Dokumentacja projektu

Bank Krwi

Alicja Borek

Semestr zimowy 2025/2026

1 I. Projekt koncepcji i założenia

1.1 1. Zdefiniowanie tematu projektu

Celem projektu jest stworzenie kompletnego systemu informatycznego wspierającego działanie banku krwi. Aplikacja ma na celu cyfryzację procesów związanych z oddawaniem krwi, zarządzaniem magazynem oraz dystrybucją krwi do szpitali.

Główne cele i zadania:

- Rejestracja i ewidencja dawców krwi.
- Obsługa procesu zgłaszania chęci oddania krwi (weryfikacja terminów).
- Rejestrowanie faktycznych oddań oraz wyników badań laboratoryjnych.
- Zarządzanie zapotrzebowaniami zgłaszanymi przez jednostki medyczne (szpitale).
- Monitorowanie stanu magazynowego krwi z uwzględnieniem daty ważności.

1.2 Analiza wymagań użytkownika

W systemie wyróżniono cztery role użytkowników, z których każda posiada dedykowane funkcjonalności:

- **Dawca**: Może rejestrować się w systemie, zarządzać danymi osobowymi, przeglądać historię oddań, wyniki badań oraz zgłaszać chęć oddania krwi (zapisywać się na termin).
- **Pracownik banku krwi**: Posiada uprawnienia do wprowadzania nowych oddań, wpisywania wyników badań (HIV, HCV itp.), zarządzania stanem magazynowym oraz realizowania zapotrzebowań szpitali.
- **Szpital**: Konto instytucjonalne, które umożliwia zgłaszanie zapotrzebowania na konkretną grupę krwi i śledzenie statusu realizacji (oczekujące/zrealizowane).
- **Administrator**: Zarządza użytkownikami systemu (zakładanie kont pracowniczych i szpitalnych oraz dawcó)w, posiada dostęp do pełnych statystyk systemu.

1.3 Zaprojektowanie funkcji

Baza danych realizuje następujące funkcje podstawowe:

- **Przechowywanie danych**: Bezpieczne składowanie haseł (szyfrowanie ‘pgcrypto‘) i danych wrażliwych.
- **Logika biznesowa (Triggers)**: Automatyczne wyliczanie daty ważności krwi (35 dni), blokowanie zbyt częstych oddań (56 dni przerwy), zmiana statusu krwi.
- **Raportowanie (Views)**: Generowanie widoków dla magazynu, historii badań oraz statystyk zapotrzebowań.

2 Projekt diagramów (konceptualny)

2.1 Budowa i analiza diagramu przepływu danych (DFD)

Analiza przepływu danych (Data Flow Diagram - DFD) na poziomie 1 obrazuje logiczne przetwarzanie informacji w systemie Banku Krwi. Model ten identyfikuje źródła danych, procesy transformujące te dane oraz miejsca ich przechowywania.

2.1.1 Elementy sterujące przepływem

Przepływ danych w systemie nie jest liniowy, lecz sterowany przez zdefiniowane reguły biznesowe zaimplementowane w warstwie aplikacji (Flask) oraz bazy danych (Triggery SQL). Główne elementy sterujące to:

- **Autoryzacja ról**: Mechanizm `@login_required` we Flasku decyduje, czy użytkownik ma prawo uruchomić dany proces (np. tylko Szpital może wysłać zapotrzebowanie).
- **Walidacja czasowa**: Wyzwalacze w bazie danych (np. `blokuj_wczesne_zgloszenie`) sterują przepływem, odrzucając próby wprowadzenia błędnych danych.
- **Wyniki badań**: Warunek logiczny w procesie przetwarzania oddania – wynik pozytywny automatycznie zmienia ścieżkę przepływu na "Utylizacja/Blokada", zamiast "Magazyn dostępny".

2.1.2 Specyfikacja przepływów (Wejścia i Wyjścia)

System podzielono na trzy główne procesy przetwarzania danych:

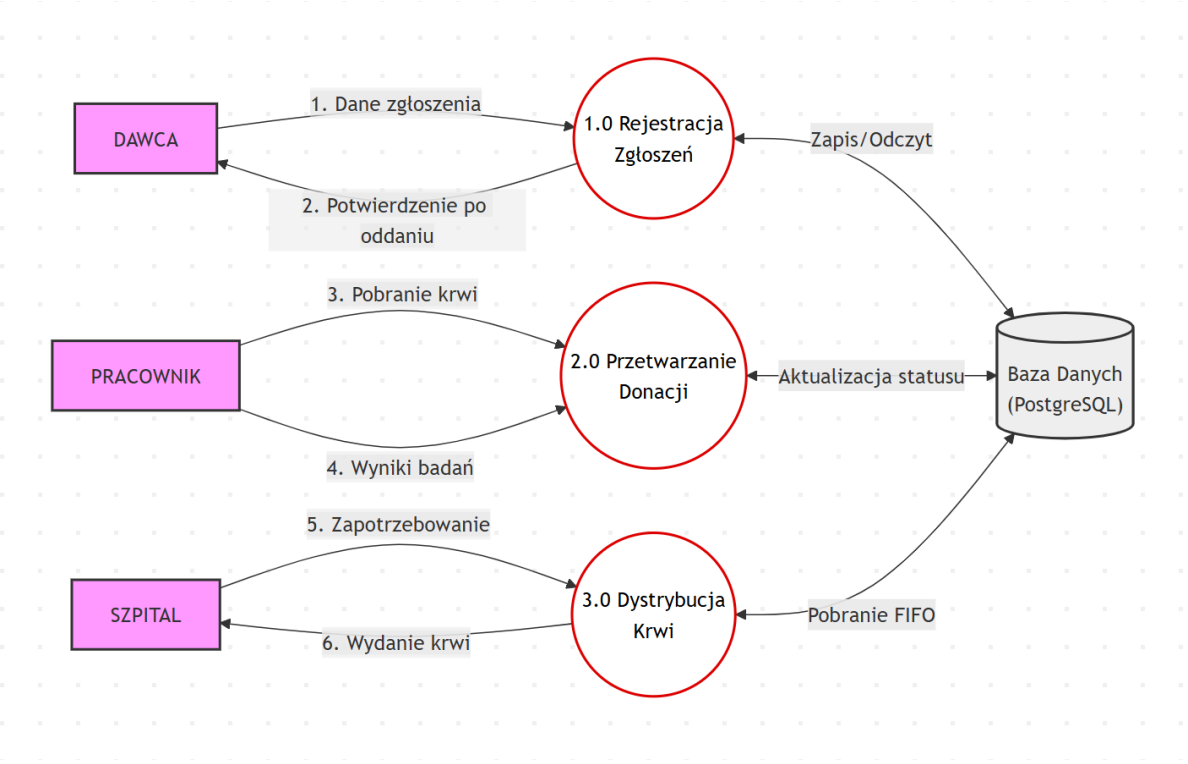
1. **Proces 1.0: Obsługa Dawcy i Zgłoszeń**
   - **Wejście**: Dane osobowe dawcy, deklarowana data wizyty.
   - **Operacje**: Weryfikacja historii w tabeli `Oddania_krwi`, sprawdzenie swoich zgłoszeń, rejestracja w tabeli `Zgloszenia`.
   - **Wyjście**: Zarejestrowanie terminu wizyty ze statusem oczekujące lub komunikat błędu (blokada).
   - **Magazyn danych**: Tabela `Dawcy`, Tabela `Zgloszenia`.

2. **Proces 2.0: Przetwarzanie Donacji i Badań**

   - **Wejście**: Fizyczna jednostka krwi (ilość ml), PESEL dawcy, wyniki badań laboratoryjnych.
   - **Operacje**: Rejestracja oddania przez Pracownika, przypisanie daty ważności (Trigger +35 dni), walidacja wyników badań (Trigger `oznacz_odrzucone_badanie`).
   - **Wyjście**: Jednostka krwi o statusie `'dostepne'` gotowa do wydania lub `'odrzucone'` niezdatna do wydania.
   - **Magazyn danych**: Tabela `Oddania_krwi`, Tabela `Badania`.

3. **Proces 3.0: Dystrybucja i Obsługa Szpitali**

   - **Wejście**: Zapotrzebowanie szpitala (Grupa krwi, Rh, ilość).
   - **Operacje**: Algorytm FIFO (First-In-First-Out) wyszukujący najstarsze pasujące jednostki krwi, aktualizacja stanów magazynowych (zmniejszenie `ilosc_pozostala`), zmiana statusu na `'zrealizowane'`.
   - **Wyjście**: Raport wydania krwi, aktualizacja panelu "Komu pomogłem?" (`Dawca_Szpital`).
   - **Magazyn danych**: Tabela `Zapotrzebowania`, Tabela `Oddanie_Zapotrzebowanie`.



Rysunek 1: Diagram przepływu danych (DFD) poziomu 1 dla systemu Banku Krwi.

2.2 Zdefiniowanie encji i atrybutów

W systemie zdefiniowano następujące klasy encji:

1. **Użytkownicy** (`Uzytkownicy`): Encja nadrzędna przechowująca dane logowania dla wszystkich ról systemowych.
    - Atrybuty: `login`, `haslo` (hash), `rola` (ENUM: ADMIN, PRACOWNIK, SZPITAL, DAWCA), `data_rejestracji`.
2. **Dawcy** (`Dawcy`): Rozszerzenie konta użytkownika o dane medyczne.
    - Atrybuty: `imie`, `nazwisko`, `pesel` (unikalny), `grupa_krwi`, `rh`, `kontakt`, `cel_ml`.
3. **Pracownicy Banku** (`Pracownicy_banku`):
    - Atrybuty: `imie`, `nazwisko`, `stanowisko` (lekarz/laborant).
4. **Szpitale** (`Szpitale`): Placówki zamawiające krew.
    - Atrybuty: `nazwa`, `adres`.
5. **Zgłoszenia** (`Zgloszenia`): Rezerwacje terminów wizyt przez dawców.
    - Atrybuty: `data_zgloszenia`, `status` (oczekujące/zaakceptowane).
6. **Oddania Krwi** (`Oddania_krwi`): Fizyczne jednostki krwi w magazynie.
    - Atrybuty: `data_oddania`, `ilosc_ml`, `data_waznosci`, `status` (dostępne, zużyte, przeterminowane, odrzucone), `ilosc_pozostala`.
7. **Badania** (`Badania`): Wyniki testów wirusologicznych dla oddań.
    - Atrybuty: `rodzaj_badania` (HIV, HCV...), `wynik`, `data_badania`.
8. **Zapotrzebowania** (`Zapotrzebowania`): Zamówienia składane przez szpitale.
    - Atrybuty: `grupa_krwi`, `rh`, `ilosc_ml`, `status`, `data_wydania`.

2.3 Zaprojektowanie relacji pomiędzy encjami

2.3.1 Klucze i powiązania (Relacje)

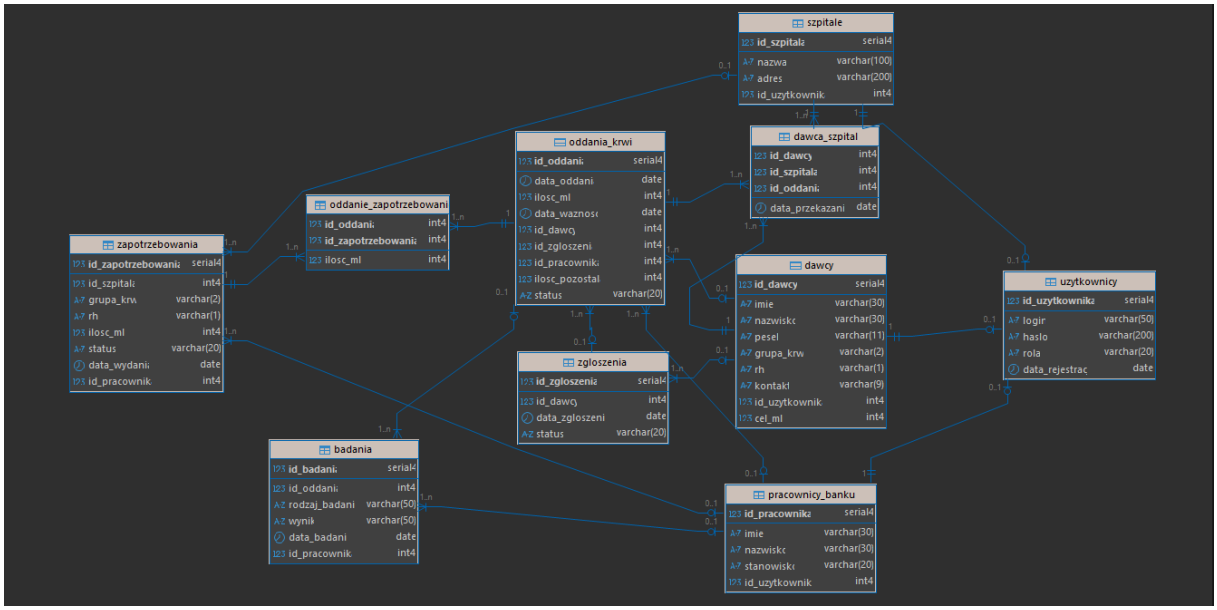
System opiera się na relacjach realizowanych poprzez klucze obce (*Foreign Keys*):

- **Relacja 1:1 (Dziedziczenie ról)**: Tabele `Dawcy`, `Pracownicy_banku` oraz `Szpitale` są powiązane z tabelą `Uzytkownicy` poprzez klucz obcy `id_uzytkownika`, który jest jednocześnie kluczem unikalnym. Pozwala to na oddzielenie danych autoryzacyjnych od danych osobowych.
- **Relacja 1:N (Jeden do wielu)**:
    - Jeden **Dawca** może mieć wiele **Oddań** oraz **Zgłoszeń**.
    - Jedno **Oddanie** może mieć przypisane wiele **Badań**.
    - Jeden **Szpital** może złożyć wiele **Zapotrzebowań**.

2.3.2 Eliminacja powiązań wiele-do-wielu (Tabele asocjacyjne)

Zidentyfikowano obszary, w których zachodziła relacja N:M, i rozwiązano je poprzez wprowadzenie tabel pośredniczących:

- **Realizacja zamówień** (`Oddanie_Zapotrzebowanie`): Pojedyncze zapotrzebowanie (np. 1000 ml) może wymagać krwi z kilku różnych oddań (np. 2 worki po 450 ml + 100 ml z trzeciego). Z kolei jedno oddanie może zostać rozdzielone na kilka mniejszych zapotrzebowań. Tabela ta łączy `id_oddania` z `id_zapotrzebowania`, przechowując informację o konkretnej ilości przekazanej krwi (`ilosc_ml`).
- **Historia przepływu** (`Dawca_Szpital`): Tabela służąca do celów raportowych i śledzenia drogi "Komu pomogłem?". Łączy Dawcę ze Szpitalem poprzez konkretne Oddanie, umożliwiając Dawcy sprawdzenie, do jakiej placówki trafiła jego krew.



Rysunek 2: Diagram związków encji (ERD) obrazujący strukturę bazy danych Banku Krwi.

3 Projekt logiczny

3.1 Projektowanie tabel (Struktura SQL)

Poniżej przedstawiono kod SQL tworzący główne struktury danych (reszta tabel jest w skrypcie sql). Wykorzystano typy danych dopasowane do domeny (np. 'VARCHAR(11)' dla PESEL, 'CHECK' dla grup krwi).

3.1.1 Tabela Użytkownicy

```
create table Uzytkownicy (
    id_uzytkownika serial primary key,
    login varchar(50) not null unique,
```

```
    haslo varchar(200) not null,
    rola varchar(20) check (rola in ('ADMIN','PRACOWNIK','SZPITAL
        ','DAWCA')),
    data_rejestracji date default current_date
);
```

3.1.2 Tabela Dawcy

```
create table Dawcy (
    id_dawcy serial primary key,
    imie varchar(30) not null,
    nazwisko varchar(30) not null,
    pesel varchar(11) unique not null,
    grupa_krwi varchar(2) check (grupa_krwi in ('A','B','AB','0')
        ),
    rh varchar(1) check (rh in ('+','-')),
    kontakt varchar(9),
    id_uzytkownika int unique references Uzytkownicy(
        id_uzytkownika),
    cel_ml int
);
```

3.1.3 Tabela Oddania Krwi

```
create table Oddania_krwi (
    id_oddania serial primary key,
    data_oddania date not null,
    ilosc_ml int check (ilosc_ml between 200 and 500),
    data_waznosci date,
    id_dawcy int references Dawcy(id_dawcy),
    id_zgloszenia int references Zgloszenia(id_zgloszenia),
    id_pracownika int references Pracownicy_banku(id_pracownika),
    ilosc_pozostala int default 0,
    status varchar(20) default 'dostepne' check (status in('
        dostepne','zuzyte','przeterminowane','odrzucone_badanie'))
);
```

3.2 Słowniki danych

W bazie zastosowano ograniczenia (‘CHECK constraints‘), które pełnią rolę słowników danych, zapewniając integralność informacji.

Atrybut	**Tabela**	**Dozwolone wartości / Ograniczenia**
rola	Uzytkownicy	’ADMIN’, ’PRACOWNIK’, ’SZPITAL’, ’DAWCA’
grupa_krwi	Dawcy, Zapotrzebowania	’A’, ’B’, ’AB’, ’0’

rh	Dawcy, Zapotrzebowania	'+', '-'
stanowisko	Pracownicy_banku	'lekarz', 'laborant'
status	Zgloszenia	'oczekujace', 'zaakceptowane'
status	Oddania_krwi	'dostepne', 'zuzyte', 'przeterminowane', 'odrzucone_badanie'
ilosc_ml	Oddania_krwi	Przedział 200 - 500
ilosc_ml	Zapotrzebowania, Oddanie_Zapotrzebowanie	Większe od 0

3.3 Normalizacja tabel

Projekt bazy danych został poddany procesowi normalizacji, aby uniknąć redundancji i anomalii.

- **1NF (Pierwsza postać normalna)**: Wszystkie atrybuty są atomowe (np. imię i nazwisko są w osobnych kolumnach, adres szpitala jest pojedynczym ciągiem znaków traktowanym jako całość w tym kontekście).
- **2NF (Druga postać normalna)**: Każda tabela posiada klucz główny, a wszystkie atrybuty niekluczowe zależą od całego klucza. W tabelach asocjacyjnych (np. 'Oddanie_Zapotrzebowanie') atrybuty takie jak 'ilosc_ml' zależą od obu kluczy obcych tworzących klucz główny.
- **3NF (Trzecia postać normalna)**: Usunięto zależności przechodnie. Na przykład dane szczegółowe o dawcy nie są powielane w tabeli 'Oddania_krwi' – znajduje się tam tylko klucz obcy 'id_dawcy'. Adres szpitala znajduje się w tabeli 'Szpitale', a nie w 'Zapotrzebowania'.

3.4 Denormalizacja

W celu optymalizacji zapytań raportowych wprowadzono elementy denormalizacji w widokach, jednak struktura fizyczna tabel pozostaje znormalizowana. Wyjątkiem optymalizacyjnym jest kolumna 'ilosc_pozostala' w tabeli 'Oddania_krwi', która jest aktualizowana na bieżąco, aby uniknąć każdorazowego sumowania wydań przy sprawdzaniu stanu magazynu.

3.5 Zaprojektowanie operacji na danych (Kwerendy)

Zdefiniowano widoki (Views) w celu uproszczenia warstwy logiki aplikacji i realizacji najczęstsze operacje wyszukiwania. Przykłady:

Stan magazynu:

```
CREATE OR REPLACE VIEW widok_stan_krwi AS
SELECT
    d.grupa_krwi,
    d.rh,
    SUM(o.ilosc_pozostala) AS dostepne_ml
```

```
FROM oddania_krwi o
JOIN dawcy d ON o.id_dawcy = d.id_dawcy
WHERE o.status = 'dostepne' AND o.ilosc_pozostala > 0
GROUP BY d.grupa_krwi, d.rh;
```

Raportowanie dużych zapotrzebowań (Agregacja)
Widok ten wykorzystuje klauzulę `HAVING`, aby filtrować dane już po pogrupowaniu. Służy do identyfikacji szpitali zgłaszających zapotrzebowanie powyżej 1000 ml na daną grupę krwi.

```
create view Widok_zapotrzebowania_duze as
select s.nazwa as szpital,
       z.grupa_krwi,
       z.rh,
       sum(z.ilosc_ml) as suma_ml
from Zapotrzebowania z
join Szpitale s on z.id_szpitala = s.id_szpitala
group by s.nazwa, z.grupa_krwi, z.rh
having sum(z.ilosc_ml) > 1000;
```

4 Projekt funkcjonalny

4.1 Interfejsy do prezentacji i edycji danych

System został zaprojektowany jako aplikacja webowa. Struktura formularzy obejmuje:

- **Formularz logowania**: Wspólny dla wszystkich ról, dodatkowo dla dawców opcja rejestracji.
- **Panel Dawcy**: Tabela z historią oddań i badań, formularz "Dodaj zgłoszenie".
- **Panel Pracownika**: Formularze "Dodaj oddanie", "Wprowadź wynik badania", tabela zarządzania zapotrzebowaniami oraz magazynem krwi.
- **Panel Szpitala**: Formularz "Zgłoś zapotrzebowanie".

4.2 Wizualizacja danych (Raporty)

Baza danych dostarcza gotowe zestawy danych do raportów poprzez widoki, które wspierają podejmowanie decyzji logistycznych::

- `widok_pracownicy_aktywnosc`: Raport wydajności pracowników (liczba obsłużonych oddań, badań i zapotrzebowań).
- `widok_zapotrzebowania_duze`: Raport dla szpitali zgłaszających duże zapotrzebowanie (> 1000ml). Występuje w formie alertu na panelu pracownika.
- `widok_statystyki_badan`: Zestawienie wyników pozytywnych i negatywnych (np. HIV/HCV).

4.3 Panel sterowania aplikacji

Aplikacja posiada główny Dashboard, który dynamicznie zmienia zawartość w zależności od roli zalogowanego użytkownika (‘id_uzytkownika‘ powiązane z tabelami ról).

4.4 Makropolecenia (Funkcje i Triggery)

W środowisku PostgreSQL rolę makropoleceń i automatyzacji pełnią funkcje składowane (Stored Functions) oraz wyzwalacze (Triggers). Zapewniają one automatyzację logiki biznesowej bez ingerencji użytkownika.

Przykład 1: Automatyczne ustawienie daty ważności (35 dni)

```
CREATE OR REPLACE FUNCTION ustaw_date_waznosci()
RETURNS trigger AS $$
BEGIN
    NEW.data_waznosci := NEW.data_oddania + INTERVAL '35 days';
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Przykład 2: Blokada zbyt częstych oddań (56 dni przerwy) Funkcja sprawdza datę ostatniego oddania dla danego dawcy przed dodaniem nowego zgłoszenia.

```
CREATE OR REPLACE FUNCTION blokuj_wczesne_zgloszenie()
RETURNS trigger AS $$
DECLARE
    ostatnie DATE;
BEGIN
    SELECT data_oddania INTO ostatnie FROM oddania_krwi
    WHERE id_dawcy = NEW.id_dawcy ORDER BY data_oddania DESC
       LIMIT 1;

    IF ostatnie IS NOT NULL AND NEW.data_zgloszenia < ostatnie +
       INTERVAL '56 days' THEN
        RAISE EXCEPTION 'Możesz oddać krew dopiero po %',
           ostatnie + INTERVAL '56 days';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

5 Dokumentacja

5.1 Wprowadzanie danych

Dane są wprowadzane do systemu na trzy sposoby:

1. **Manualnie**: Poprzez formularze aplikacji webowej (rejestracja, zgłoszenia).
2. **Automatycznie (Triggery)**: System sam uzupełnia pola takie jak ‘data_rejestracji‘ (DEFAULT current_date), ‘data_waznosci‘ czy ‘status‘.

3. **Skrypt inicjalizujący**: Baza została zasilona danymi testowymi przy użyciu instrukcji ‘INSERT‘ zawartych w pliku ‘bank_krwi.sql‘ (m.in. utworzenie kont użytkowników).

5.2 Dokumentacja użytkownika (Instrukcja)

Krótka ścieżka obsługi dla Dawcy:

1. Wejdź na stronę główną i wybierz "Zarejestruj się" lub "Zaloguj się" jeśli masz konto.
2. Po zalogowaniu wybierz zakładkę "Zgłoszenia i historia".
3. Kliknij "Wyślij zgłoszenie" – system sprawdzi, czy upłynął wymagany czas od ostatniego oddania.
4. Po wizycie w punkcie pobrań, historia oddania pojawi się w tej samej zakładce co powyżej.

Krótka ścieżka obsługi dla Pracownika:

1. Zaloguj się na konto pracownicze i sprawdź Panel (system wyświetli alerty o dużych zapotrzebowaniach).
2. Aby przyjąć krew: wejdź w zakładkę "Oddania", wyszukaj dawcę po peselu i dodaj nowe oddanie (system automatycznie ustawi datę ważności).
3. Aby uzupełnić wyniki: przejdź do zakładki "Badania", wybierz oddanie i wprowadź wynik (np. HIV/HCV) – w przypadku wyniku pozytywnego system zablokuje jednostkę.
4. Aby wydać krew: w sekcji "Zapotrzebowania" sprawdź listę oczekujących zamówień i po wydaniu krwi zmień status na "Zrealizowane".

Krótka ścieżka obsługi dla Szpitala:

1. Zaloguj się danymi placówki medycznej.
2. Wybierz opcję "Zgłoś zapotrzebowanie".
3. Wypełnij formularz określając grupę krwi, czynnik Rh oraz ilość (ml).
4. Śledź status swojego zamówienia w tabeli "Moje zapotrzebowania" – gdy pracownik banku wyda krew, status zmieni się z "Oczekujące" na "Zrealizowane".

5.3 Opracowanie dokumentacji technicznej

Projekt zrealizowano w architekturze klient-serwer, uruchamianej w środowisku lokalnym (localhost).

- **Baza danych**: PostgreSQL 14+. Wykorzystano rozszerzenie ‘pgcrypto‘ do bezpiecznego haszowania haseł.
- **Backend**: Python (Flask) łączący się z bazą przez sterownik ‘psycopg2‘.

- **Struktura plików**:
  - `bank_krwi.sql` – kompletny skrypt inicjalizujący bazę danych. Zawiera definicje DDL (tworzenie tabel, relacji, indeksów), implementację logiki biznesowej po stronie serwera SQL (funkcje PL/pgSQL, wyzwalacze/triggery), definicje widoków raportowych oraz instrukcje DML zasilające system danymi startowym.
  - `db/connection.py` – moduł odpowiedzialny za nawiązywanie połączenia z bazą danych. Wykorzystuje bibliotekę `psycopg2` do komunikacji z PostgreSQL. Centralizuje konfigurację parametrów połączenia (host, port, poświadczenia) oraz automatycznie ustawia ścieżkę wyszukiwania (`search_path`) na schemat `bank_krwi`, co upraszcza zapytania SQL w aplikacji.
  - `app.py` – główny plik wykonywalny aplikacji oparty na frameworku Flask. Pełni rolę kontrolera: obsługuje routing (ścieżki URL), zarządza sesjami użytkowników, realizuje mechanizmy autoryzacji (sprawdzanie ról) oraz pośredniczy w wymianie danych między formularzami a bazą danych.
  - `templates/` – katalog zawierający szablony widoków HTML. Wykorzystuje silnik szablonów Jinja2 do dynamicznego renderowania danych pobranych z bazy (np. tabele wyników, alerty). Warstwa prezentacji jest zintegrowana z frameworkiem Bootstrap 5, co zapewnia responsywność interfejsu.
  - `static/style.css` – arkusz stylów definiujący unikalną szatę graficzną aplikacji. Rozszerza standardowe komponenty frameworka Bootstrap (takie jak tabele i przyciski), nadpisując ich zmienne i wprowadzając dedykowany motyw kolorystyczny oparty na głębokiej czerwieni (`#8a0303`), co zapewnia interfejsowi wygląd adekwatny do tematyki Banku Krwi.

5.4 Wykaz literatury

- Dokumentacja PostgreSQL: `https://www.postgresql.org/docs/`
- Dokumentacja Flask: `https://flask.palletsprojects.com/`
- Materiały z wykładów przedmiotu "Bazy Danych I".
- Dokumentacja Bootstrap 5: `https://getbootstrap.com/`
- Wikipedia: `https://pl.wikipedia.org/wiki/Posta%C4%87_normalna_(bazy_danych)`