МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ВІДДІЛЕННЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП’ЮТЕРНІ НАУКИ»

ЦИКЛОВА КОМІСІЯ СПЕЦІАЛЬНОСТІ
«КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня фаховий молодший бакалавр
за спеціальністю 122 Комп'ютерні науки
на тему:

«Інформаційна система обліку донорів крові»

Виконав здобувач освіти групи П-43стн
Мельнійчук Петро Вікторович

Керівник роботи:
Студзінський Володимир Вікторович

Рецензент:

Житомир – 2026

Зміст

Вступ 6

Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ 9

1.1 Основні задачі 9

1.2 Огляд існуючих рішень 9

1.3 Функціональні вимоги до програми 10

1.4 Вибір технології 10

1.5 Постановка задачі 11

Висновки до розділу 1 11

Розділ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ 12

2.1 Модель бази даних 12

2.2 Основні форми застосунку та їх призначення 21

2.3 Архітектура програмного засобу 33

2.4 Програмна реалізація 39

2.5 Тестування програми 48

Висновки до розділу 2 50

Розділ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ 52

3.1 Мінімальні вимоги до системи 52

3.2 Інструкція для роботи з програмою 53

Висновки до розділу 3 61

ВИСНОВКИ 62

Перелік літературних джерел 64

Додаток А. Код для роботи з базою даних 67

РЕФЕРАТ

Кваліфікаційна робота: 68 стор., 16 рис., 10 табл., 1 додаток, 20 джерел.

Об’єкт дослідження — процеси обліку донорів, реєстрації донацій та управління запасами компонентів крові в медичних закладах.

Мета роботи — розробка інформаційної системи для автоматизації діяльності центрів переливання крові, що забезпечує контроль термінів придатності матеріалів та оперативний пошук сумісних донорів.

Методи дослідження — об’єктно-орієнтоване проектування, використання архітектурного патерну MVVM для розподілу бізнес-логіки та інтерфейсу.

У роботі спроектовано та реалізовано десктопний додаток на базі платформи **.NET 9** та технології **WPF**. Для збереження даних використано реляційну СУБД **SQLite** з використанням **Entity Framework Core**. Система дозволяє вести детальний облік донорів, автоматично додавати результати донацій до інвентарю, відстежувати терміни придатності за принципом FIFO та формувати вихідну документацію (накладні, звіти) у форматі PDF за допомогою бібліотеки QuestPDF.

Результатом розробки є функціональне програмне забезпечення, яке мінімізує вплив людського фактора при підборі крові та підвищує ефективність управління критично важливими медичними ресурсами.

Ключові слова: *ДОНОР, КРОВ, .NET 9, WPF, MVVM, SQLITE, АВТОМАТИЗАЦІЯ, ІНВЕНТАРИЗАЦІЯ, PDF-ЗВІТНІСТЬ.*

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД — база даних;

ПЗ — програмне забезпечення;

ІС — інформаційна система;

Гр. крові — група крові донора або реципієнта;

АПІ (API) — інтерфейс прикладного програмування (Application Programming Interface);

АСУ — автоматизована система управління;

ПІБ — прізвище, ім'я та по батькові;

АТ — артеріальний тиск;

Hb — рівень гемоглобіну в крові;

Rh — резус-фактор (Rh-фактор);

C# — об'єктно-орієнтована мова програмування;

FIFO — «першим прийшов — першим пішов» (First In, First Out), метод управління запасами за датою придатності;

MVVM — архітектурний патерн проектування (Model-View-ViewModel);

NET 9 — кросплатформна середа розробки від Microsoft;

PDF — формат файлу для представлення документів (Portable Document Format);

SQLite — компактна вбудована реляційна система управління базами даних;

WPF — підсистема для побудови графічних інтерфейсів користувача (Windows Presentation Foundation);

XAML — декларативна мова розмітки для опису інтерфейсу додатка.

ВСТУП

Донорство крові є одним із ключових елементів системи охорони здоров'я, що забезпечує можливість проведення хірургічних втручань, лікування онкологічних захворювань, надання допомоги постраждалим у надзвичайних ситуаціях та підтримки пацієнтів із хронічними захворюваннями крові. За даними Всесвітньої організації охорони здоров'я, щорічно у світі заготовлюється понад 118 мільйонів одиниць донорської крові, однак потреба в ній залишається незадоволеною — особливо в країнах із недостатньо розвиненою інфраструктурою служб крові. В умовах збройного конфлікту та зростаючого навантаження на медичну систему України питання автоматизації обліку донорів крові та управління запасами набуває особливої практичної значущості.

Сучасний стан більшості регіональних станцій переливання крові характеризується фрагментарністю облікових процесів: дані про донорів ведуться в паперових журналах або розрізнених електронних таблицях, відсутній механізм автоматичного відстеження придатності запасів, а опрацювання запитів від медичних закладів здійснюється без належного інструментарію для пошуку сумісних донорів і формування супровідної документації. Це призводить до збільшення часу реагування на термінові запити, підвищення ризику використання протермінованих компонентів крові та загальної неефективності управлінських процесів.

Актуальність теми зумовлена необхідністю розробки спеціалізованої інформаційної системи, яка б об'єднала в єдиному програмному середовищі функції обліку донорів, реєстрації донацій, управління запасами компонентів крові та опрацювання запитів від лікувальних закладів. Впровадження такої системи дозволить скоротити витрати часу персоналу на рутинні операції, мінімізувати людський фактор при формуванні документів та підвищити загальну ефективність роботи служби крові.

Метою кваліфікаційної роботи є проєктування та розробка інформаційної системи обліку донорів крові з функціями реєстрації донацій,

автоматичного управління запасами компонентів крові, опрацювання запитів медичних закладів та формування супровідної документації у форматі PDF.

Для досягнення поставленої мети визначено такі **завдання**:

1. проаналізувати предметну область донорства крові та існуючі програмні рішення для автоматизації відповідних процесів;
2. визначити функціональні та нефункціональні вимоги до розроблюваної системи;
3. спроєктувати архітектуру застосунку із застосуванням шаблону проєктування MVVM та принципів розділення відповідальностей;
4. реалізувати модулі обліку донорів, донацій, запасів крові та запитів медичних закладів;
5. забезпечити автоматичне оновлення запасів при реєстрації донацій та механізм списання компонентів крові при виконанні запитів за принципом FIFO;
6. реалізувати формування звітної документації — списку донорів та накладної на видачу крові — у форматі PDF;
7. провести тестування розробленої системи та оцінити її відповідність поставленим вимогам.

Об'єктом дослідження є процеси обліку донорів крові, реєстрації донацій та управління запасами компонентів крові в установах служби крові.

Предметом дослідження є методи та засоби автоматизації зазначених процесів із використанням сучасних технологій розробки настільних застосунків.

У процесі виконання роботи використано такі **методи дослідження**: аналіз предметної області та нормативно-правової бази донорства крові в Україні; методи об'єктно-орієнтованого проєктування; шаблони архітектурного проєктування програмного забезпечення; методи реляційного моделювання даних.

Практична значущість роботи полягає в розробці повнофункціонального програмного продукту, який може бути впроваджений

у роботу регіональних станцій переливання крові або центрів крові лікарняних закладів. Система реалізована з використанням платформи .NET 9, технології WPF для побудови графічного інтерфейсу користувача, Entity Framework Core для роботи з базою даних SQLite, бібліотеки MaterialDesignThemes для забезпечення сучасного інтерфейсу відповідно до концепції Material Design, а також бібліотеки QuestPDF для формування документів.

Структура роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проведено аналіз предметної області та існуючих рішень. Другий розділ присвячено проєктуванню архітектури системи та моделюванню даних. У третьому розділі описано практичну реалізацію основних модулів системи. Четвертий розділ містить результати тестування та оцінку розробленого програмного продукту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Основні задачі

Автоматизація обліку в центрах переливання крові є критично важливою для забезпечення оперативності медичної допомоги. Основними задачами у цій сфері є:

- **Централізація даних про донорів**

  Зберігання персональних даних, групової приналежності та контактів в єдиній базі.

- **Моніторинг запасів**

  Автоматичний облік наявних об’ємів крові та її компонентів (плазми, еритроцитів, тромбоцитів).

- **Контроль придатності**

  Відстеження термінів зберігання біоматеріалів для запобігання використанню протермінованих компонентів.

- **Обробка термінових запитів**

  Швидкий пошук сумісних донорів для задоволення потреб медичних закладів.

- **Медичний аудит**

  Фіксація показників здоров'я донорів (гемоглобін, тиск) під час кожної процедури.

1.2 Огляд існуючих рішень

На ринку існують як комплексні госпітальні інформаційні системи (HIS), так і спеціалізовані рішення для банків крові (наприклад, eDelphyn або вітчизняні реєстри).

- **Переваги**

  Висока функціональність, інтеграція з державними реєстрами.

- **Недоліки**

  Висока вартість ліцензій, складність інтерфейсу для невеликих пунктів прийому, жорсткі вимоги до апаратного забезпечення.

- **Обґрунтування розробки**

  Створення локального, швидкого та інтуїтивно зрозумілого додатка, що працює офлайн (через SQLite) та забезпечує автоматичну генерацію супровідної документації (накладних, списків).

1.3 Функціональні вимоги до програми

На основі аналізу потреб користувачів сформовано такі вимоги:

- **Ведення бази донорів**

  додавання, редагування та видалення карток з історією донацій.

- **Реєстрація процедур**

  облік об’єму та типу отриманого матеріалу.

- **Управління складом**

  автоматичне додавання до запасів після донації та списання за принципом FIFO (First In, First Out).

- **Документообіг**

  експорт накладних на видачу крові та списків донорів у формат PDF.

- **Візуалізація стану**

  відображення статистики активності на головному екрані (дежборді).

1.4 Вибір технології

Для реалізації обрано наступний технологічний стек:

- **Платформа**

  .NET 9 та WPF (Windows Presentation Foundation) для створення сучасного десктопного інтерфейсу.

- **Архітектурний патерн**

  MVVM (Model-View-ViewModel), що забезпечує чіткий розподіл логіки та представлення.

- **База даних**

  SQLite — легка та надійна реляційна БД, що не потребує окремого сервера.

- **Бібліотеки UI**

  Material Design in XAML для забезпечення сучасної естетики та

ергономіки.

- **Звітність**

  QuestPDF для програмної генерації складних PDF-документів.

1.5 Постановка задачі

Метою роботи є розробка інформаційної системи «Донори Крові» для автоматизації роботи персоналу станцій переливання крові.

Об’єкт дослідження: процеси обліку донорів та руху запасів крові.

Предмет дослідження: методи та засоби розробки ПЗ на базі .NET для медичних інформаційних систем.

Основні завдання:

1. Проектування структури бази даних для зберігання інформації про донорів, донації та запити.
2. Реалізація механізму перевірки сумісності груп крові при обробці замовлень від лікарень.
3. Розробка алгоритму автоматичного списання запасів з контролем термінів придатності.
4. Створення зручного інтерфейсу з підтримкою темної/світлої тем та адаптивної навігації.

Висновки до розділу 1

В ході аналізу предметної галузі встановлено, що існуючі рішення часто є занадто складними для локальних пунктів переливання крові. Визначено основні бізнес-процеси, які потребують автоматизації: від первинної реєстрації донора до формування видаткової накладної. Обраний технологічний стек (.NET 9, WPF, SQLite) дозволяє створити швидку, надійну та незалежну від сервера систему з високим рівнем юзабіліті завдяки використанню Material Design. Сформовані вимоги та постановка задачі є базою для подальшого проектування архітектури та розробки програмного продукту.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

2.1 Модель бази даних

2.1.1 Опис предметної галузі та основних сутностей

Предметна галузь системи охоплює діяльність служби крові, пов'язану з реєстрацією донорів, обліком донацій, управлінням запасами компонентів крові та опрацюванням запитів від медичних закладів. Для коректного відображення цих процесів у реляційній базі даних виділено п'ять основних сутностей, кожна з яких відповідає окремому об'єкту предметної галузі.

Сутність «Донор» (Donors) є центральною в усій моделі. Вона зберігає персональні дані фізичної особи, яка виявила бажання або вже брала участь у здачі крові чи її компонентів. До ключових атрибутів належать: прізвище, ім'я, по батькові, дата народження, стать, група крові, резус-фактор, контактні дані, місце проживання, а також статус донора (активний, тимчасово відсторонений, неактивний). Додатково зберігаються агреговані показники — кількість виконаних донацій та дата останньої донації, що використовуються для швидкого відображення в інтерфейсі без додаткових обчислювальних запитів до бази даних.

Сутність «Донація» (Donations) фіксує кожен окремий факт здачі крові або її компонентів конкретним донором. Атрибути сутності включають: дату донації, тип зданого матеріалу (цільна кров, плазма, тромбоцити, еритроцити), об'єм у мілілітрах, місце проведення, прізвище лікаря, стан здоров'я донора під час здачі, показники гемоглобіну та артеріального тиску. Ця сутність є сполучною між донором та запасами крові, оскільки реєстрація нової донації автоматично ініціює поповнення запасів відповідним компонентом.

Сутність «Запаси крові» (BloodInventory) відображає наявний на складі обсяг конкретного компонента крові визначеної групи та резус-фактора. Кожен запис характеризується: групою крові, резус-фактором, типом компонента, кількістю в мілілітрах, датою заготівлі, датою закінчення терміну придатності та статусом (придатна або протермінована). Модель передбачає зберігання окремих записів для кожної заготовленої одиниці, що уможливлює

реалізацію принципу FIFO при видачі — першими використовуються компоненти з найближчим терміном придатності.

Сутність «Запит на кров» (BloodRequests) представляє офіційне звернення медичного закладу з проханням надати певний компонент крові визначеної групи та об'єму. Атрибути включають: назву лікарні, контактну особу, номер телефону, групу крові та резус-фактор, рівень терміновості (висока, середня, низька), необхідний об'єм, дату подання запиту, бажану дату отримання, поточний статус (активний, виконаний, скасований) та примітки. Виконання запиту супроводжується автоматичним списанням відповідних запасів і формуванням накладної.

Сутність «Сповіщення» (Notifications) призначена для зберігання інформації про повідомлення, що надсилаються донорам — зокрема, нагадування про можливість наступної донації або термінові запити. Атрибути: тип сповіщення, дата, текст повідомлення та ознака відправки.

2.1.2 ER-діаграма (Entity-Relationship Diagram)

ER-діаграма розробленої системи містить п'ять сутностей із зв'язками.

Таблиця 2.1.2.1 – Типи зв'язку

Сутність А	Сутність Б	Тип зв'язку	Назва зв'язку	Пояснення
Donors	Donations	1 : N	«здійснює»	Один донор може мати необмежену кількість донацій; кожна донація належить лише одному донору
Donors	Notifications	1 : N	«отримує»	Одному донору може бути надіслано декілька сповіщень
BloodRequests	BloodInventory	—	«списує»	Логічний зв'язок реалізується програмно: виконання запиту зменшує відповідні записи запасів

Зв'язки N:M у цій моделі відсутні, що спрощує реляційну схему та виключає необхідність у додаткових таблицях-асоціаціях. Зв'язок між запитами та запасами свідомо не моделюється як FK-зв'язок у базі даних, оскільки одне звернення може торкатися довільної кількості записів запасів

різних компонентів, а процес списання є транзакційною операцією прикладного рівня.

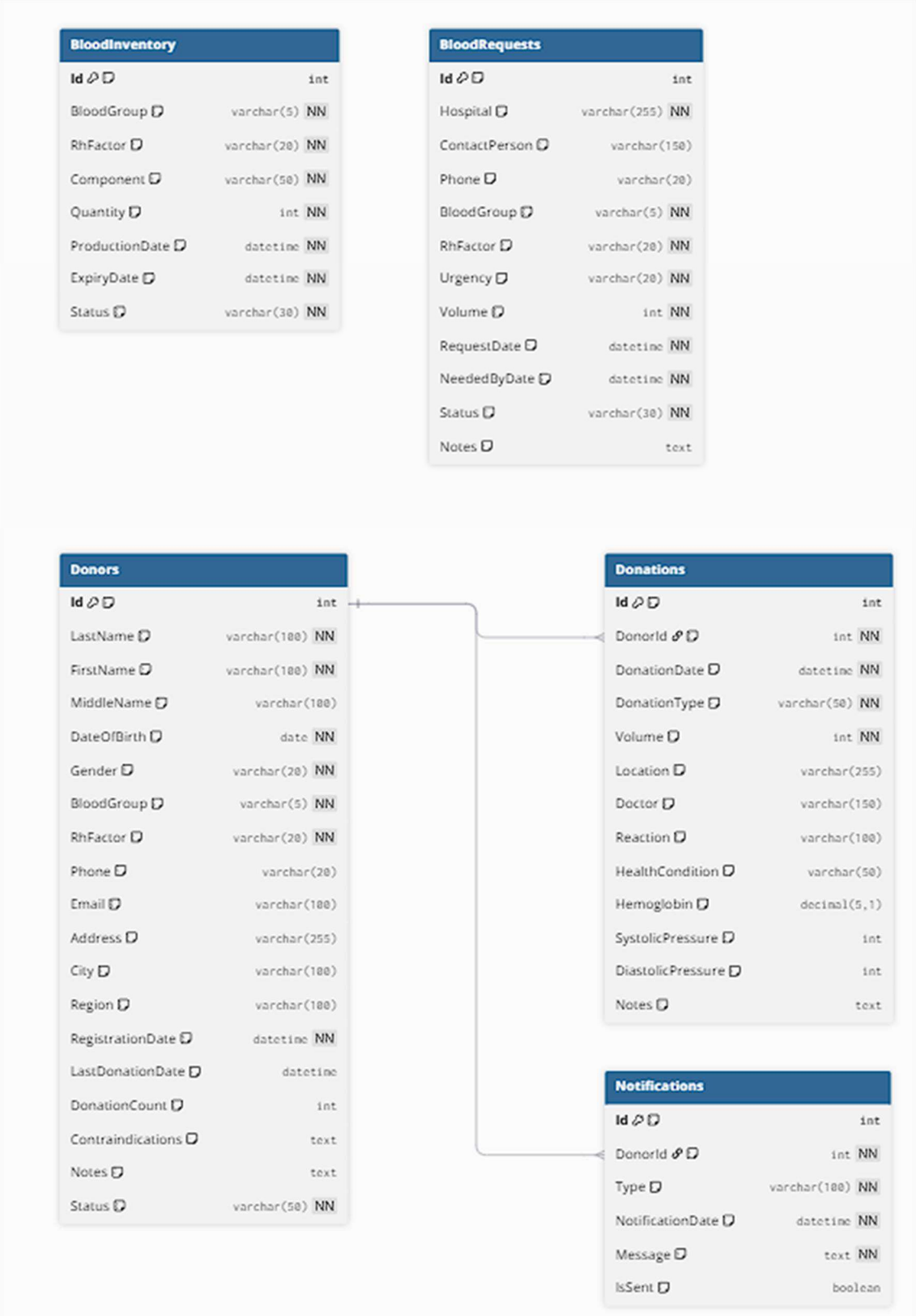

Рисунок 2.1.1. – ER-діаграма

Таблиця 2.1.2.2 – Класифікація атрибутів сутності Donors:

Атрибут	Тип	Клас	Примітка
Id	int	Простий, однозначний	Первинний ключ
LastName, FirstName, MiddleName	varchar	Прості, однозначні	Разом утворюють складений атрибут «ПІБ»
DateOfBirth	date	Простий, однозначний	Вік обчислюється похідно
BloodGroup + RhFactor	varchar	Прості, однозначні	Разом визначають сумісність
Phone, Email	varchar	Прості, однозначні	Контактні дані
Address, City, Region	varchar	Частини складеного атрибуту «Місцезнаходження»	—
DonationCount	int	Похідний (агрегований)	Денормалізація для продуктивності
LastDonationDate	datetime	Похідний (агрегований)	Оновлюється при кожній донації
Status	varchar	Простий, перелічуваний	Обмежений список значень

Таблиця 2.1.2.3 – Класифікація атрибутів сутності Donations:

Атрибут	Тип	Клас	Примітка
Id	int	Простий, однозначний	Первинний ключ
DonorId	int	Простий, однозначний	Зовнішній ключ
DonationType	varchar	Простий, перелічуваний	4 допустимі значення
Volume	int	Простий, однозначний	У мілілітрах
Hemoglobin, SystolicPressure, DiastolicPressure	decimal/int	Прості, однозначні	Медичні показники
HealthCondition	varchar	Простий, перелічуваний	—

Таблиця 2.1.2.4 – Класифікація атрибутів сутності BloodInventory:

Атрибут	Тип	Клас	Примітка
Id	int	Простий, однозначний	Первинний ключ
BloodGroup + RhFactor	varchar	Прості, однозначні	Ідентифікація типу крові
Component	varchar	Простий, перелічуваний	Тип компонента
Quantity	int	Простий, однозначний	Змінюється при списанні
ProductionDate, ExpiryDate	datetime	Прості, однозначні	Використовуються для FIFO
Status	varchar	Простий, похідний	Визначається порівнянням ExpiryDate з поточною датою

Забезпечення цілісності даних та уникнення надмірності

Модель реалізує кілька механізмів підтримки цілісності та мінімізації надмірності даних.

Референційна цілісність забезпечується зовнішніми ключами: Donations.DonorId → Donors.Id та Notifications.DonorId → Donors.Id. Entity Framework Core автоматично генерує відповідні обмеження на рівні SQLite, що унеможливлює збереження запису донації без прив'язки до існуючого донора.

Уникнення надмірності досягається виділенням кожної предметної сутності в окрему таблицю. Зокрема, персональні дані донора зберігаються виключно в таблиці Donors і не дублюються в таблиці Donations — до них звертаються через JOIN при потребі. Аналогічно, характеристики крові (група, резус) не дублюються між таблицями Donations, BloodInventory та BloodRequests, а зберігаються незалежно в кожній таблиці лише в обсязі, необхідному для функціонування відповідного модуля.

Денормалізація з обґрунтуванням. Атрибути DonationCount та LastDonationDate у таблиці Donors є свідомим відступом від третьої нормальної форми — їхні значення теоретично можна отримати агрегуванням таблиці Donations. Однак така денормалізація є обґрунтованою: вона забезпечує миттєве відображення зведених показників у списку донорів без виконання агрегатних запитів, що суттєво підвищує продуктивність інтерфейсу при великій кількості записів. Цілісність цих полів підтримується програмно — вони оновлюються транзакційно одночасно зі збереженням нової донації.

Відсутність зв'язку між BloodRequests та BloodInventory на рівні схеми бази даних є архітектурним рішенням, що відображає природу процесу видачі крові: один запит може бути задоволений шляхом часткового списання з кількох записів запасів різних партій. Операція списання реалізована як транзакційний алгоритм на прикладному рівні, що зберігає гнучкість моделі та уникає ускладнення схеми проміжними таблицями.

2.1.3 Фізична модель бази даних

Фізична модель бази даних розроблена для СКБД SQLite і реалізована засобами Entity Framework Core із підходом Code First. Усі таблиці створюються автоматично при першому запуску застосунку через метод EnsureCreatedAsync(). Нижче наведено опис кожної таблиці з повним переліком полів, типів даних та обмежень.

Таблиця 2.1.3.1 – Donors (Донори)

№	Поле	Тип SQLite	Обмеження	Опис
1	Id	INTEGER	PRIMARY KEY, AUTOINCREMENT, NOT NULL	Унікальний ідентифікатор
2	LastName	TEXT	NOT NULL	Прізвище
3	FirstName	TEXT	NOT NULL	Ім'я
4	MiddleName	TEXT	NULL	По батькові
5	DateOfBirth	TEXT	NOT NULL	Дата народження (ISO 8601)
6	Gender	TEXT	NOT NULL, DEFAULT 'Чоловіча'	Стать
7	BloodGroup	TEXT	NOT NULL, DEFAULT 'A+'	Група крові
8	RhFactor	TEXT	NOT NULL, DEFAULT 'Позитивний'	Резус-фактор
9	Phone	TEXT	NULL	Номер телефону
10	Email	TEXT	NULL	Електронна пошта
11	Address	TEXT	NULL	Адреса проживання
12	City	TEXT	NULL	Місто
13	Region	TEXT	NULL	Область
14	RegistrationDate	TEXT	NOT NULL	Дата реєстрації
15	LastDonationDate	TEXT	NULL	Дата останньої донації
16	DonationCount	INTEGER	NOT NULL, DEFAULT 0	Загальна кількість донацій
17	Contraindications	TEXT	NULL	Медичні протипоказання
18	Notes	TEXT	NULL	Примітки
19	Status	TEXT	NOT NULL, DEFAULT 'Активний'	Статус донора

Таблиця 2.1.3.2 – Donations (Донації)

№	Поле	Тип SQLite	Обмеження	Опис
1	Id	INTEGER	PRIMARY KEY, AUTOINCREMENT, NOT NULL	Унікальний ідентифікатор
2	DonorId	INTEGER	NOT NULL, FOREIGN KEY → Donors(Id) ON DELETE CASCADE	Посилання на донора
3	DonationDate	TEXT	NOT NULL	Дата донації (ISO 8601)
4	DonationType	TEXT	NOT NULL, DEFAULT 'Цільна кров'	Тип донації
5	Volume	INTEGER	NOT NULL, DEFAULT 450	Об'єм у мілілітрах
6	Location	TEXT	NULL	Місце проведення
7	Doctor	TEXT	NULL	Прізвище лікаря
8	Reaction	TEXT	NULL	Реакція організму
9	HealthCondition	TEXT	NOT NULL, DEFAULT 'Добре'	Стан здоров'я
10	Hemoglobin	REAL	NULL	Гемоглобін, г/л
11	SystolicPressure	INTEGER	NULL	Систолічний тиск, мм рт. ст.
12	DiastolicPressure	INTEGER	NULL	Діастолічний тиск, мм рт. ст.
13	Notes	TEXT	NULL	Примітки

Таблиця 2.1.3.3 – BloodRequests (Запити на кров)

№	Поле	Тип SQLite	Обмеження	Опис
1	Id	INTEGER	PRIMARY KEY, AUTOINCREMENT, NOT NULL	Унікальний ідентифікатор
2	Hospital	TEXT	NOT NULL	Назва медичного закладу
3	ContactPerson	TEXT	NOT NULL	Контактна особа
4	Phone	TEXT	NOT NULL	Телефон
5	BloodGroup	TEXT	NOT NULL, DEFAULT 'A+'	Запитувана група крові
6	RhFactor	TEXT	NOT NULL, DEFAULT 'Позитивний'	Резус-фактор
7	Urgency	TEXT	NOT NULL, DEFAULT 'Середня'	Терміновість
8	Volume	INTEGER	NOT NULL	Необхідний об'єм, мл
9	RequestDate	TEXT	NOT NULL	Дата подання запиту
10	NeededByDate	TEXT	NOT NULL	Бажана дата отримання
11	Status	TEXT	NOT NULL, DEFAULT 'Активний'	Поточний статус
12	Notes	TEXT	NULL	Примітки

Таблиця 2.1.3.4 – Notifications (Сповіщення)

№	Поле	Тип SQLite	Обмеження	Опис
1	Id	INTEGER	PRIMARY KEY, AUTOINCREMENT, NOT NULL	Унікальний ідентифікатор
2	DonorId	INTEGER	NOT NULL, FOREIGN KEY → Donors(Id) ON DELETE CASCADE	Посилання на донора
3	Type	TEXT	NOT NULL, DEFAULT 'Наступна донація'	Тип сповіщення
4	NotificationDate	TEXT	NOT NULL	Запланована дата
5	Message	TEXT	NOT NULL	Текст повідомлення
6	IsSent	INTEGER	NOT NULL, DEFAULT 0	Ознака відправки (0/1)

Таблиця 2.1.3.5 – BloodInventory (Запаси крові)

№	Поле	Тип SQLite	Обмеження	Опис
1	Id	INTEGER	PRIMARY KEY, AUTOINCREMENT, NOT NULL	Унікальний ідентифікатор
2	BloodGroup	TEXT	NOT NULL, DEFAULT 'A+'	Група крові
3	RhFactor	TEXT	NOT NULL, DEFAULT 'Позитивний'	Резус-фактор
4	Component	TEXT	NOT NULL, DEFAULT 'Цільна кров'	Тип компонента
5	Quantity	INTEGER	NOT NULL	Кількість у мілілітрах
6	ProductionDate	TEXT	NOT NULL	Дата заготівлі
7	ExpiryDate	TEXT	NOT NULL	Термін придатності
8	Status	TEXT	NOT NULL, DEFAULT 'Придатна'	Статус запасу

Схема бази даних подано рисунку 2.1.2. На ній відображено всі таблиці, їхні поля з типами даних, а також зв'язки між ними із зазначенням кардинальності.

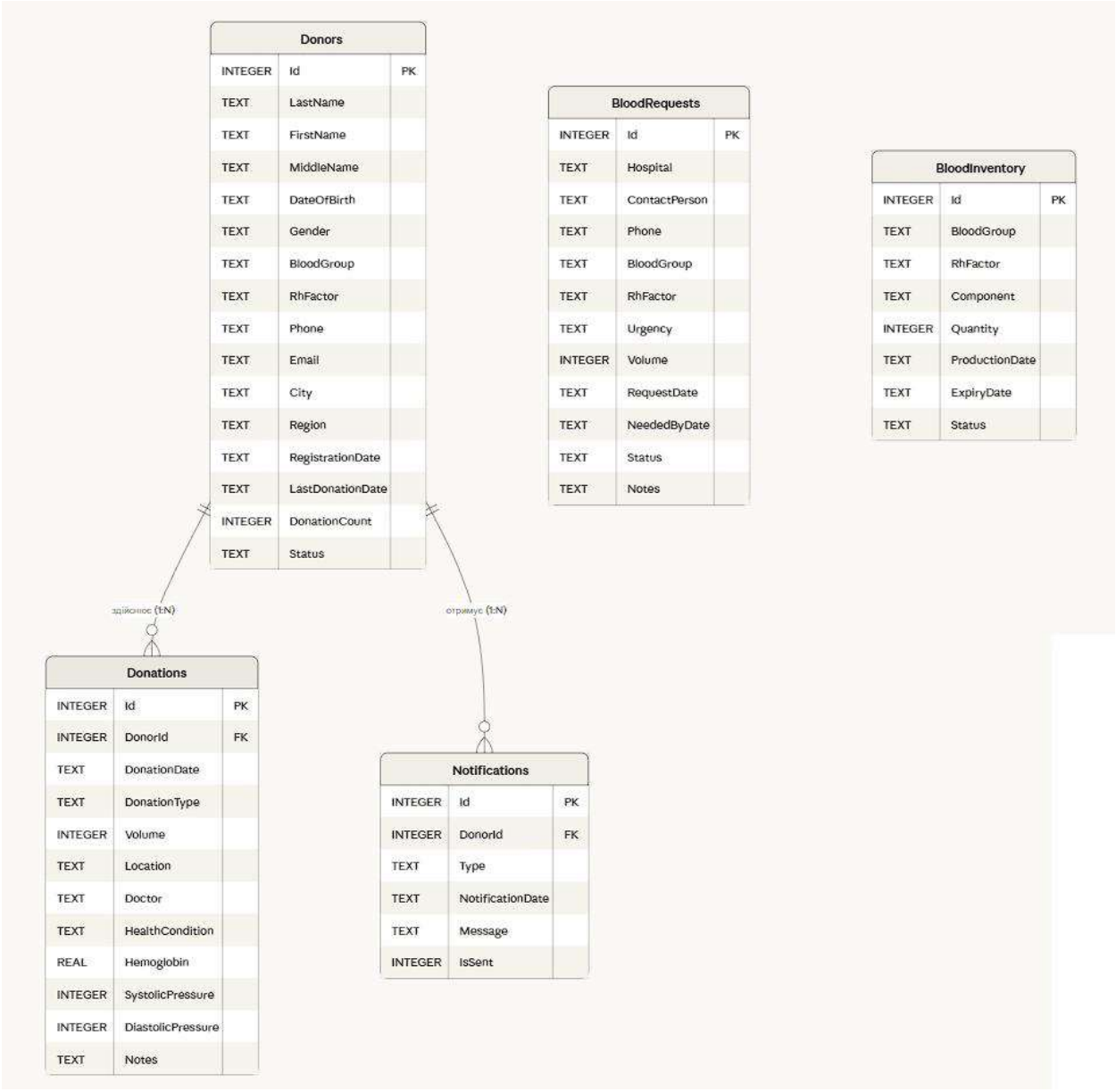

Рисунок 2.1.2. – Схема бази даних інформаційної системи

Таблиця 2.1.3.6 — Структура таблиць бази даних

Таблиця	Призначення	К-сть полів	Первинний ключ	Зовнішні ключі	Каскадне видалення
Donors	Персональні дані донорів	19	Id (INTEGER)	—	—
Donations	Факти здачі крові	13	Id (INTEGER)	DonorId → Donors.Id	Так
BloodRequests	Запити медичних закладів	12	Id (INTEGER)	—	—
Notifications	Сповіщення донорам	6	Id (INTEGER)	DonorId → Donors.Id	Так
BloodInventory	Запаси компонентів крові	8	Id (INTEGER)	—	—

Усі первинні ключі визначені як INTEGER PRIMARY KEY AUTOINCREMENT, що відповідає рекомендованому підходу для SQLite і забезпечує автоматичне призначення унікальних цілочисельних ідентифікаторів. Поля дати та часу зберігаються у форматі тексту згідно зі стандартом ISO 8601 (YYYY-MM-DDTHH:MM:SS), що є стандартною практикою для SQLite через відсутність окремого типу даних для дат. Entity Framework Core автоматично виконує перетворення між типами DateTime у C# та текстовим представленням у базі даних. Логічне значення поля IsSent у таблиці Notifications зберігається як INTEGER зі значеннями 0 (не відправлено) та 1 (відправлено), що відповідає обмеженням типової системи SQLite.

2.2 Основні форми застосунку та їх призначення

Розроблений програмний засіб реалізований у вигляді настільного Windows-застосунку з єдиним головним вікном, усередині якого здійснюється навігація між функціональними розділами. Інтерфейс побудований за принципом «одне вікно — один робочий простір»: ліва панель містить постійне навігаційне меню, а права частина динамічно відображає сторінку обраного розділу. Такий підхід забезпечує цілісність сприйняття застосунку та мінімізує когнітивне навантаження на користувача. Загальна структура інтерфейсу складається з семи функціональних форм, кожна з яких відповідає окремому аспекту діяльності служби крові.

2.2.1 Головне вікно та панель навігації

Головне вікно застосунку є контейнером для всіх інших форм і залишається незмінним протягом усього сеансу роботи. Воно розділене на дві зони: вертикальну навігаційну панель фіксованої ширини та основну робочу область змінного вмісту. Навігаційна панель відображає логотип системи, перелік розділів з відповідними піктограмами, а також кнопку переходу до налаштувань у нижній частині. У верхній смузі основної зони відображається назва поточного розділу та дата у форматі «дд місяць рррр» відповідно до

українського мовного стандарту. Вибір пункту меню призводить до заміни вмісту правої частини без перезавантаження вікна, що реалізовано через механізм прив'язки даних між MainViewModel та шаблонами відображення DataTemplate.

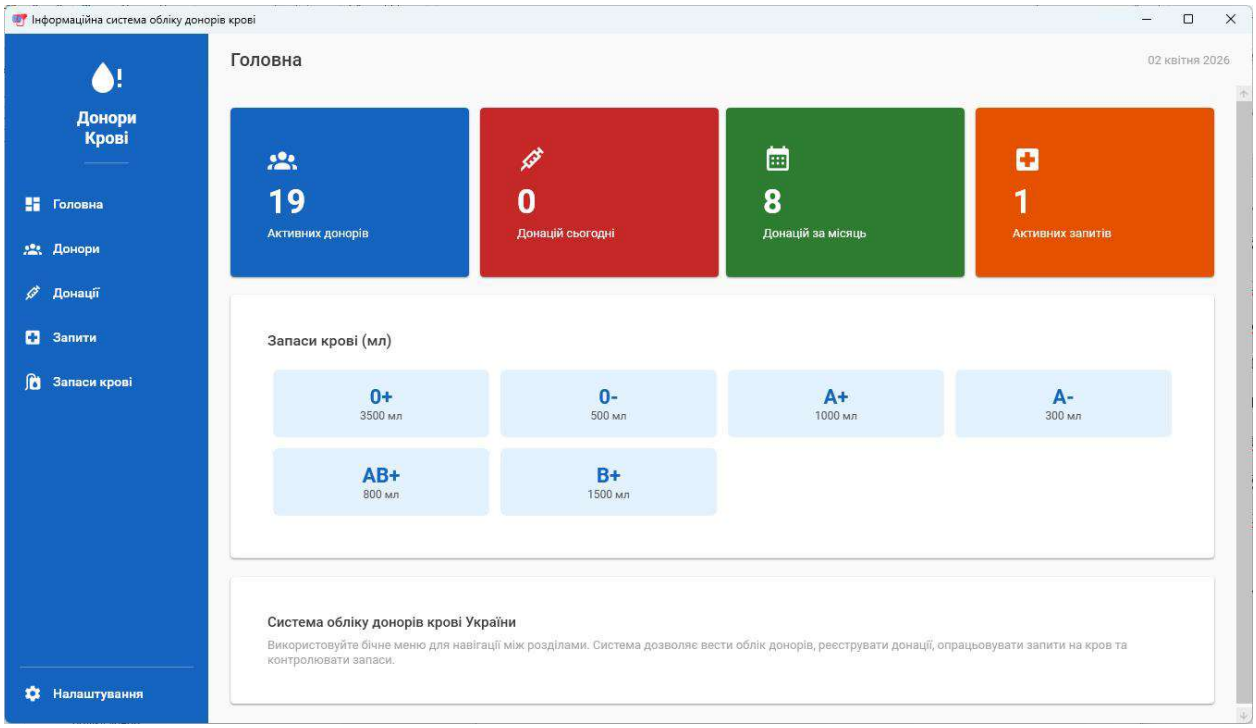

Рисунок 2.2.1. – Головне вікно застосунку з навігаційною панеллю

2.2.2 Форма «Головна» (Dashboard)

Форма головної сторінки призначена для відображення зведених статистичних показників роботи служби крові та забезпечує оперативний огляд поточного стану системи без необхідності переходу до окремих розділів. Дані на дашборді оновлюються автоматично щоразу при переході до цього розділу.

У верхній частині форми розміщено чотири інформаційні картки, кожна з яких відображає один ключовий показник: кількість активних донорів, кількість донацій за поточний день, кількість донацій за поточний місяць та кількість активних запитів від медичних закладів. Усі показники обчислюються в реальному часі шляхом агрегатних запитів до бази даних. Нижче розташована панель зведених запасів крові, що містить інформацію про наявний об'єм у мілілітрах у розрізі восьми груп крові. Дані запасів

групуються за полем BloodGroup таблиці BloodInventory з підсумовуванням поля Quantity лише для записів зі статусом «Придатна».

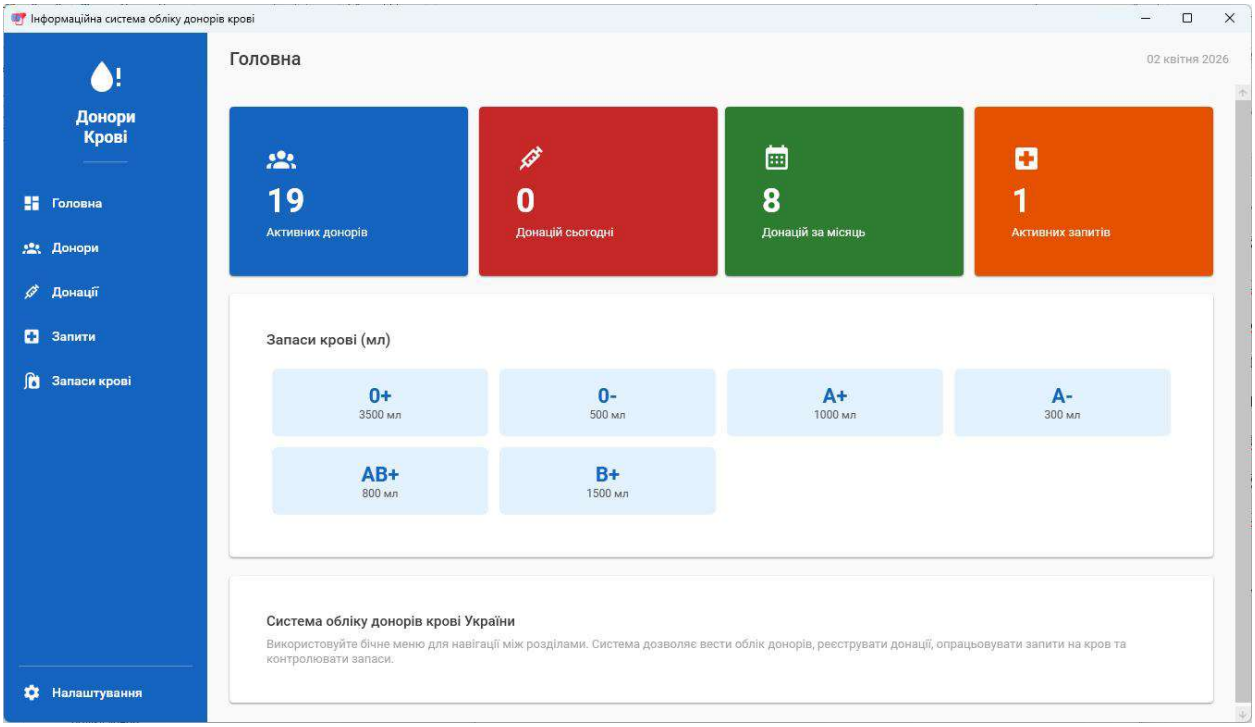

Рисунок 2.2.2. – Форма «Головна» із зведеними показниками та станом запасів

2.2.3 Форма «Донори»

Форма управління донорами є однією з центральних у системі і забезпечує повний цикл роботи з реєстром донорів: перегляд, пошук, додавання, редагування, видалення записів та перегляд індивідуальної історії донацій. Форма реалізована у вигляді двох станів: режим перегляду списку та режим редагування запису, між якими відбувається перемикання без переходу на нову сторінку.

У режимі перегляду списку верхня панель інструментів містить три елементи фільтрації. Перше поле — текстове, призначене для пошуку за прізвищем, ім'ям, по батькові або номером телефону; пошук виконується після натискання клавіші Enter або кнопки «Знайти». Друге поле — випадаючий список груп крові з дев'ятьма значеннями: «Всі», «A+», «A−», «B+», «B−», «AB+», «AB−», «0+», «0−». Третє поле — випадаючий список статусів із чотирма значеннями: «Всі», «Активний», «Тимчасово відсторонений»,

«Неактивний». Праворуч від полів фільтрації розміщено три кнопки дій: «Знайти», «Додати» та кнопки редагування й видалення обраного запису.

Основна таблиця відображає наступні поля донора: прізвище, ім'я, по батькові, вік (обчислюється автоматично з дати народження), стать, група крові, резус-фактор, номер телефону, місто, кількість донацій, дата останньої донації у форматі «дд.мм.рррр» та поточний статус. Рядки донорів зі статусом «Тимчасово відсторонений» підсвічуються жовтим фоном, а зі статусом «Неактивний» — сірим, що дозволяє швидко ідентифікувати проблемні записи без додаткових фільтрів.

У нижній частині форми розміщено розгортальну панель «Історія донацій вибраного донора». При виборі рядка в основній таблиці система автоматично завантажує всі пов'язані записи з таблиці Donations та відображає їх у вигляді вкладеної таблиці з такими полями: дата, тип донації, об'єм у мілілітрах, місце проведення, прізвище лікаря, стан здоров'я та рівень гемоглобіну.

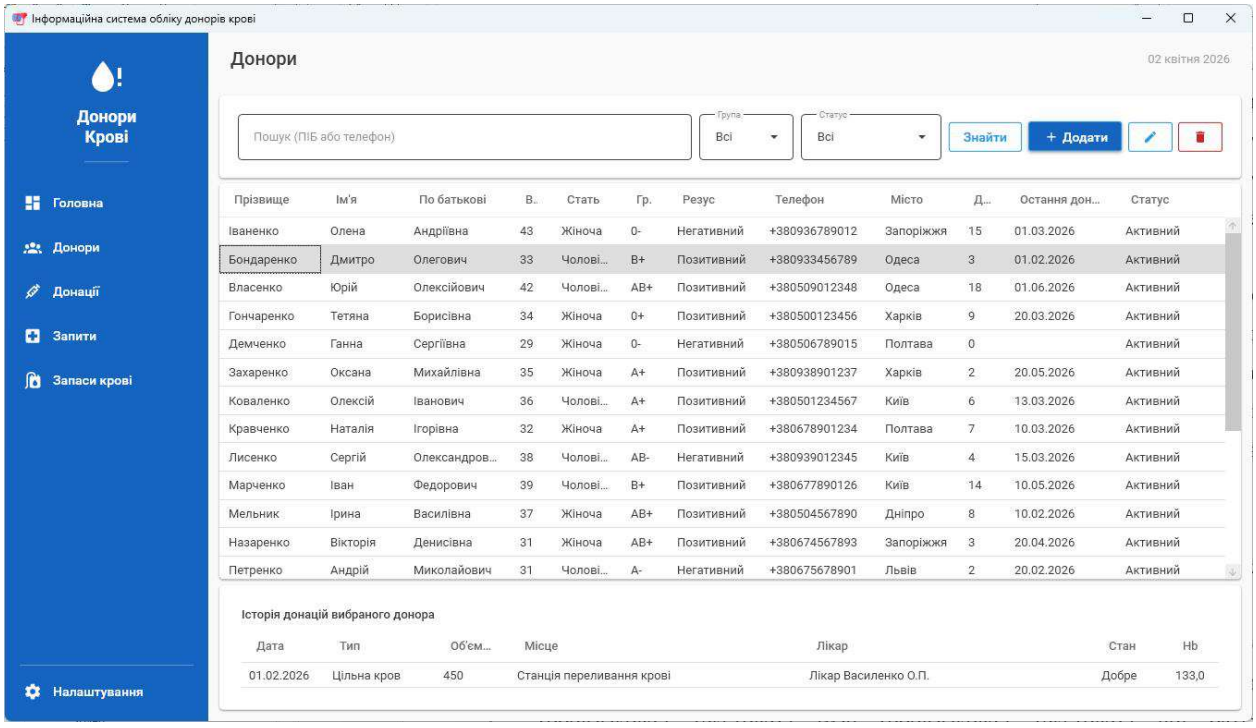

Рисунок 2.2.3. – Форма «Донори»

У режимі редагування форма відображає структуровану картку донора, розбиту на тематичні блоки полів. Особисті дані включають: прізвище (обов'язкове, текстове), ім'я (обов'язкове, текстове), по батькові

(необов'язкове, текстове), дату народження (вибір через календарний контрол DatePicker у форматі «дд.мм.рррр»). Медичні дані включають: стать (випадаючий список «Чоловіча» / «Жіноча»), групу крові (вісім значень), резус-фактор («Позитивний» / «Негативний»), статус (три значення). Контактні дані включають: номер телефону у довільному текстовому форматі, адресу електронної пошти, адресу проживання, місто та область. Додатково передбачені поля «Протипоказання» та «Примітки» для зберігання довільного медичного тексту. Збереження запису виконується через кнопку «Зберегти»; при додаванні нового донора присвоюється автоматичний ідентифікатор, при редагуванні — оновлюються поля існуючого запису методом FindAsync з наступним копіюванням значень у відстежуваний Entity Framework Core об'єкт.

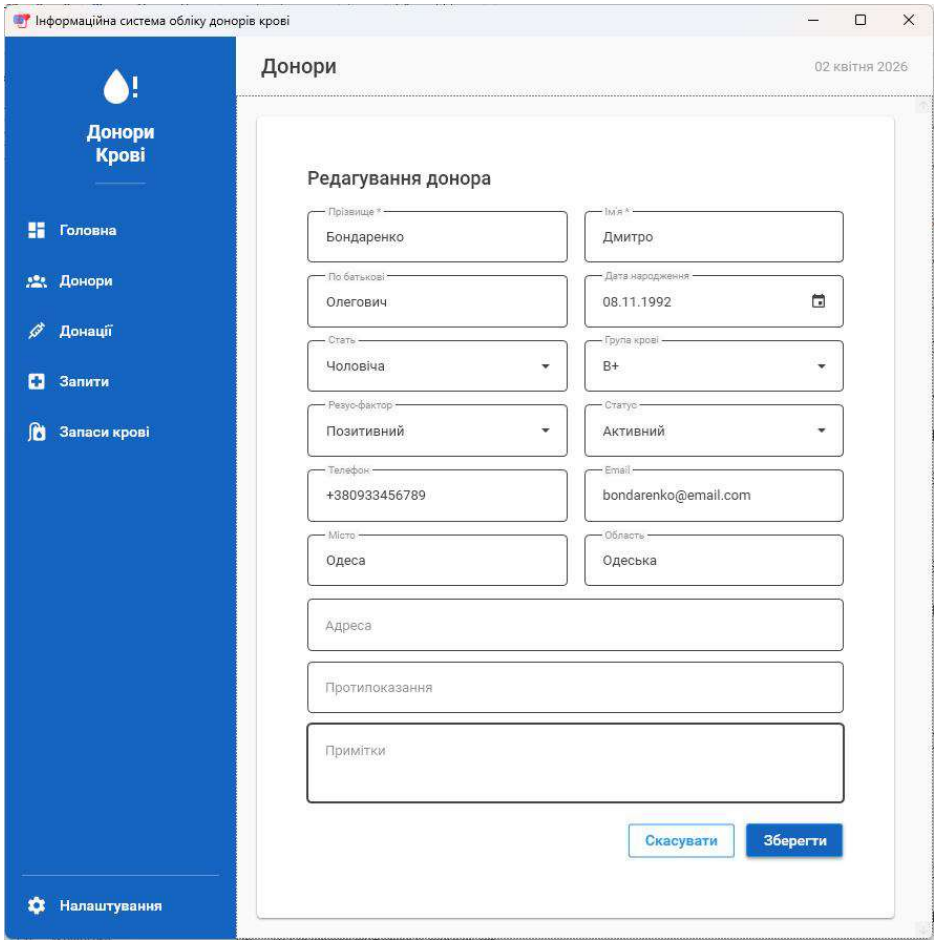

Рисунок 2.2.4. – Форма редагування запису донора

2.2.4 Форма «Донації»

Форма обліку донацій призначена для реєстрації фактів здачі крові або її компонентів, перегляду загального журналу донацій із фільтрацією за датою, а також редагування і видалення існуючих записів. Форма, аналогічно до форми донорів, реалізована у двох станах: перегляд журналу та форма введення / редагування донації.

У режимі журналу верхня панель містить два поля вибору дати — «Від» та «До» — для обмеження часового діапазону відображуваних записів. Зміна будь-якого з цих полів автоматично ініціює перезавантаження таблиці без додаткового підтвердження, що реалізовано через методи OnFilterFromChanged та OnFilterToChanged у DonationsViewModel. За замовчуванням встановлено діапазон від трьох місяців назад до поточної дати. Поруч із полями дат розміщено кнопки «Додати», «Редагувати» та «Видалити». Таблиця журналу відображає: дату донації, повне ім'я донора, групу крові, тип донації, об'єм, місце, лікаря, стан здоров'я та рівень гемоглобіну. Подвійний клік на рядку відкриває форму редагування обраної донації.

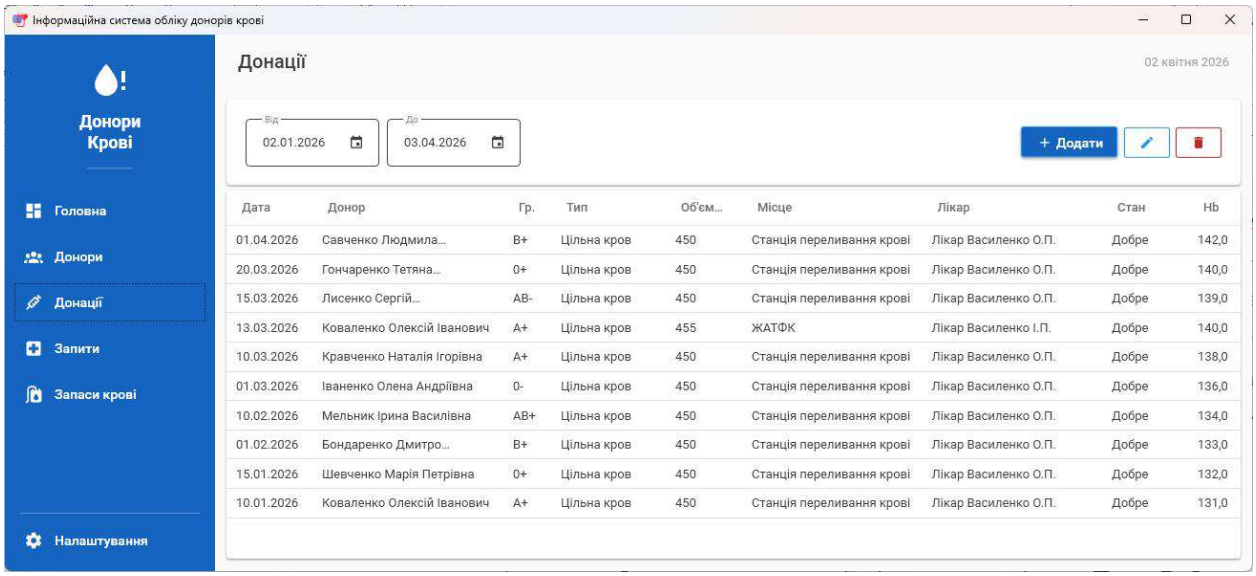

Рисунок 2.2.5. – Форма «Донації» у режимі журналу

Форма введення донації містить наступні поля. Поле «Вибрати донора» реалізоване як ComboBox з відображенням повного імені та групи крові донора — перелік заповнюється з таблиці Donors і містить усіх зареєстрованих

донорів. Після вибору донора, який вже здавав кров раніше, система автоматично відображає інформаційний рядок із датою наступної можливої донації, що обчислюється відповідно до типу попередньої донації: 60 днів для цільної крові, 14 днів для плазми та тромбоцитів, 90 днів для еритроцитів. Поле «Дата донації» вводиться через DatePicker. Поле «Тип донації» — випадаючий список із чотирма значеннями: «Цільна кров», «Плазма», «Тромбоцити», «Еритроцити». Поле «Об'єм» — числове, у мілілітрах, за замовчуванням 450 мл. Поле «Стан здоров'я» — випадаючий список «Добре» / «Задовільно» / «Погано». Поля «Місце проведення» та «Лікар» — текстові, необов'язкові. Медичні показники «Гемоглобін» (у г/л, тип REAL) та «АТ систолічний» (у мм рт. ст., тип INTEGER) — числові, необов'язкові. При збереженні нової донації система автоматично оновлює поля DonationCount та LastDonationDate у записі відповідного донора, а також створює новий запис у таблиці BloodInventory з відповідним компонентом, об'ємом та розрахованим терміном придатності.

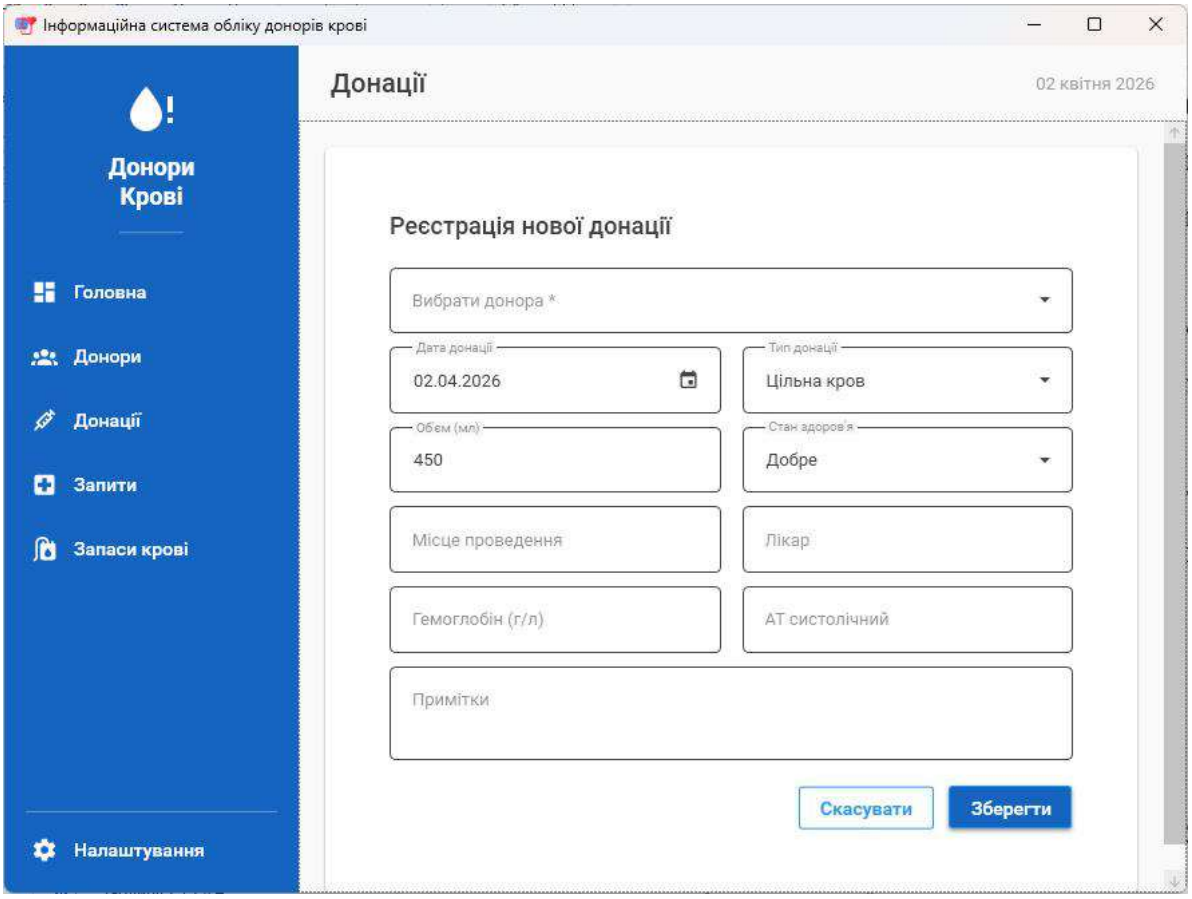

Рисунок 2.2.6. – Форма реєстрації нової донації

2.2.5 Форма «Запити на кров»

Форма управління запитами від медичних закладів призначена для реєстрації звернень лікарень, відстеження їхнього статусу, пошуку відповідних донорів, формування PDF-списку донорів для оповіщення, а також виконання запиту зі списанням запасів і формуванням накладної на видачу крові. Форма поділена на дві зони: ліву — таблицю запитів — та праву — панель пошуку донорів.

Таблиця запитів відображає такі поля: назва лікарні, група крові, рівень терміновості, необхідний об'єм у мілілітрах, бажана дата отримання та поточний статус. Рядки з терміновістю «Висока» підсвічуються червоним фоном, виконані запити — зеленим, скасовані — відображаються сірим кольором. Над таблицею розміщено панель інструментів із кнопками «Новий запит», «Редагувати» (олівець), «Видалити» (кошик) та «Виконати запит». Перед таблицею відображається інформаційний рядок із даними про доступний об'єм запасів крові відповідної групи та резус-фактора для вибраного запиту. Кнопка «Виконати запит» активна лише тоді, коли наявних придатних запасів достатньо для задоволення всього об'єму запиту, що контролюється властивістю CanFulfill.

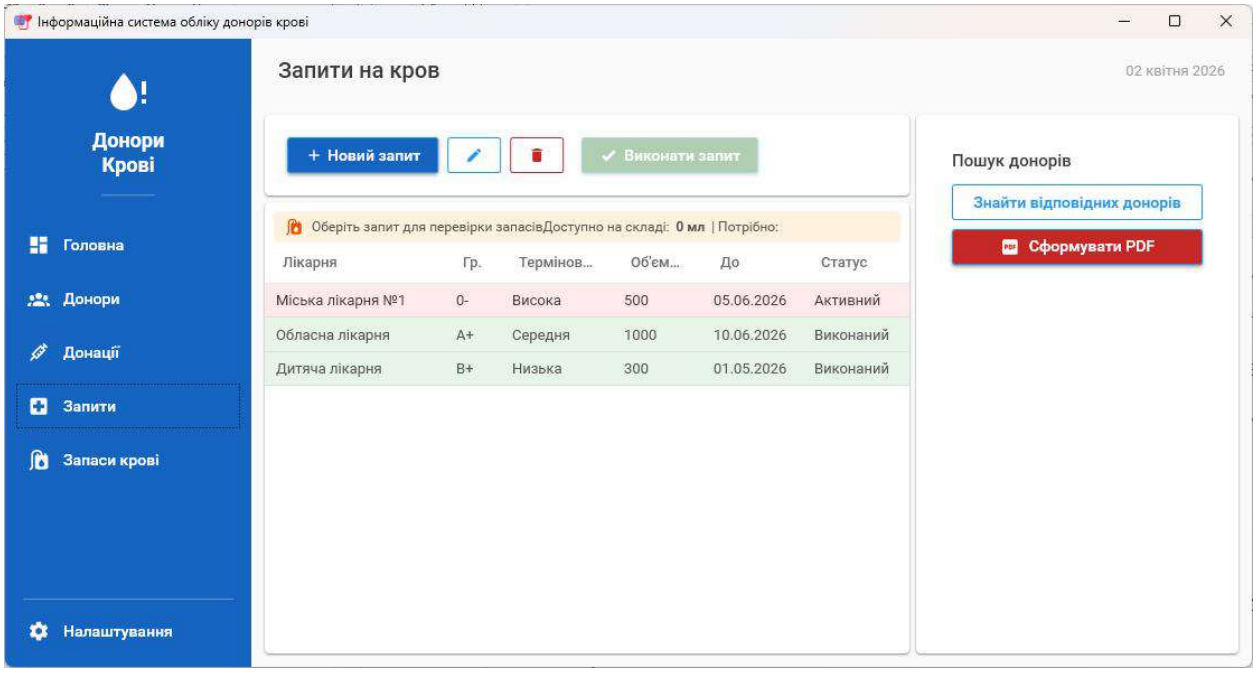

Рисунок 2.2.7. – Форма «Запити на кров»

Форма введення нового запиту або редагування існуючого містить такі поля: «Лікарня» — текстове, обов'язкове; «Контактна особа» — текстове; «Телефон» — текстове; «Група крові» — випадаючий список із вісьмома значеннями; «Резус-фактор» — «Позитивний» / «Негативний»; «Терміновість» — «Висока» / «Середня» / «Низька»; «Статус» — «Активний» / «Виконаний» / «Скасований»; «Об'єм (мл)» — числове; «Потрібно до» — дата через DatePicker; «Примітки» — текстове, необов'язкове.

Права панель «Пошук донорів» дозволяє знайти активних донорів із сумісною групою крові. Після натискання «Знайти відповідних донорів» система формує список сумісних донорів на основі таблиці сумісності груп крові. Кнопка «Сформувати PDF» відкриває системний діалог збереження файлу та генерує PDF-документ зі списком донорів та деталями запиту. При виконанні запиту через кнопку «Виконати запит» система виконує транзакційне списання запасів за принципом FIFO (першими використовуються компоненти з найближчим терміном придатності), змінює статус запиту на «Виконаний» та генерує накладну на видачу крові у форматі PDF із полями для підписів відповідальних осіб.

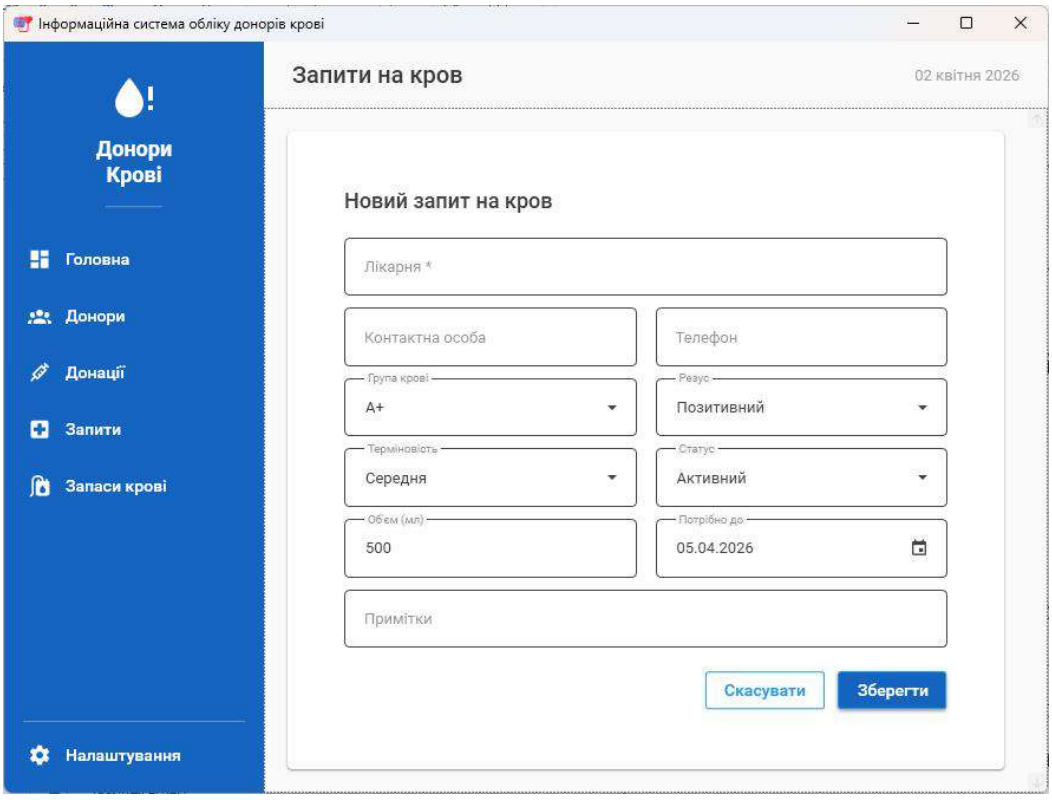

Рисунок 2.2.8. – Форма введення нового запиту на кров

2.2.6 Форма «Запаси крові»

Форма управління запасами крові призначена для відображення поточного стану складу компонентів крові, а також для ручного додавання, редагування і видалення записів запасів. Передбачено два шляхи поповнення запасів: автоматичний — при реєстрації нової донації система автоматично створює відповідний запис у таблиці BloodInventory, та ручний — через форму введення, що призначена для обліку компонентів, отриманих з інших медичних закладів або заготовлених поза системою.

Таблиця запасів відображає такі поля: група крові, резус-фактор, тип компонента, кількість у мілілітрах, дата заготівлі, дата закінчення терміну придатності у форматі «дд.мм.рррр» та статус. Записи зі статусом «Протермінована» підсвічуються червоним фоном з темно-червоним текстом, що дозволяє миттєво ідентифікувати непридатні компоненти. Статус визначається автоматично під час перегляду шляхом порівняння поля ExpiryDate з поточною датою в методі GetAvailableAsync. Над таблицею розміщено панель із кнопками «Додати запас», «Редагувати» та «Видалити».

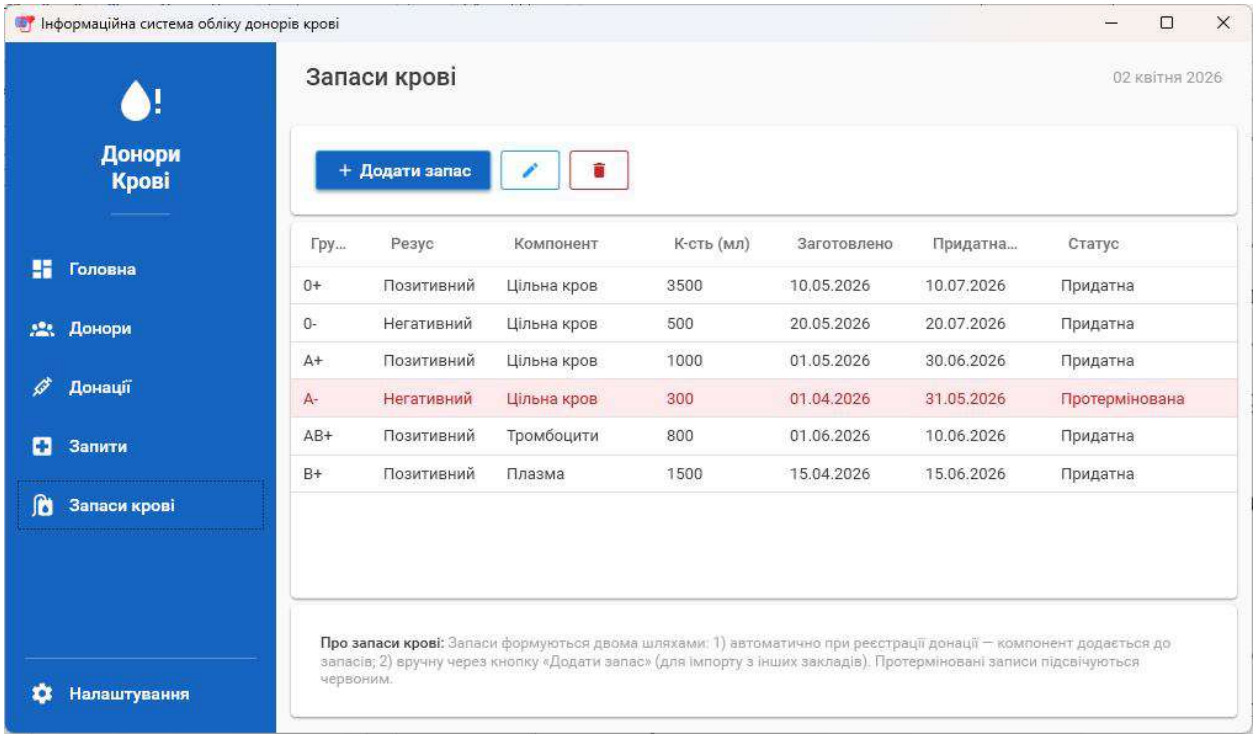

Рисунок 2.2.9. – Форма «Запаси крові» з таблицею компонентів

Форма введення або редагування запасу містить такі поля: «Група крові» — випадаючий список із вісьмома значеннями; «Резус-фактор» —

«Позитивний» / «Негативний»; «Компонент» — «Цільна кров» / «Еритроцити» / «Плазма» / «Тромбоцити»; «Кількість (мл)» — числове поле типу INTEGER; «Дата заготівлі» та «Термін придатності» — через DatePicker; «Статус» — «Придатна» / «Протермінована». Терміни придатності за типами компонентів для довідки: цільна кров та еритроцити — 42 доби, плазма — 365 діб, тромбоцити — 5 діб.

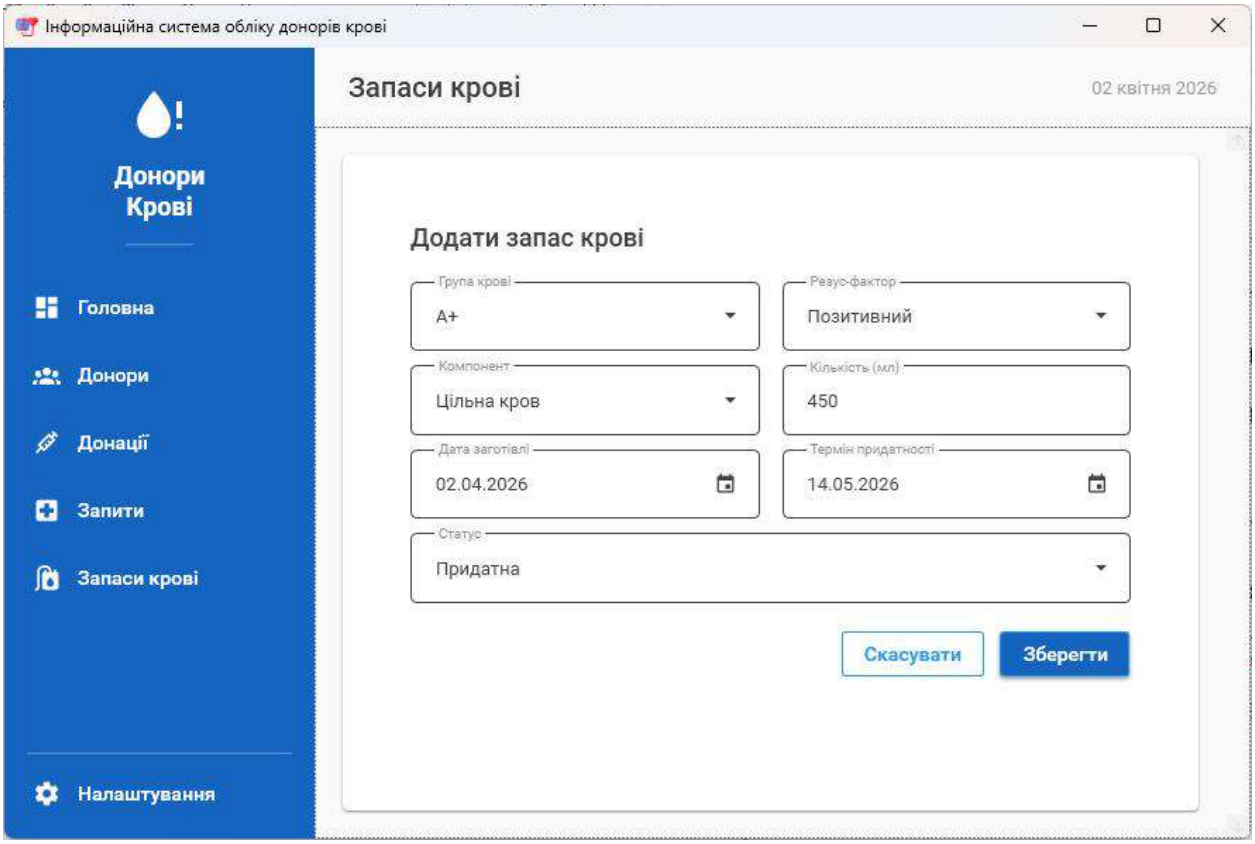

Рисунок 2.2.10. – Форма введення нового запасу крові

2.2.7 Форма «Налаштування»

Форма налаштувань є службовою і надає користувачеві доступ до конфігураційних параметрів застосунку, не пов'язаних безпосередньо з оперативними даними. Форма складається з трьох карток.

Картка «Зовнішній вигляд» містить єдиний елемент керування — перемикач «Темна тема» типу ToggleButton. При активації перемикача система викликає ThemeService, який через PaletteHelper бібліотеки MaterialDesignThemes змінює базову тему застосунку з «Світлої» на «Темну»

і навпаки. Стан теми зберігається лише на час поточного сеансу та не записується до бази даних.

Картка «База даних» відображає у полі лише для читання повний шлях до файлу бази даних SQLite. Шлях формується динамічно як Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "BloodDonorSystem.db") і вказує на директорію, де розміщено виконуваний файл застосунку. Кнопка «Резервна копія бази даних» відкриває системний діалог вибору місця збереження і копіює файл бази даних до обраної директорії. Після успішного копіювання відображається повідомлення з підтвердженням.

Картка «Про систему» містить статичну довідкову інформацію: повну назву системи, версію (1.0.0) та перелік використаних технологій — .NET 9, WPF, MVVM, SQLite, Material Design.

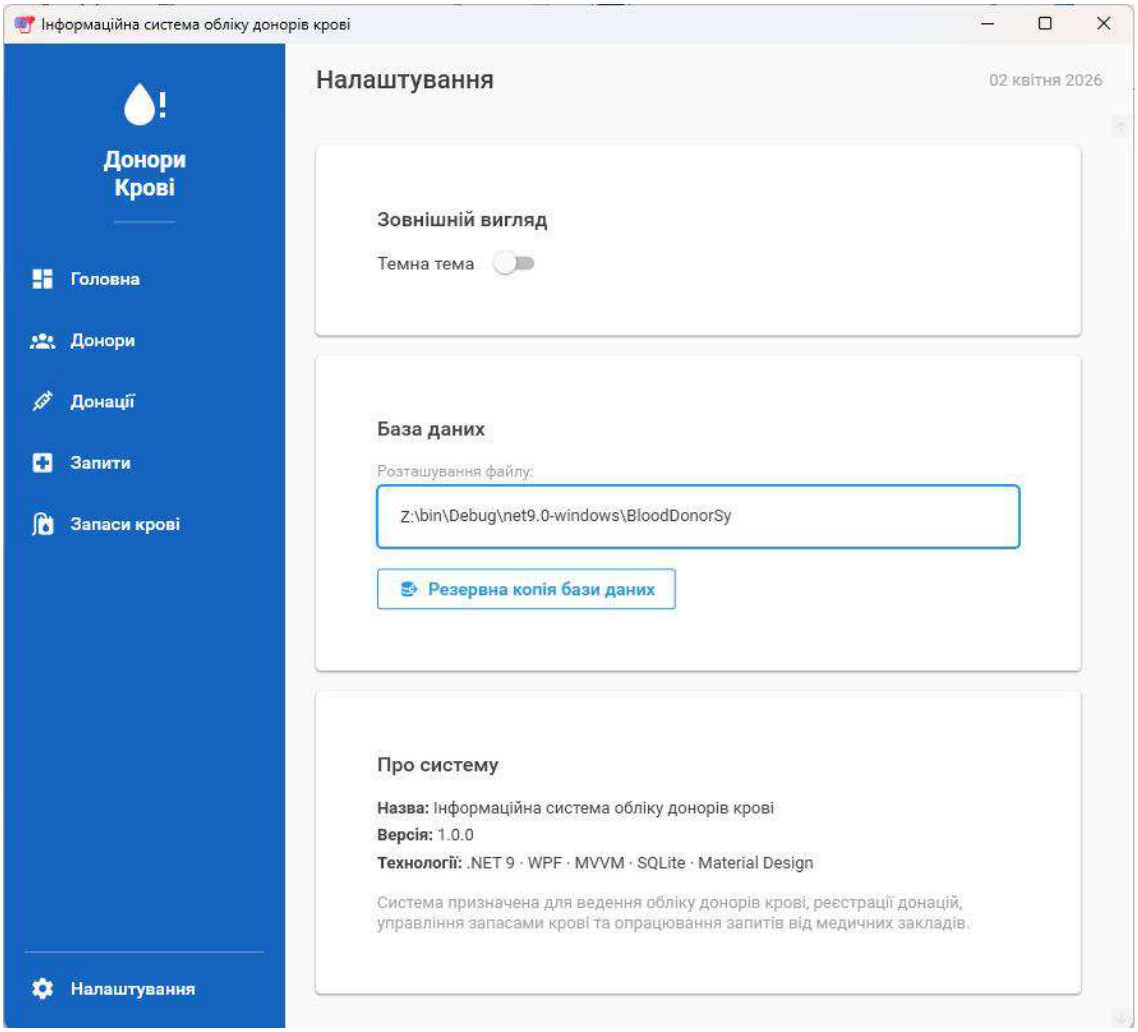

Рисунок 2.2.11. – Форма «Налаштування»

2.3 Архітектура програмного засобу

2.3.1 Загальні архітектурні принципи

Архітектура розробленого програмного засобу базується на поєднанні кількох усталених шаблонів проєктування, що забезпечують чіткий розподіл відповідальностей між компонентами, слабке зчеплення модулів та зручність супроводу й розширення системи. Основним архітектурним шаблоном обрано MVVM (Model-View-ViewModel), який є рекомендованим підходом для WPF-застосунків і органічно підтримується платформою .NET завдяки механізму прив'язки даних. Додатково застосовано шаблон Repository для абстрагування доступу до бази даних, а також шаблон Dependency Injection для управління залежностями між компонентами. Сукупність цих підходів утворює чотирирівневу структуру застосунку, де кожен рівень взаємодіє лише із суміжним, що унеможливлює безпосередній доступ рівня відображення до бази даних і навпаки.

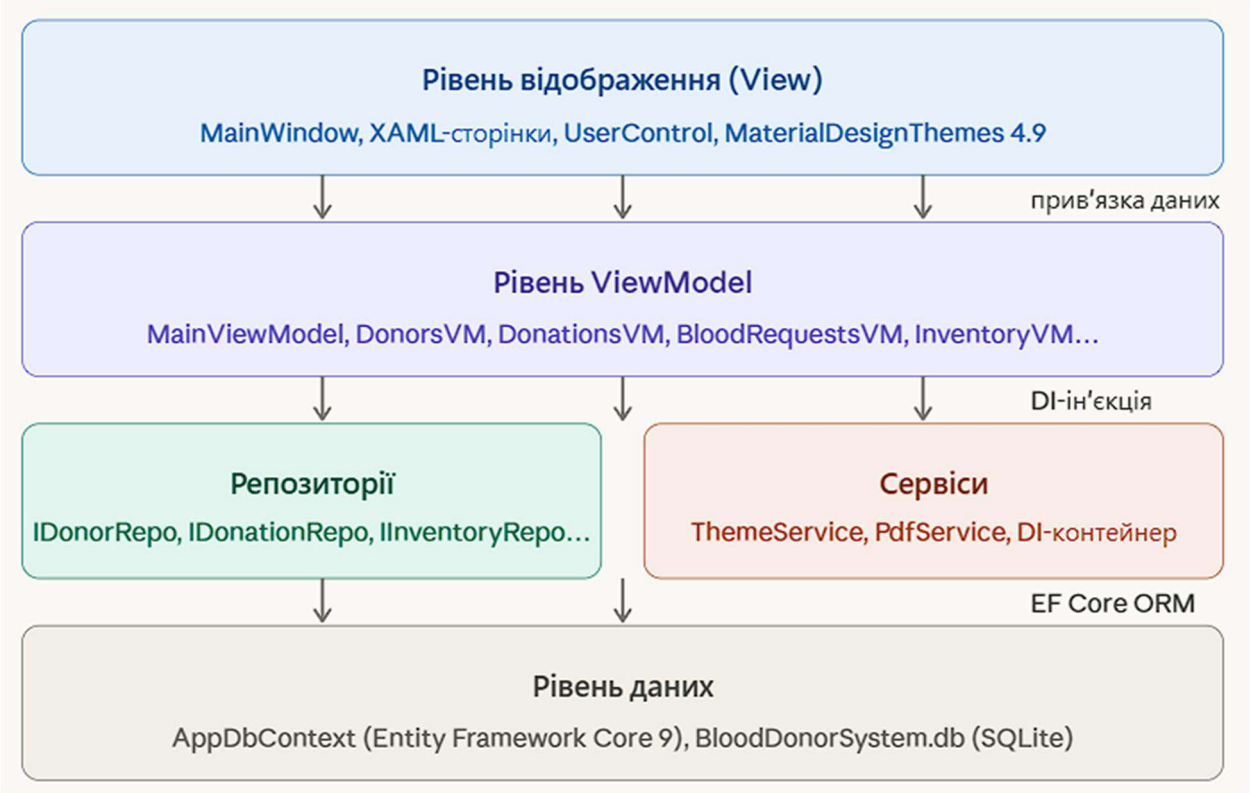

Рисунок 2.3.1. – Рівнева архітектура застосунку

2.3.2 Шаблон MVVM та механізм прив'язки даних

Шаблон MVVM розділяє застосунок на три логічні складові. Компонент Model представлений класами сутностей у просторі імен BloodDonorSystem.Models: Donor, Donation, BloodRequest, BloodInventory та Notification. Ці класи є чистими об'єктами даних (POCO) і не містять жодної логіки відображення. Компонент View представлений XAML-файлами сторінок та головного вікна, які декларативно описують інтерфейс без жодного коду бізнес-логіки у відповідних файлах *.xaml.cs. Компонент ViewModel є сполучною ланкою між двома попередніми і реалізований у шести класах сторінок та одному кореневому MainViewModel.

Усі класи ViewModel успадковують від ObservableObject бібліотеки CommunityToolkit.Mvvm, що надає реалізацію інтерфейсу INotifyPropertyChanged. Властивості, зміни яких повинні відображатися в інтерфейсі, декларуються через атрибут [ObservableProperty] над приватним полем, після чого генератор вихідного коду автоматично створює публічну властивість із викликом OnPropertyChanged. Команди інтерфейсу декларуються через атрибут [RelayCommand] над методом і генерують відповідні об'єкти IRelayCommand, до яких прив'язуються кнопки та інші елементи керування у XAML через атрибут Command. Зворотний зв'язок від ViewModel до View здійснюється виключно через механізм прив'язки даних WPF — ViewModel не має жодного посилання на об'єкти View.

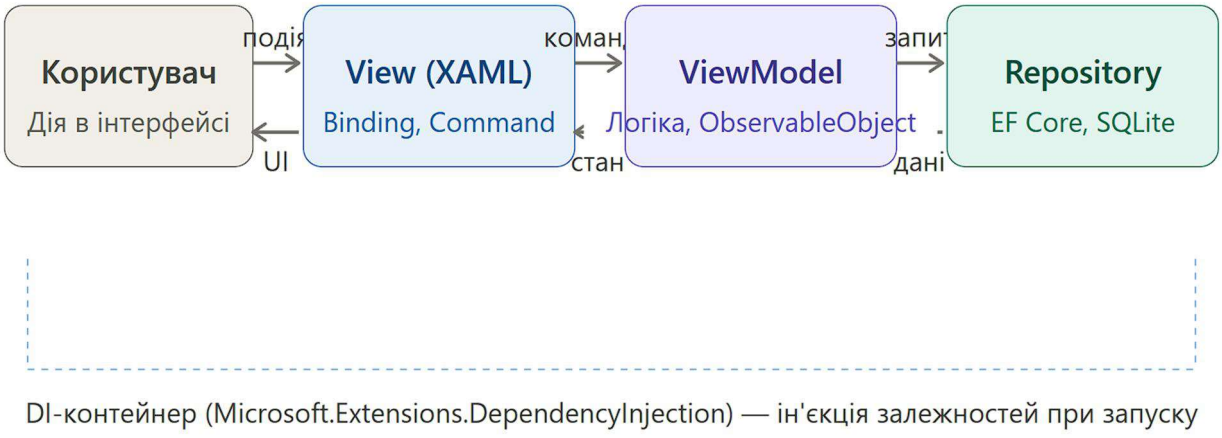

Рисунок 2.3.2. – Потік даних у шаблоні MVVM

2.3.3 Система навігації між сторінками

Навігація між функціональними розділами реалізована через механізм DataTemplate у WPF без використання фреймів (Frame) чи навігаційних сервісів. Клас MainViewModel зберігає властивість CurrentPage типу object, до якої прив'язаний елемент ContentControl у головному вікні. Відповідні DataTemplate у ресурсах MainWindow.xaml декларують, який клас UserControl відображати для кожного типу ViewModel: наприклад, якщо CurrentPage містить об'єкт типу DonorsViewModel, WPF автоматично відображає DonorsPage. Такий підхід дозволяє повністю зберігати стан кожної сторінки між переходами, оскільки всі ViewModel-об'єкти зареєстровані у DI-контейнері як Transient і створюються один раз на час сеансу роботи.

При натисканні кнопки навігаційного меню відповідна [RelayCommand]-команда у MainViewModel присвоює значення CurrentPage потрібному ViewModel-екземпляру та асинхронно викликає метод LoadAsync() для завантаження актуальних даних. Таким чином, перехід до будь-якого розділу завжди ініціює оновлення відображуваних даних, що гарантує актуальність інформації після будь-яких змін, виконаних в інших розділах. Властивість CurrentPageTitle оновлюється одночасно з CurrentPage і прив'язана до заголовка у верхній смузі головного вікна.

2.3.4 Контейнер впровадження залежностей

Керування залежностями між компонентами здійснюється через контейнер IServiceCollection бібліотеки Microsoft.Extensions.DependencyInjection, що є стандартним DI-фреймворком платформи .NET. Конфігурація контейнера зосереджена в методі ConfigureServices класу App і виконується один раз при запуску застосунку. Контейнер реєструє такі компоненти: AppDbContext — з часом існування Transient, що означає створення нового екземпляра контексту для кожного репозиторію (це усуває проблему відстеження сутностей між різними операціями); чотири інтерфейси репозиторіїв разом із їхніми реалізаціями —

також як Transient; IThemeService та ThemeService — як Singleton, оскільки тема є глобальним станом застосунку; усі шість ViewModel-класів сторінок та MainViewModel — як Transient; клас MainWindow — як Transient.

Після побудови контейнера (BuildServiceProvider) клас App запитує екземпляр MainWindow через GetRequiredService<MainWindow>(), і весь ланцюжок залежностей вирішується автоматично: MainWindow отримує MainViewModel, MainViewModel отримує шість ViewModel-об'єктів, кожен ViewModel отримує відповідні репозиторії, кожен репозиторій отримує AppDbContext. Жоден клас у застосунку не створює залежності через оператор new — усі залежності передаються через конструктор, що забезпечує тестованість і слабке зчеплення компонентів.

2.3.5 Шаблон Repository та рівень доступу до даних

Рівень доступу до бази даних реалізований через чотири інтерфейси репозиторіїв: IDonorRepository, IDonationRepository, IBloodRequestRepository та IBloodInventoryRepository. Кожен інтерфейс визначає набір асинхронних методів для виконання CRUD-операцій відповідно до потреб конкретного модуля. Реалізації інтерфейсів у класах DonorRepository, DonationRepository тощо отримують екземпляр AppDbContext через конструктор і виконують запити засобами LINQ to Entities.

Усі операції читання виконуються з розширенням AsNoTracking(), що вимикає механізм відстеження змін Entity Framework Core для прочитаних об'єктів. Це запобігає виникненню помилки дублювання відстежуваних сутностей (InvalidOperationException: A TwoWay or OneWayToSource binding...), яка виникала б при спробі оновити клон об'єкта, якщо оригінал вже відстежується тим самим контекстом. Операції оновлення реалізовані через патерн FindAsync + копіювання полів: метод UpdateAsync спочатку завантажує відстежуваний об'єкт із бази даних за первинним ключем, після чого копіює значення полів із переданого DTO і зберігає зміни через SaveChangesAsync. Такий підхід гарантує коректну роботу механізму

відстеження змін EF Core незалежно від того, чи прив'язаний переданий об'єкт до будь-якого контексту.

Метод GetAvailableAsync репозиторію BloodInventoryRepository має особливе значення для бізнес-логіки виконання запитів: він повертає лише придатні, непротерміновані записи конкретної групи та резус-фактора, відсортовані за терміном придатності за зростанням (FIFO). Саме цей порядок сортування забезпечує правильну послідовність списання запасів при виконанні запиту від медичного закладу.

2.3.6 Сервісний рівень

Поряд із репозиторіями в архітектурі системи виділено два сервісні класи, що реалізують функціональність, не пов'язану безпосередньо з операціями над базою даних.

ThemeService реалізує інтерфейс IThemeService і забезпечує перемикання між світлою та темною темами застосунку в реальному часі. Сервіс використовує клас PaletteHelper бібліотеки MaterialDesignThemes 4.9, який надає доступ до поточної теми через метод GetTheme(). Зміна теми виконується через виклик theme.SetBaseTheme(dark ? Theme.Dark : Theme.Light) та застосовується до всього застосунку через paletteHelper.SetTheme(theme) без перезавантаження вікон.

PdfService є статичним класом, що реалізує формування PDF-документів двох типів: списку донорів для оповіщення при опрацюванні запиту (GenerateDonorListPdf) та накладної на видачу крові при виконанні запиту (GenerateInvoicePdf). Обидва методи використовують бібліотеку QuestPDF 2024.10.4 з Community-ліцензією. Перед збереженням файлу відкривається системний діалог SaveFileDialog, що дозволяє користувачеві обрати місце збереження. Сформований документ автоматично відкривається засобами операційної системи через Process.Start після успішного збереження. Клас PdfService не реєструється в DI-контейнері, оскільки є статичним і не має власних залежностей, що потребували б впровадження.

2.3.7 Ініціалізація та запуск застосунку

Точкою входу в застосунок є клас App, успадкований від Application. Метод OnStartup виконується замість стандартного запуску через StartupUri (який видалений з App.xaml). Послідовність ініціалізації охоплює чотири кроки. По-перше, встановлюється українська культура uk-UA для поточного потоку та всіх елементів WPF через FrameworkElement.LanguageProperty.OverrideMetadata, що забезпечує правильну локалізацію компонентів DatePicker та форматування дат. По-друге, формується і будується DI-контейнер. По-третє, через окремий scope ініціалізується база даних: метод DbInitializer.InitializeAsync викликає EnsureCreatedAsync для автоматичного створення схеми SQLite і, якщо таблиця Donors порожня, заповнює її тестовими даними — двадцятьма донорами з реальними українськими іменами, містами та медичними показниками. По-четверте, з контейнера запитується та відображається MainWindow, після чого асинхронно викликається InitializeAsync для переходу на головну сторінку та завантаження початкових даних.

Рисунок 2.3.3. –Послідовність ініціалізації застосунку

2.3.8 Структура проєкту

Файлова структура проєкту організована у відповідності до функціональних рівнів архітектури. Кореневий каталог містить файли конфігурації проєкту (BloodDonorSystem.csproj), точку входу (App.xaml та App.xaml.cs) та головне вікно. Кожна архітектурна складова відокремлена у власній директорії: Models містить класи сутностей, Data — клас AppDbContext та DbInitializer, Repositories — інтерфейси та реалізації репозиторіїв, ViewModels — сім класів ViewModel, Views — клас головного вікна та піддиректорію Pages із шістьма класами сторінок, Services — ThemeService та PdfService, Helpers — конвертери значень та допоміжні класи обчислення дат, Themes — словник ресурсів конвертерів.

Такий поділ відповідає принципу єдиної відповідальності і спрощує орієнтацію в кодовій базі: зміни у логіці доступу до даних локалізовані виключно у директорії Repositories, зміни в бізнес-логіці — у ViewModels, а зміни у відображенні — у Views. Взаємодія між рівнями здійснюється виключно через інтерфейси та механізм DI, тому заміна будь-якої реалізації (наприклад, перехід з SQLite на інший СКБД) вимагає зміни лише у відповідному класі репозиторію та реєстрації у DI-контейнері без модифікації ViewModel-рівня.

2.4 Програмна реалізація

2.4.1 Реалізація моделей даних

Модельний рівень застосунку реалізований у файлі Models/Models.cs і містить п'ять класів-сутностей, що відображають предметну галузь системи. Усі класи є стандартними POCO-об'єктами без жодної залежності від інфраструктурних бібліотек. Клас Donor є центральним у моделі і містить 19 властивостей, серед яких дві є обчислюваними: FullName повертає повне ім'я донора конкатенацією трьох рядкових полів, а Age обчислює поточний вік на основі дати народження з урахуванням того, чи відбувся вже день народження

у поточному році. Навігаційні властивості Donations та Notifications типу ICollection<T> дозволяють Entity Framework Core автоматично формувати SQL-запити з JOIN при явному виклику Include().

```
public string FullName => $"{LastName} {FirstName} {MiddleName}".Trim();
public int Age => DateTime.Today.Year - DateOfBirth.Year -
    (DateTime.Today < DateOfBirth.AddYears(
        DateTime.Today.Year - DateOfBirth.Year) ? 1 : 0);
```

Клас BloodInventory зберігає кожну заготовлену одиницю компонента крові як окремий запис, що є свідомим архітектурним рішенням. Такий підхід дозволяє реалізувати принцип FIFO при списанні: оскільки кожна партія має власний термін придатності, їх можна відсортувати за полем ExpiryDate і споживати у порядку зростання цього значення. Поле Status зберігається явно, а не обчислюється, що дозволяє відображати його в таблиці без додаткових запитів до бази даних. Клас BloodRequest не має зовнішніх ключів до жодної іншої таблиці — зв'язок із запасами є суто логічним і реалізується на рівні бізнес-логіки у BloodRequestsViewModel.

2.4.2 Конфігурація Entity Framework Core

Клас AppDbContext успадковує від DbContext і визначає п'ять властивостей типу DbSet<T>, кожна з яких відповідає окремій таблиці SQLite. Шлях до файлу бази даних формується динамічно у статичній властивості та вказує на директорію поруч із виконуваним файлом застосунку:

```
private static string DbPath =>
    Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
        "BloodDonorSystem.db");

protected override void OnConfiguring(DbContextOptionsBuilder o)
    => o.UseSqlite($"Data Source={DbPath}");
```

Метод OnModelCreating явно конфігурує два зовнішні ключі з каскадним видаленням: при видаленні донора автоматично видаляються всі пов'язані донації та сповіщення. Це реалізується через OnDelete(DeleteBehavior.Cascade) і відповідає вимозі референційної цілісності

бази даних. Entity Framework Core на основі цієї конфігурації та атрибуту [Required] автоматично генерує правильну DDL-схему SQLite при першому запуску через EnsureCreatedAsync.

Статичний клас DbInitializer відповідає за початкове заповнення бази даних тестовими даними. Метод InitializeAsync перевіряє, чи вже містить таблиця Donors записи, і якщо ні — додає 20 донорів з реальними українськими прізвищами та іменами, містами восьми регіонів України, різними групами крові та статусами. Для кожного донора, який вже здавав кров, автоматично створюється відповідний запис у таблиці Donations. Додатково заповнюються таблиці BloodRequests та BloodInventory з тестовими даними, що дозволяє одразу продемонструвати всі функції системи без ручного введення даних.

2.4.3 Реалізація репозиторіїв

Кожен репозиторій реалізує відповідний інтерфейс і отримує AppDbContext через конструктор. Ключовою особливістю реалізації є застосування AsNoTracking() до всіх запитів на читання, що запобігає кешуванню прочитаних об'єктів у пам'яті контексту. Завдяки цьому об'єкт, прочитаний у методі списку, може бути клонований у ViewModel без конфліктів із механізмом відстеження EF Core.

Метод UpdateAsync у всіх репозиторіях реалізований за патерном «знайди та оновлюй», що є надійним способом уникнення помилки InvalidOperationException про дублювання відстежуваних сутностей:

```
public async Task UpdateAsync(Donor donor)
{
    var tracked = await _db.Donors.FindAsync(donor.Id);
    if (tracked == null) return;
    tracked.LastName   = donor.LastName;
    tracked.FirstName  = donor.FirstName;
    // ... копіювання решти полів
    await _db.SaveChangesAsync();
}
```

Метод FindAsync звертається до кешу контексту або виконує SELECT за первинним ключем і повертає вже відстежуваний об'єкт. Поля переписуються безпосередньо в цей відстежуваний об'єкт, після чого SaveChangesAsync генерує UPDATE-запит лише для полів, значення яких змінились. Такий підхід є безпечним і ефективним порівняно з _db.Update(donor), який намагається прикріпити до контексту сторонній об'єкт і може конфліктувати з вже відстежуваними екземплярами.

Репозиторій BloodInventoryRepository містить спеціалізований метод GetAvailableAsync, що відіграє ключову роль у логіці виконання запитів:

```
public async Task<List<BloodInventory>> GetAvailableAsync(
    string bloodGroup, string rhFactor) =>
    await _db.BloodInventory
        .Where(i => i.BloodGroup == bloodGroup
                 && i.RhFactor   == rhFactor
                 && i.Status     == "Придатна"
                 && i.Quantity   > 0
                 && i.ExpiryDate >= DateTime.Today)
        .OrderBy(i => i.ExpiryDate)
        .ToListAsync();
```

Сортування за ExpiryDate у порядку зростання забезпечує реалізацію принципу FIFO на рівні запиту до бази даних: при ітерації отриманого списку перші елементи завжди матимуть найкоротший термін придатності і будуть списані першими.

2.4.4 Реалізація ViewModel-рівня

Клас DonorsViewModel демонструє типовий підхід до реалізації ViewModel у системі. Властивості, прив'язані до елементів інтерфейсу, декларуються через [ObservableProperty], що генерує публічні властивості з автоматичним повідомленням про зміни. Команди декларуються через [RelayCommand] над асинхронними методами. Для реагування на зміну вибраного рядка в таблиці донорів перевизначається частковий метод OnSelectedDonorChanged, що генерується CommunityToolkit.Mvvm:

```
partial void OnSelectedDonorChanged(Donor? value)
{
    if (value != null)
        _ = LoadDonorHistoryAsync(value.Id);
    else
    {
        SelectedDonorDonations.Clear();
        HistoryStatus =
            "Оберіть донора для перегляду історії донацій";
    }
}
```

Клас DonationsViewModel реалізує автоматичне оновлення списку донацій при зміні фільтрів дати через аналогічний механізм часткових методів. Крім того, метод Save реалізує складнішу бізнес-логіку: при реєстрації нової донації він не лише зберігає запис у таблицю Donations, але й виконує дві додаткові операції — оновлює агреговані поля LastDonationDate та DonationCount у записі донора, а також автоматично створює відповідний запис у таблиці BloodInventory:

```
var shelf = ShelfDays.GetValueOrDefault(component, 42);

await _inventoryRepo.AddAsync(new BloodInventory
{
    BloodGroup     = donorBloodGroup,
    RhFactor       = donorRhFactor,
    Component      = component,
    Quantity       = EditDonation.Volume,
    ProductionDate = EditDonation.DonationDate,
    ExpiryDate     = EditDonation.DonationDate.AddDays(shelf),
    Status         = "Придатна"
});
```

Термін придатності обчислюється на основі словника ShelfDays, що відображає тип компонента на кількість діб зберігання відповідно до

медичних стандартів: цільна кров та еритроцити — 42 доби, плазма — 365 діб, тромбоцити — 5 діб.

2.4.5 Алгоритм виконання запиту та FIFO-списання

Метод FulfillRequest у BloodRequestsViewModel є найбільш складним алгоритмом у системі. Він реалізує транзакційну операцію списання запасів за принципом FIFO і складається з чотирьох послідовних кроків.

На першому кроці виконується перевірка достатності запасів: метод GetAvailableAsync повертає список придатних одиниць потрібної групи та резус-фактора, відсортованих за терміном придатності. Якщо загальний об'єм менший за потрібний, операція переривається з відповідним повідомленням.

На другому кроці виконується ітераційне списання по одиницях із найближчим терміном придатності:

```
int remaining = SelectedRequest.Volume;
foreach (var item in stock)
{
    if (remaining <= 0) break;
    int take = Math.Min(item.Quantity, remaining);
    remaining -= take;

    invoiceItems.Add(new InvoiceItem(
        item.BloodGroup, item.RhFactor,
        item.Component, take,
        item.ProductionDate, item.ExpiryDate));

    if (item.Quantity - take == 0)
        await _inventoryRepo.DeleteAsync(item.Id);
    else
    {
        item.Quantity -= take;
        await _inventoryRepo.UpdateAsync(item);
    }
}
```

Якщо весь об'єм одиниці було використано, запис видаляється з бази даних. Якщо частково — об'єм зменшується і запис оновлюється. Список invoiceItems накопичує деталі кожного списаного елемента для подальшого включення у накладну.

На третьому кроці статус запиту змінюється на «Виконаний» і зберігається через UpdateAsync. На четвертому кроці викликається PdfService.GenerateInvoicePdf для формування накладної.

2.4.6 Реалізація сервісу генерації PDF

Клас PdfService містить два публічні статичні методи для генерації PDF-документів засобами бібліотеки QuestPDF 2024.10.4. Ліцензія Community встановлюється в статичному конструкторі при першому зверненні до класу.

Метод GenerateDonorListPdf формує звіт зі списком відповідних донорів для обраного запиту. Документ структурований за допомогою Fluent API бібліотеки QuestPDF: через Document.Create будується сторінка формату A4 з полями 2 см, власними стилями тексту шрифтом Arial. Вміст складається з картки деталей запиту, таблиці донорів із восьма колонками та підсумкового рядка. Таблиця донорів генерується в циклі з чергуванням кольорів рядків для покращення читабельності.

Метод GenerateInvoicePdf формує офіційну накладну на видачу крові. Документ містить шапку із заголовком та номером накладної, блок реквізитів відправника й отримувача, зведений рядок із характеристиками виданої крові, таблицю виданих компонентів та блок підписів:

```
col.Item().PaddingTop(20).Row(row =>
{
    void Sig(RowDescriptor r, string role)
    {
        r.RelativeItem().Column(c =>
        {
            c.Item().Text(role).FontSize(9).FontColor("#888");
            c.Item().PaddingTop(16)
                    .BorderBottom(1).BorderColor("#999").Text("");
            c.Item().PaddingTop(2)
                    .Text("(підпис / ПІБ)").FontSize(8)
                    .FontColor("#AAA");
        });
    }
    Sig(row, "Відпустив:");
    row.ConstantItem(24);
    Sig(row, "Прийняв:");
    row.ConstantItem(24);
    Sig(row, "Лікар:");
});
```

Перед збереженням обидва методи відкривають системний діалог SaveFileDialog з попередньо сформованим ім'ям файлу, що включає групу крові та поточну дату. Після успішного збереження документ відкривається у програмі перегляду PDF за замовчуванням через Process.Start з параметром UseShellExecute = true.

2.4.7 Реалізація перетворювачів значень

Файл Helpers/Converters.cs містить три реалізації інтерфейсу IValueConverter, що використовуються у XAML-прив'язках. InverseBoolToVisibilityConverter перетворює булеве значення у Visibility з інверсією: true → Collapsed, false → Visible. Цей конвертер застосовується для відображення або приховування таблиці при перемиканні у режим форми редагування. NullToVisibilityConverter перетворює null та порожній рядок у Collapsed, а будь-яке непусте значення — у Visible, що використовується для умовного показу панелі деталей вибраного запису та інформаційних повідомлень. IdToTitleConverter перетворює числове значення ідентифікатора: 0 відповідає режиму додавання нового запису і повертає рядок «Новий донор»,

будь-яке інше значення відповідає режиму редагування і повертає «Редагування донора».

Усі конвертери реєструються як статичні ресурси у файлі Themes/Converters.xaml і підключаються до кожної сторінки через ResourceDictionary. Стандартний конвертер WPF BooleanToVisibilityConverter також доступний через ресурси MaterialDesignThemes і використовується для прямого перетворення булевих значень у видимість елементів.

2.4.8 Допоміжна логіка обчислення дат донацій

Клас DonationHelper у просторі імен BloodDonorSystem.Helpers реалізує бізнес-правило визначення мінімального інтервалу між донаціями. Статичний метод GetNextDonationDate приймає тип донації та дату останньої і повертає найранішу дату наступної допустимої донації:

```
public static DateTime GetNextDonationDate(
    string donationType, DateTime lastDate) =>
    donationType switch
    {
        "Цільна кров"  => lastDate.AddDays(60),
        "Плазма"       => lastDate.AddDays(14),
        "Тромбоцити"   => lastDate.AddDays(14),
        "Еритроцити"   => lastDate.AddDays(90),
        _              => lastDate.AddDays(60)
    };
```

Клас BloodCompatibility реалізує таблицю сумісності груп крові. Метод GetCompatibleDonorGroups повертає список груп крові донорів, яких можна використати для задоволення запиту на певну групу реципієнта. Логіка заснована на медичних стандартах сумісності: наприклад, для реципієнта з групою AB+ підходять донори всіх восьми груп, а для реципієнта з групою 0− підходять лише донори групи 0−. Цей метод використовується у BloodRequestsViewModel при пошуку відповідних донорів для опрацювання запиту.

2.5 Тестування програми

Напиши висновок по тестуванню програми. 20 речень. Напиши академічно, без списків та нумерації.

У процесі розробки інформаційної системи обліку донорів крові було проведено комплексне тестування всіх функціональних модулів застосунку з метою перевірки відповідності реалізованого програмного продукту висунутим вимогам. Тестування охоплювало функціональну перевірку кожного розділу системи, перевірку коректності роботи з базою даних, а також контроль правильності виконання ключових бізнес-алгоритмів. За результатами проведеного тестування встановлено, що всі основні функції системи працюють коректно відповідно до задуманої логіки та проєктних вимог.

Модуль управління донорами успішно пройшов перевірку операцій додавання, редагування та видалення записів, а фільтрація за групою крові, статусом та текстовий пошук за прізвищем і номером телефону повертають очікувані результати. Виявлена у процесі тестування помилка дублювання відстежуваних сутностей Entity Framework Core при збереженні відредагованого запису була усунута шляхом переходу до патерну FindAsync з ручним копіюванням полів у методі UpdateAsync всіх репозиторіїв. Коректність автоматичного завантаження історії донацій при виборі рядка в таблиці донорів підтверджено для записів із різною кількістю пов'язаних донацій, включаючи донорів без жодної донації, у яких відображається відповідне інформаційне повідомлення.

Під час тестування модуля реєстрації донацій було підтверджено автоматичне оновлення двох залежних таблиць при збереженні нового запису: поля DonationCount та LastDonationDate у таблиці Donors оновлюються коректно, а до таблиці BloodInventory додається новий запис із правильно обчисленим терміном придатності відповідно до типу компонента. Автоматичне оновлення списку при зміні дат фільтрів через механізм

часткових методів OnFilterFromChanged та OnFilterToChanged працює без затримок і не потребує додаткового підтвердження від користувача.

Алгоритм FIFO-списання запасів при виконанні запиту на кров протестовано на сценаріях із різними комбінаціями об'ємів і кількістю записів у таблиці запасів. Підтверджено, що при наявності кількох записів відповідної групи крові система завжди обирає одиниці з найближчим терміном придатності, а при частковому використанні запасу коректно зменшує поле Quantity замість видалення запису. Перевірку блокування кнопки «Виконати запит» при недостатньому об'ємі запасів проведено для граничних значень, включаючи ситуації рівності наявного та потрібного об'ємів.

Генерація PDF-документів обох типів — списку донорів та накладної на видачу крові — перевірена на коректність формування структури документа, відображення всіх необхідних полів та правильність розрахунків підсумкових значень. Збереження файлу через системний діалог і автоматичне відкривання у програмі перегляду після збереження функціонує коректно на цільовій операційній системі Windows 10 та Windows 11.

Локалізація інтерфейсу перевірена на коректність відображення українських назв місяців і днів тижня у компонентах DatePicker після встановлення культури uk-UA глобально через FrameworkElement.LanguageProperty.OverrideMetadata. Перемикання між світлою та темною темами інтерфейсу протестоване на всіх сторінках застосунку, і встановлено, що всі елементи коректно адаптують своє забарвлення завдяки використанню динамічних ресурсів MaterialDesignThemes замість статично заданих кольорів. Резервне копіювання бази даних через форму налаштувань успішно копіює файл BloodDonorSystem.db до обраної директорії зі збереженням цілісності даних.

За підсумками тестування підтверджено, що розроблений програмний засіб відповідає всім функціональним вимогам, сформованим на етапі проєктування, забезпечує цілісність даних при виконанні складних

транзакційних операцій і може бути рекомендований до практичного впровадження у роботу служби крові.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи виконано комплекс проєктувальних і реалізаційних завдань, результатом яких стала повнофункціональна інформаційна система обліку донорів крові, побудована на сучасному технологічному стеці .NET 9 і WPF.

На етапі моделювання бази даних визначено п'ять основних сутностей предметної галузі — Donors, Donations, BloodRequests, BloodInventory та Notifications — і встановлено зв'язки між ними. Реляційна схема побудована з дотриманням принципів нормалізації, референційна цілісність забезпечується зовнішніми ключами з каскадним видаленням, а свідома часткова денормалізація у вигляді агрегованих полів DonationCount та LastDonationDate обґрунтована вимогами продуктивності інтерфейсу. Фізична модель реалізована засобами Entity Framework Core із підходом Code First, що дозволило автоматично генерувати схему SQLite при першому запуску застосунку без окремих міграційних сценаріїв.

Архітектура програмного засобу побудована на основі шаблону MVVM, що забезпечує повне розділення логіки відображення та бізнес-логіки. Застосування шаблону Repository абстрагує доступ до бази даних і локалізує всі SQL-взаємодії в окремому рівні, незалежному від ViewModel-рівня. Контейнер впровадження залежностей Microsoft.Extensions.DependencyInjection вирішує всі залежності між компонентами автоматично при запуску і унеможливлює жорстке зчеплення між рівнями архітектури. Така організація коду відповідає принципу єдиної відповідальності та забезпечує зручність супроводу й подальшого розширення системи.

У процесі програмної реалізації вирішено низку практичних інженерних задач. Проблему дублювання відстежуваних сутностей Entity Framework Core

усунуто через патерн FindAsync з ручним копіюванням полів у всіх методах UpdateAsync. Автоматичне оновлення запасів крові при реєстрації донації реалізовано транзакційно в методі збереження DonationsViewModel із коректним обчисленням терміну придатності за типом компонента. Алгоритм FIFO-списання запасів при виконанні запиту на кров реалізовано через сортування відповідних записів BloodInventory за полем ExpiryDate у порядку зростання з подальшою ітераційною деструкцією. Генерація PDF-документів двох типів реалізована засобами бібліотеки QuestPDF 2024.10.4 із застосуванням Fluent API для декларативного опису структури документів.

Тестування підтвердило коректність роботи всіх функціональних модулів системи: управління донорами, реєстрації донацій, опрацювання запитів на кров, управління запасами та формування PDF-документів. Усі виявлені в процесі розробки дефекти усунуто до завершення розділу. Результатом виконаної роботи є готовий до практичного використання програмний засіб, що реалізує повний цикл автоматизації діяльності служби крові відповідно до сформованих вимог.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1 Мінімальні вимоги до системи

Розроблений програмний засіб є настільним Windows-застосунком, побудованим на платформі .NET 9 із використанням технології WPF, що обумовлює певні вимоги до апаратного та програмного забезпечення комп'ютера, на якому планується його експлуатація. Дотримання наведених нижче вимог гарантує стабільну та коректну роботу всіх функціональних модулів системи.

З точки зору операційної системи застосунок функціонує виключно в середовищі Microsoft Windows, оскільки технологія WPF є Windows-ексклюзивною і не підтримується на платформах macOS чи Linux. Мінімально підтримуваною версією операційної системи є Windows 10 версії 1903 (збірка 18362) або новіша, включаючи Windows 11 усіх актуальних версій. Рекомендованою версією є Windows 10 22H2 або Windows 11 23H2 із встановленими останніми оновленнями безпеки, що забезпечує сумісність із середовищем виконання .NET 9.

Обов'язковою умовою для запуску застосунку є встановлення середовища виконання .NET 9 Desktop Runtime для архітектури x64. Без цього компонента запуск виконуваного файлу неможливий, оскільки застосунок скомпільовано під цільову платформу net9.0-windows. Середовище виконання доступне для безкоштовного завантаження з офіційного сайту Microsoft за адресою https://dotnet.microsoft.com/download/dotnet/9.0 і не потребує придбання ліцензії.

Щодо апаратних вимог, мінімально необхідний обсяг оперативної пам'яті становить 512 МБ, однак рекомендованим є обсяг не менше 2 ГБ для комфортної паралельної роботи з іншими застосунками. Процесор має підтримувати архітектуру x64 із тактовою частотою не нижче 1 ГГц; рекомендованою є частота від 2 ГГц і вище. Для зберігання виконуваних файлів застосунку та бази даних SQLite необхідно не менше 100 МБ вільного дискового простору, хоча в разі активного тривалого використання системи з

великими обсягами донацій та запасів крові рекомендується резервувати до 500 МБ з урахуванням зростання розміру файлу бази даних і збережених PDF-документів.

Мінімальна підтримувана роздільна здатність екрана становить 1024 × 768 пікселів, однак за такої роздільної здатності частина інтерфейсу може відображатися з горизонтальною смугою прокрутки. Оптимальною є роздільна здатність 1366 × 768 пікселів і вище, за якої головне вікно розміром 1280 × 768 пікселів відображається повністю без прокрутки. Для роботи з функцією генерації та відкривання PDF-документів на комп'ютері має бути встановлена будь-яка програма перегляду PDF-файлів, що зареєстрована в операційній системі як обробник файлів із розширенням .pdf, наприклад Adobe Acrobat Reader, Foxit PDF Reader або вбудований переглядач Microsoft Edge. Підключення до мережі Інтернет для роботи застосунку не потрібне, оскільки база даних SQLite зберігається локально поруч із виконуваним файлом і жодних зовнішніх сервісів система не використовує.

3.2 Інструкція для роботи з програмою

3.2.1 Запуск застосунку та початок роботи

Перед першим запуском необхідно переконатися, що на комп'ютері встановлено середовище виконання .NET 9 Desktop Runtime для архітектури x64. Для запуску застосунку слід двічі клацнути лівою кнопкою миші на файлі BloodDonorSystem.exe, розміщеному у директорії встановлення. При першому запуску система автоматично створює файл бази даних BloodDonorSystem.db у тій самій директорії, де знаходиться виконуваний файл, та заповнює його початковими демонстраційними даними — двадцятьма донорами, кількома донаціями, запитами на кров та записами запасів крові. Жодних додаткових дій для ініціалізації від користувача не вимагається.

Після запуску відображається головне вікно застосунку розміром 1280 × 768 пікселів, розділене на дві зони: ліворуч — вертикальна навігаційна панель синього кольору з піктограмами та назвами розділів, праворуч — основна робоча область із вмістом обраного розділу. У верхній частині робочої зони

завжди відображається назва поточного розділу та поточна дата у форматі «дд місяць рррр» українською мовою. За замовчуванням при запуску відкривається розділ «Головна» із зведеними статистичними показниками.

3.2.2 Навігація між розділами

Для переходу між розділами необхідно натиснути відповідний пункт у навігаційному меню лівої панелі. Система містить шість основних розділів: «Головна», «Донори», «Донації», «Запити», «Запаси крові» та «Налаштування». При кожному переході до розділу дані автоматично оновлюються із бази даних, тому після внесення змін в одному розділі вони будуть відображатися в інших після переходу до них.

3.2.3 Робота з розділом «Головна»

Розділ «Головна» призначений для швидкого перегляду поточного стану системи і не потребує жодного введення даних. Після переходу до розділу автоматично відображаються чотири інформаційні картки у верхній частині екрана: кількість активних донорів, кількість донацій за поточний день, кількість донацій за поточний місяць та кількість активних запитів на кров. Нижче відображається панель зведених запасів крові з об'ємами у мілілітрах у розрізі груп крові — лише для компонентів зі статусом «Придатна». Всі показники обчислюються автоматично і не потребують ручного оновлення.

3.2.4 Робота з розділом «Донори»

Перегляд та пошук донорів. При відкритті розділу таблиця відображає всіх зареєстрованих донорів, відсортованих за прізвищем. Для фільтрації списку використовується панель інструментів у верхній частині. Поле «Пошук» приймає частину прізвища, імені, по батькові або номер телефону — для виконання пошуку після введення тексту необхідно натиснути клавішу Enter або кнопку «Знайти». Випадаючий список «Група» дозволяє відфільтрувати донорів за групою крові (значення «Всі» знімає фільтр). Випадаючий список «Статус» дозволяє відобразити лише донорів з певним

статусом: «Активний», «Тимчасово відсторонений» або «Неактивний». Фільтри можна комбінувати між собою.

У таблиці донорів рядки підсвічуються кольором залежно від статусу: жовтий фон позначає тимчасово відсторонених, сірий текст — неактивних донорів. При натисканні на рядок у нижній частині екрана автоматично відкривається панель «Історія донацій вибраного донора» зі списком усіх донацій обраного донора.

Додавання нового донора. Для додавання нового донора необхідно натиснути кнопку «Додати» (синя кнопка з піктограмою «+») на панелі інструментів. Відкриється форма введення з такими полями:

- «Прізвище» та «Ім'я» — обов'язкові текстові поля; без їх заповнення збереження неможливе;
- «По батькові» — необов'язкове текстове поле;
- «Дата народження» — вибір через календарний компонент; для відкриття календаря слід клацнути на поле або піктограму календаря праворуч;
- «Стать» — вибір зі списку «Чоловіча» або «Жіноча»;
- «Група крові» — вибір зі списку восьми значень: A+, A−, B+, B−, AB+, AB−, 0+, 0−;
- «Резус-фактор» — вибір зі списку «Позитивний» або «Негативний»;
- «Статус» — вибір зі списку «Активний», «Тимчасово відсторонений» або «Неактивний»;
- «Телефон» — текстове поле, рекомендований формат +380XXXXXXXXX;
- «Email» — текстове поле для адреси електронної пошти;
- «Місто» та «Область» — текстові поля;
- «Адреса» — текстове поле для повної поштової адреси;
- «Протипоказання» — текстове поле для медичних протипоказань;
- «Примітки» — багаторядкове текстове поле для довільних нотаток.

Після заповнення полів слід натиснути кнопку «Зберегти» для збереження запису або «Скасувати» для відмови від введення. Після збереження система повертається до таблиці і новий донор відображається у списку.

Редагування запису донора. Для редагування існуючого донора необхідно виділити його рядок у таблиці одним натисканням, після чого натиснути кнопку з піктограмою олівця на панелі інструментів, або виконати подвійне клацання на рядку. Відкриється та сама форма, що і при додаванні, але з уже заповненими полями. Після внесення змін слід натиснути «Зберегти».

Видалення донора. Для видалення слід виділити рядок донора та натиснути кнопку з піктограмою кошика (червона рамка). Видалення відбувається негайно без додаткового підтвердження, тому перед натисканням необхідно переконатися у правильності вибору. При видаленні донора автоматично видаляються всі пов'язані з ним донації та сповіщення.

3.2.5 Робота з розділом «Донації»

Перегляд журналу донацій. Журнал відображає донації за обраний часовий діапазон. За замовчуванням встановлено фільтр «від трьох місяців назад до сьогодні». Для зміни діапазону слід клацнути на поле «Від» або «До» і обрати дату через календар — список донацій оновлюється автоматично одразу після зміни дати без натискання додаткових кнопок. У таблиці відображаються дата донації, повне ім'я та група крові донора, тип донації, об'єм, місце проведення, лікар та стан здоров'я.

Реєстрація нової донації. Для реєстрації нової донації слід натиснути кнопку «Додати» на панелі інструментів. У формі, що відкриється, необхідно:

1. Обрати донора зі списку «Вибрати донора» — список містить усіх зареєстрованих донорів із зазначенням групи крові. Після вибору донора, який вже здавав кров, система автоматично відображає жовтий

інформаційний рядок із датою наступної допустимої донації відповідно до типу попередньої.

2. Вказати дату донації через поле «Дата донації».
3. Обрати тип донації зі списку: «Цільна кров», «Плазма», «Тромбоцити» або «Еритроцити».
4. Вказати об'єм у мілілітрах у полі «Об'єм (мл)» — за замовчуванням встановлено 450 мл.
5. Обрати стан здоров'я зі списку: «Добре», «Задовільно» або «Погано».
6. За необхідності заповнити поля «Місце проведення», «Лікар», «Гемоглобін (г/л)» та «АТ систолічний» — ці поля є необов'язковими.
7. Натиснути «Зберегти».

Після збереження система автоматично оновлює кількість донацій та дату останньої донації у картці відповідного донора, а також додає новий запис до таблиці запасів крові з відповідним компонентом, об'ємом та розрахованим терміном придатності.

Редагування та видалення донації. Для редагування донації слід виділити рядок у таблиці та натиснути кнопку олівця, або виконати подвійне клацання. При редагуванні існуючої донації запаси крові автоматично не перераховуються. Для видалення слід виділити рядок і натиснути кнопку кошика.

3.2.6 Робота з розділом «Запити»

Перегляд списку запитів. Таблиця відображає всі запити на кров від медичних закладів. Рядки підсвічуються кольором: червоний фон позначає запити з терміновістю «Висока», зелений фон — виконані запити, сірий текст — скасовані. Над таблицею відображається інформаційний рядок із даними про доступний об'єм придатних запасів крові відповідної групи для вибраного запиту.

Додавання нового запиту. Для додавання слід натиснути кнопку «Новий запит». У формі необхідно заповнити поля: «Лікарня» (обов'язкове),

«Контактна особа», «Телефон», «Група крові», «Резус-фактор», «Терміновість» («Висока», «Середня» або «Низька»), «Статус» («Активний», «Виконаний» або «Скасований»), «Об'єм (мл)», «Потрібно до» (дата через календар) та «Примітки». Після заповнення слід натиснути «Зберегти».

Пошук відповідних донорів. Для пошуку донорів, сумісних із запитом, необхідно виділити запит у таблиці та натиснути кнопку «Знайти відповідних донорів» на правій панелі. Система відобразить список активних донорів із сумісними групами крові із зазначенням контактних даних. Для формування PDF-списку знайдених донорів слід натиснути кнопку «Сформувати PDF» — відкриється системний діалог збереження файлу, після якого документ буде відкрито автоматично.

Виконання запиту та списання запасів. Для виконання запиту слід виділити його у таблиці. Якщо доступного об'єму придатних запасів достатньо для задоволення запиту, кнопка «Виконати запит» (зелена) стане активною. При натисканні на кнопку система виконує такі дії: списує необхідний об'єм крові з таблиці запасів за принципом FIFO (першими використовуються компоненти з найближчим терміном придатності), змінює статус запиту на «Виконаний» та відкриває діалог збереження накладної PDF. Якщо запасів недостатньо, кнопка залишається неактивною і відображається повідомлення з наявним об'ємом.

Редагування та видалення запиту. Для редагування слід виділити запит і натиснути кнопку олівця або виконати подвійне клацання. Для видалення — виділити запит і натиснути кнопку кошика.

3.2.7 Робота з розділом «Запаси крові»

Перегляд запасів. Таблиця відображає всі записи запасів крові із зазначенням групи крові, резус-фактора, типу компонента, кількості у мілілітрах, дати заготівлі, терміну придатності та статусу. Протерміновані записи підсвічуються червоним фоном і темно-червоним текстом для негайного візуального виявлення.

Запаси поповнюються двома способами: автоматично при реєстрації нової донації у розділі «Донації» та вручну через форму додавання. Ручне додавання призначене для обліку компонентів, отриманих з інших медичних закладів або заготовлених поза основною системою.

Ручне додавання запасу. Для додавання слід натиснути кнопку «Додати запас». У формі необхідно: обрати групу крові та резус-фактор зі списків; обрати тип компонента («Цільна кров», «Еритроцити», «Плазма» або «Тромбоцити»); вказати кількість у мілілітрах; вказати дату заготівлі та термін придатності через поля з календарем; обрати статус («Придатна» або «Протермінована»). Орієнтовні терміни придатності для довідки: цільна кров та еритроцити — 42 доби, плазма — 365 діб, тромбоцити — 5 діб. Після заповнення слід натиснути «Зберегти».

Редагування та видалення запасу. Для редагування запису слід виділити його у таблиці та натиснути кнопку олівця або виконати подвійне клацання. Для видалення — виділити запис і натиснути кнопку кошика. Видалення запасів відбувається також автоматично при виконанні запиту на кров, якщо весь об'єм одиниці було повністю використано.

3.2.8 Робота з розділом «Налаштування»

Перемикання теми інтерфейсу. У картці «Зовнішній вигляд» розміщено перемикач «Темна тема». При переведенні перемикача у положення «Увімкнено» інтерфейс застосунку миттєво змінює кольорову схему на темну. Повторне перемикання повертає світлу тему. Стан теми зберігається лише на час поточного сеансу роботи і при наступному запуску скидається до стандартної світлої теми.

Резервне копіювання бази даних. У картці «База даних» відображається повний шлях до файлу бази даних. Для створення резервної копії слід натиснути кнопку «Резервна копія бази даних» — відкриється системний діалог вибору папки та імені файлу для збереження копії. Після успішного збереження під кнопкою з'явиться зелене підтвердження. Резервне

копіювання рекомендується виконувати регулярно — не рідше одного разу на тиждень — а також перед будь-якими масовими змінами в даних. Для відновлення із резервної копії достатньо замінити файл BloodDonorSystem.db у директорії застосунку збереженою копією при закритому застосунку.

3.2.9 Типові сценарії роботи

Сценарій 1. Реєстрація нового донора та першої донації.

Перейдіть до розділу «Донори» та натисніть «Додати». Заповніть обов'язкові поля «Прізвище», «Ім'я», групу крові та резус-фактор, встановіть статус «Активний». Натисніть «Зберегти». Перейдіть до розділу «Донації» та натисніть «Додати». Оберіть щойно доданого донора зі списку, вкажіть дату, тип донації та об'єм. Натисніть «Зберегти». Після цього у профілі донора автоматично оновляться поля «Кількість донацій» та «Дата останньої донації», а у розділі «Запаси крові» з'явиться новий запис.

Сценарій 2. Опрацювання термінового запиту на кров.

Перейдіть до розділу «Запити» та натисніть «Новий запит». Введіть назву лікарні, вкажіть групу крові «0−», терміновість «Висока» та необхідний об'єм, встановіть дату «Потрібно до». Натисніть «Зберегти». Виділіть створений запит у таблиці — система відобразить наявний об'єм запасів групи 0−. Якщо запасів достатньо, натисніть зелену кнопку «Виконати запит». У діалозі збережіть накладну PDF та передайте документ відповідальній особі. Перейдіть до розділу «Запаси крові» і переконайтесь, що об'єм відповідних записів зменшився.

Сценарій 3. Формування списку донорів для оповіщення.

Перейдіть до розділу «Запити» та виділіть активний запит. На правій панелі натисніть «Знайти відповідних донорів» — система відобразить список активних донорів із сумісними групами крові. Натисніть «Сформувати PDF», вкажіть місце збереження файлу та передайте документ персоналу для зв'язку з донорами.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи викладено повне керівництво користування розробленою інформаційною системою обліку донорів крові, що охоплює технічні вимоги до середовища розгортання та детальні інструкції щодо роботи з кожним функціональним модулем застосунку.

У підрозділі мінімальних вимог до системи визначено перелік апаратних і програмних умов, за яких гарантується коректне функціонування програмного засобу. Встановлено, що єдиною обов'язковою зовнішньою залежністю є середовище виконання .NET 9 Desktop Runtime для архітектури x64, яке доступне для безкоштовного завантаження з офіційного ресурсу Microsoft. Відсутність вимоги до мережевого підключення та використання локальної бази даних SQLite суттєво спрощує розгортання системи і робить її придатною до експлуатації на ізольованих від мережі комп'ютерах медичних установ, що є поширеною практикою в закладах охорони здоров'я України.

Інструкція користувача охоплює повний цикл роботи з системою: від першого запуску та автоматичної ініціалізації бази даних до виконання складних операцій, таких як списання запасів крові при виконанні запиту та формування PDF-документів. Для кожного розділу застосунку детально описано призначення елементів інтерфейсу, допустимі формати введення даних та послідовність дій для виконання типових завдань. Три типові сценарії роботи — реєстрація нового донора з першою донацією, опрацювання термінового запиту на кров та формування списку донорів для оповіщення — демонструють практичне застосування системи в умовах реальної діяльності служби крові і можуть слугувати основою для навчання персоналу під час впровадження.

Викладені матеріали свідчать про те, що розроблений програмний засіб має інтуїтивно зрозумілий інтерфейс, що не потребує спеціальної технічної підготовки від персоналу, а типові операції з донорами, донаціями та запасами крові виконуються за мінімальну кількість дій завдяки автоматизації супутніх процесів на рівні бізнес-логіки системи.

ВИСНОВКИ

У кваліфікаційній роботі виконано проєктування та розробку інформаційної системи обліку донорів крові, що реалізує повний цикл автоматизації діяльності служби крові — від реєстрації донорів і донацій до опрацювання запитів медичних закладів і управління запасами компонентів крові. Досягнення поставленої мети підтверджується послідовним виконанням усіх визначених завдань у межах кожного з розділів роботи.

За результатами аналізу предметної галузі встановлено, що автоматизація обліку донорів крові є актуальною практичною потребою, зумовленою недостатнім рівнем інформатизації регіональних служб крові та зростаючим навантаженням на медичну систему України в умовах збройного конфлікту. Проведений огляд існуючих рішень виявив відсутність доступних спеціалізованих вітчизняних систем, орієнтованих на малі та середні заклади служби крові, що підтвердило доцільність самостійної розробки.

На етапі проєктування розроблено реляційну модель бази даних із п'яти сутностей, сформовано ER-діаграму та фізичну модель на основі СКБД SQLite, визначено зв'язки між таблицями з підтримкою каскадного видалення та референційної цілісності. Архітектура застосунку побудована на поєднанні шаблонів проєктування MVVM, Repository та Dependency Injection, що забезпечило чітке розділення відповідальностей між компонентами, слабке зчеплення рівнів та зручність подальшого супроводу системи.

У процесі програмної реалізації розроблено сім класів ViewModel, чотири репозиторії з повним набором CRUD-операцій, два сервісні класи та набір допоміжних конвертерів значень. Реалізовано алгоритм автоматичного поповнення запасів крові при реєстрації донації з обчисленням терміну придатності за медичними стандартами для кожного типу компонента. Алгоритм FIFO-списання запасів при виконанні запиту на кров забезпечує пріоритетне використання компонентів із найближчим терміном придатності, що мінімізує втрати від протермінування. Реалізовано формування PDF-документів двох типів — списку донорів для оповіщення та накладної на

видачу крові — засобами бібліотеки QuestPDF із коректною структурою, підсумковими полями та блоком підписів відповідальних осіб.

Вирішено ряд практичних інженерних задач, зокрема усунено помилку дублювання відстежуваних сутностей Entity Framework Core через запровадження патерну FindAsync з ручним копіюванням полів, забезпечено локалізацію інтерфейсу українською мовою через глобальне перевизначення культури, реалізовано автоматичне реагування на зміну дат фільтрів без ручного підтвердження.

Проведене тестування підтвердило коректність роботи всіх функціональних модулів системи, відповідність реалізованих алгоритмів бізнес-правилам предметної галузі та стабільність застосунку при виконанні типових і граничних сценаріїв використання. Розроблений програмний засіб відповідає сформованим функціональним і нефункціональним вимогам, має зрозумілий україномовний інтерфейс у стилі Material Design та не потребує спеціальної технічної підготовки від кінцевого користувача.

Практична значущість роботи полягає у тому, що розроблений застосунок є готовим до впровадження програмним продуктом, який може бути використаний у роботі регіональних станцій переливання крові, центрів крові лікарняних закладів або волонтерських організацій донорства крові для підвищення ефективності управлінських процесів і скорочення часу реагування на термінові запити. Подальший розвиток системи може передбачати реалізацію модуля автоматичного надсилання сповіщень донорам засобами електронної пошти або SMS, розробку веб-версії застосунку для забезпечення багатокористувацького доступу, а також інтеграцію з реєстрами медичних закладів для автоматичного імпорту даних про запити на кров.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Про донорство крові та її компонентів : Закон України від 23 черв. 1995 р. № 239/95-ВР : із змінами, внес. згідно із Законом № 2802-IX від 01.12.2022. Відомості Верховної Ради України. 1995. № 23. Ст. 183.
2. Про затвердження Порядку медичного обстеження донорів крові та (або) її компонентів : Наказ Міністерства охорони здоров'я України від 01 серп. 2005 р. № 385. Офіційний вісник України. 2005. № 32.
3. Настанова ВООЗ щодо вимог до служб крові : WHO/EHT/04.08. Женева : Всесвітня організація охорони здоров'я, 2004. 48 с.
4. World Health Organization. Blood Safety and Availability. Geneva : WHO, 2023. URL: https://www.who.int/news-room/fact-sheets/detail/blood-safety-and-availability (дата звернення: 10.02.2026).
5. MacDonald M. Pro WPF 4.5 in C# : Windows Presentation Foundation in .NET 4.5. 4th ed. New York : Apress, 2012. 1094 p.
6. Smith J. P. Entity Framework Core in Action. 2nd ed. Shelter Island : Manning Publications, 2021. 584 p.
7. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.
8. Skeet J. C# in Depth. 4th ed. Shelter Island : Manning Publications, 2019. 528 p.
9. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.
10. Microsoft Corporation. Entity Framework Core Documentation. 2024. URL: https://learn.microsoft.com/en-us/ef/core (дата звернення: 15.02.2025).
11. Microsoft Corporation. .NET 9 Documentation. 2024. URL: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-9/overview (дата звернення: 15.05.2026).
12. Microsoft Corporation. Windows Presentation Foundation Documentation for .NET. 2024. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf (дата звернення: 18.05.2026).

13. Microsoft Corporation. The MVVM Pattern. 2023. URL: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm (дата звернення: 20.05.2026).
14. CommunityToolkit Contributors. CommunityToolkit.Mvvm : Source Generators and Base Classes for MVVM. 2024. URL: https://github.com/CommunityToolkit/dotnet (дата звернення: 22.02.2025).
15. Kozak J., Vykydalova A. Material Design in XAML Toolkit. 2024. URL: https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit (дата звернення: 25.05.2026).
16. QuestPDF Contributors. QuestPDF : Modern .NET Library for PDF Document Generation. 2024. URL: https://www.questpdf.com/documentation (дата звернення: 01.03.2026).
17. SQLite Consortium. SQLite Documentation. 2024. URL: https://www.sqlite.org/docs.html (дата звернення: 05.03.2026).
18. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 533 p.
19. Інформаційні технології. Настанова щодо документування комп'ютерних програм : ДСТУ 3918-99. Київ : Держстандарт України, 1999. 38 с.
20. Бібліографічне посилання. Загальні положення та правила складання : ДСТУ 8302:2015. Київ : ДП «УкрНДНЦ», 2016. 16 с.