolean	–	–
comment	Коментар до донації	character	255	–
donor_id	Ідентифікатор донора, який здійснив донацію	bigint	–	FK
worker_id	Ідентифікатор співробітника, який провів донацію	bigint	–	FK

Таблиця 4.6 – таблиця повні результати аналізів крові

Назва поля	Призначення поля	Тип	Довжина	Ключ
id	Ідентифікатор запису	bigint	20	PK
syphilis	Наявність сифіліс	boolean	–	–
hiv_aids	Наявність ВІЛ/СНІДу	boolean	–	–
hepatitis_b	Наявність гепатиту B	boolean	–	–
hepatitis_c	Наявність гепатиту C	boolean	–	–

Продовження таблиці 4.6

comment	Коментар	character	255	–
blood_common_result_id	Ідентифікатор загального результату крові	bigint	–	FK
donation_id	Ідентифікатор донації	bigint	–	FK
donor_id	Ідентифікатор донора	bigint	–	FK

Таблиця 4.7 – таблиця загальні результати аналізів крові

Назва поля	Призначення поля	Тип	Довжина	Ключ
id	Ідентифікатор запису	bigint	–	PK
hgb	Гемоглобін	double	–	–
rbc	Еритроцити	double	–	–
wbc	Лейкоцити	double	–	–
plt	Тромбоцити	double	–	–
hct	Гематокрит	double	–	–
mpv	Середній об'єм тромбоцитів	double	–	–
mcv	Середній об'єм еритроцитів	double	–	–
mch	Середня маса еритроцитів	double	–	–
mchc	Середня концентрація гемоглобіну в еритроцитах	double	–	–
lym_percent	Відсоток лімфоцитів	double	–	–
mon_percent	Відсоток моноцитів	double	–	–
neut_percent	Відсоток нейтрофілів	double	–	–
lym_number	Кількість лімфоцитів	double	–	–
mon_number	Кількість моноцитів	double	–	–
neut_number	Кількість нейтрофілів	double	–	–
rdw_sd	Стандартне відхилення ширини розподілу еритроцитів	double	–	–
rdw_cv	Коефіцієнт варіації ширини розподілу еритроцитів	double	–	–

Продовження таблиці 4.7

rdw	Ширина розподілу еритроцитів	double	–	–
p_lcr	Процент лейкоцитів, що не мають сегментів	double	–	–
pct	Відсоток тромбоцитів, що мають великі ядра	double	–	–
donor_id	Ідентифікатор донора	bigint	–	FK
id	Ідентифікатор запису	bigint	–	PK

Таблиця 4.8 – таблиця записи на прийом

Назва поля	Призначення поля	Тип	Довжина	Ключ
id	Ідентифікатор запису	bigint	–	PK
day	День	date	–	–
time	Час	character	–	–
time_ordered	Дата та час створення запису	timestamp	–	–
user_id	Ідентифікатор користувача	integer	–	FK

На рисунку 4.1 зображено структуру БД зі зв’язками.

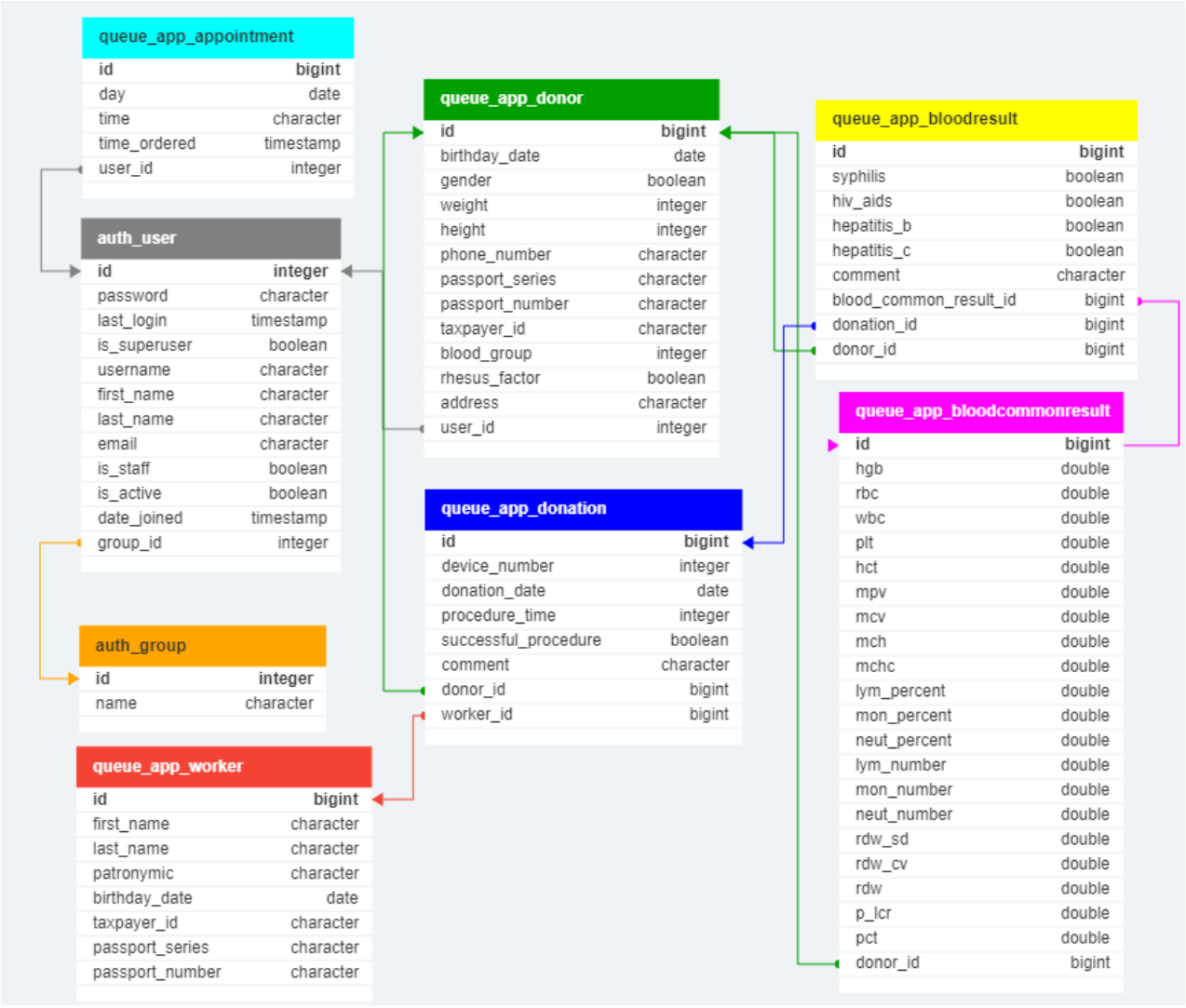


Рисунок 4.1 – структурна схема БД

Структура веб-сайту була розроблена з урахуванням потреб 4 основних категорій користувачів:

– клієнти, які виступають у ролі донорів;

– працівники, включаючи персонал реєстратури, терапевтів та відділу прийому донорів.

Для функціонування та надання працівникам інструменту фільтрації інформації на сторінках сайту, було задіяно мову програмування JavaScript.

На рисунках 4.2 – 4.3 зображено структуру файлів сайту.

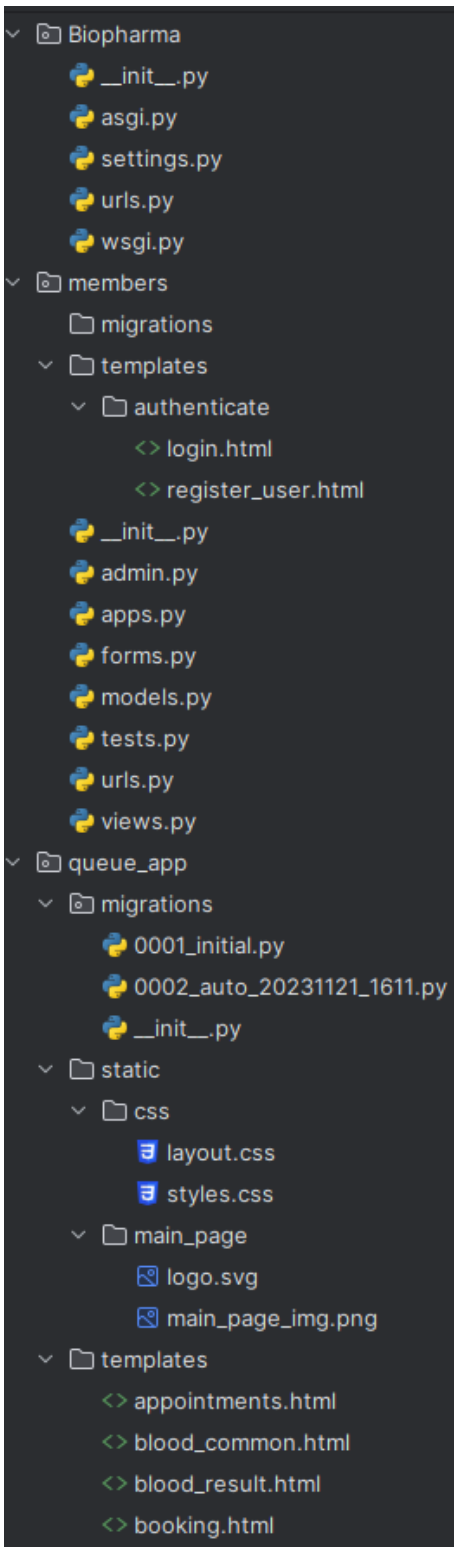


Рисунок 4.2 – Структура файлів сайту

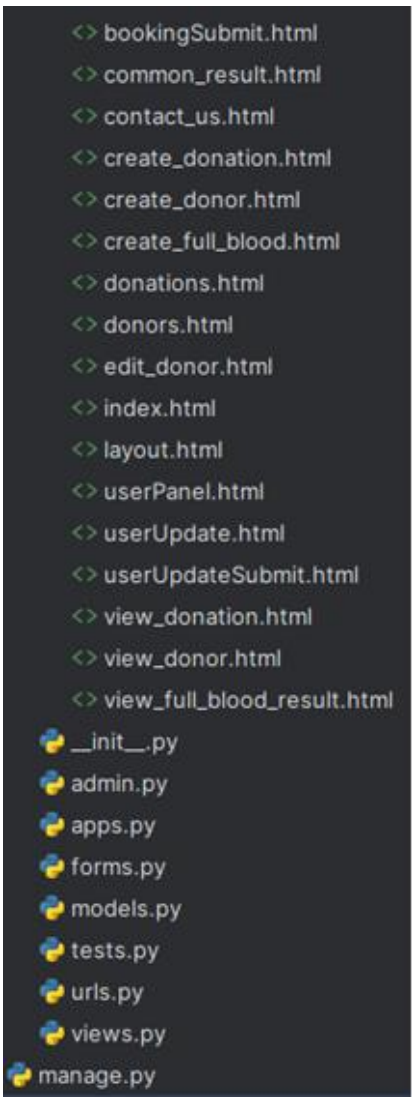


Рисунок 4.3 – Продовження структури файлів сайту

На рисунках 4.4 – 4.7 структура сайту для різних категорій користувачів.

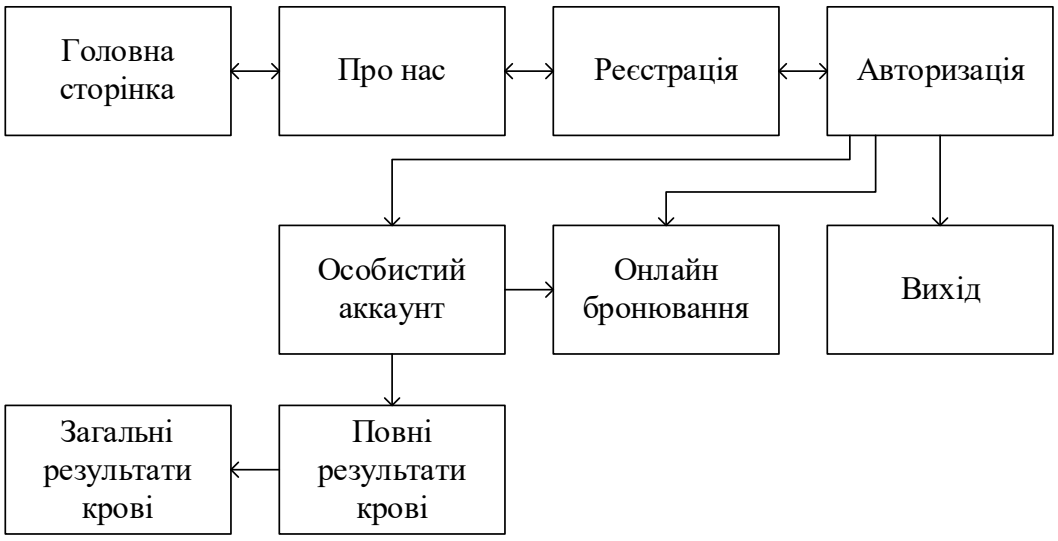


Рисунок 4.4 – Структура сайту для клієнтів

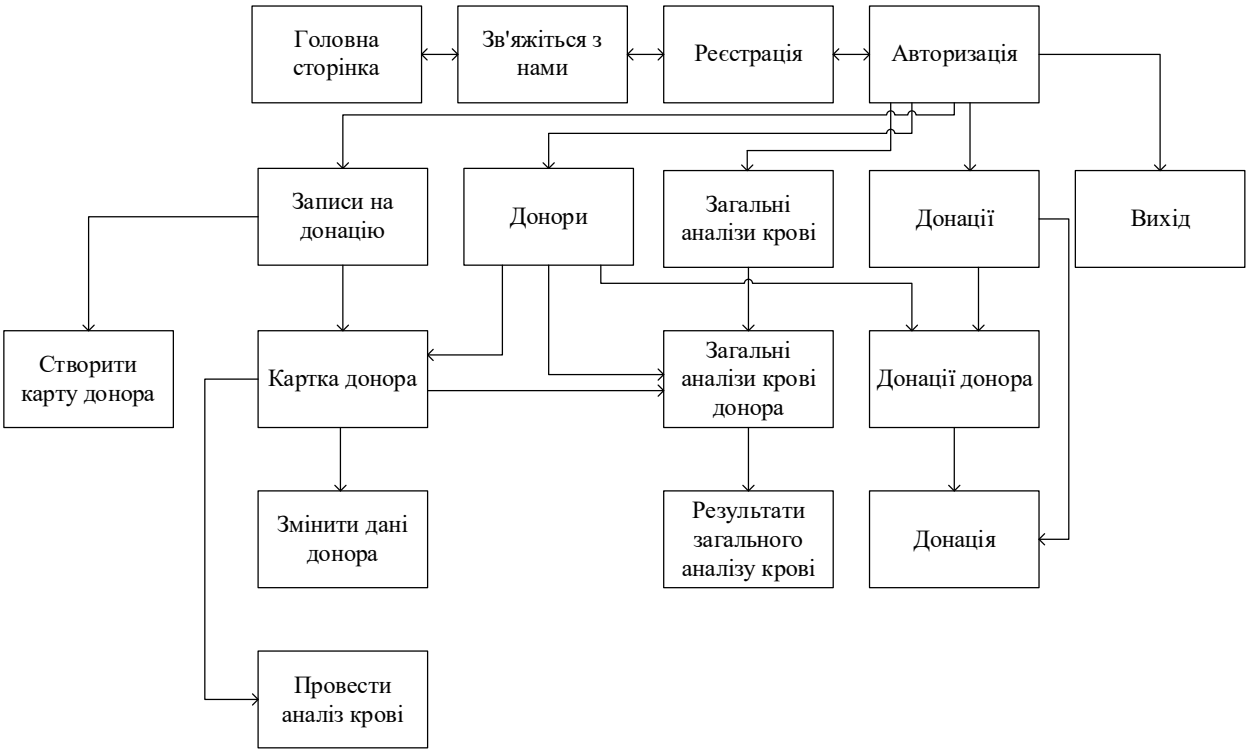


Рисунок 4.5 – Структура сайту для Реєстратури

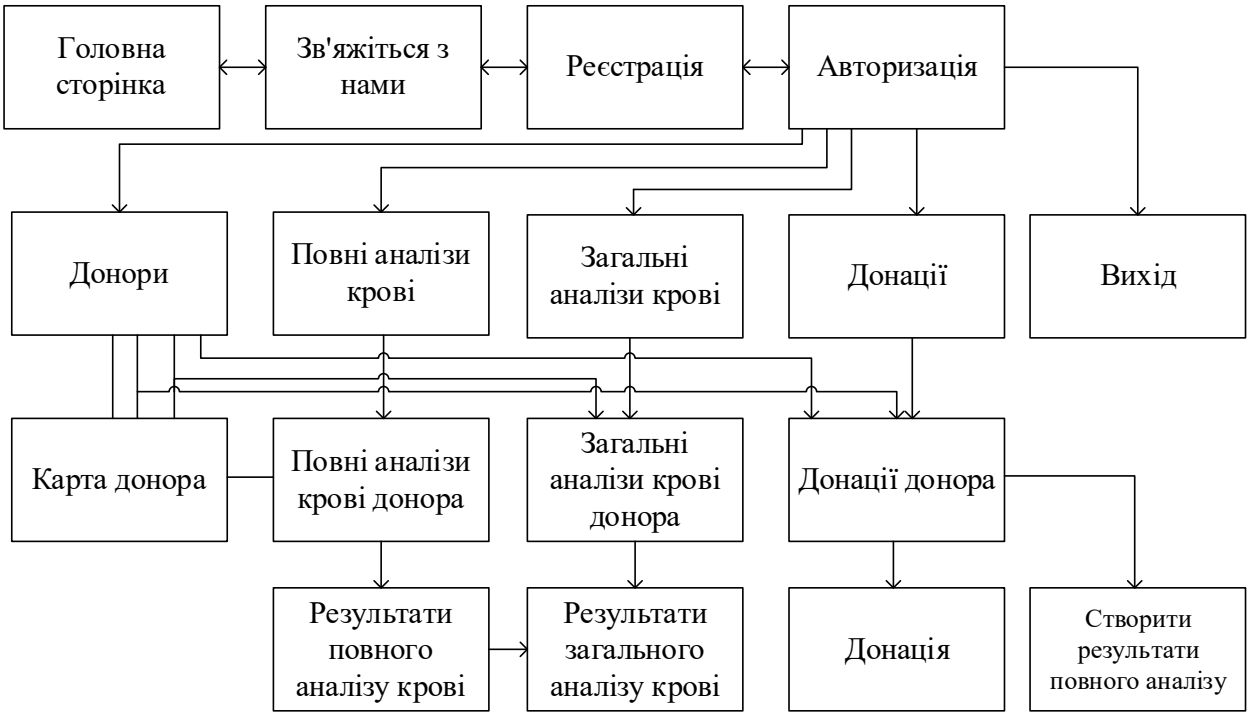


Рисунок.4.6 – Структура сайту для Терапевтів

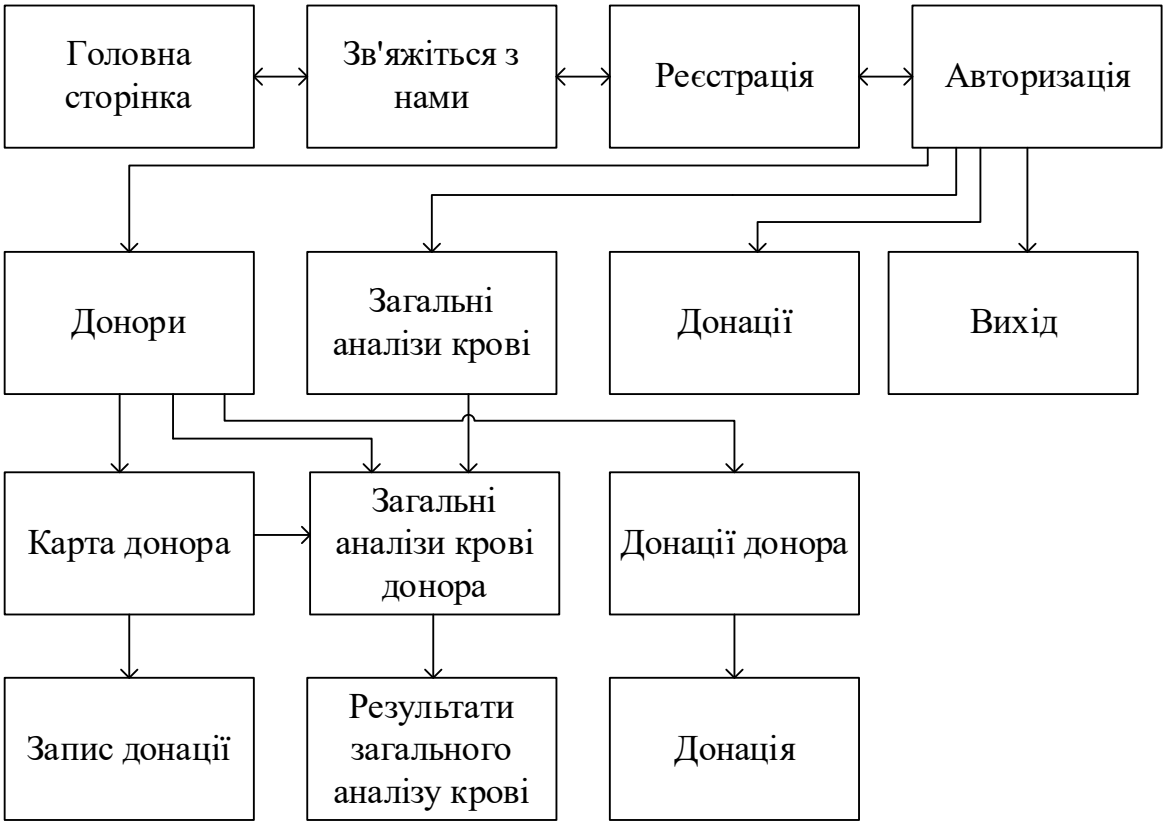


Рисунок 4.7 – Структура сайту для Відділу прийому донорів

4.3.2 Виклик та завантаження

Процес виклику відбувається шляхом формування запиту до сервера за конкретною URL адресою або завдяки локальному хосту. Запит, що походить від браузера, ініціює цю взаємодію із сервером, спрямовуючи запит на необхідні ресурси чи дані. Натомість, завантаження є наступним етапом, де відбувається процес отримання відповіді від сервера до браузера. Під час завантаження браузер отримує та обробляє цю відповідь, яка містить необхідні дані чи ресурси, призначені для відображення вмісту веб-сторінки на екрані користувача.

4.4 Висновки

У результаті розробки програми DonorPlus для управління донорським центром було створено веб-застосунок, який націлений як для персоналу центру, так і для самих донорів. Цей веб-застосунок, побудований на фреймворку Django та використовуючи базу даних PostgreSQL, спрямований

на ефективне зберігання та обробку різноманітної інформації про донорів. Програма має широкий спектр функцій, включаючи реєстрацію, внесення, оновлення та контроль доступу до даних, що забезпечує конфіденційність особистої інформації. Крім того, вона пропонує донорам можливість перегляду власних даних та легкої реєстрації на донації, сприяючи зручності та ефективності участі у процесі донорства.

5 ЕКСПЕРЕМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Формулювання вимог до експерименту

Для проведення експерименту з перевірки працездатності веб-додатку донорського центру необхідно провести оцінку працездатності функціоналу, а саме:

а) коректність інтерфейсу:

1) реєстрація, авторизація та вихід:
   - перевірка функціональності реєстраційної форми, правильність введення та збереження даних;
   - перевірка входу в систему за допомогою авторизації;
   - перевірка процедури виходу з особистого облікового запису, коректне завершення сесії користувача.
2) загальнодоступні сторінки:
   - оцінка доступності та коректності інформації на загальнодоступних сторінках для неавторизованих користувачів.
3) перевірка розділення на групи:
   - перевірка функціоналу, який розділяє користувачів на групи залежно від їхніх прав і статусів.
4) перевірка сторінок відповідно до груп:
   - перевірка доступу до певних сторінок в залежності від групової належності користувачів, впевненість у точності інформації та функціоналів, доступних для кожної групи.

б) перевірка коректності бази даних:

1) перевірка правильності збереження, оновлення та видалення даних в базі даних у контексті дій, виконаних користувачами в інтерфейсі програмного забезпечення;
2) перевірка цілісності та узгодженості даних між інтерфейсом та базою даних.

Мета цього експерименту полягає у перевірці дослідним шляхом функціональності та адекватності веб-застосунку та взаємодію з базою даних. Перевірка орієнтована на встановлення відповідності полів у веб-інтерфейсі користувача та відповідних полів у базі даних, а також перевірка функціоналу правильності веб-інтерфейсу. Крім того, експеримент передбачає перевірку правильності функціоналу веб-інтерфейсу для визначення його коректності в роботі.

5.2 Опис експерименту

У рамках експерименту передбачається тестування розробленого ПЗ, яке було описане у четвертому розділі.

Для тестування зі стаціонарного комп'ютера планується використання програмного забезпечення, призначеного для цієї мети, на операційній системі Windows 10. Крім того, для перевірки функціональності веб-інтерфейсу через стаціонарний комп'ютер, використовуватиметься браузер Google Chrome.

Щодо тестування з ноутбука, передбачається використання вбудованого браузера Microsoft Edge для взаємодії з веб-інтерфейсом на операційній системі Windows 10. Для забезпечення доступу до сайту через динамічну URL-адресу планується використання налаштованого тунельного з'єднання з локальним портом пристрою розробника за допомогою ngrok.

5.3 Проведення експерименту

5.3.1 Підготовка до проведення експерименту

У контексті викликів та завантаження веб-застосунку, існують відмінності у методах, що використовуються для комп'ютера розробника та ноутбука.

У випадку комп'ютера розробника передбачається виклик веб-застосунку та завантаження проходить через локальний хост. Цей процес здійснюється за допомогою спеціалізованих інструментів розробника,

спрямованих на взаємодію з програмою, що виконується на цьому конкретному комп'ютері.

У випадку ноутбука використовується динамічна URL-адреса для завантаження веб-сайту та його взаємодії через відповідний браузер.

5.3.2 Результати досліджень

5.3.2.1 Загальнодоступні сторінки

Відкриваємо у браузері веб-застосунок та завантажуємо головну сторінку сайту, зображено на рисунку 5.1.

Рисунок 5.1 – Відображення головної сторінки

Перехід на сторінку «зв’яжіться з нами», зображено на рисуноку 5.2.

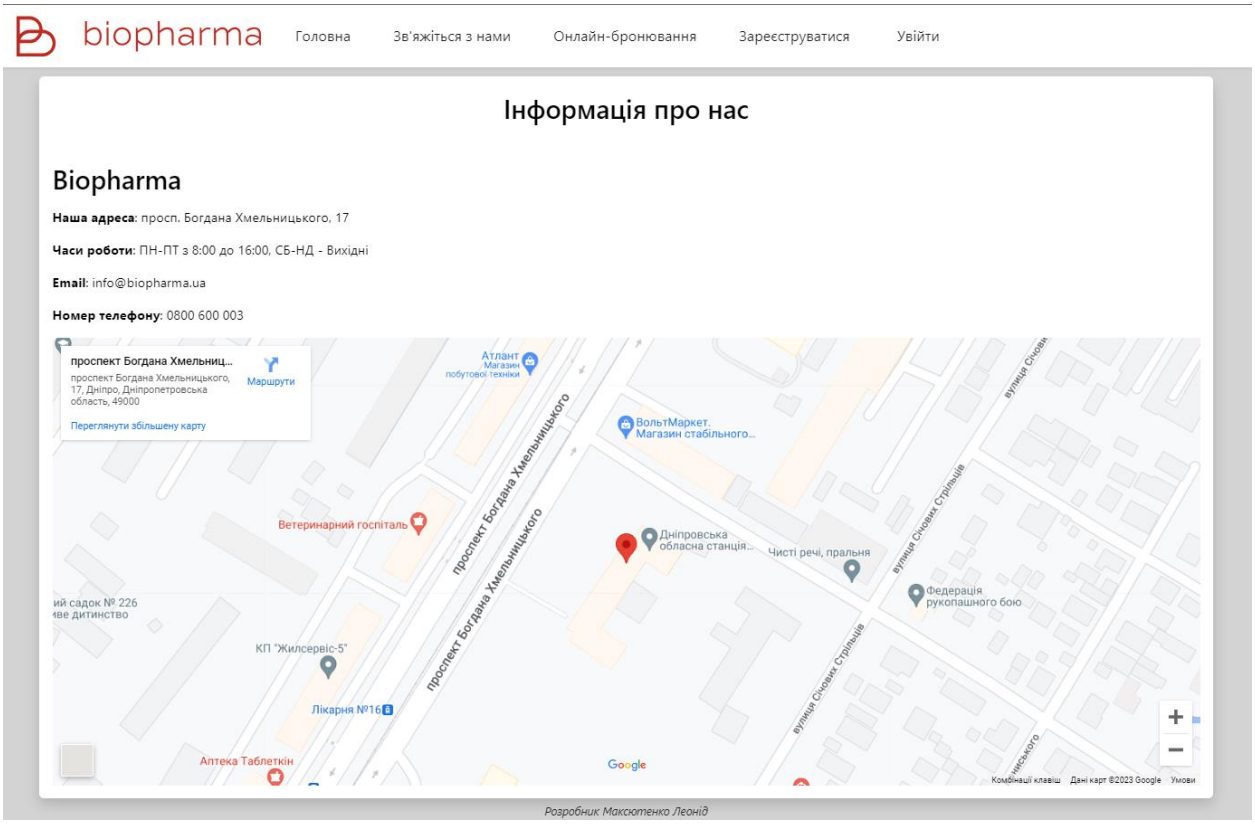


Рисунок 5.2 – Відображення сторінки «зв’яжіться з нами»

5.3.2.2 Реєстрація, авторизація та вихід

Вводимо дані правильно введені дані для форми реєстрації, зображено на рисунку 5.3.

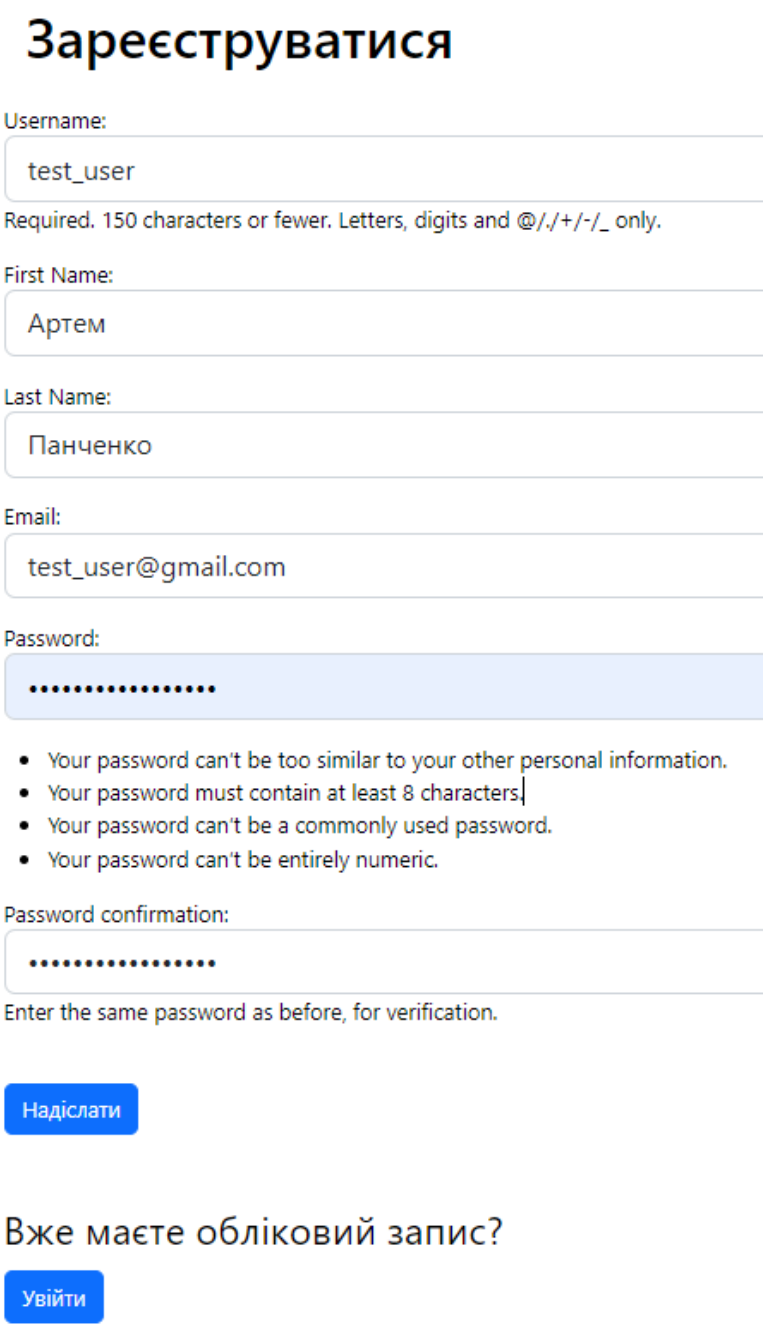


Рисунок 5.3 – Відображення заповнення форми реєстрації

Після вдалої реєстрації відображається повідомлення про результат реєстрації, зображено на рисунку 5.4

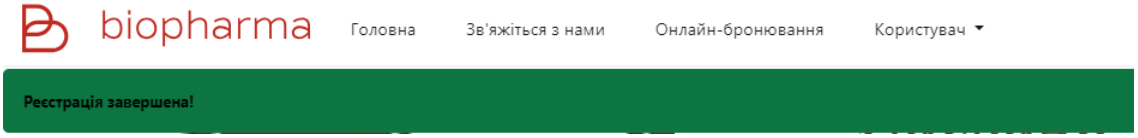


Рисунок 5.4 – Відображення вдалої реєстрації

Спроба зареєструватися, коли пароль «Qwerty123», зображено на рисунку 5.5.

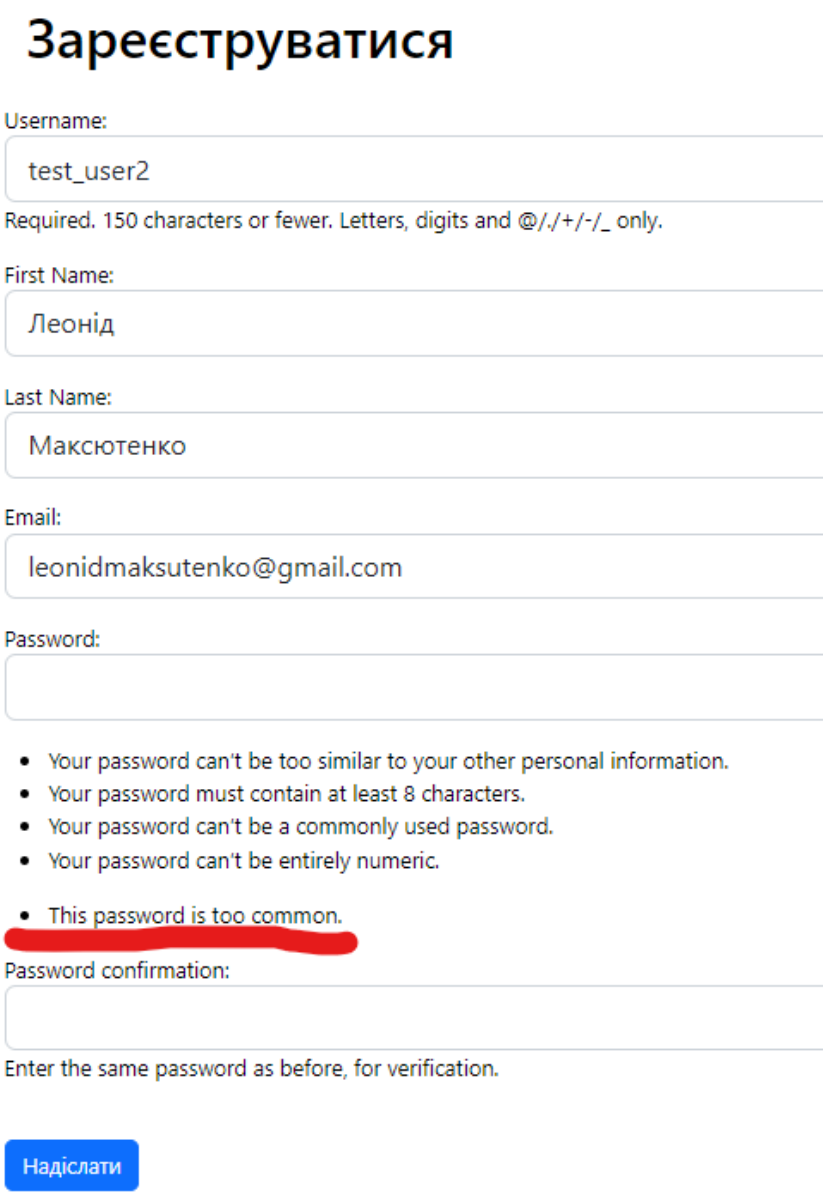


Рисунок 5.5 – Відображення не вдалої реєстрації, неправильний пароль

Спроба авторизуватися, коли пароль був введений неправильно, зображено на рисунку 5.6.

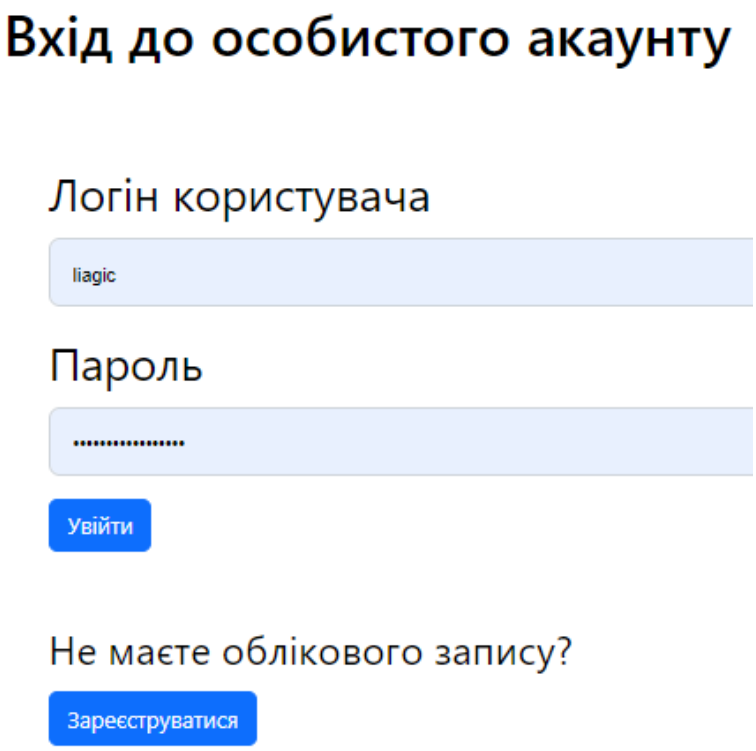


Рисунок 5.6 – Авторизація

Повідомлення про те, що данні були введені неправильно зображено на рисунку 5.7.

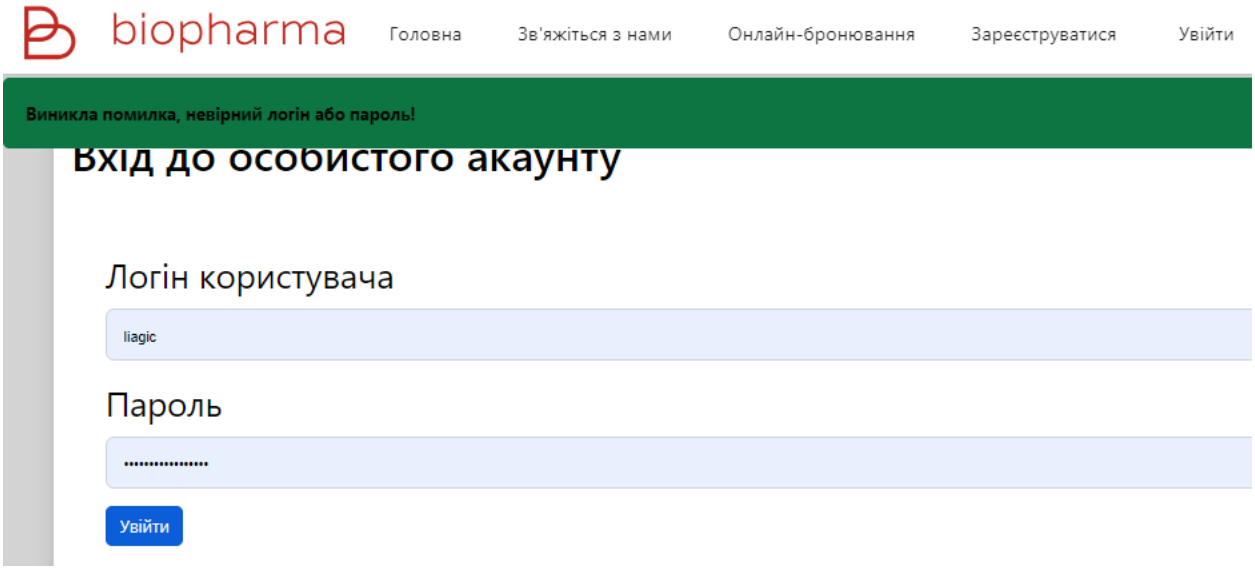


Рисунок 5.7 – Невдала авторизація

На рисунку 5.8 зображено, результат вдалої авторизації, відображається на панелі навігації додаткові посилання на панель користувача, та кнопка вихід із системи.

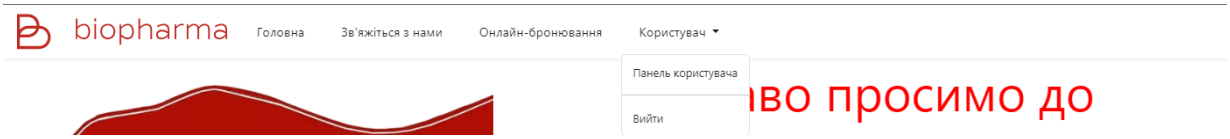


Рисунок 5.8 – Після вдалої авторизації

На рисунку 5.9 відображено повідомлення, яке вказує на вдалий вихід із системи.

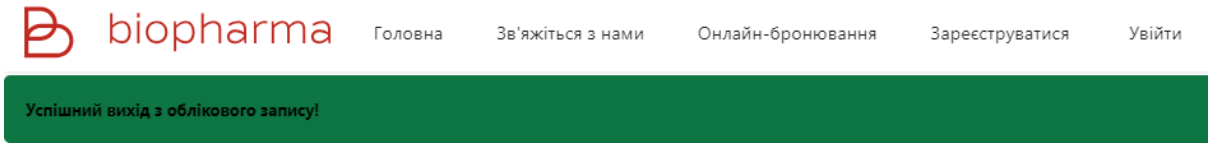


Рисунок 5.9 – Вихід з системи

5.3.2.3 Перевірка розділення на групи та їх функціоналу

Для тестування клієнтської частини, необхідно попередньо авторизуватись як клієнт. Після цього переходимо до панелі користувача, що зображено на рисунку 5.10.

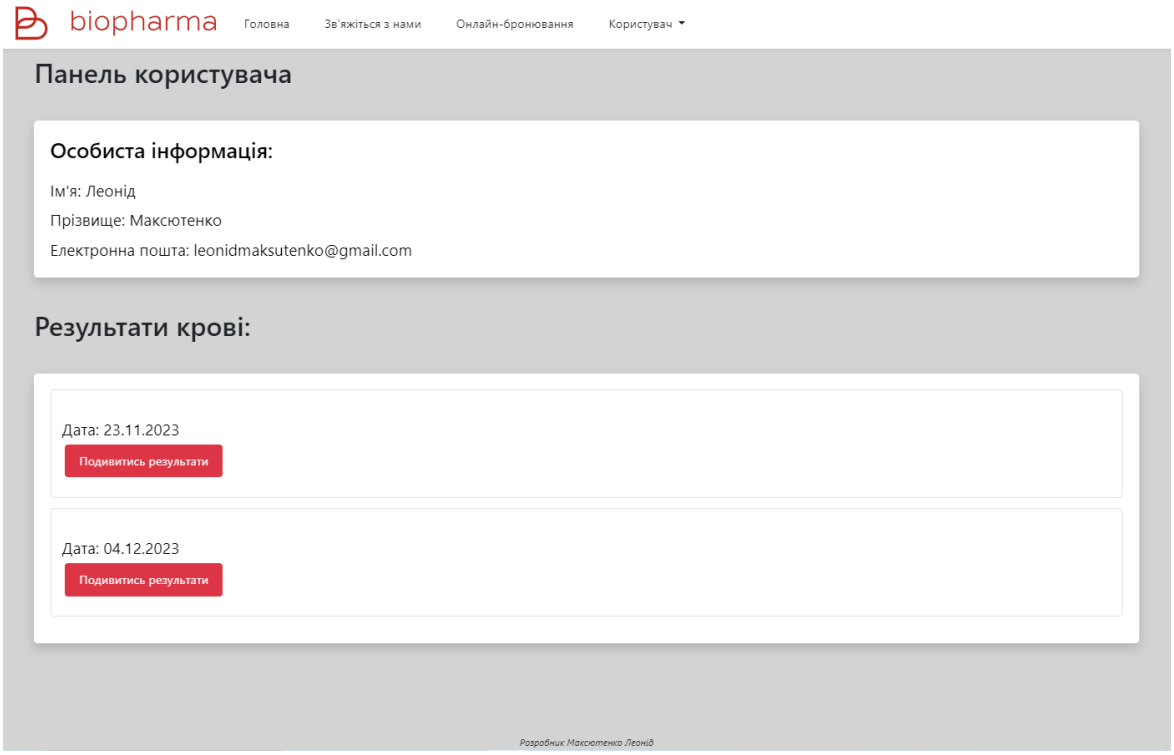


Рисунок 5.10 – Панель користувача

Для створення запису на донацію, необхідно перейти на сторінку «Онлайн бронювання» і обрати бажаний день, зображено на рисунку 5.11.

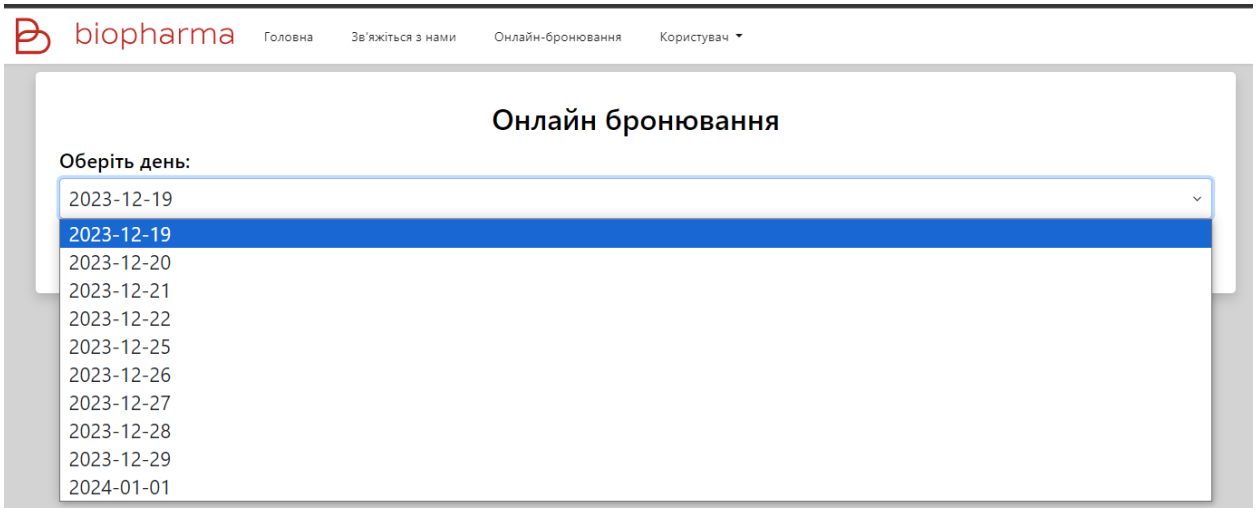


Рисунок 5.11 – Сторінка сайту «Онлайн бронювання»

Після підтвердження обраного дня, сайт перенаправляє користувача на вибір часу для запису, зображено на рисунку 5.12.

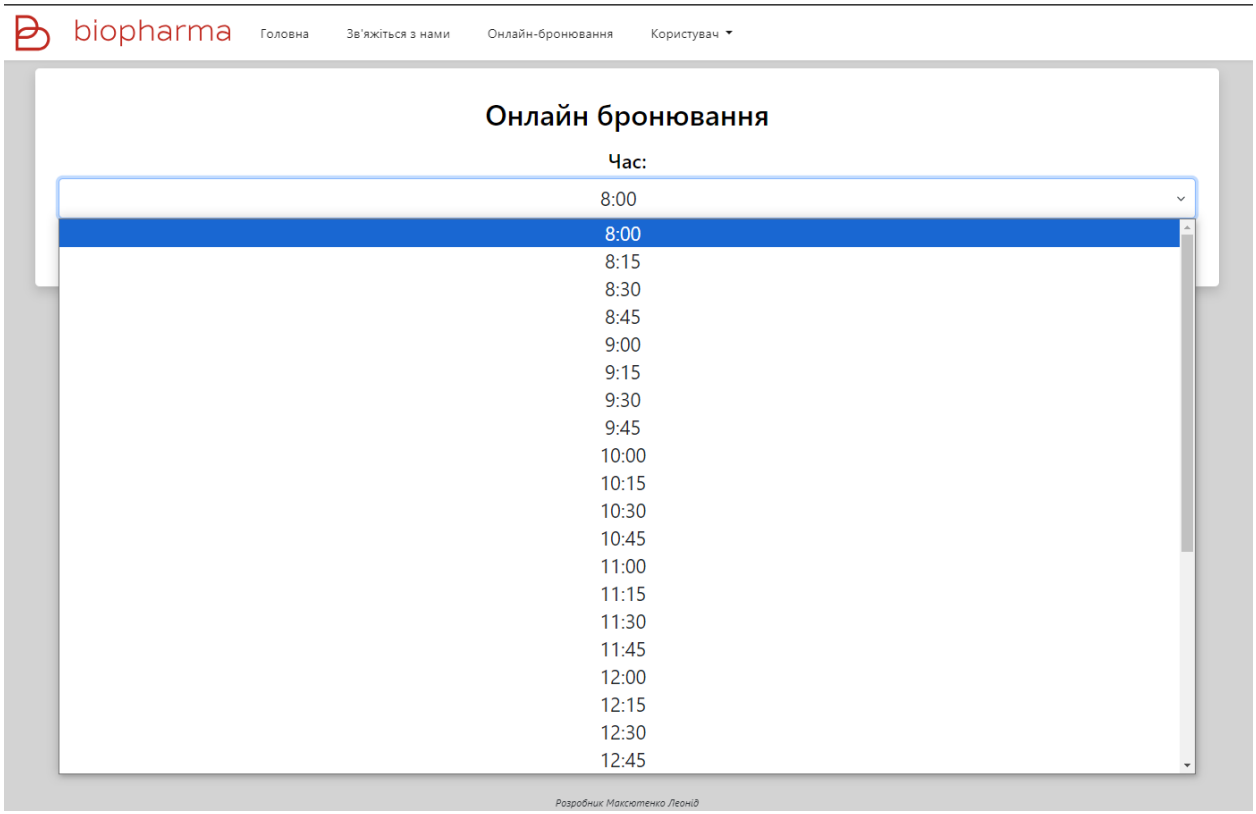


Рисунок 5.12 – Продовження онлайн бронювання

Після підтвердження запису переходимо на панель користувача, де відображено створений запис на донацію, зображено на рисунку 5.13.

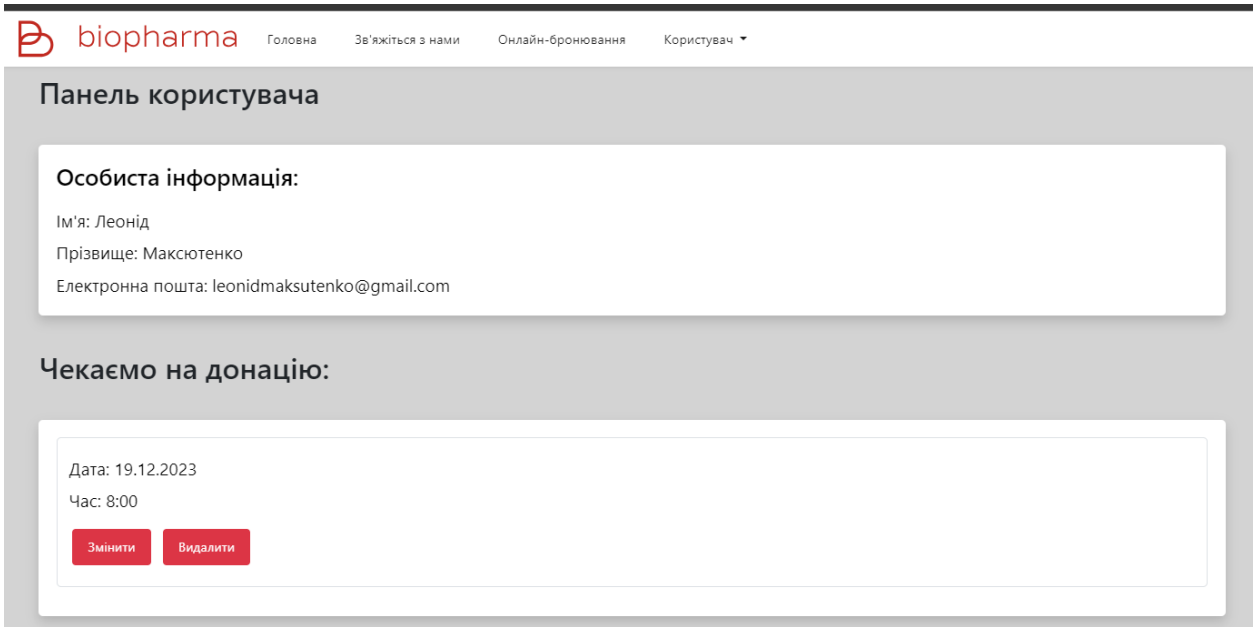


Рисунок 5.13 – Відображення запису на донацію в панелі користувача

Для редагування дати та часу донації необхідно натиснути на кнопку «Змінити», після натискання додаток перенаправить користувача на форму редагування запису на донацію, зображенно на рисунку 5.14.

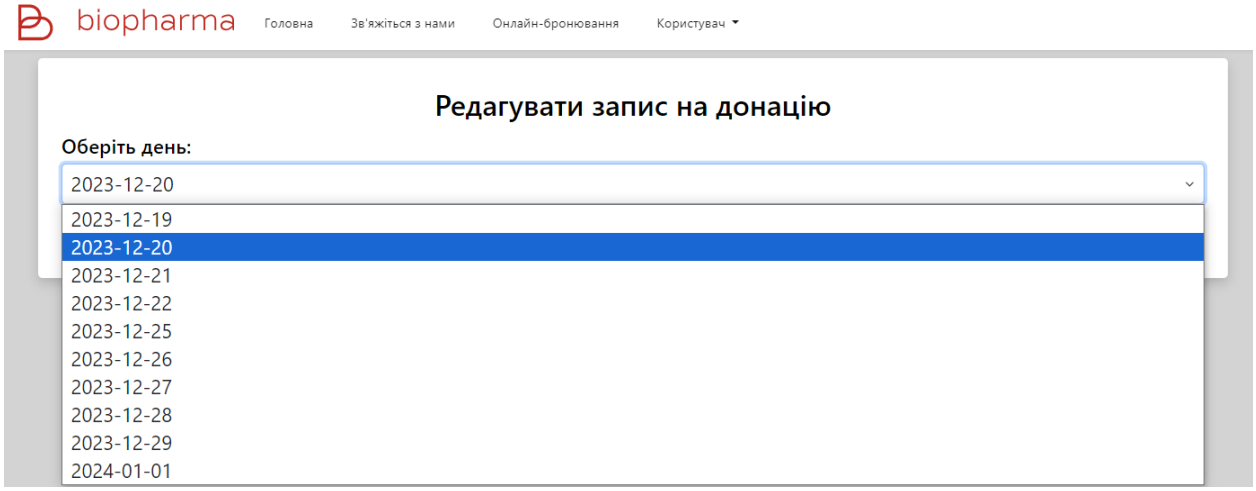


Рисунок 5.14 – Зміна запису на донацію

Після редагування дати на донацію, користувача перенаправляє на форму вибору часу на донацію, зображено на рисунку 5.15.

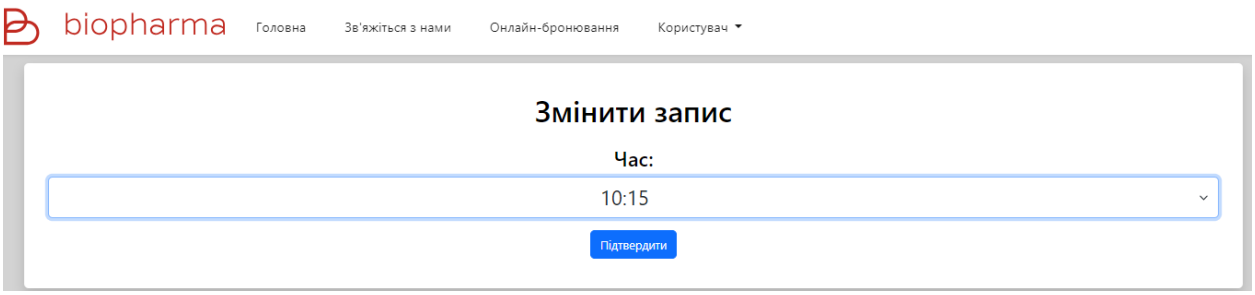


Рисунок 5.15 – Продовження зміни запису на донацію

Після зміни запису на донацію, користувача може переглянути зміни на панелі користувача. Зображено на рисунку 5.16.

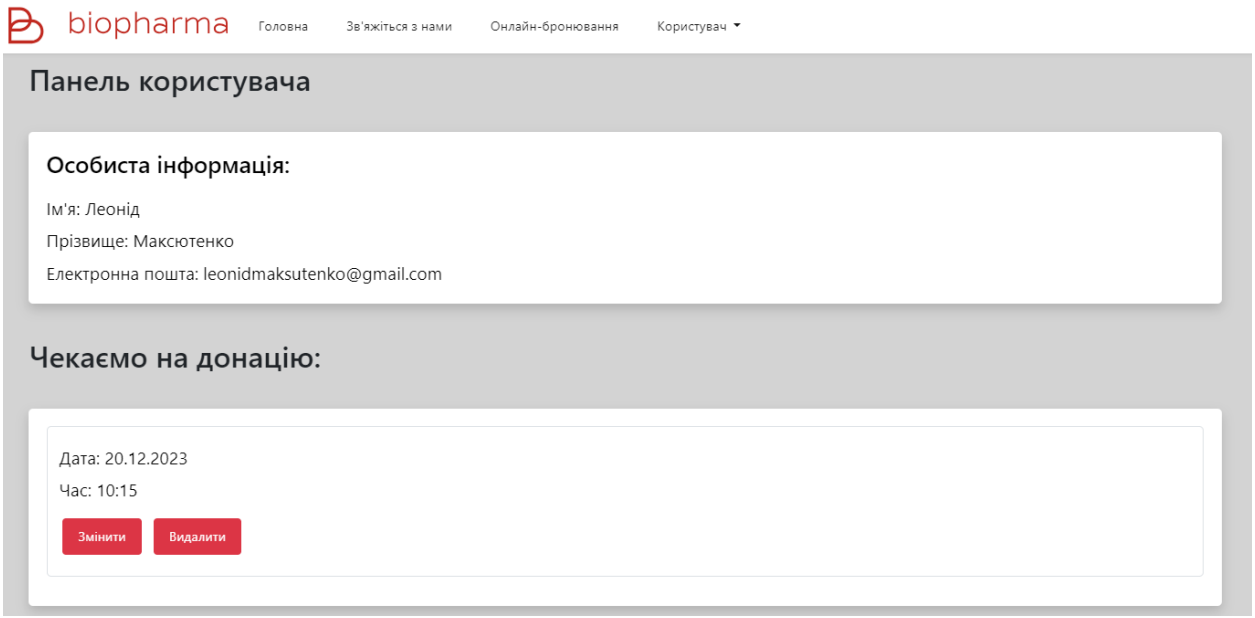


Рисунок 5.16 – перегляд зміни запису на донацію

При спробі створити другий запис на донацію, виведеться повідомлення, що не можна створити більше одного запису на донацію, це зроблено щоб користувачі не могли цілеспрямовано порушити діяльність донорського центру, зображено на рисунку 5.17.

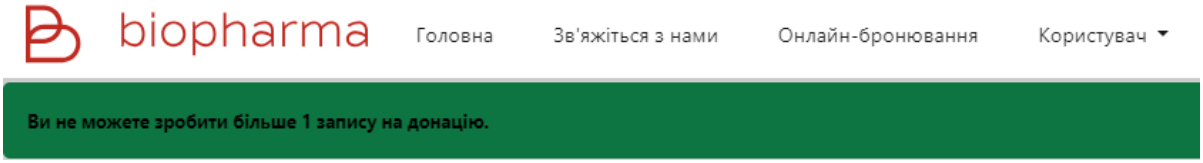


Рисунок 5.17 – Спроба створити другий запис на донацію

При видаленні запису на донацію, викликається повідомлення при вдалому видалені запису, зображено на рисунку 5.18.

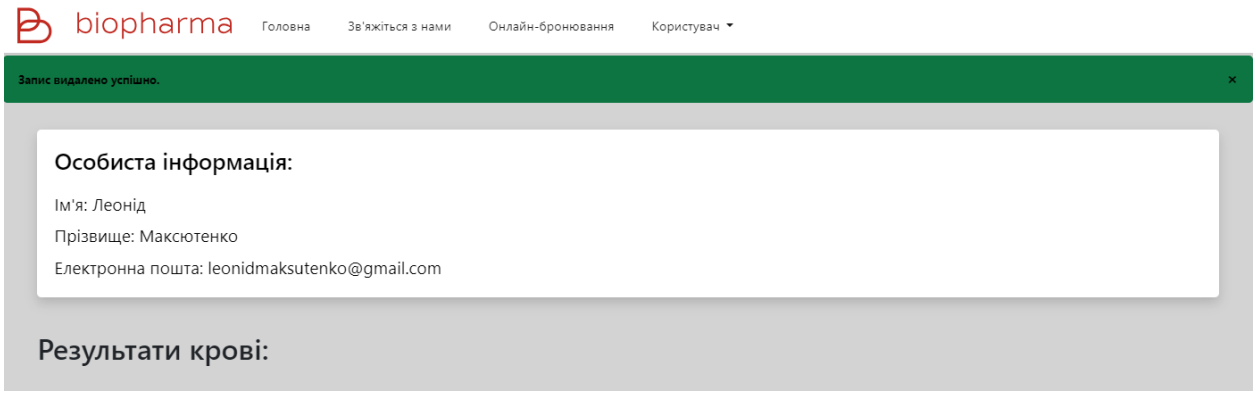


Рисунок 5.18 – Видалення запису на донацію

Після формування терапевтом результатів донації клієнт може подивитись свої результати крові в панелі користувача. Результати крові зображено на рисунках 5.19 – 5.21.

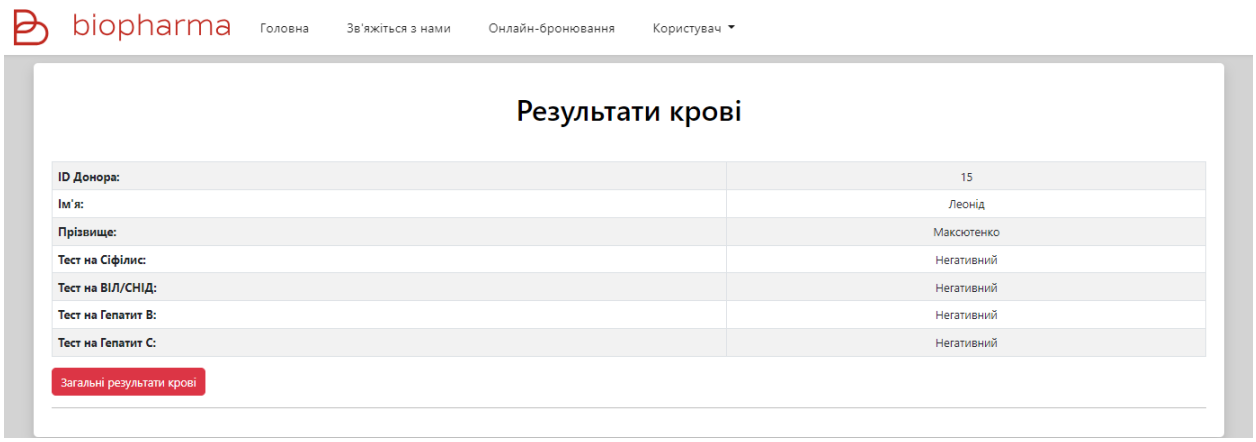
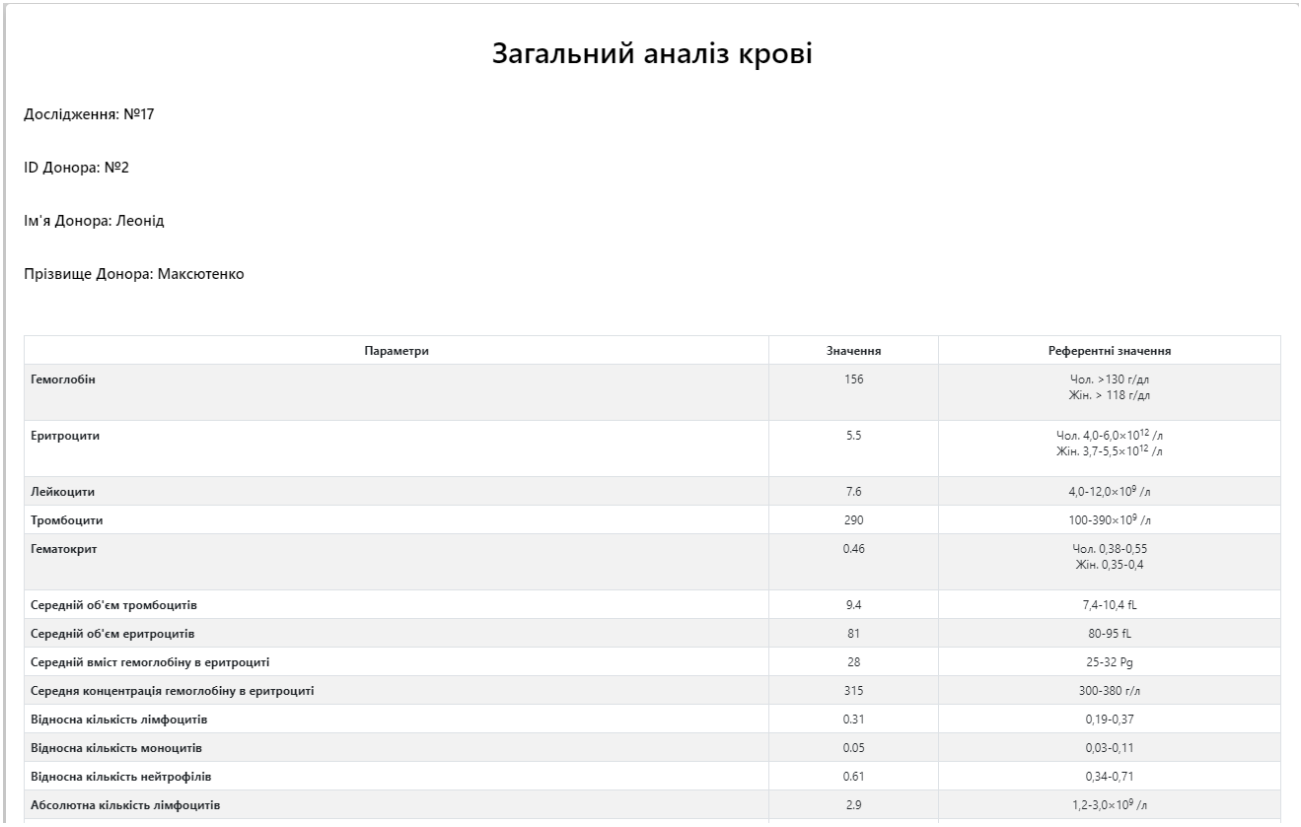


Рисунок 5.19 – перегляд результатів крові після донації



Загальний аналіз крові

Дослідження: №17

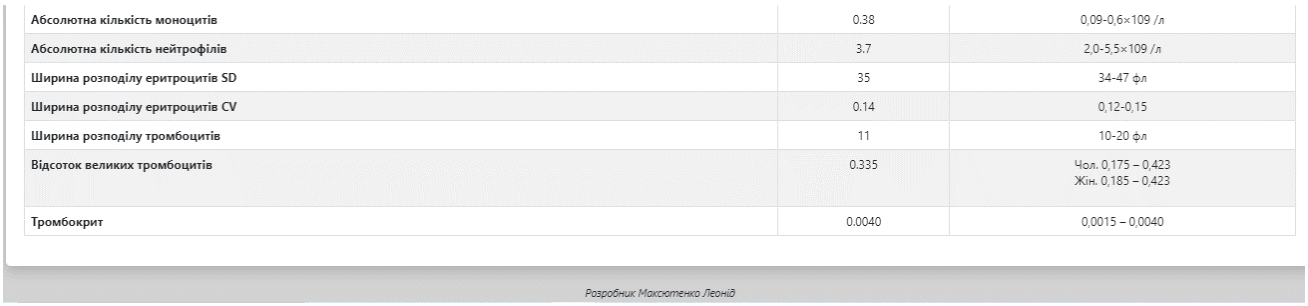
ID Донора: №2

Ім'я Донора: Леонід

Прізвище Донора: Максютенко

Параметри	Значення	Референтні значення
Гемоглобін	156	Чол. >130 г/дл Жін. > 118 г/дл
Еритроцити	5.5	Чол. 4,0-6,0×10^{12} /л Жін. 3,7-5,5×10^{12} /л
Лейкоцити	7.6	4,0-12,0×10^{9} /л
Тромбоцити	290	100-390×10^{9} /л
Гематокрит	0.46	Чол. 0,38-0,55 Жін. 0,35-0,4
Середній об'єм тромбоцитів	9.4	7,4-10,4 fL
Середній об'єм еритроцитів	81	80-95 fL
Середній вміст гемоглобіну в еритроциті	28	25-32 Pg
Середня концентрація гемоглобіну в еритроциті	315	300-380 г/л
Відносна кількість лімфоцитів	0.31	0,19-0,37
Відносна кількість моноцитів	0.05	0,03-0,11
Відносна кількість нейтрофілів	0.61	0,34-0,71
Абсолютна кількість лімфоцитів	2.9	1,2-3,0×10^{9} /л

Рисунок 5.20 – продовження перегляду результатів крові



Абсолютна кількість моноцитів	0.38	0,09-0,6×109 /л
Абсолютна кількість нейтрофілів	3.7	2,0-5,5×109 /л
Ширина розподілу еритроцитів SD	35	34-47 фл
Ширина розподілу еритроцитів CV	0.14	0,12-0,15
Ширина розподілу тромбоцитів	11	10-20 фл
Відсоток великих тромбоцитів	0.335	Чол. 0,175 – 0,423 Жін. 0,185 – 0,423
Тромбокрит	0.0040	0,0015 – 0,0040

Розробник Максютенко Леонід

Рисунок 5.21 – кінець сторінки перегляду загальних результатів крові

Для перевірки функціоналу групи користувачів «Реєстратура», необхідно авторизуватись як працівник даного відділу. Після авторизації відображається сторінки до яких може перейти працівник, зображено на рисунку 5.22.

Рисунок 5.22 – Відображення доступних сторінок для реєстратури

На рисунку 5.23 зображено таблиця з записами на донацію, де є можливість переглянути картку донора та видалити запис, після того, як прийняли цього донора.

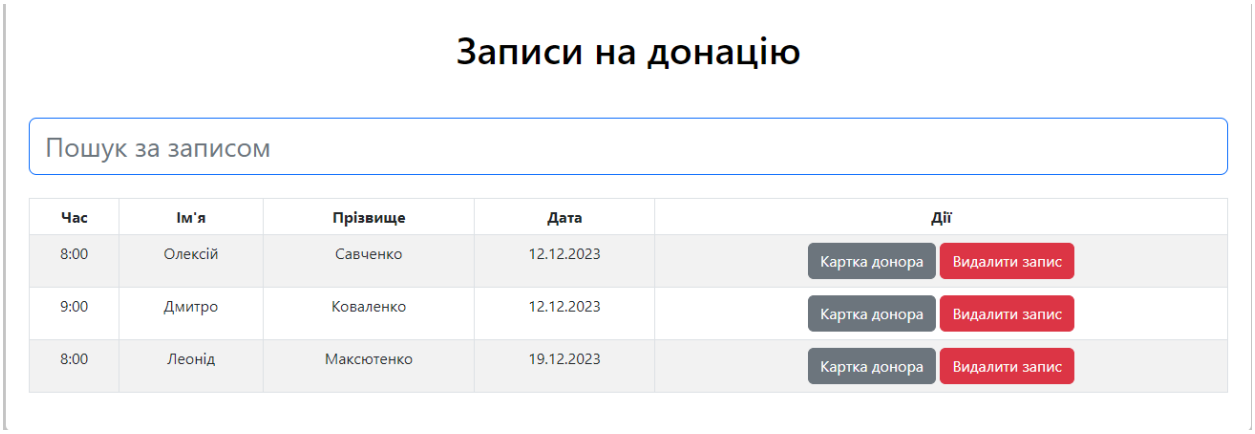


Рисунок 5.23 – Відображення записів на донацію

У веб-інтерфейсі реалізовано функціонал пошуку за різними критеріями. Якщо користувач вводить значення поля, якого не існує, програма не поверне жодних результатів. Однак, при введенні існуючого значення поля, програма відображає відповідне поле. Цей функціонал проілюстровано на рисунку 5.24.

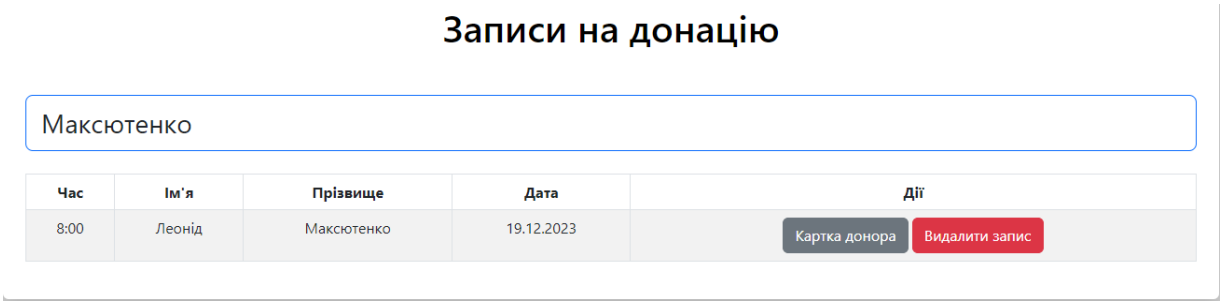


Рисунок 5.24 – Перевірка пошуку в записах на донацію

Під час активації кнопки, веб-додаток проводить перевірку наявності картки донора. У випадку наявності в системі відповідної картки донора, користувачеві надсилається відповідна інформація. Однак, у випадку відсутності такої картки, додаток автоматично перенаправляє користувача на форму для створення нової картки донора. Ці функціональні можливості проілюстровані на рисунках 5.24 і 5.25.

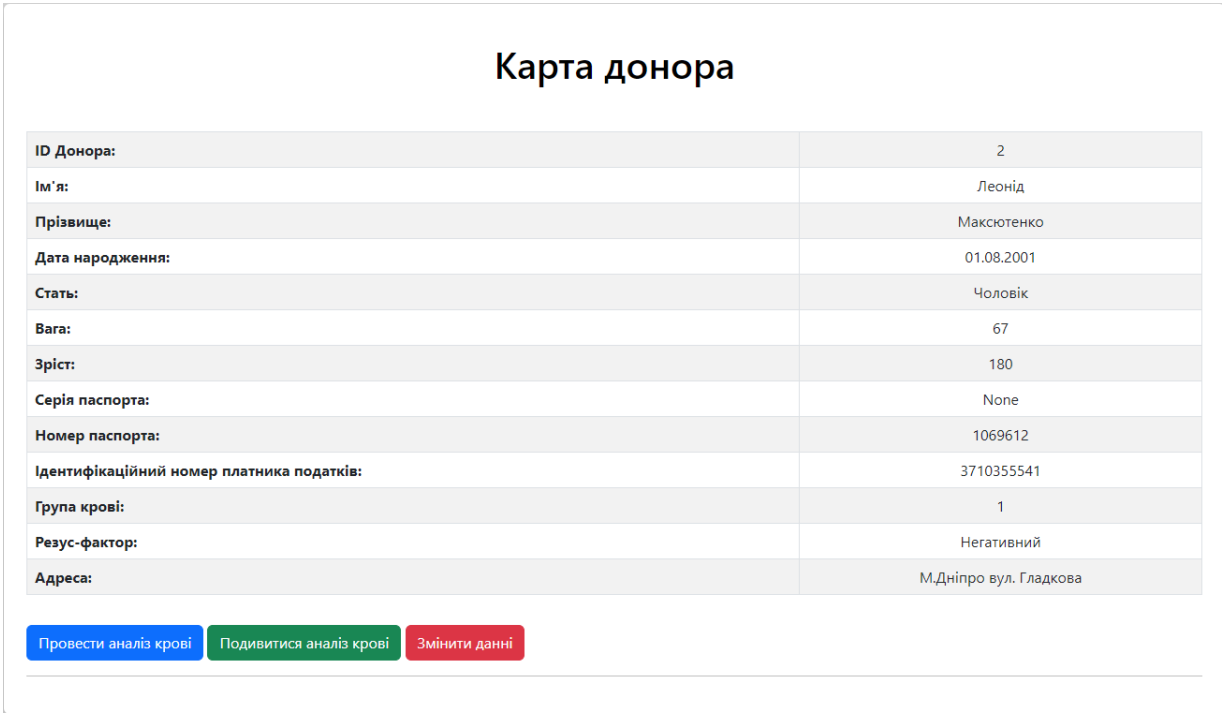


Рисунок 5.24 – Перегляд картки донора користувачем з групи «Реєстратура»

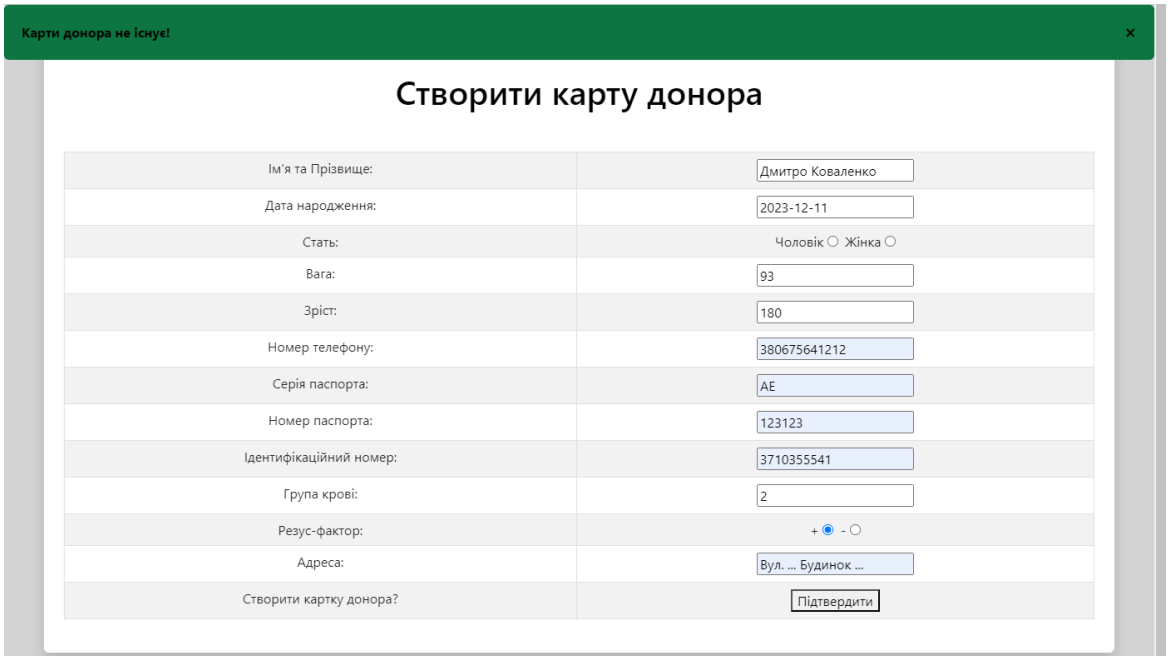


Рисунок 5.25 – Створення картки донора

Для зміни інформації в картці донора користувач перенаправляється на відповідну форму картки донора, зображено на рисунку 5.26

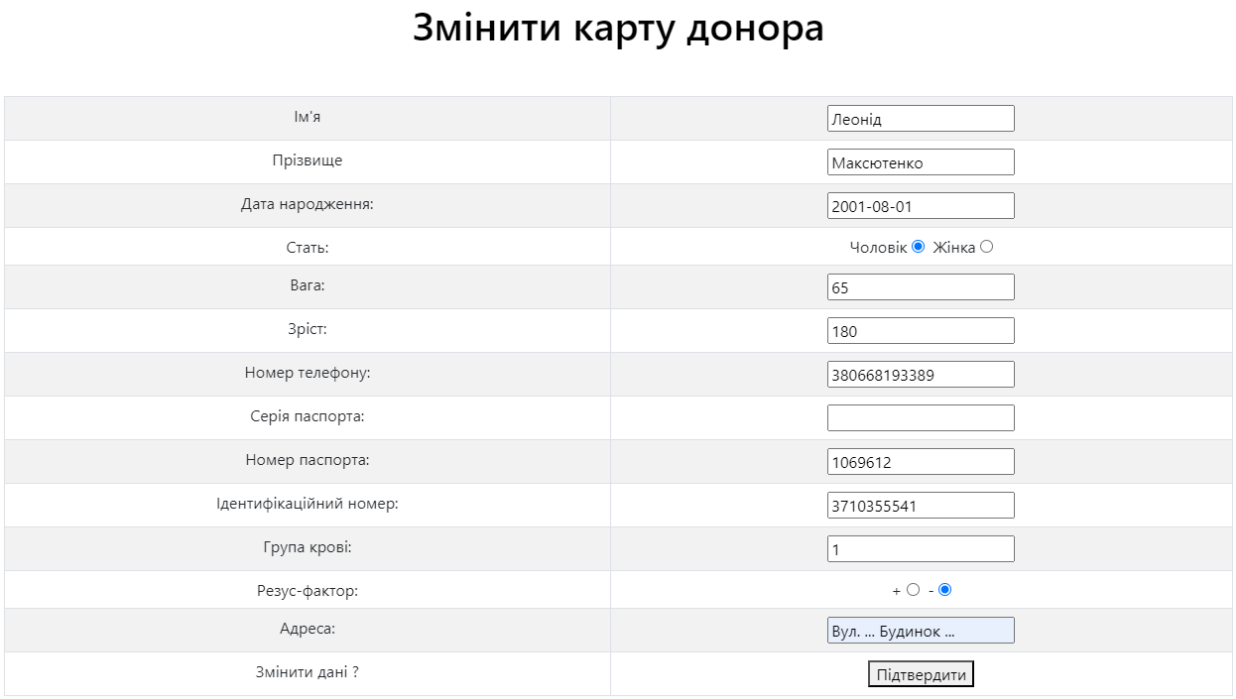


Рисунок 5.26 – Змінна даних в картці донора

Для проведення загального аналізу крові, працівник з групи «Реєстратура» відправляє запит на сервер, що ініціює запит на отримання аналізу крові. Після завершення процедури аналізатор крові надсилає дані на сервер та сервер зберігає їх у базі даних. У випадку успішного завершення аналізу, сервер надсилає повідомлення про успішно проведений аналіз крові. Вдалий результат зображено на рисунку 5.27.

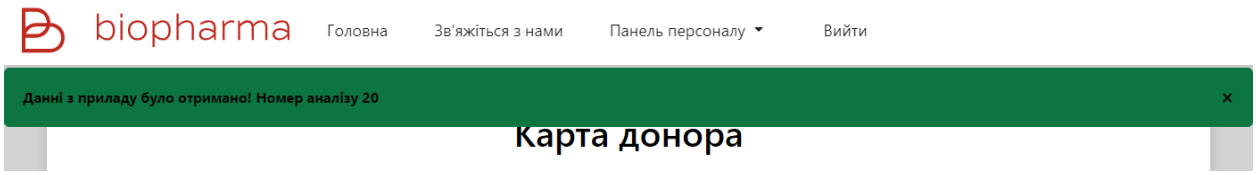


Рисунок 5.27 – проведення загального аналізу донора

Для доступу до загальних результатів аналізу крові необхіно натиснути на кнопку «Подивитися загальні аналізи», що спрямовує користувача до

переліку результатів крові певного донора. Далі, для перегляду конкретних результатів крові, слід натиснути відповідну кнопку в таблиці, що призведе до переадресації користувача до відповідної інформації. Цей процес ілюструється на рисунках 5.28 – 5.30.

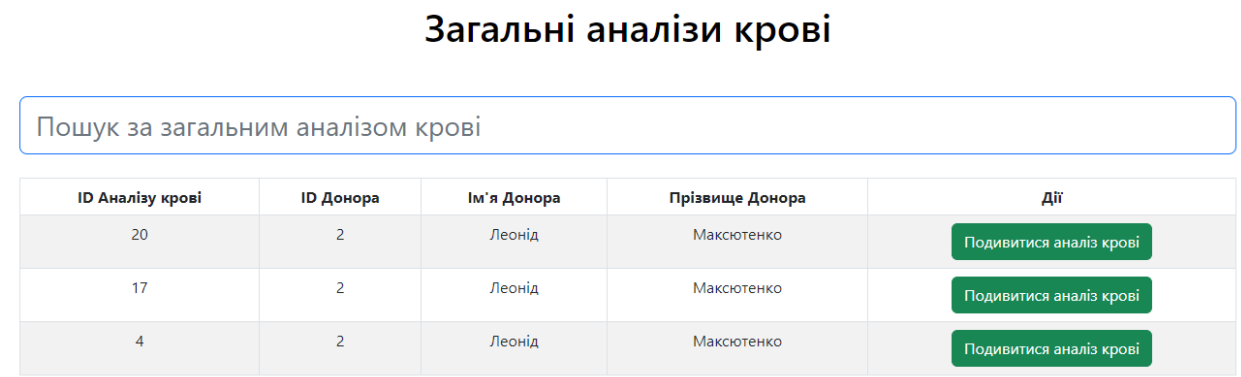


Рисунок 5.28 – Перегляд списку аналізів крові відповідного донора

Рисунок 5.29 – Результати аналізу крові

Середній об'єм тромбоцитів	8.1	7,4-10,4 fL
Середній об'єм еритроцитів	87	80-95 fL
Середній вміст гемоглобіну в еритроциті	30	25-32 Pg
Середня концентрація гемоглобіну в еритроциті	304	300-380 г/л
Відносна кількість лімфоцитів	0.22	0,19-0,37
Відносна кількість моноцитів	0.06	0,03-0,11
Відносна кількість нейтрофілів	0.55	0,34-0,71
Абсолютна кількість лімфоцитів	1.3	$1,2\text{-}3,0\times10^9$ /л
Абсолютна кількість моноцитів	0.35	0,09-0,6×109 /л
Абсолютна кількість нейтрофілів	4.4	2,0-5,5×109 /л
Ширина розподілу еритроцитів SD	43	34-47 фл
Ширина розподілу еритроцитів CV	0.13	0,12-0,15
Ширина розподілу тромбоцитів	15	10-20 фл
Відсоток великих тромбоцитів	0.330	Чол. 0,175 – 0,423 Жін. 0,185 – 0,423
Тромбокрит	0.0032	0,0015 – 0,0040

Рисунок 5.30 – Продовження результату аналізу крові

Для процедури видалення запису щодо донації необхідно відвідати відповідну сторінку з інформацією про цю донацію і натиснути кнопку «Видалити запис». Після цього з'являється повідомлення, що запитує підтвердження бажання видалити запис про донацію. Після підтвердження цього запиту, запис видаляється, і на екрані з'являється повідомлення про успішне видалення. Це проілюстровано на рисунках 5.31 – 5.32.

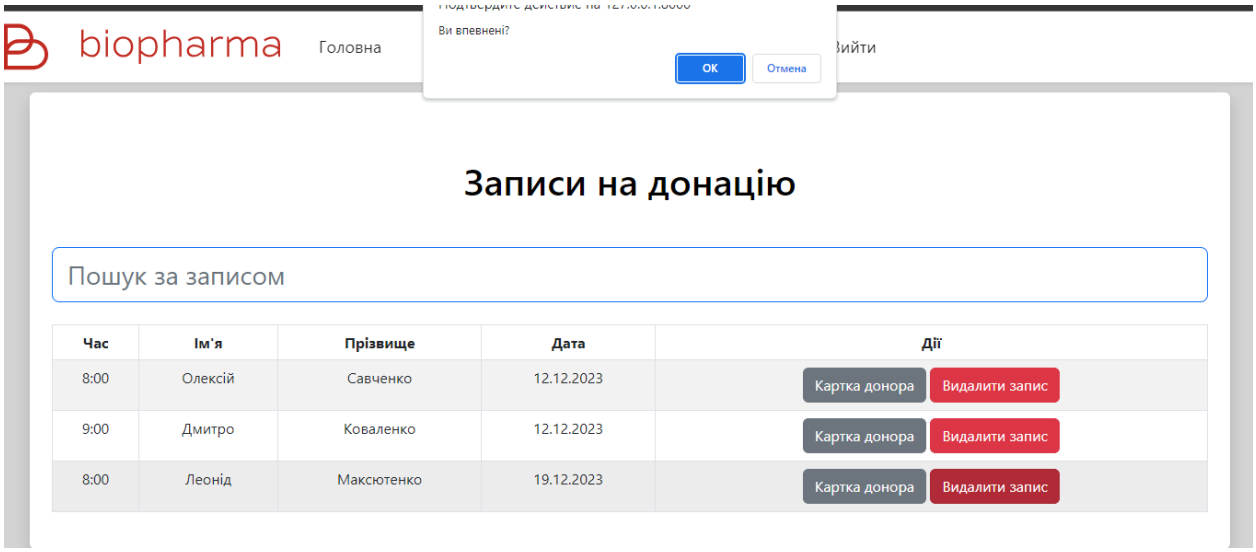


Рисунок 5.31 – Видалення запису на донацію

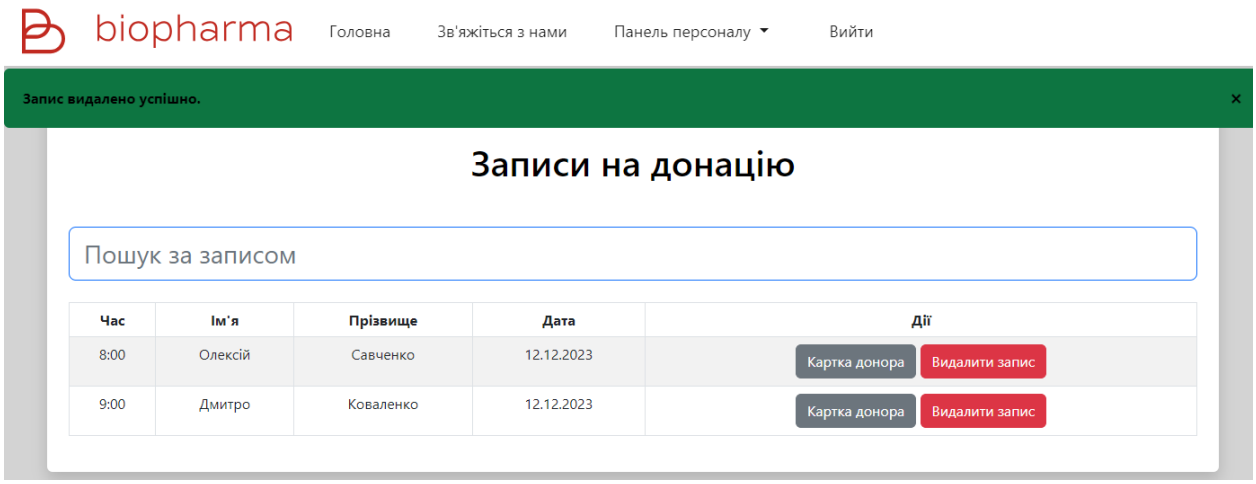


Рисунок 5.32 – Після видалення запису на донацію

Для перегляду існуючих донорів необхідно в панелі персоналу перейти на відповідну сторінку «Донори», це проілюстровано на рисунку 5.33.

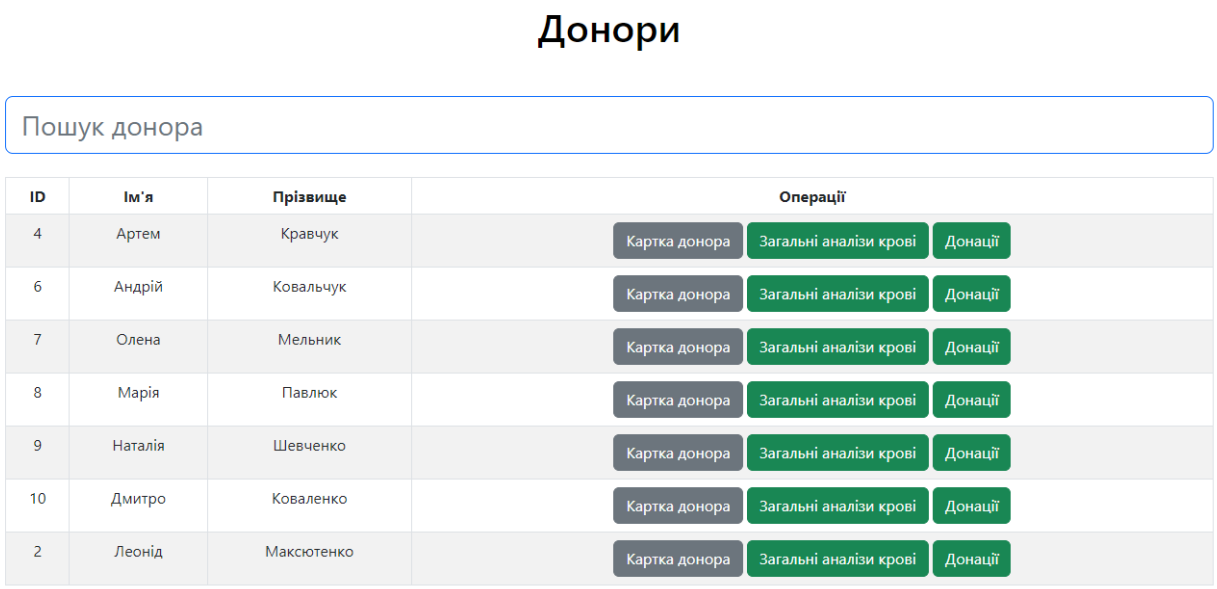


Рисунок 5.33 – Перегляд донорів

Для перегляду донацій, переходимо в панелі користувача на «Донації», проілюстровано на рисунку 5.34.

Донації

Пошук за донаціями

ID Донації	ID Донора	Ім'я Донора	Прізвище Донора	Дата	Дії
15	7	Олена	Мельник	04.12.2023	Детальніше
14	8	Марія	Павлюк	04.12.2023	Детальніше
13	2	Леонід	Максютенко	04.12.2023	Детальніше
11	6	Андрій	Ковальчук	04.12.2023	Детальніше
10	4	Артем	Кравчук	04.12.2023	Детальніше
4	2	Леонід	Максютенко	23.11.2023	Детальніше

Рисунок 5.34 – Перегляд донацій

Для тестування можливостей групи користувачів 2Відділ прийому донорів» потрібно увійти в систему як співробітник цього відділу. Після успішної авторизації відобразиться перелік сторінок, доступних для перегляду та використання працівником, як показано на рисунку 5.35.

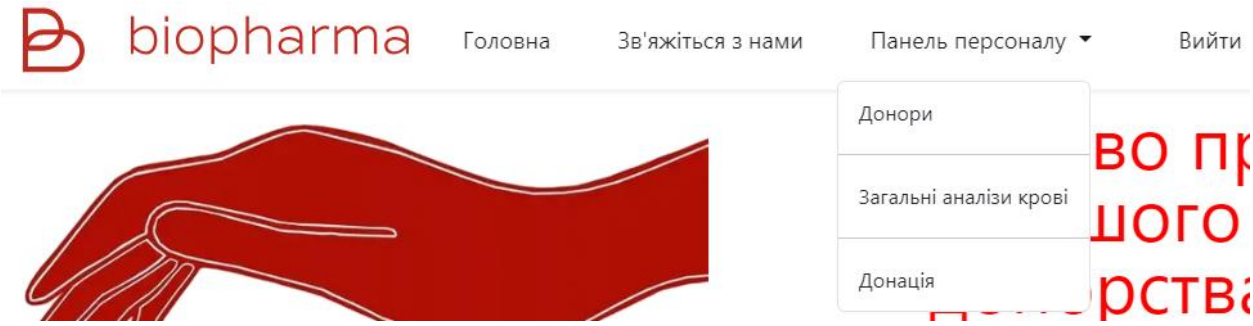


Рисунок 5.35 – Відображення доступних сторінок для відділу прийому донорів

Для перегляду загальних результатів аналізу крові слід натиснути кнопку «Подивитися загальні аналізи», що перенаправить користувача на сторінку з переліком результатів крові для певного донора. На рисунку 5.36 показана таблиця зі загальними результатами аналізу крові.

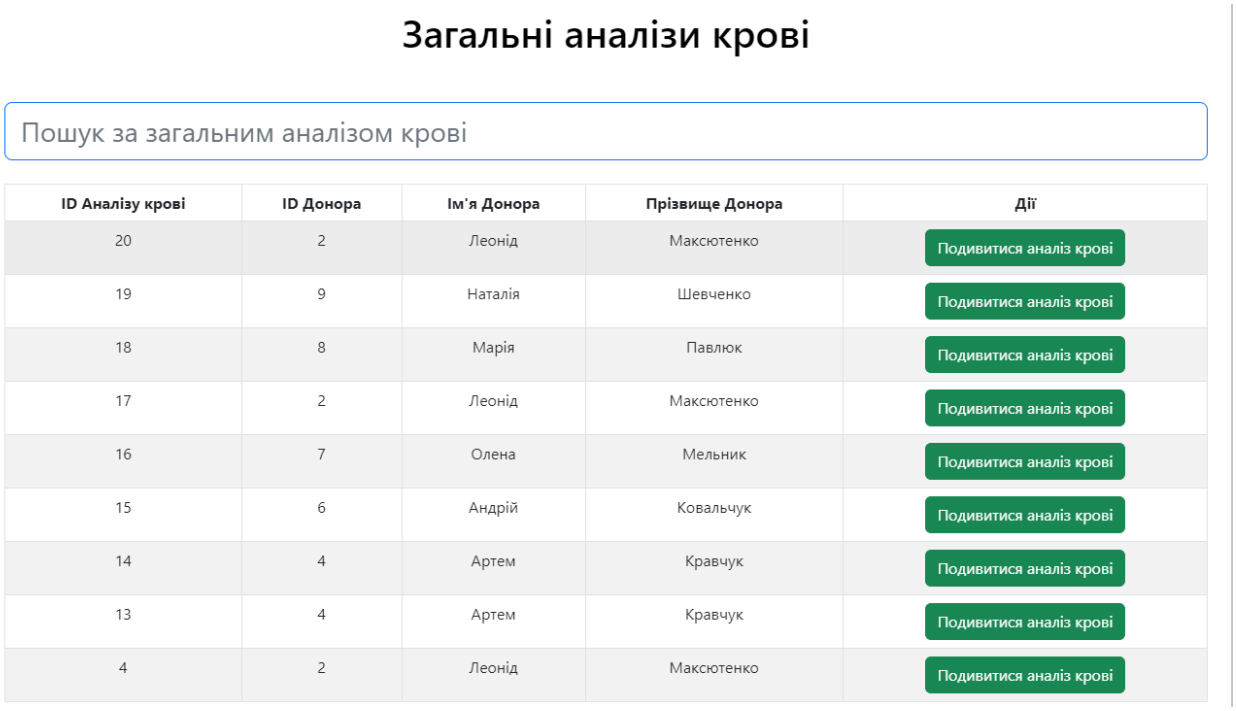


Рисунок 5.36 – Сторінка з загальними аналізами крові

Для перегляду інформації про донації потрібно у панелі персоналу обрати опцію «Донації», що перенаправить вас на сторінку зі списком доступних донацій. Результат процесу показано на рисунку 5.37.

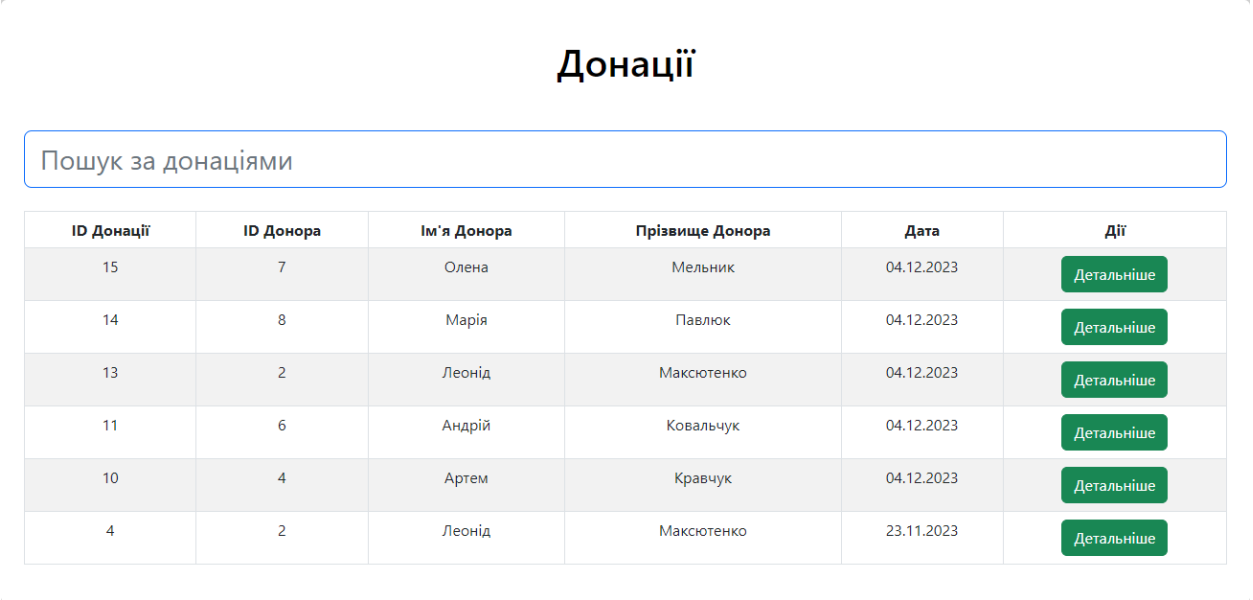


Рисунок 5.37 – Сторінка з донаціями

Для доступу до інформації про донорів потрібно вибрати на сторінці панелі керування опцію «Донори», що веде до сторінки зі списком даних про донорів. Це зображено на рисунку 5.38.

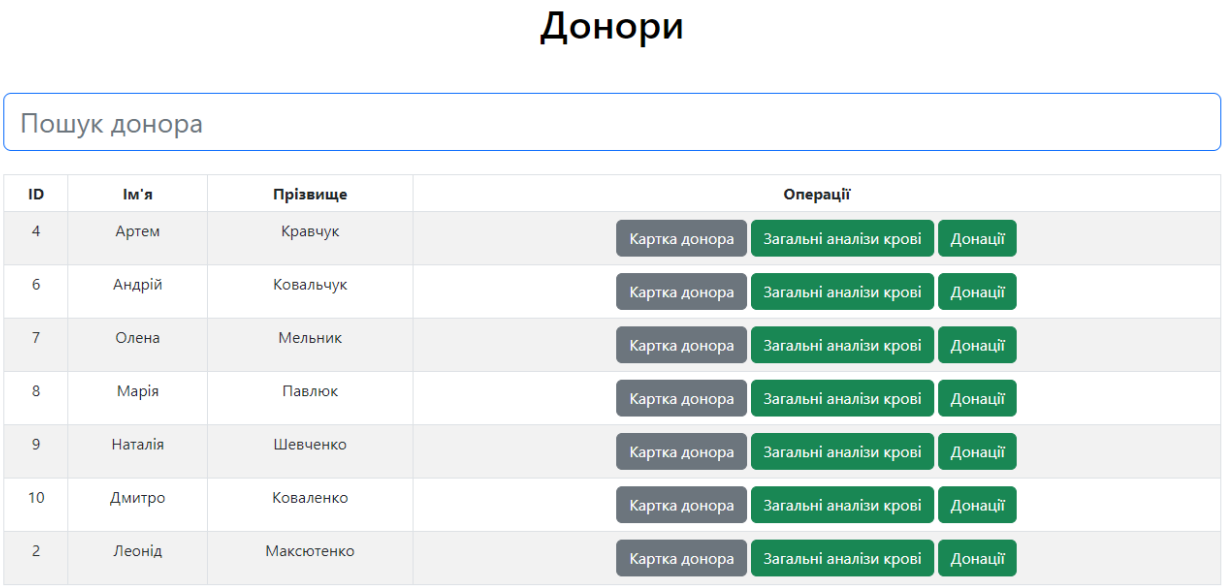


Рисунок 5.38 – Сторінка з донорами

У списку донорів є кнопка «Картка донора», яка при натисканні перенаправляє на індивідуальну картку донора. Це зображено на рисунку 5.39.

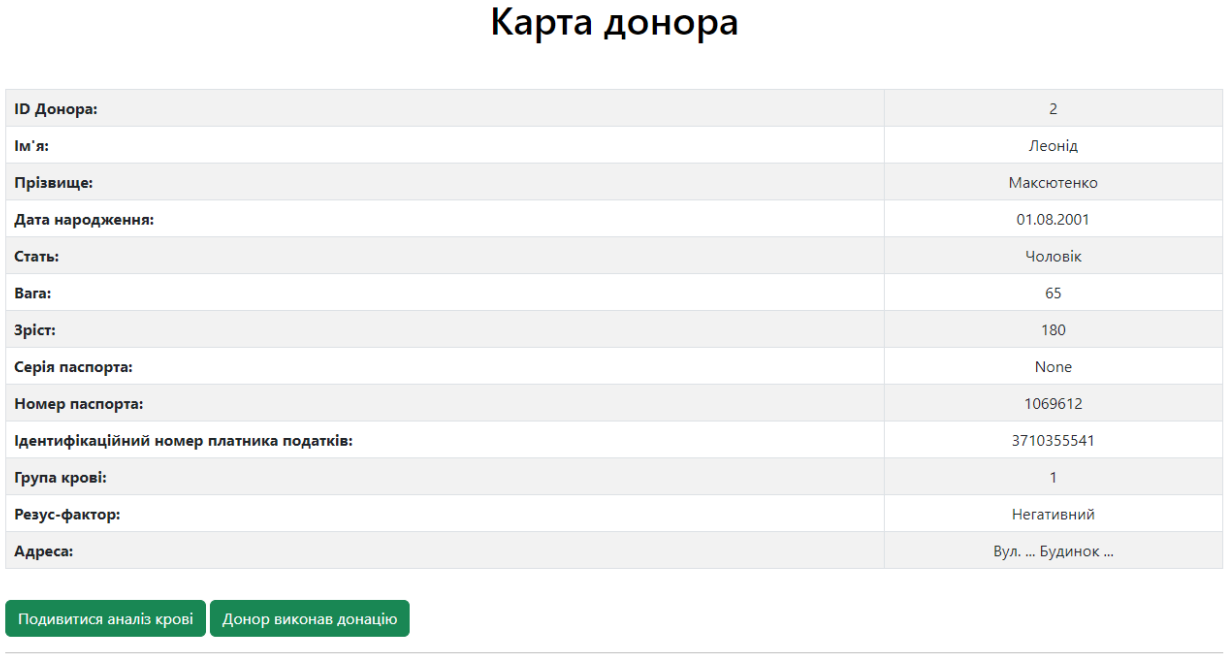


Рисунок 5.39 – Картка донора

У картці донора для групи «Відділ прийому донорів» є кнопка «Донор виконав донацію», яка перенаправляє на форму створення донації. Процес успішного створення донації зображено на рисунках 5.40 – 5.42.

Донор виконав донацію

Worker:	Worker object (2)
Ім'я та Прізвище:	Леонід Максютенко
Device number:	3
Donation date:	2023-12-11
Procedure time:	76
Successful procedure:	☑
Comment:	Вдала донація, 800 мл
Підтвердити донацію?	Підтвердити

Рисунок 5.40 – Форма створення донації

Донації

Пошук за донаціями

ID Донації	ID Донора	Ім'я Донора	Прізвище Донора	Дата	Дії
16	2	Леонід	Максютенко	11.12.2023	Детальніше
15	7	Олена	Мельник	04.12.2023	Детальніше
14	8	Марія	Павлюк	04.12.2023	Детальніше
13	2	Леонід	Максютенко	04.12.2023	Детальніше
11	6	Андрій	Ковальчук	04.12.2023	Детальніше
10	4	Артем	Кравчук	04.12.2023	Детальніше
4	2	Леонід	Максютенко	23.11.2023	Детальніше

Рисунок 5.41 – Відображення створеної донації

Донація

Параметри	Значення
Номер донації	№ 16
Дата донації	11.12.2023
ID донора	2
Донор	Леонід Максютенко
ID працівника	2
Працівник	Ольга Коваленко
№ пристрою	3
Час процедури	76 хвилин
Результат процедури	Вдала донація
Коментар	Вдала донація, 800 мл

Рисунок 5.42 – Відображення інформації створеної донації

Щоб протестувати можливості групи користувачів «Терапевт», потрібно увійти в систему як працівник цього відділу. Після успішної авторизації відкриється список сторінок, доступних для перегляду та використання працівником, як показано на рисунку 5.43.

Рисунок 5.44 – Відображення доступних сторінок для групи «Терапевти»

Терапевт має право доступу до карток донорів, повних аналізів крові, загальних аналізів крові та донацій. Для переходу на ці сторінки використовується навігаційна панель сайту. Результати цих переходів представлені на рисунках 5.45 – 5.48.

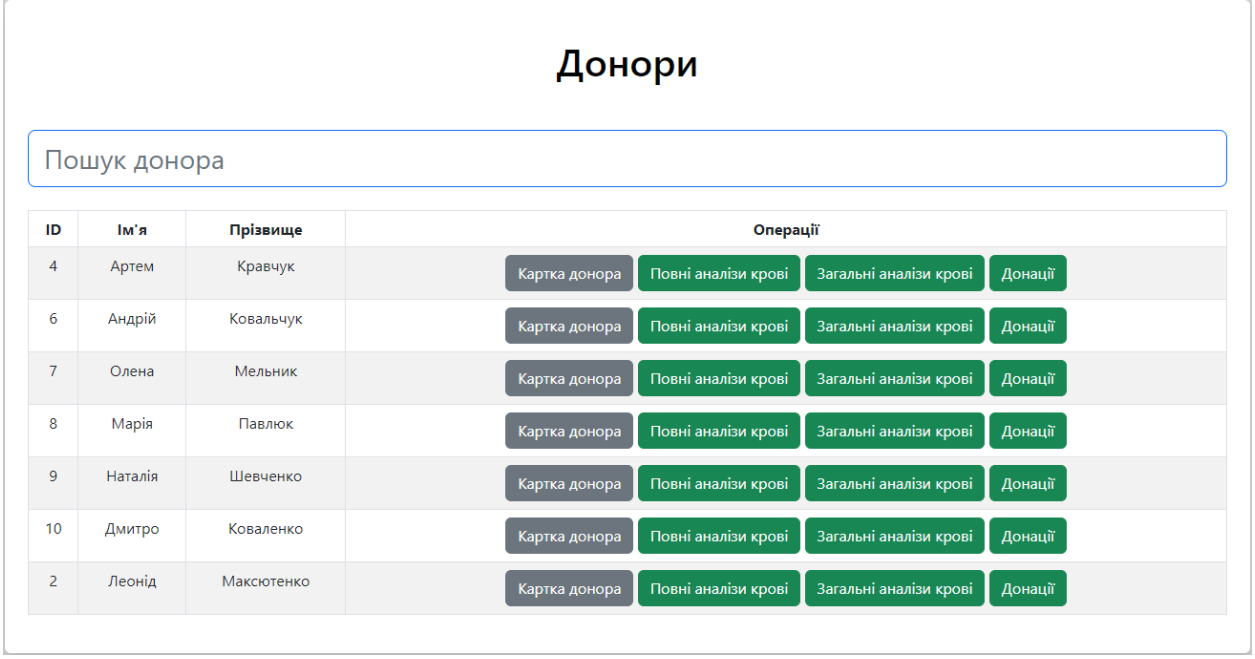


Рисунок 5.45 – Перехід до сторінки з донорами

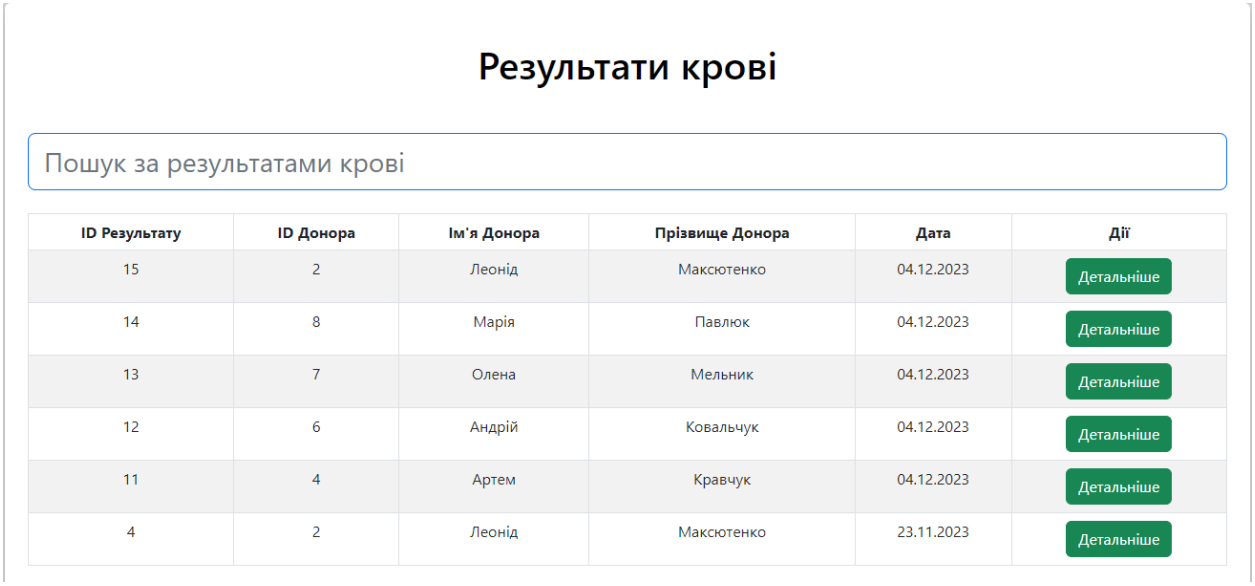


Рисунок 5.46 – Перехід до сторінки з повними результатами крові

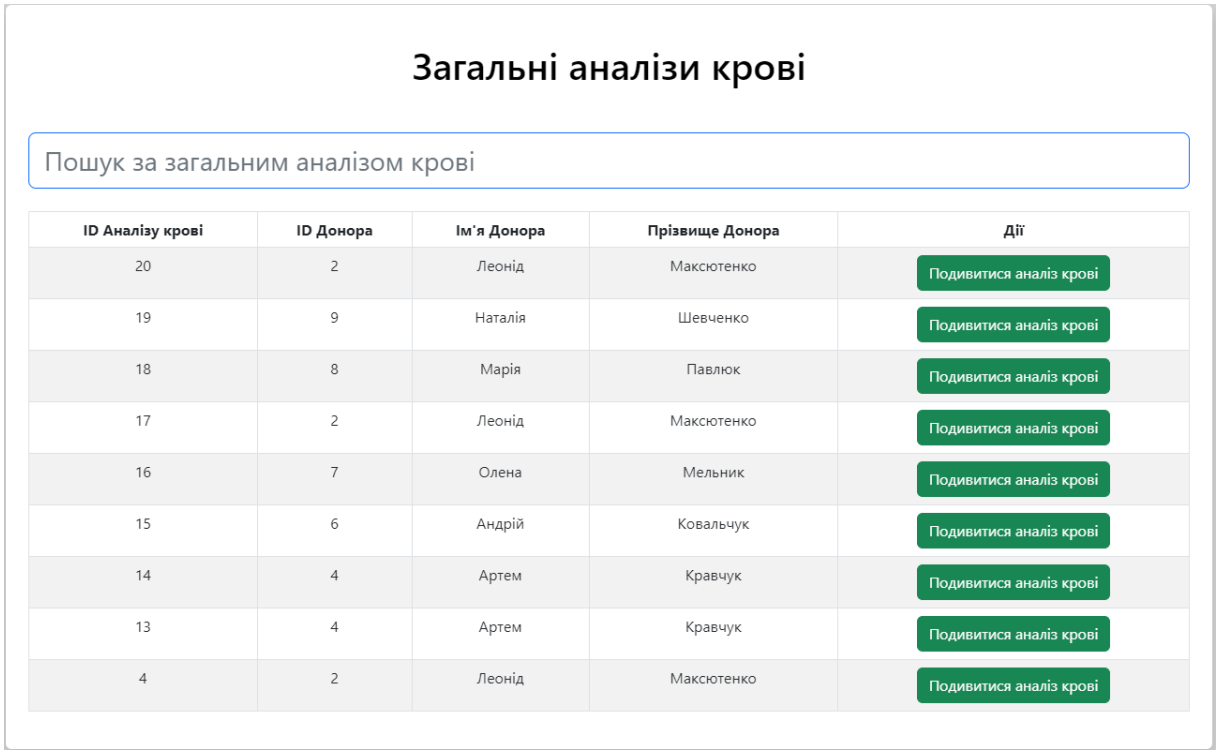


Рисунок 5.47 – Перехід до сторінки з загальними результатами крові

На сторінці з донаціями, донації, для яких ще не було створено повного результату крові, позначаються жовтою кнопкою. Коли користувач з роллю «Терапевт» завершує повний аналіз, кнопка змінюється на «Переглянути повний результат кров» На сторінці 5.48 показано сторінку «Донації», де

присутня одна донація, для якої ще не був створений повний результат крові. На сторінці 5.49 відображено форму створення повного результату крові.

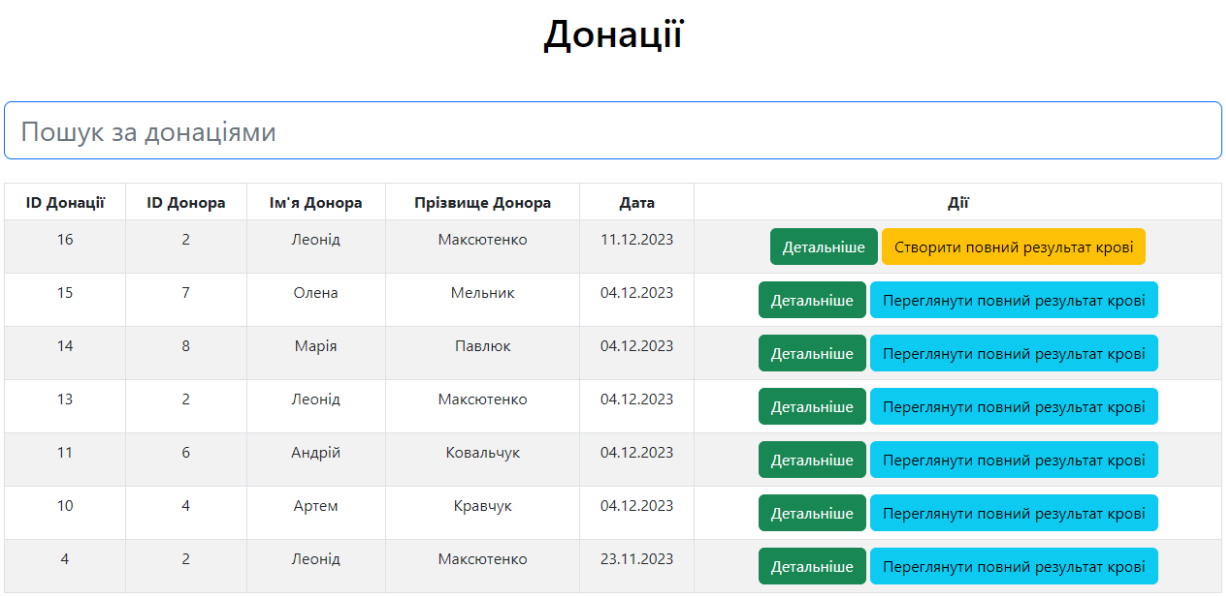


Рисунок 5.48 – Сторінка з донаціями, де є донація, до якої ще не було створено повний результат крові

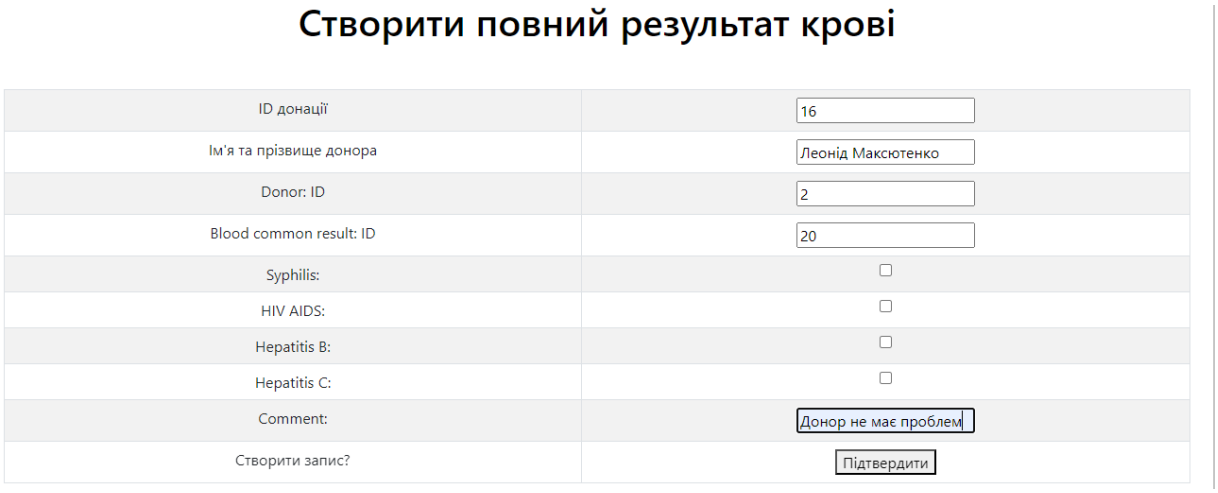


Рисунок 5.49 – Форма створення повного результату крові

На зображені 5.50 відображено зміну кнопки «Створити повний результат крові» на кнопку «Переглянути повний результат крові».

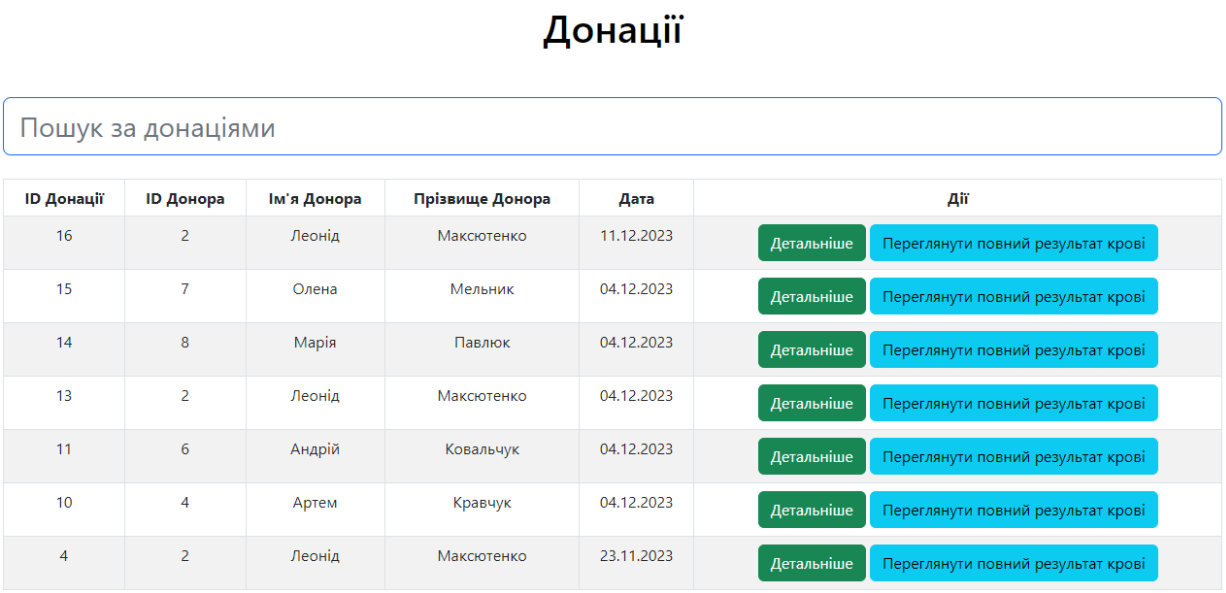


Рисунок 5.50 – Сторінка донації, після створення повного результату крові

На рисунку 5.51 зображено повний результат крові.

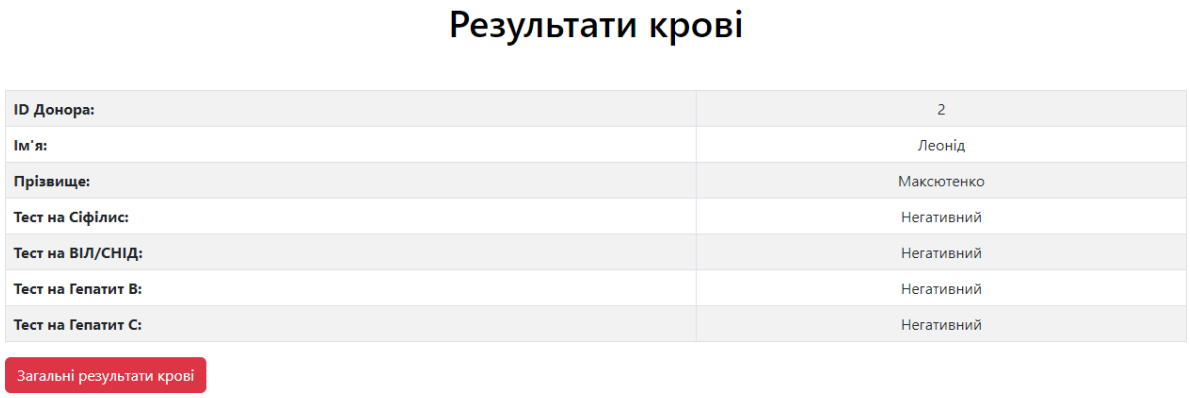


Рисунок 5.51 – Сторінка повного результату крові

5.3.2.4 Перевірка коректності записаної інформації в БД

Для перегляду записаної інформації в БД буде використовуватися pgAdmin4. На рисунках 5.52 – 5.53 зображено таблиці отримані з БД, які відображають дані, до створення нового користувача.

Data Output | Messages | Notifications

	id [PK] integer	password character varying (128)	last_login timestamp with time zone	is_superuser boolean	username character varying (150)	first_name character var	last_name character varyir	email character varying (254)	is_staff boolean	is_active boolean
1	1	pbkdf2_sha256$720000$Rk...	2023-12-11 22:51:30.351398+02	true	Maksiutenko				true	true
2	2	pbkdf2_sha256$720000$Qy...	2023-12-11 11:54:44.191548+02	false	liagic	Леонід	Максютенко	leonidmaksutenko@gmail.com	false	true
3	5	pbkdf2_sha256$600000$UP...	2023-12-03 12:44:19.031127+02	false	Kravchuk	Артем	Кравчук	Kravchuk@i.ua	false	true
4	11	pbkdf2_sha256$720000$os...	2023-12-11 19:42:42.047582+02	false	Ohrimchuk	Яна	Охрімчук		true	true
5	12	pbkdf2_sha256$720000$hw...	2023-12-11 19:41:32.199649+02	false	Butko	Бутко	Юліанна		true	true
6	14	pbkdf2_sha256$720000$BK...	2023-12-11 18:41:29.064624+02	false	savch_85	Олексій	Савченко	savch_85@i.ua	false	true
7	15	pbkdf2_sha256$720000$zk...	[null]	false	maria_pavlyuk	Марія	Павлюк	maria_pavlyuk90@gmail.com	false	true
8	16	pbkdf2_sha256$720000$gS...	[null]	false	andriy_kovalchuk	Андрій	Ковальчук	andriy_kovalchuk89@gmail.com	false	true
9	17	pbkdf2_sha256$720000$tA...	[null]	false	olena_melnyk	Олена	Мельник	olena_melnyk87@i.ua	false	true
10	18	pbkdf2_sha256$720000$S2...	[null]	false	nataliya_shevchenko	Наталія	Шевченко	nataliya_shevchenko80@gmail.com	false	true
11	19	pbkdf2_sha256$720000$yr...	2023-12-11 22:51:47.019168+02	false	olga_kovalenko70	Ольга	Коваленко	olga_kovalenko70@example.com	true	true
12	21	pbkdf2_sha256$720000$LG...	2023-12-04 11:04:53.468349+02	false	test_user	Артем	Панченко	test_user@gmail.com	false	true
13	22	pbkdf2_sha256$720000$jn...	2023-12-11 14:22:05.247781+02	false	L_user	Дмитро	Коваленко	Leonid232@i.ua	false	true

Рисунок 5.52 – Таблиця з БД до створення користувача

	id [PK] bigint	day date	time character varying (10)	time_ordered timestamp with time zone	user_id integer
1	37	2023-12-12	8:00	2023-12-11 11:56:13.20806+02	14
2	38	2023-12-12	9:00	2023-12-11 14:22:17.231506+02	22

Рисунок 5.53 – Таблиця з БД до створення запису на донацію новим користувачем

На рисунках 5.54 – 5.55 зображено створення нового користувача, та його запису на донацію.

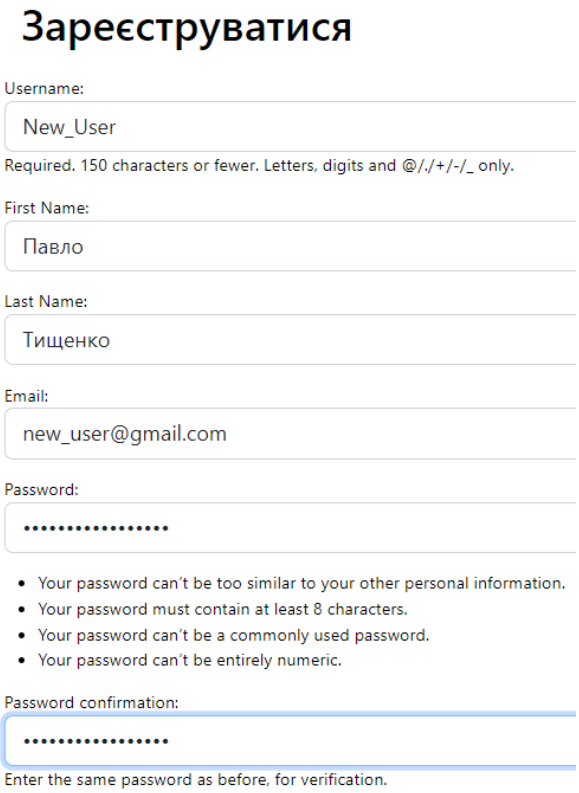


Рисунок 5.54 – Форма створення користувача

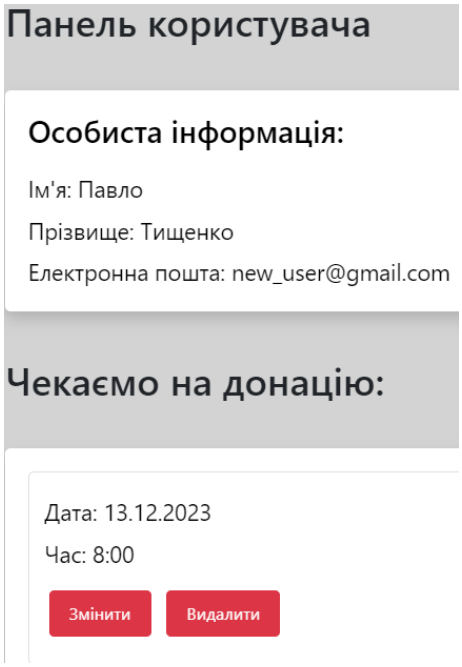


Рисунок 5.55 – створений запис на донацію

На рисунках 5.56 – 5.57 зображено таблиці з БД, в яких відображено створеного користувача, та його запису на донацію.

Data Output | Messages | Notifications

	id [PK] integer	password character varying (128)	last_login timestamp with time zone	is_superuser boolean	username character varying (150)	first_name character var	last_name character vary	email character varying (254)	is_staff boolean	is_active boolean
1	1	pbkdf2_sha256$720000$RkU...	2023-12-11 22:51:30.351398+02	true	Maksiutenko				true	true
2	2	pbkdf2_sha256$720000$Qyy...	2023-12-11 11:54:44.191548+02	false	liagic	Леонід	Максютен...	leonidmaksutenko@gmail.com	false	true
3	5	pbkdf2_sha256$600000$UP...	2023-12-03 12:44:19.031127+02	false	Kravchuk	Артем	Кравчук	Kravchuk@i.ua	false	true
4	11	pbkdf2_sha256$720000$os9...	2023-12-11 19:42:42.047582+02	false	Ohrimchuk	Яна	Охрімчук		true	true
5	12	pbkdf2_sha256$720000$hw...	2023-12-11 19:41:32.199649+02	false	Butko	Бутко	Юліанна		true	true
6	14	pbkdf2_sha256$720000$BK...	2023-12-11 18:41:29.064624+02	false	savch_85	Олексій	Савченко	savch_85@i.ua	false	true
7	15	pbkdf2_sha256$720000$zk...	[null]	false	maria_pavlyuk	Марія	Павлюк	maria_pavlyuk90@gmail.com	false	true
8	16	pbkdf2_sha256$720000$gS...	[null]	false	andriy_kovalchuk	Андрій	Ковальчук	andriy_kovalchuk89@gmail.com	false	true
9	17	pbkdf2_sha256$720000$tAc...	[null]	false	olena_melnyk	Олена	Мельник	olena_melnyk87@i.ua	false	true
10	18	pbkdf2_sha256$720000$S2...	[null]	false	nataliya_shevchenko	Наталія	Шевченко	nataliya_shevchenko80@gmail.com	false	true
11	19	pbkdf2_sha256$720000$yrV...	2023-12-11 22:51:47.019168+02	false	olga_kovalenko70	Ольга	Коваленко	olga_kovalenko70@example.com	true	true
12	21	pbkdf2_sha256$720000$LG6...	2023-12-04 11:04:53.468349+02	false	test_user	Артем	Панченко	test_user@gmail.com	false	true
13	22	pbkdf2_sha256$720000$jnR...	2023-12-11 14:22:05.247781+02	false	L_user	Дмитро	Коваленко	Leonid232@i.ua	false	true
14	23	pbkdf2_sha256$720000$0zP...	2023-12-12 09:20:41.561187+02	false	New_User	Павло	Тищенко	new_user@gmail.com	false	true

Рисунок 5.56 – Відображення таблиці, в якій додався користувач New_User

	id [PK] bigint	day date	time character varying (10)	time_ordered timestamp with time zone	user_id integer
1	37	2023-12-12	8:00	2023-12-11 11:56:13.20806+02	14
2	38	2023-12-12	9:00	2023-12-11 14:22:17.231506+02	22
3	40	2023-12-13	8:00	2023-12-12 09:24:40.522145+02	23

Рисунок 5.57 – Відображення таблиці, в якій новий користувач створив запис на донацію

5.3.2.4 Перевірка веб-інтерфейсу на іншому пристрої

Для проведення тестування на ноутбуці завантаження відбулося через динамічну URL-адресу, що автоматично перенаправило на головну сторінку веб-додатку. На рисунку 5.58 показано головну сторінку.

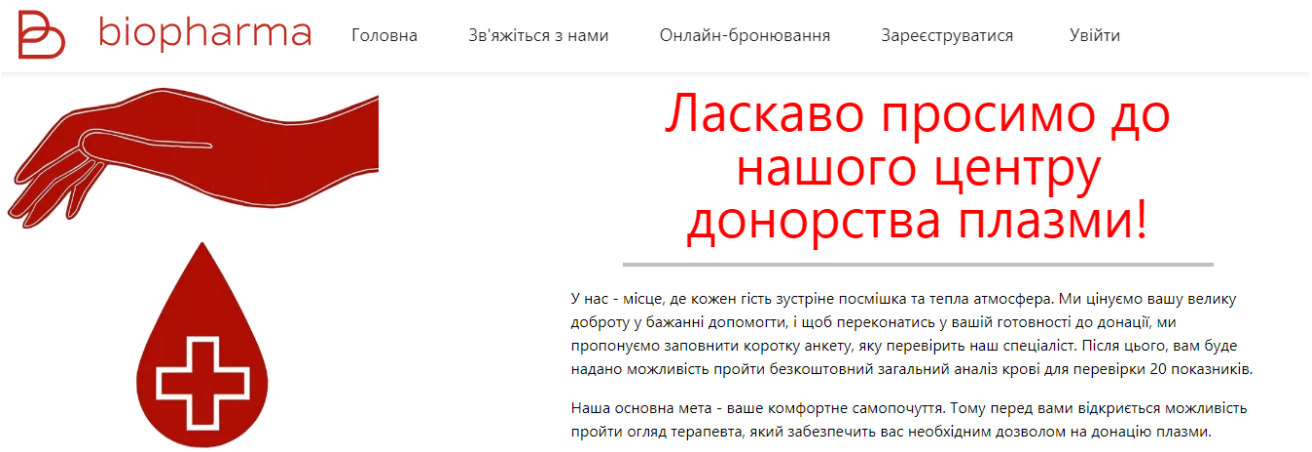


Рисунок 5.58 – Головна сторінка

Після авторизації переглядаємо панель користувача, зображено на рисунку 5.59.

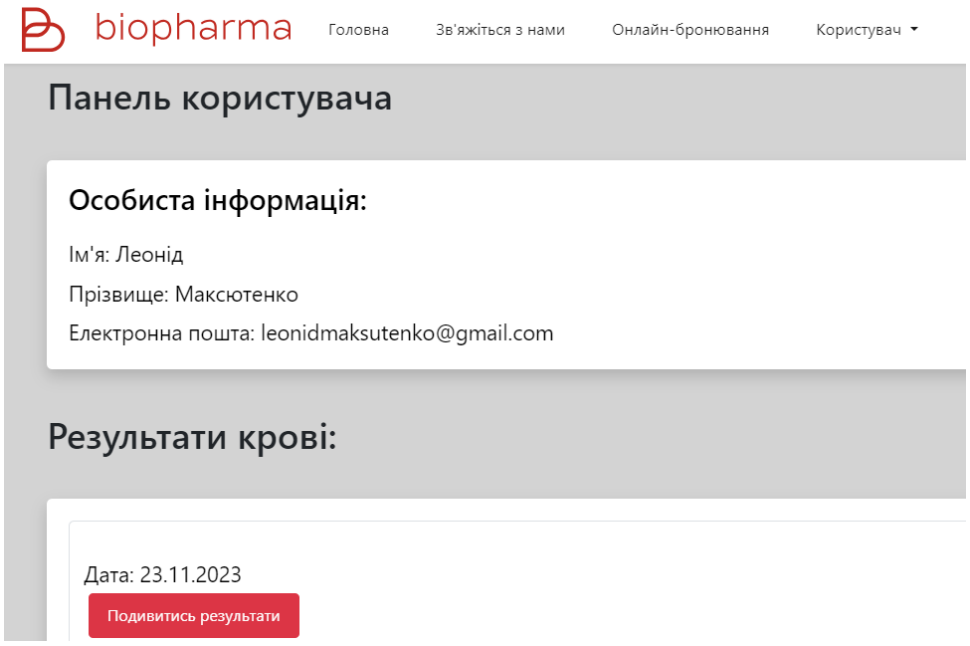


Рисунок 5.59 – Панель користувача

Обмеження для онлайн-бронювання донацій було встановлене на 14 днів з урахуванням вихідних, що є стандартною практикою для донації плазми. Форму для створення запису на донацію показано на рисунку 5.60.

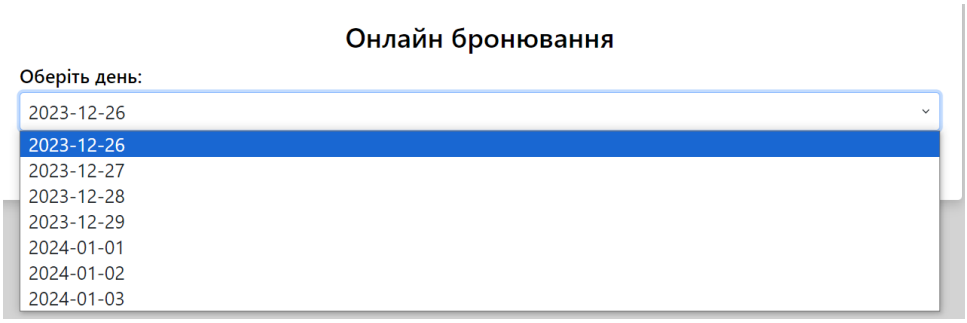


Рисунок 5.60 – Вибір дати на донацію

Після створення запису на донацію, в особистому кабінеті користувача відображається інформація про створений запис, що свідчить про можливість створення записів на сервері з іншого пристрою. Це показано на рисунку 5.61.

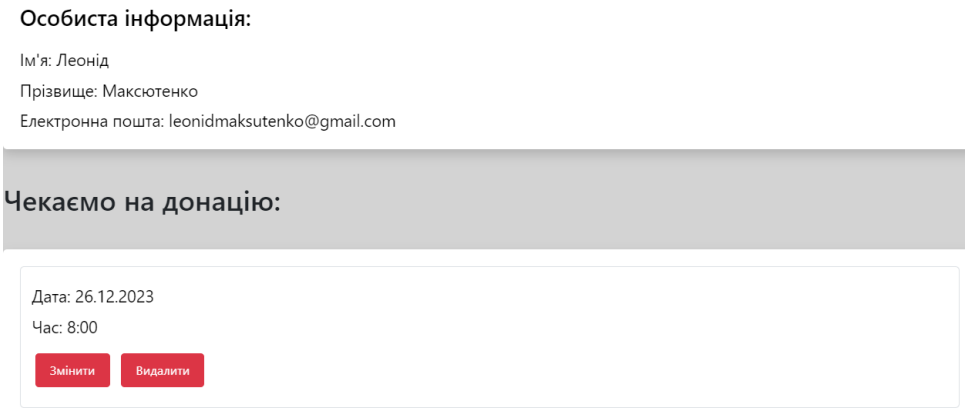


Рисунок 5.61 – Створений запис на донацію в панелі користувача

На рисунку 5.62 зображено вдалий вихід з особистого акаунту.

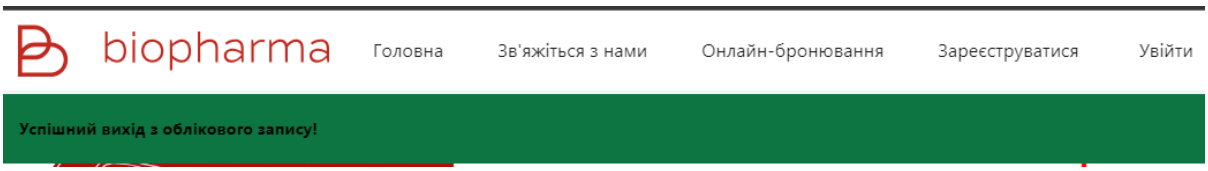


Рисунок 5.62 – Вихід з особистого аккаунту

5.3.3 Аналіз результатів експерименту

За результатами проведеного тестування виявлено, що отримані результати відповідають вимогам до програмного забезпечення. Тестування різних груп у веб-застосунку підтвердило правильний розподіл, який відповідає їхнім можливостям.

Клієнти, як передбачалося, можуть створювати, змінювати та видаляти записи про донації, а також переглядати свої результати аналізів крові.

Реєстратура має можливість переглядати та видаляти записи про донації, створювати, переглядати та редагувати дані карток донорів, а також проводити аналіз крові.

Відділ прийому донорів може створювати донації, переглядати їх, переглядати картки донорів та результати загальних аналізів крові.

Терапевти мають можливість переглядати картки донорів, повні та загальні аналізи крові, донації, а також створювати повні аналізи крові.

Увесь функціонал працює згідно з очікуваннями та плануванням. Записи в БД відповідають тому, що відображається у веб-інтерфейсі. Також підтверджено коректну роботу функціоналу під час використання іншого пристрою.

5.4 Висновки

В результаті проведеного експерименту було підтверджено, що розроблене ПЗ для донорського центру у формі веб-застосунку виявилося адекватним. Це ПЗ сприяє прискоренню обслуговування донорів та швидкому доступу до необхідної інформації завдяки використанню БД. Проведене тестування повністю підтверджує правильність та ефективність розробленого веб-застосунку.

ВИСНОВКИ

Кваліфікаційна робота є завершеною науковою роботою в якій було вирішена науково-практична задача, а саме обґрунтування та розробка комп’ютерної системи донорського центру «Biopharma» із детальною проробкою інтерфейсу користувача та відповідної бази даних.

Основні висновки і результати роботи полягають у наступному:

1. Проведено аналіз проблеми функціонування донорського центру, виявлено його недоліки та визначено вимоги до подальшої розробки.
2. Проведено огляд актуальних тенденцій у сфері ПЗ, включаючи аналіз та вибір мов програмування, аналіз веб-фреймворків, обґрунтування використання реляційних СКБД та визначення вимог до комп'ютерної системи та програмного забезпечення.
3. Розроблено структурну та функціональну схему комп'ютерної системи донорського центру, що чітко відображає всі компоненти центру та їх взаємозв'язки.
4. Описано веб-застосунок «DonorPlus», розроблений на базі фреймворку Django та використовуючи СКБД PostgreSQL, який забезпечує ефективне управління інформацією про донорів та контроль доступу, включаючи функції реєстрації, внесення та оновлення даних.
5. Проведено тестування веб-додатку для оцінки його ефективності, коректності роботи інтерфейсу та бази даних. Результати тестування підтвердили адекватність та ефективність розробленого веб-застосунку, що сприяє полегшенню доступу до інформації та прискоренню обслуговування донорів. донорства.

Усі ці кроки в процесі дослідження та розробки вказують на успішну реалізацію та практичне застосування розробленої комп’ютерної системи для оптимізації роботи донорського центру, що впливає на покращення обслуговування донорів та сприяє зручнішому доступу до інформації.

ПЕРЕЛІК ДЖЕРЕЛ І ПОСИЛАНЬ

1. Чому Україні треба відмовитись від стихійного донорства. ДонорUA. URL: https://www.donor.ua/pages/2634 (дата звернення: 26.09.2023)

2. Донорство компонентів крові в Україні | Biopharma. Biopharma Plasmacenters. URL: https://biopharmaplasma.ua/ (дата звернення: 26.09.2023)

3. The Different Types of Databases - Overview with Examples. Prisma's Data Guide. URL: http://surl.li/ofixr (дата звернення: 09.12.2023)

4. Aijaz M. An analysis of the use of different programming languages in web development. LinkedIn. URL: http://surl.li/ofizo (дата звернення: 01.11.2023)

5. Sharma A. Express JS Tutorial. Simplilearn. URL: http://surl.li/ofjbg (дата звернення: 13.12.2023)

6. What is JavaScript? - Learn web development. MDN Web Docs. URL: http://surl.li/ofjck (дата звернення: 13.12.2023)

7. AdonisJS Framework Tutorial. Educative. URL: http://surl.li/ofjcz (дата звернення: 13.12.2023)

8. What is Django?. IBM. URL: https://www.ibm.com/topics/django (дата звернення: 13.12.2023)

9. What is Flask?. Learn Python Programming. URL: https://pythonbasics.org/what-is-flask-python/ (дата звернення: 13.12.2023)

10. Introduction to Flask Framework. Analytics Vidhya. URL: https://www.analyticsvidhya.com/blog/2021/10/flask-python/ (дата звернення: 13.12.2023)

11. What is MySQL?. Dev MySQL. URL: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html (дата звернення: 12.12.2023)

12. PostgreSQL: About. PostgreSQL. URL: https://www.postgresql.org/about/ (дата звернення: 11.12.2023);

13. What is PostgreSQL?. Guru99. URL: http://surl.li/ofjha (дата звернення: 12.12.2023)

14. PostgreSQL vs MySQL. Amazon Web Services. URL: http://surl.li/ofjha (дата звернення: 12.12.2023)

15. Cross A. R. Blood Donor. Android Apps on Google Play. URL: http://surl.li/ofjij (дата звернення: 13.12.2023)

16. Home. NHS Blood Donation. URL: https://www.blood.co.uk/ (дата звернення: 13.12.2023)

17. Гематологічний аналізатор Sysmex XP-300. servislab. URL: http://surl.li/ofjjs (дата звернення: 12.12.2023)

18. Сервер ARTLINE Business R19v26. rozetka. URL: https://rozetka.com.ua/artline-r19v26/p396394140/ (дата звернення: 12.12.2023)

19. Сервер ARTLINE Business R36 (R36v22). rozetka. URL: https://rozetka.com.ua/artline-r36v22/p396392490/ (дата звернення: 12.12.2023)

Додаток А

Текст програми «DonorPlus»

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
«DonorPlus»

Текст програми
804.02070743.23016-01 12 01
Листів 32

2023

АНОТАЦІЯ

Цей текст програми на мові Python розроблено для веб-додатку, що використовується донорським центром «Biopharma». Програмне забезпечення має на меті забезпечення доступу до інформації про донорів, результатів аналізів крові та донацій плазми. Воно надає функціональні можливості для управління даними, пов'язаними із здійсненням донацій, а також забезпечує можливість перегляду та редагування інформації про донорів. Розробка цієї програми спрямована на оптимізацію та полегшення процесів управління даними, пов'язаними із донорською діяльністю, що дозволяє ефективніше використовувати ці дані для подальшого аналізу та вжиття відповідних заходів.

ЗМІСТ

1 ПРОЕКТ DONORPLUS

1.1 Файл urls.py

```
#Імпортує необхідні модулі для налаштування маршрутів Django
from django.contrib import admin
from django.urls import path, include

#Створює список urlpatterns, який містить шляхи веб-додатку
urlpatterns = [
    #Шлях для адміністративної панелі Django, доступної за адресою /admin/
    path('admin/', admin.site.urls),
    # Шлях, який вказує на включення URL адрес з додатку "queue_app".
    path('', include("queue_app.urls")),
    # Шлях для обробки URL адрес з додатку "members", доступний за адресою /user
    path('user', include("members.urls")),
]
```

2 ДОДАТОК MEMBERS ПРОЕКТУ DONORPLUS

2.1 Файл apps.py

```
#Імпортує клас AppConfig з модуля django.apps для конфігурації додатку
from django.apps import AppConfig
#Створює клас MembersConfig, який успадковує властивості від AppConfig
class MembersConfig(AppConfig):
    #Встановлює поле default_auto_field для використання в базі даних.
    default_auto_field = 'django.db.models.BigAutoField'
    #Встановлює ім'я додатку як 'members'.
    name = 'members'
```

2.2 Файл forms.py

```
#Імпортує необхідні модулі та класи з Django для створення форми реєстрації користувача
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from django import forms
from django.db import models
class RegisterUserForm(UserCreationForm):
    #Додає додаткові поля до форми для введення email, ім'я та прізвища користувача
    email = forms.EmailField(label='Email', widget=forms.EmailInput(attrs={'class':'form-control form-control-lg', 'placeholder': 'example@gmail.com'}))
    first_name = forms.CharField(label='First Name', max_length=50, widget=forms.TextInput(attrs={'class':'form-control form-control-lg'}))
    last_name = forms.CharField(label='Last Name', max_length=50, widget=forms.TextInput(attrs={'class':'form-control form-control-lg'}))

    class Meta:
        #Вказує модель, з якою працює форма, та перелік полів для відображення у формі
        model = User
        fields = ('username', 'first_name', 'last_name', 'email', 'password1', 'password2')
```

```
    def __init__(self, *args, **kwargs):
        super(RegisterUserForm, self).__init__(*args, **kwargs)
        #Налаштовує відображення класів для полів у формі
        self.fields['username'].widget.attrs['class'] = 'form-control form-control-lg'
        self.fields['password1'].widget.attrs['class'] = 'form-control form-control-lg'
        self.fields['password2'].widget.attrs['class'] = 'form-control form-control-lg'
```

2.3 Файл urls.py

#Імпортує необхідні функції та класи з модуля django.urls для визначення шляхів URL-адрес

```
from django.urls import path
from . import views

#Список urlpatterns, який містить шляхи URL-адрес для відповідних views
urlpatterns = [
    #URL-адреса для входу користувача
    path('login_user', views.login_user, name='login'),

    #URL-адреса для виходу користувача
    path('logout_user', views.logout_user, name='logout'),

    #URL-адреса для реєстрації нового користувача
    path('register_user', views.register_user, name='register'),
]
```

2.4 Файл views.py

#Імпортує необхідні функції та класи з Django для обробки запитів та автентифікації користувачів

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from .forms import RegisterUserForm
from django.views.decorators.csrf import csrf_exempt
```

```
@csrf_exempt
def login_user(request):
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            #Перенаправлення на сторінку успішного входу
            return redirect('index')
        else:
            #Повернення повідомлення про невірний логін або пароль
            messages.error(request, "Виникла помилка!")
            return redirect('login')
    else:
        return render(request, 'authenticate/login.html', {})

@csrf_exempt
def logout_user(request):#Вихід з системи
    logout(request)
    messages.success(request, "Успішний вихід з облікового запису!")
    return redirect('index')

@csrf_exempt
def register_user(request): #Реєстрація користувача
    if request.method == "POST":
        form = RegisterUserForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data['username']
            password = form.cleaned_data['password1']
            user = authenticate(username=username, password=password)
            login(request, user)
```

```
            messages.success(request, "Реєстрація завершена!")
            return redirect('index')
    else:
        form = RegisterUserForm()
    return render(request, 'authenticate/register_user.html', {
        'form': form,
    })
```

3 ДОДАТОК QUEUE_APP ПРОЕКТУ DONORPLUS

3.1 Файл apps.py

```
from django.apps import AppConfig
class QueueAppConfig(AppConfig):
    #Встановлює тип поля для автоматичного збільшення у моделях за замовчуванням
    default_auto_field = 'django.db.models.BigAutoField'
    # Вказує назву додатку
    name = 'queue_app'
```

3.2 Файл forms.py

```
from django import forms
from .models import *

#Форма для додавання донорів
class DonorForm(forms.ModelForm):
    class Meta:
        model = Donor
        fields = '__all__'
        labels = {
            'birthday_date': 'Дата народження',
            'gender': 'Стать',
            'weight': 'Вага',
            'height': 'Зріст',
            'phone_number': 'Номер телефону',
            'passport_series': 'Серія паспорта',
            'passport_number': 'Номер паспорта',
            'taxpayer_id': 'Ідентифікаційний номер',
            'blood_group': 'Група крові',
            'rhesus_factor': 'Резус-фактор',
            'address': 'Адреса',
        }

#Форма для додавання інформації про донацію
class DonationForm(forms.ModelForm):
```

```
    class Meta:
        model = Donation
        fields = '__all__'

#Форма для введення результатів аналізу крові
class FullBloodForm(forms.ModelForm):
    class Meta:
        model = BloodResult
        fields = '__all__'

#Форма для редагування даних про донора
class EditDonorForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ['first_name', 'last_name']
```

3.3 Файл models.py

```
from django.db import models
from datetime import datetime, date
from django.contrib.auth.models import User


#Вибір часу для запису на прийом
TIME_CHOICES = (
    ("8:00", "8:00"),
    ("8:15", "8:15"),
    ("8:30", "8:30"),
    ("8:45", "8:45"),
    ("9:00", "9:00"),
    ("9:15", "9:15"),
    ("9:30", "9:30"),
    ("9:45", "9:45"),
    ("10:00", "10:00"),
    ("10:15", "10:15"),
```

```
        ("10:30", "10:30"),
        ("10:45", "10:45"),
        ("11:00", "11:00"),
        ("11:15", "11:15"),
        ("11:30", "11:30"),
        ("11:45", "11:45"),
        ("12:00", "12:00"),
        ("12:15", "12:15"),
        ("12:30", "12:30"),
        ("12:45", "12:45"),
        ("13:00", "13:00"),
        ("13:15", "13:15"),
        ("13:30", "13:30"),
        ("13:45", "13:45"),
        ("14:00", "14:00"),
        ("14:15", "14:15"),
        ("14:30", "14:30"),
        ("14:45", "14:45"),
        ("15:00", "15:00"),
        ("15:15", "15:15"),
        ("15:30", "15:30"),
        ("15:45", "15:45"),
        ("16:00", "16:00"),
    )

    #Модель для запису на прийом
    class Appointment(models.Model):
        user = models.ForeignKey(User, on_delete=models.CASCADE, null=True, blank=True)
        day = models.DateField(default=datetime.now)
        time = models.CharField(max_length=10, choices=TIME_CHOICES)
        time_ordered = models.DateTimeField(default=datetime.now, blank=True)
        def __str__(self):
            return f"{self.user.username} | day: {self.day} | time: {self.time}"
```

```
#Модель для донора
class Donor(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    birthday_date = models.DateField(null=False, default=datetime.now)
    gender = models.BooleanField(null=False)
    weight = models.IntegerField(null=False)
    height = models.IntegerField(null=False)
    phone_number = models.CharField(max_length=12, null=False, blank=False)
    passport_series = models.CharField(max_length=2, null=True, blank=True)
    passport_number = models.CharField(max_length=15)
    taxpayer_id = models.CharField(max_length=15)
    blood_group = models.IntegerField(null=False)
    rhesus_factor = models.BooleanField(null=False)
    address = models.CharField(max_length=255)

#Модель для працівника
class Worker(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    patronymic = models.CharField(max_length=50, null=False)
    birthday_date = models.DateField(null=False, default=datetime.now)
    taxpayer_id = models.CharField(max_length=15)
    passport_series = models.CharField(max_length=2, null=True)
    passport_number = models.CharField(max_length=15)
#Модель для донації
class Donation(models.Model):
    worker = models.ForeignKey(Worker, on_delete=models.CASCADE)
    donor = models.ForeignKey(Donor, on_delete=models.CASCADE)
    device_number = models.IntegerField(default=0)
    donation_date = models.DateField(default=datetime.now)
    procedure_time = models.IntegerField()
    successful_procedure = models.BooleanField(default=True)
    comment = models.CharField(max_length=255, null=True)
```

```
        #Модель для загальних результатів аналізу крові
        class BloodCommonResult(models.Model):
            donor = models.ForeignKey(Donor, on_delete=models.CASCADE)
            HGB = models.FloatField()
            RBC = models.FloatField()
            WBC = models.FloatField()
            PLT = models.FloatField()
            HCT = models.FloatField()
            MPV = models.FloatField()
            MCV = models.FloatField()
            MCH = models.FloatField()
            MCHC = models.FloatField()
            LYM_percent = models.FloatField()
            MON_percent = models.FloatField()
            NEUT_percent = models.FloatField()
            LYM_number = models.FloatField()
            MON_number = models.FloatField()
            NEUT_number = models.FloatField()
            RDW_SD = models.FloatField()
            RDW_CV = models.FloatField()
            RDW = models.FloatField()
            P_LCR = models.FloatField()
            PCT = models.FloatField()
        #Модель для повних результатів аналізу крові
        class BloodResult(models.Model):
            donation = models.OneToOneField(Donation,
on_delete=models.CASCADE)#DONATION ID
            donor = models.ForeignKey(Donor, on_delete=models.CASCADE)
            blood_common_result = models.OneToOneField(BloodCommonResult,
on_delete=models.CASCADE)
            syphilis = models.BooleanField(default=False)
            HIV_AIDS = models.BooleanField(default=False)
            hepatitis_B = models.BooleanField(default=False)
            hepatitis_C = models.BooleanField(default=False)
```

```
    comment = models.CharField(max_length=255, null=True)
```

3.4 Файл urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    #Головна сторінка
    path('', views.index, name='index'),

    #Сторінка бронювання, де обирається дата
    path('booking', views.booking, name='booking'),

    #Обробник форми бронювання, де обирається час
    path('booking-submit', views.bookingSubmit, name='bookingSubmit'),

    #Панель користувача
    path('user-panel', views.userPanel, name='userPanel'),

    #Оновлення бронювання користувача за ID бронювання
    path('user-update/<int:id>', views.userUpdate, name='userUpdate'),

    #Відправка оновленої інформації бронювання на сервер за його ID
    path('user-update-submit/<int:id>', views.userUpdateSubmit,
name='userUpdateSubmit'),

    #Сторінка з усіма записами на прийом
    path('appointments', views.appointments, name='appointments'),

    #Сторінка контактів
    path('contact_us', views.contact_us, name='contactUs'),
    #Перегляд інформації про донора за його ID
    path('view_donor/<int:donor_id>/', views.view_donor, name='view_donor'),
    #Список донорів
```

```
        path('donors', views.list_donor, name='donors'),
        #Створення донора за ID користувача
        path('create_donor/<int:user_id>/', views.create_donor, name='create_donor_with_id'),
        # Видалення запису на прийом за його ID
        path('delete_appointment/<int:appointment_id>/', views.delete_appointment,
name='delete_appointment'),
        #Заповнення результатів аналізу крові для донора за його ID
        path('fill_blood_results/<int:donor_id>/', views.fill_blood_results,
name='conduct_blood_analysis'),
        #Перегляд загальних результатів аналізу крові
        path('view_common_blood_analysis', views.view_common_blood_analysis,
name='view_common_blood_analysis'),
        path('view_common_blood_analysis/<int:donor_id>/',
views.view_common_blood_analysis,  name='view_common_blood_analysis_with_id'),

        #Перегляд даних загальних результатів аналізу крові за його ID
        path('common_result/<int:blood_id>', views.common_blood_result,
name='common_result'),

        #Редагування інформації про донора за його ID
        path('edit_donor/<int:donor_id>', views.edit_donor, name='edit_donor'),

        #Сторінка з усіма донаціями
        path('donations', views.donations, name='donations'),
        path('donations/<int:donor_id>', views.donations, name='donations_with_id'),

        # Створення донації за ID донора
        path('create_donation/<int:donor_id>', views.create_donation,
name='create_donation'),

        # Перегляд донацій за його ID
        path('view_donation/<int:donation_id>', views.view_donation, name='view_donation'),

        #Створення повних результатів аналізу крові для донора ID донації
```

```
    path('create_full_blood/<int:donation_id>', views.create_full_blood_result, name='create_full_blood'),

    #Сторінка з усіма результатами аналізу крові
    path('blood_result', views.full_blood_results, name='blood_result'),
    path('blood_result/<int:donor_id>', views.full_blood_results, name='blood_result_with_id'),

    #Перегляд повних результатів аналізу крові за його ID
    path('view_full_blood_result/<int:full_blood_result_id>', views.view_full_blood_result, name='view_full_blood_result')
]
```

3.5 Файл views.py

```
from django.shortcuts import render, redirect, get_object_or_404, reverse
from datetime import datetime, timedelta
from .models import *
from .forms import *
from django.contrib import messages
from django.conf.urls.static import static
import random
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt

import time
# Глобальна змінна times, що містить доступні часи для бронювання
global times
times = [
    "8:00",
    "8:15",
    "8:30",
    "8:45",
    "9:00",
    "9:15",
    "9:30",
```

```
    "9:45",
    "10:00",
    "10:15",
    "10:30",
    "10:45",
    "11:00",
    "11:15",
    "11:30",
    "11:45",
    "12:00",
    "12:15",
    "12:30",
    "12:45",
    "13:00",
    "13:15",
    "13:30",
    "13:45",
    "14:00",
    "14:15",
    "14:30",
    "14:45",
    "15:00",
    "15:15",
    "15:30",
    "15:45",
    "16:00",
]
```

#Головна сторінка/файл index.html

def index(request):#request об'єкт, що представляє HTTP-запит, який надсилає користувач.

return render(request, "index.html")#Відповідь сервера зі згенерованою HTML-сторінкою для відображення клієнту.

```
    #Сторінка про місце розташування та номер телефону/файл contact_us.html
    def contact_us(request):
        return render(request, 'contact_us.html')


    #Відображає інформацію користувача, результати крові та запис на донацію/файл userPanel.html / Має функціонал до зміни запису донації, видаленю та перегляду результатів крові
    @csrf_exempt
    def userPanel(request):
        user = request.user
        delta_time = datetime.today() + timedelta(days=21)
        max_date = delta_time.strftime('%Y-%m-%d')
        try:
            donor = Donor.objects.get(user_id=user.pk)
            blood_results = BloodResult.objects.filter(donor= donor)
            appointments = Appointment.objects.filter(
                user_id=user.id,
                day__range=[datetime.today().strftime('%Y-%m-%d'), max_date]
            ).order_by('day', 'time')
            return render(request, 'userPanel.html', {
                'user': user,
                'appointments': appointments,
                'blood_results': blood_results
            })
        except Donor.DoesNotExist:
            appointments = Appointment.objects.filter(
                user_id=user.id,
                day__range=[datetime.today().strftime('%Y-%m-%d'), max_date]
            ).order_by('day', 'time')
            return render(request, 'userPanel.html', {
                'user': user,
                'appointments': appointments,


            })
```

```
#Сторінка для запису на донацію, обирається дата/booking.html
@csrf_exempt
def booking(request):
    if not request.user.is_authenticated:
        return redirect('login')

    weekdays = validWeekday(22)
    validateWeekdays = isWeekdayValid(weekdays)

    available_weekdays = additional_check(request.user.id, validateWeekdays)
    if Appointment.objects.filter(user=request.user).count() >= 1:
        messages.error(request, 'Ви не можете зробити більше 1 запису на донацію.')
        return redirect('userPanel')

    if request.method == 'POST':
        day = request.POST.get('day')
        request.session['day'] = day


        if not available_weekdays:
            messages.error(request, 'Немає доступних днів на донацію.')
            return redirect('index')

        return redirect('bookingSubmit')

    return render(request, 'booking.html', {
        'weekdays': weekdays,
        'validateWeekdays': available_weekdays,
    })
```

#Сторінка підтвердження запису на донацію з обиранням часу /файл bookingSubmit.html

```
@csrf_exempt
def bookingSubmit(request):
    user = request.user
    today = datetime.now()
    minDate = today.date()
    maxDate = today + timedelta(days=21)

    day = request.session.get('day')

    hour = checkTime(times, day)
    if request.method == 'POST':
        time = request.POST.get("time")
        date = dayToWeekday(day)

        if day and isinstance(day, str):  # Перевірка, що день це строка
            day = datetime.strptime(day, '%Y-%m-%d').date()

        if day <= maxDate.date() and day >= minDate:
            if date != 'Saturday' and date != 'Sunday':
                if Appointment.objects.filter(day=day).count() < 11:
                    if Appointment.objects.filter(day=day, time=time).count() < 1:
                        AppointmentForm = Appointment.objects.get_or_create(
                            user=user,
                            day=day,
                            time=time,
                        )
                        messages.success(request, " Запис збережено!")
                        return redirect('index')
                    else:
                        messages.error(request, "Обраний час було зарезервовано раніше!")
                else:
                    messages.error(request, "Обраний день зайнятий!")
            else:
                messages.error(request, "Обрана дата невірна")
```

```
        return render(request, 'bookingSubmit.html', {
            'times': hour,
        })


    #Для персоналу, перегляд записів на донацію/ appointments.html / view_donor_card, delete_appointments and create_donor_card
    @csrf_exempt
    def appointments(request):
        today = datetime.today()
        min_date = today.strftime('%Y-%m-%d')
        delta_time = today + timedelta(days=21)
        max_date = delta_time.strftime('%Y-%m-%d')

        appointments = Appointment.objects.filter(day__range=[min_date, max_date]).order_by('day', 'time')

        return render(request, 'appointments.html', {
            'items': appointments
        })

    #Видалення запису на донацію/ delete_appointment.html / userPanel or appointments
    @csrf_exempt
    def delete_appointment(request, appointment_id):
        try:
            appointment = Appointment.objects.get(pk=appointment_id)
            user = request.user
            if user == appointment.user:
                appointment.delete()
                messages.success(request, 'Запис видалено успішно.')
            elif user.is_staff:
                appointment.delete()
                messages.success(request, 'Запис видалено успішно.')
```

```
                return render(request, 'appointments.html')
            else:
                messages.error(request, 'Запис не відповідає користувачу')
                return redirect('userPanel')
        except Appointment.DoesNotExist:
            messages.error(request, 'Запис не існує або вже був видалений.')
        return redirect('userPanel')

    #Оновлення запису на донацію/ userUpdate.html / userUpdateSubmit
    @csrf_exempt
    def userUpdate(request, id):
        appointment = Appointment.objects.get(pk=id)
        userdatepicked = appointment.day
        today = datetime.today()
        #обмежує, якщо донор бажає змінити час на донацію за 1 годину до донації
        delta = (userdatepicked).strftime('%Y-%m-%d') >= (today + timedelta(hours=1)).strftime('%Y-%m-%d')
        #Перевірка доступних днів
        weekdays = validWeekday(22)
        validateWeekdays = isWeekdayValid(weekdays)
        available_weekdays = additional_check(request.user.id, validateWeekdays)

        if request.method == 'POST':
            day = request.POST.get('day')
            request.session['day'] = day
            return redirect('userUpdateSubmit', id=id)

        return render(request, 'userUpdate.html', {
            'weekdays': weekdays,
            'validateWeekdays': available_weekdays,
            'delta': delta,
            'id': id,
        })
```

```
#Вибір часу для донації, для оновлення запису/ userUpdateSubmit.html / userUpdate if params is not correct or index
@csrf_exempt
def userUpdateSubmit(request, id):
    user = request.user
    today = datetime.now()
    minDate = today.strftime('%Y-%m-%d')
    deltatime = today + timedelta(days=21)
    strdeltatime = deltatime.strftime('%Y-%m-%d')
    maxDate = strdeltatime

    day = request.session.get('day')

    #Показує лише час дня, який не був обраний раніше, та час, який редагується:
    hour = checkEditTime(times, day, id)
    appointment = Appointment.objects.get(pk=id)
    userSelectedTime = appointment.time
    if request.method == 'POST':
        time = request.POST.get("time")
        date = dayToWeekday(day)
        if day <= maxDate and day >= minDate:
            if date != 'Saturday' or date != 'Sunday':
                if Appointment.objects.filter(day=day).count() < 11:
                    if Appointment.objects.filter(day=day, time=time).count() < 1 or userSelectedTime == time:
                        AppointmentForm = Appointment.objects.filter(pk=id).update(
                            user=user,
                            day=day,
                            time=time,
                        )
                        messages.success(request, "Запис на донацію відредаговано!")
                        return redirect('index')
                    else:
```

```
                    messages.success(request, "Обраний час вже був зарезервований раніше!")
                else:
                    messages.success(request, "Обраний день заповнений!")
            else:
                messages.success(request, "Обраний день некоректний")
        else:
            messages.success(request, "Обрана дата не знаходиться в правильному періоді!")

        return redirect('userPanel')

    return render(request, 'userUpdateSubmit.html', {
        'times': hour,
        'id': id,
    })


#Створення карти донора, якщо не існує самої карти для данного користувача/ create_donor.html / list_donor
@csrf_exempt
def create_donor(request, user_id):
    user_create_donor = get_object_or_404(User, pk=user_id)
    if request.method == 'POST':
        form = DonorForm(request.POST)
        if form.is_valid():
            donor = form.save(commit=False)
            donor.save()
            return redirect('donors')
        else:
            messages.error(request, "Форма не правильно заповненна!")

    else:
        form = DonorForm()
```

```
        return render(request, 'create_donor.html', {'form': form, 'user_create_donor': user_create_donor})

    #Перегляд карти донора / view_donor.html
    @csrf_exempt
    def view_donor(request, donor_id):
        try:
            donor = Donor.objects.get(user_id=donor_id)
            return render(request, 'view_donor.html',
                          {'donor': donor})
        except Donor.DoesNotExist:
            messages.error(request, "Карти донора не існує!")
            return redirect('create_donor_with_id', user_id=donor_id)

    #Зміна інформації в картці донора/ edit_donor.html
    @csrf_exempt
    def edit_donor(request, donor_id):
        donor = get_object_or_404(Donor, pk=donor_id)
        user_create_donor = donor.user

        if request.method == 'POST':
            donor_form = DonorForm(request.POST, instance=donor)
            user_form = EditDonorForm(request.POST, instance=user_create_donor)

            if user_form.is_valid() and donor_form.is_valid():
                user = user_form.save(commit=False)
                user.save()
                donor_form.save()
                return redirect('view_donor', donor_id=donor.user.id)
            else:
                messages.error(request, "Форма не правильно заповненна!")
        else:
            donor_form = DonorForm(instance=donor)
            user_form = EditDonorForm(instance=user_create_donor)
```

```
    return render(request, 'edit_donor.html', {'donor_form': donor_form, 'user_form': user_form, 'user_create_donor': user_create_donor})


#Список донорів / donors.html
@csrf_exempt
def list_donor(request):
    donors = Donor.objects.filter()

    return render(request, 'donors.html', {
        'donors': donors
    })


#Перегляд повних результатів крові у вигляді списку, або якщо вказати id донора, тільки його результати/ blood_common.html / common_blood_result
@csrf_exempt
def view_common_blood_analysis(request, donor_id=None):
    blood_common = BloodCommonResult.objects.all().order_by('-id')
    if donor_id is not None:
        blood_common = blood_common.filter(donor_id=donor_id).order_by('-id')
    return render(request, 'blood_common.html', {'blood_commons': blood_common})

#Перегляд результатів загального аналізу крові / common_result.html / ENDPOINT
@csrf_exempt
def common_blood_result(request, blood_id):
    user = request.user
    blood = get_object_or_404(BloodCommonResult, pk=blood_id)
    if user.pk == blood.donor.user.pk or user.is_staff:
        return render(request, 'common_result.html', {'bloods': blood})
    else:
        messages.error(request, "Немає доступу до запису!")
        return redirect("index")
```

```
#simulation of taking blood results from SYSMEX XP-300
@csrf_exempt
def fill_blood_results(request, donor_id):
    donor = get_object_or_404(Donor, pk=donor_id)
    ranges = {
        'HGB': (130.0, 200.0),
        'RBC': (4.0, 6.0),
        'WBC': (4.0, 12.0),
        'PLT': (100.0, 390.0),
        'HCT': (0.38, 0.55),
        'MPV': (7.4, 10.4),
        'MCV': (80.0, 95.0),
        'MCH': (25.0, 32.0),
        'MCHC': (300.0, 380.0),
        'LYM_percent': (0.19, 0.37),
        'MON_percent': (0.03, 0.11),
        'NEUT_percent': (0.34, 0.71),
        'LYM_number': (1.2, 3.0),
        'MON_number': (0.09, 0.6),
        'NEUT_number': (2.0, 5.5),
        'RDW_SD': (34.0, 47.0),
        'RDW_CV': (0.12, 0.15),
        'RDW': (10.0, 20.0),
        'P_LCR': (0.175, 0.423),
        'PCT': (0.0015, 0.0040),
    }
    blood_result = BloodCommonResult(donor=donor)
    for field, (min_val, max_val) in ranges.items():
        random_value = random.uniform(min_val, max_val)
        setattr(blood_result, field, random_value)
    blood_result.save()
    time.sleep(5)
```

```
        messages.success(request, f"Данні з приладу було отримано! Номер аналізу {blood_result.pk}")
        return redirect('view_donor', donor_id=donor.user_id)


    #Список донацій/ donations.html / view_donation
    @csrf_exempt
    def donations(request, donor_id=None):
        donations_list = Donation.objects.filter().order_by('-id')
        if donor_id is not None:
            donations_list = donations_list.filter(donor_id=donor_id).order_by('-pk')
        has_result = []
        result_list = BloodResult.objects.filter()
        for donation in donations_list:
            blood_result = result_list.filter(donation=donation.pk).first()
            has_result.append(blood_result)
        com_list = list(zip(donations_list, has_result))
        return render(request, 'donations.html', {'com_list': com_list})

    #Створення донації / create_donation.html
    @csrf_exempt
    def create_donation(request, donor_id):
        donor = get_object_or_404(Donor, pk=donor_id)
        if request.method == 'POST':
            form = DonationForm(request.POST)
            if form.is_valid():
                donation = form.save(commit=False)
                donation.save()
                return redirect('donations')
            else:
                messages.error(request, f"Форма не правильно заповненна")
        else:
            form = DonationForm()
        return render(request, 'create_donation.html', {'form': form, 'donor': donor})
```

```
#Перегляд донації/ view_donation.html
@csrf_exempt
def view_donation(request, donation_id):
    try:
        donation = Donation.objects.get(pk=donation_id)
        return render(request, 'view_donation.html',
                      {'donation': donation})
    except Donation.DoesNotExist:
        messages.error(request, "Запису про донацію не існує!")
        return redirect('donations')


#створення повного результату крові
@csrf_exempt
def create_full_blood_result(request, donation_id):
    donation = get_object_or_404(Donation, pk=donation_id)
    last_common_result = BloodCommonResult.objects.filter(donor=
donation.donor).order_by('-id').first()

    if request.method == 'POST':
        form = FullBloodForm(request.POST)
        if form.is_valid():
            blood_result = form.save(commit=False)
            blood_result.donation = donation
            blood_result.blood_common_result = last_common_result
            blood_result.save()
            return redirect('blood_result')
        else:
            messages.error(request, f"Результат був створений раніше")
            return redirect('blood_result')
    else:
        form = FullBloodForm()
```

```
            return render(request, 'create_full_blood.html',
                          {'form': form,
                           'donation': donation,
                           'last_common_result': last_common_result})

        #перегляд списку повних результатів крові
        @csrf_exempt
        def full_blood_results(request, donor_id=None):
            try:
                blood_results = BloodResult.objects.all().order_by('-id')
                if donor_id is not None:
                    blood_results = blood_results.filter(donor_id=donor_id).order_by('-id')
                return render(request, 'blood_result.html',
                              {'blood_results': blood_results})
            except Donation.DoesNotExist:
                messages.error(request, "Запису про результат крові не існує!")
                return redirect('blood_result')
        #перегляд повних результатів крові
        @csrf_exempt
        def view_full_blood_result(request, full_blood_result_id):#from full_blood_results: 100%
and user_panel
            user = request.user
            blood_result = get_object_or_404(BloodResult, pk=full_blood_result_id)
            if user.pk == blood_result.donor.user.pk or user.is_staff:
                return render(request, 'view_full_blood_result.html', {'blood_result': blood_result})
            else:
                messages.error(request, "Немає доступу до запису!")
                return redirect("index")




        #Функція для перетворення дати у день тижня
        def dayToWeekday(x):
```

```
    z = datetime.strptime(x, "%Y-%m-%d")
    y = z.strftime('%A')
    return y

#Функція для визначення списку робочих днів на наступні 21 день
def validWeekday(days):
    # Створення списку робочих днів на наступні 21 день
    today = datetime.now() + timedelta(days=1)
    weekdays = []
    for i in range(0, days):
        x = today + timedelta(days=i)
        y = x.strftime('%A')
        if y != 'Saturday' and y != 'Sunday':
            weekdays.append(x.strftime('%Y-%m-%d'))
    return weekdays

#Функція, що перевіряє, чи робочий день доступний для запису
def isWeekdayValid(x):
    validateWeekdays = []
    for j in x:
        #Перевірка, чи на обраний день доступно менше 10 записів
        if Appointment.objects.filter(day=j).count() < 10:
            validateWeekdays.append(j)
    return validateWeekdays

#Функція, що перевіряє доступність часів для запису на обраний день
def checkTime(times, day):
    available_times = []
    for k in times:
        # Перевірка, чи обраний час на день є доступним для запису
        if Appointment.objects.filter(day=day, time=k).count() < 1:
            available_times.append(k)
    return available_times
```

```
#Функція, що перевіряє доступність часів для редагування запису
def checkEditTime(times, day, id):
    available_times = []
    appointment = Appointment.objects.get(pk=id)
    time = appointment.time
    for k in times:
        #Перевірка, чи обраний час для редагування є доступним
        if Appointment.objects.filter(day=day, time=k).count() < 1 or time == k:
            available_times.append(k)
    return available_times


#Додаткова перевірка доступності часів для запису, з урахуванням того, що донор недавно виконав донацію
def additional_check(user_id, weekdays):
    try:
        donor = Donor.objects.get(user_id=user_id)
        latest_donation = Donation.objects.filter(donor=donor).order_by('-donation_date').first()

        if latest_donation:
            latest_donation_date = latest_donation.donation_date
            excluded_dates = [str(latest_donation_date + timedelta(days=i)) for i in range(0, 15)]
            available_weekdays = [weekday for weekday in weekdays if weekday not in excluded_dates]

            return available_weekdays

    except Donor.DoesNotExist:
        pass  #Обробка відсутності Donor з вказаним user_id

    return weekdays
```