Uniwersytet Rzeszowski

Kolegium Nauk Przyrodniczych

Kierunek:
Informatyka

Rok akademicki:
2023/2024 (4 semestr)

Przedmiot:
Bazy danych

Prowadzący:
dr inż. Piotr Grochowalski

Temat projektu:
System zarządzania placówką medyczną

Wykonanie:
Krzysztof Motas
125145

Patryk Jarosiewicz
123066

1. Opis projektu

Projekt bazy danych został stworzony z myślą o zarządzaniu informacjami medycznymi w placówce medycznej, zapewniając wsparcie w prowadzeniu dokumentacji dotyczącej personelu, lekarstw, metod leczenia oraz pacjentów.

Dzięki naszej aplikacji personel medyczny może zarządzać danymi pacjentów, śledzić historię chorób i leczenia, oraz szybko dostępować do informacji o lekarzach, lekach i gabinetach.

2. Funkcjonalności projektu

Baza danych umożliwia dokładne śledzenie historii wizyt każdego pacjenta u lekarza, w tym datę wizyty, powód wizyty oraz diagnozę lekarza. Ta funkcjonalność pozwala na skuteczne monitorowanie stanu zdrowia pacjentów oraz świadczenie im właściwej opieki medycznej.
Ważnym elementem projektu jest również moduł dotyczący leków i recept. Baza danych przechowuje informacje o dostępnych lekach, wraz z ich ceną. Ponadto, system umożliwia przypisywanie leków do konkretnych recept przepisanych pacjentom podczas wizyt, wraz z informacjami dotyczącymi dawkowania oraz odpłatności leku.

Niektóre z dostępnych funkcji:

- rejestrowanie pacjentów,
- przyjmowanie wizyt,
- tworzenie dokumentacji medycznej,
- przepisywanie leków,
- analiza danych, np. popularność przepisywanych leków, lekarze z największą liczbą wizyt, średni czas trwania wizyty, statystyki dotyczące wieku pacjentów, popularność specjalizacji lekarskich, średnia liczba wizyt w zależności od wieku pacjentów, rozkład płci pacjentów w różnych grupach wiekowych itp.,
- historia pacjentów.

Dzięki temu systemowi personel medyczny może śledzić pacjentów i zapewnić im odpowiednią opiekę medyczną.

3. Zaprojektowane tabele oraz powiązania

Na podstawie wyżej wymienionych założeń i opisu funkcjonalności projektu, stworzono następujące tabele:

- **Tabela patients (Pacjenci):**
  - Identyfikator: id (integer, klucz główny)
  - Imię: name (varchar)
  - Nazwisko: last_name (varchar)
  - Płeć: gender (varchar)
  - Adres: address (varchar)
  - Numer telefonu: phone_number (varchar)
  - Data urodzenia: date_of_birth (date)

- **Tabela doctors (Lekarze):**
  - Identyfikator: id (integer, klucz główny)
  - Imię: name (varchar)
  - Nazwisko: last_name (varchar)
  - Specjalizacja: specialization (varchar)
  - Numer telefonu: phone_number (varchar)

- **Tabela visits (Wizyty):**
  - Identyfikator: id (integer, klucz główny)
  - Identyfikator pacjenta: patient_id (integer, klucz obcy od patients.id)
  - Identyfikator lekarza: doctor_id (integer, klucz obcy od doctors.id)
  - Powód: reason (varchar)
  - Data rozpoczęcia: start_date (datetime)
  - Data zakończenia: end_date (datetime)

- **Tabela documentations (Dokumentacje):**
  - Identyfikator: id (integer, klucz główny)
  - Identyfikator wizyty: visit_id (integer, klucz obcy od visits.id)
  - Data: date (datetime)
  - Diagnoza: diagnosis (varchar)
  - Metoda leczenia: treatment_method (varchar)

- **Tabela prescriptions (Recepty):**
  - Identyfikator: id (integer, klucz główny)
  - Identyfikator wizyty: visit_id (integer, klucz obcy od visits.id)
  - Data ważności: expiration_date (datetime)
  - Kod: code (varchar)

- **Tabela medicines (Leki):**
  - Identyfikator: id (integer, klucz główny)
  - Nazwa: name (varchar)
  - Cena: price (integer)

- **Tabela prescription_medicines (Leki na receptach):**
  - Identyfikator recepty: prescription_id (integer, klucz obcy od prescriptions.id)
    Identyfikator leku: medicine_id (integer, klucz obcy od medicines.id)
    Dawkowanie: dosage (varchar)
  - Opłata: payment (integer)

Pomiędzy tabelami zostały stworzone następujące relacje:

- **patients.id (Tabela patients) < visits.patient_id (Tabela visits)**

  Jest to relacja jeden-do-wielu (<). Jeden pacjent może mieć wiele wizyt.

- **doctors.id (Tabela doctors) < visits.doctor_id (Tabela visits)**

  Jest to relacja jeden-do-wielu (<). Jeden lekarz może przeprowadzić wiele wizyt.

- **visits.id (Tabela visits) - documentations.visit_id (Tabela documentations)**

  Jest to relacja jeden-do-jednego (-). Jedna wizyta może mieć tylko jedną dokumentację medyczną.

- **prescriptions.visit_id (Tabela prescriptions) - visits.id (Tabela visits)**

  Jest to relacja jeden-do-jednego (-). Jedna wizyta może mieć tylko jedną receptę.

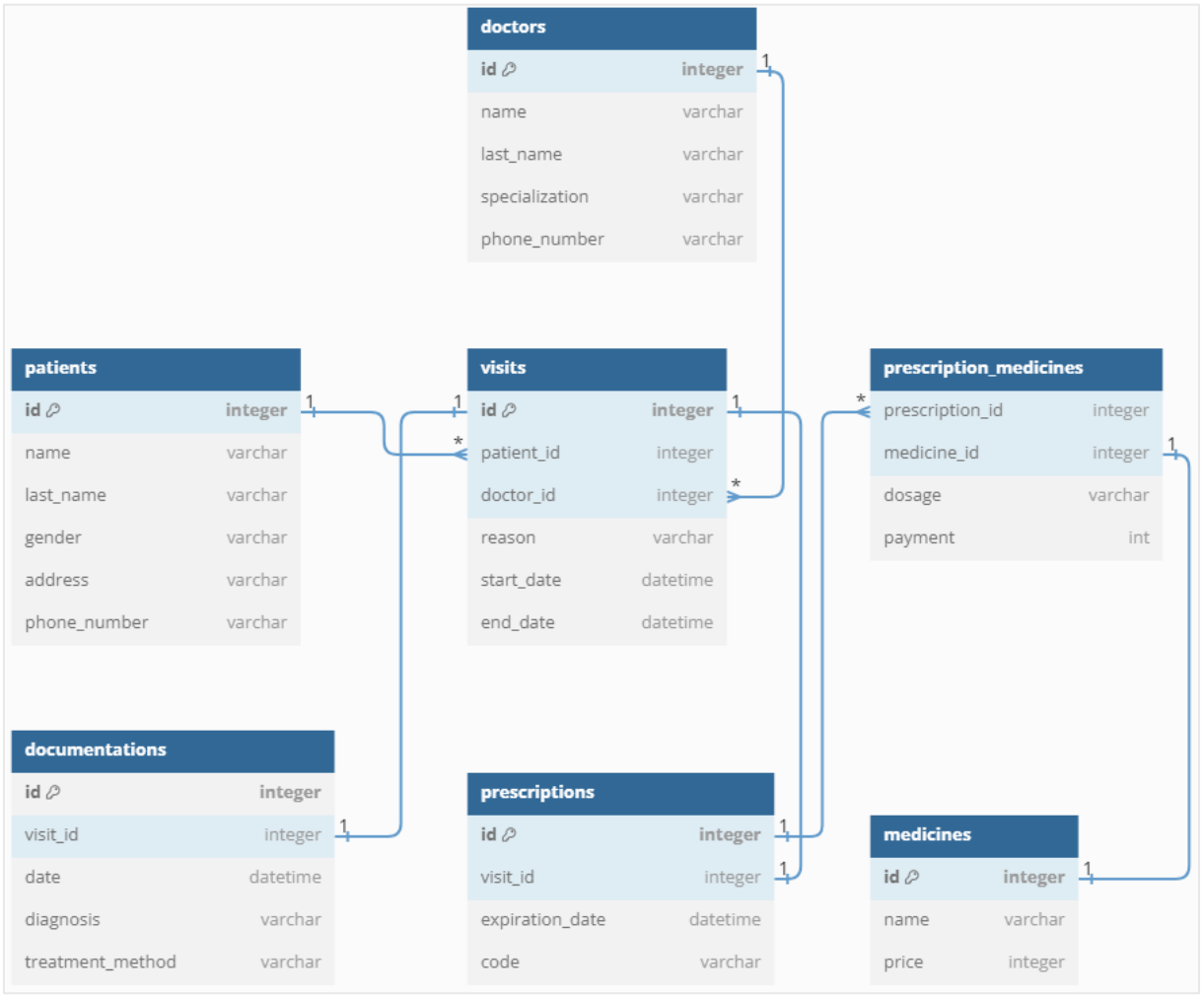
- **prescription_medicines.prescription_id (Tabela prescription_medicines) > prescriptions.id (Tabela prescriptions)**

  Relacja wiele-do-jednego (>). Wiele leków może być przypisanych do jednej recepty.

- **prescription_medicines.medicine_id (Tabela prescription_medicines) - medicines.id (Tabela medicines)**

  Relacja jeden-do-jednego (-). Oznacza to, że każde przypisanie leku na recepcie może być powiązane tylko z jednym lekiem, który jest zdefiniowany w tabeli leków.

Do wizualnego przedstawienia tabel oraz powiązań pomiędzy nimi, stworzono poniższy diagram ERD.



Zrzut ekranu 1 - Diagram ERD

4. Wybór technologii

Projekt został zrealizowany przy użyciu narzędzi i technologii. Wykorzystane technologie obejmują różne języki programowania, bazę danych, biblioteki oraz framework.

4.1. Bazy danych

- **Oracle Database:**
  - Wersja: 18c
  - Link do dokumentacji: https://docs.oracle.com/en/database/

4.2. Języki programowania

- **PHP:**
  - Wersja: PHP 8.2.12
  - Link do dokumentacji: https://www.php.net/docs.php
- **JavaScript:**
  - Wersja: ECMAScript 2023
  - Link do dokumentacji: https://developer.mozilla.org/enUS/docs/Web/JavaScript

4.3. Języki znaczników

- **HTML:**
  - Wersja: HTML 5
  - Link do dokumentacji: https://developer.mozilla.org/en-US/docs/Web/HTML

4.4. Języki arkuszy stylów

- **CSS:**
  - Wersja: CSS 5
  - Link do dokumentacji: https://developer.mozilla.org/en-US/docs/Web/CSS

4.5. Narzędzia

- **Docker:**
  - Wersja: 26.1.3
  - Link: https://docs.docker.com/

- **Composer:**
  - Wersja: 2.7.6
  - Link do dokumentacji: https://getcomposer.org/doc/

4.6. Biblioteki i frameworki

- **Laravel:**
  - Wersja: 11.0.9
  - Link do dokumentacji: https://laravel.com/docs/11.x/readme

- **Bootstrap:**
  - Wersja: 5.3.3
  - Link do dokumentacji: https://getbootstrap.com/

4.7. Środowiska programistyczne

- **Visual Studio Code:**
  - Wersja: 1.89.1
  - Link: https://code.visualstudio.com/

5. Funkcjonalności

W celu zaimplementowania głównych funkcjonalności aplikacji, stworzono wiele procedur i funkcji przy użyciu języka PL/SQL. Dzięki temu możliwe było zrealizowanie skomplikowanych operacji na bazie danych.

Nazwa	Typ	Opis
SEARCH_EXPENSIVE_MEDICINES	FUNCTION	Szuka drogich leków
CALCULATE_AVERAGE_MEDICINE_PRICE	FUNCTION	Oblicza średnią cenę leku
GENERATE_DOCTOR_PATIENT_COUNT_REPORT	FUNCTION	Generuje raport ilości pacjentów
GENERATE_TOP_DIAGNOSIS_REPORT	FUNCTION	Generuje raport najczęstszych diagnoz
GENERATE_VISIT_COUNT_BY_SPECIALIZATION_REPORT	FUNCTION	Generuje raport wizyt wg specjalizacji
GET_ALL_DOCTORS	FUNCTION	Pobiera wszystkich lekarzy
GET_ALL_PATIENTS	FUNCTION	Pobiera wszystkich pacjentów
GET_ALL_VISITS	FUNCTION	Pobiera wszystkie wizyty
GET_PRESCRIPTION_MEDICINES	FUNCTION	Pobiera leki z recepty
SEARCH_DOCTORS_BY_SPECIALIZATION	FUNCTION	Szuka lekarzy wg specjalizacji
SEARCH_TOP_PRESCRIBED_MEDICINES_BY_DOCTOR	FUNCTION	Szuka najczęściej przepisywanych leków
SEARCH_VISITS_BY_PATIENT_LAST_NAME	FUNCTION	Szuka wizyt wg nazwiska pacjenta
LOGIN_PATIENT	FUNCTION	„Loguje” pacjenta
LOGIN_DOCTOR	FUNCTION	„Loguje” lekarza
GET_ALL_MEDICINES	FUNCTION	Pobiera wszystkie leki
GET_PATIENT_ID	FUNCTION	Pobiera ID pacjenta
GET_DOCTOR_VISITS	FUNCTION	Pobiera wizyty lekarza
GET_PATIENT_PRESCRIPTIONS	FUNCTION	Pobiera recepty pacjenta
CALCULATE_AVERAGE_VISIT_TIME	FUNCTION	Oblicza średni czas wizyty
CALCULATE_AVERAGE_VISITS_BY_AGE	FUNCTION	Oblicza średnią wizyt wg wieku
GET_PATIENT_VISITS	FUNCTION	Pobiera wizyty pacjenta
ADD_DOCUMENTATION	PROCEDURE	Dodaje dokumentację
ADD_MEDICINE	PROCEDURE	Dodaje lek
ADD_PATIENT	PROCEDURE	Dodaje pacjenta
ADD_PRESCRIPTION	PROCEDURE	Dodaje receptę
ADD_PRESCRIPTION_MEDICINE	PROCEDURE	Dodaje lek do recepty
ADD_VISIT	PROCEDURE	Dodaje wizytę
DELETE_DOCUMENTATION	PROCEDURE	Usuwa dokumentację
DELETE_PRESCRIPTION	PROCEDURE	Usuwa receptę
DELETE_PRESCRIPTION_MEDICINE	PROCEDURE	Usuwa lek z recepty
UPDATE_DOCUMENTATION	PROCEDURE	Aktualizuje dokumentację
UPDATE_PATIENT	PROCEDURE	Aktualizuje dane pacjenta

UPDATE_PRESCRIPTION	PROCEDURE	Aktualizuje receptę
UPDATE_PRESCRIPTION_MEDICINE	PROCEDURE	Aktualizuje lek w recepcie
DELETE_VISIT	PROCEDURE	Usuwa wizytę
DELETE_PRESCRIPTION_MEDICINE_BY_PRESCRIPTION_ID	PROCEDURE	Usuwa lek z recepty wg ID
DELETE_VISIT_AND_ASSOCIATED_DATA	PROCEDURE	Usuwa wizytę i powiązane dane
UPDATE_VISIT	PROCEDURE	Aktualizuje wizytę

5.1. Opis przykładowych funkcji i procedur z zakresu funkcjonalności typu CRUD

- **ADD_VISIT**

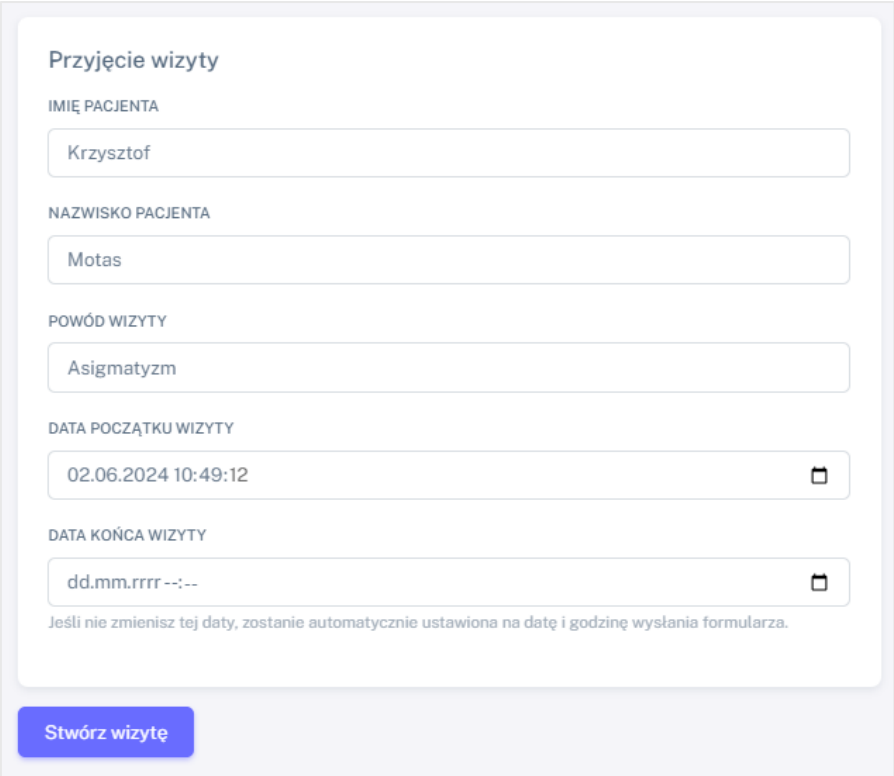
```
create or replace PROCEDURE "ADD_VISIT" (
    P_PATIENT_ID IN NUMBER,
    P_DOCTOR_ID IN NUMBER,
    P_REASON IN VARCHAR2,
    P_START_DATE IN TIMESTAMP,
    P_END_DATE IN TIMESTAMP
)
AS
BEGIN
    INSERT INTO VISITS (PATIENT_ID, DOCTOR_ID, REASON, START_DATE, END_DATE)
    VALUES (P_PATIENT_ID, P_DOCTOR_ID, P_REASON, P_START_DATE, P_END_DATE);
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
```

Zrzut ekranu 2 - Implementacja ADD_VISIT

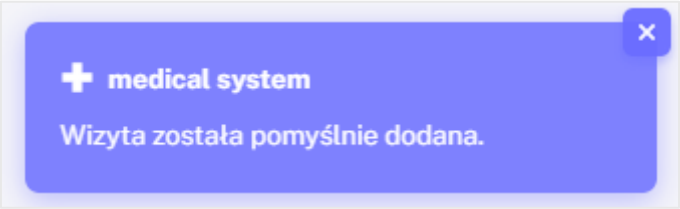
Procedura służy do dodawania nowych wizyt do systemu medycznego. Przyjmuje ona parametry dotyczące identyfikatora pacjenta, identyfikatora lekarza, powodu wizyty oraz daty rozpoczęcia i zakończenia wizyty. Następnie procedura dodaje te informacje do tabeli "VISITS" w bazie danych i potwierdza transakcję. W przypadku wystąpienia błędu, procedura wykonuje rollback transakcji i przekazuje wyjątek do obsługi wyżej.

```
DB::statement("CALL ADD_VISIT(?, ?, ?, TO_TIMESTAMP(?, 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP(?, 'YYYY-MM-DD HH24:MI:SS'))", [
    $patientId,
    $doctorId,
    $validatedData['reason'],
    str_replace('T', ' ', $validatedData['start_date']),
    str_replace('T', ' ', $validatedData['end_date'] ?? getDateTime())
]);
```

Zrzut ekranu 3 - Wykorzystanie procedury w kodzie PHP



Zrzut ekranu 4 - Formularz tworzenia wizyty



Zrzut ekranu 5 - Potwierdzenie dodania wizyty

- **GET_PATIENT_PRESCRIPTIONS**

```
create or replace FUNCTION GET_PATIENT_PRESCRIPTIONS (
    p_patient_id IN NUMBER
) RETURN SYS_REFCURSOR
AS
    prescriptions_cursor SYS_REFCURSOR;
BEGIN
    OPEN prescriptions_cursor FOR
        SELECT pr.*,
               v.START_DATE,
               d.NAME AS DOCTOR_NAME,
               d.LAST_NAME AS DOCTOR_LAST_NAME,
               d.SPECIALIZATION AS DOCTOR_SPECIALIZATION
        FROM VISITS v
        JOIN PRESCRIPTIONS pr ON v.ID = pr.VISIT_ID
        JOIN DOCTORS d ON v.DOCTOR_ID = d.ID
        WHERE v.PATIENT_ID = p_patient_id
        ORDER BY v.START_DATE DESC;

    RETURN prescriptions_cursor;
END;
```

Zrzut ekranu 6 - Implementacja funkcji

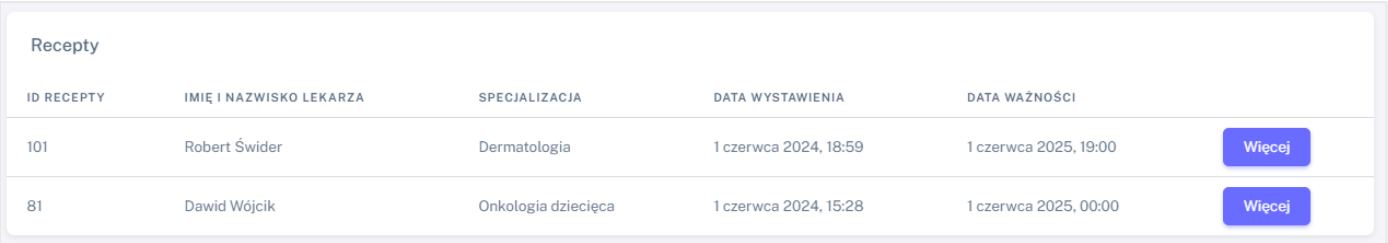
Funkcja zwraca kursor typu SYS_REFCURSOR zawierający informacje o receptach pacjenta o określonym identyfikatorze. Wewnętrznie, funkcja otwiera kursor i wykonuje zapytanie SQL, które pobiera wszystkie informacje o receptach pacjenta, łącząc je z informacjami o wizytach (takimi jak data rozpoczęcia wizyty) oraz danymi lekarza, który przepisał receptę. Następnie kursor jest zwracany jako wynik funkcji.

```php
public function prescriptions()
{
    $user = Auth::user();
    $prescriptions = executeFunctionWithCursor('GET_PATIENT_PRESCRIPTIONS', [$user->table_id]);

    foreach ($prescriptions as &$prescription) {
        $medicines = executeFunctionWithCursor('GET_PRESCRIPTION_MEDICINES', [$prescription['ID']]);
        $prescription['medicines'] = $medicines;
    }

    return view('patient.components.prescriptions', compact('prescriptions'));
}
```

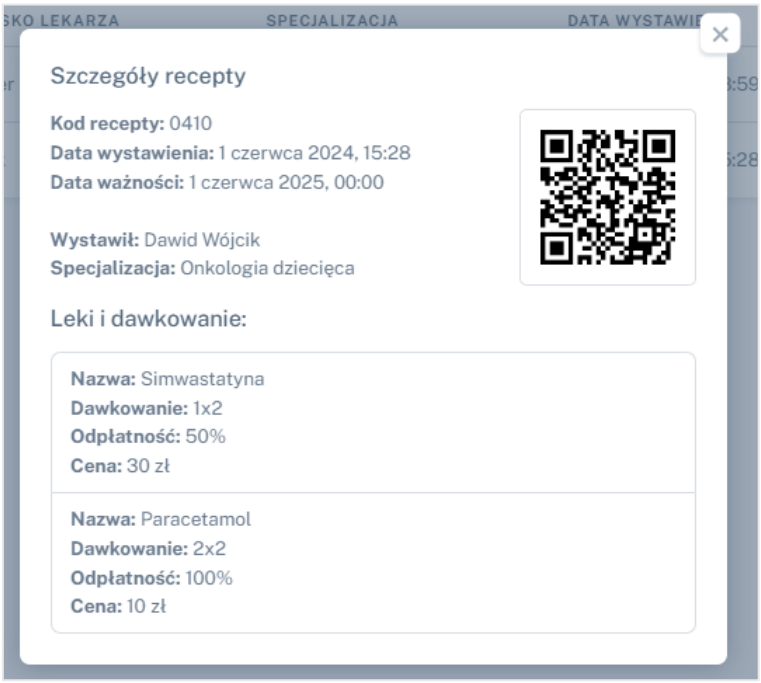
Zrzut ekranu 7 - Wykorzystanie funkcji w kodzie PHP

```php
function executeFunctionWithCursor($procedureName, $params = [])
{
    $conn = DB::getPdo()->getResource();
    $sql = "BEGIN :result := $procedureName(";
    $placeholders = [];

    foreach ($params as $index => $param) {
        $placeholders[] = ":param$index";
    }

    $sql .= implode(', ', $placeholders) . "); END;";
    $stmt = oci_parse($conn, $sql);
    $cursor = oci_new_cursor($conn);

    oci_bind_by_name($stmt, ':result', $cursor, -1, OCI_B_CURSOR);

    foreach ($params as $index => $param) {
        oci_bind_by_name($stmt, ":param$index", $params[$index]);
    }

    oci_execute($stmt);
    oci_execute($cursor, OCI_DEFAULT);

    $results = [];
    while (($row = oci_fetch_assoc($cursor)) != false) {
        $results[] = $row;
    }

    oci_free_statement($stmt);
    oci_free_statement($cursor);
    oci_close($conn);

    return $results;
}
```

Zrzut ekranu 8 - Implementacja funkcji w PHP obsługującej funkcję zwracającą kursor



Zrzut ekranu 9 - Lista wszystkich recept dla danego pacjenta



Zrzut ekranu 10 - Szczegóły danej recepty

- **UPDATE_PATIENT**

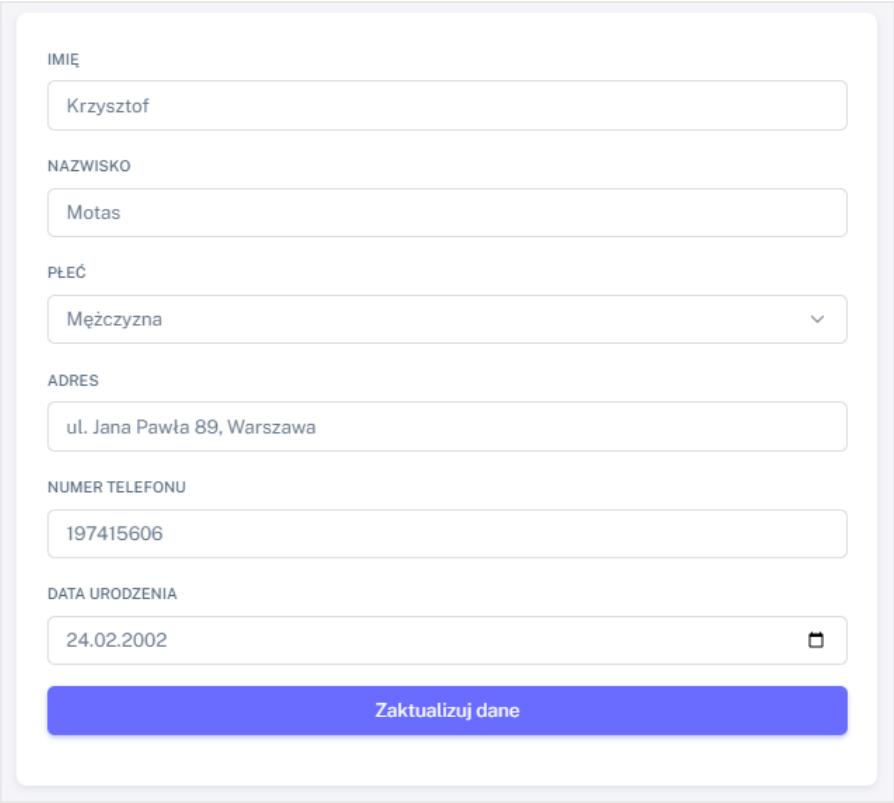
```
create or replace PROCEDURE UPDATE_PATIENT (
    P_PATIENT_ID IN NUMBER,
    P_NAME IN VARCHAR2,
    P_LAST_NAME IN VARCHAR2,
    P_GENDER IN VARCHAR2,
    P_ADDRESS IN VARCHAR2,
    P_PHONE_NUMBER IN VARCHAR2,
    P_DATE_OF_BIRTH IN DATE
)
AS
BEGIN
    UPDATE PATIENTS
    SET NAME = P_NAME,
        LAST_NAME = P_LAST_NAME,
        GENDER = P_GENDER,
        ADDRESS = P_ADDRESS,
        PHONE_NUMBER = P_PHONE_NUMBER,
        DATE_OF_BIRTH = P_DATE_OF_BIRTH
    WHERE ID = P_PATIENT_ID;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
```

Zrzut ekranu 11 - Implementacja procedury

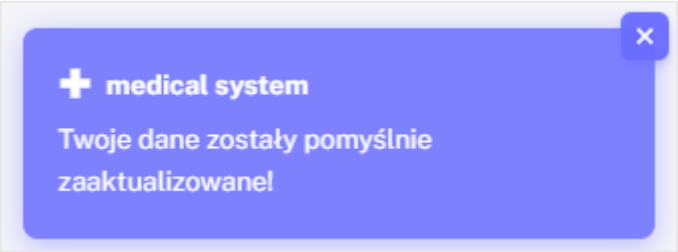
Procedura do aktualizacji danych pacjenta w systemie medycznym. Przyjmuje ona parametry, takie jak identyfikator pacjenta, imię, nazwisko, płeć, adres, numer telefonu oraz datę urodzenia. Następnie procedura wykonuje zapytanie UPDATE, aktualizując dane pacjenta w tabeli "PATIENTS" zgodnie z przekazanymi parametrami. Po udanej aktualizacji, zmiany są zatwierdzane przez COMMIT. W przypadku wystąpienia błędu, procedura wykonuje rollback transakcji i przekazuje wyjątek do obsługi wyżej.

```
DB::statement("CALL UPDATE_PATIENT(?, ?, ?, ?, ?, ?, TO_DATE(?, 'YYYY-MM-DD'))", [
    $user->table_id,
    $name,
    $last_name,
    $request->input('gender'),
    $request->input('address'),
    $request->input('phone_number'),
    $request->input('date_of_birth')
]);
```

Zrzut ekranu 12 - Wykorzystanie procedury w kodzie PHP



Zrzut ekranu 13 - Ustawienia konta pacjenta



Zrzut ekranu 14 - Powiadomienie o pomyślnym zaktualizowaniu ustawień

- **DELETE_VISIT_AND_ASSOCIATED_DATA**

```
create or replace PROCEDURE "DELETE_VISIT_AND_ASSOCIATED_DATA" (
    P_VISIT_ID IN NUMBER
)
AS
    V_DOCUMENTATION_ID NUMBER;
    V_PRESCRIPTION_ID NUMBER;
BEGIN
    -- Pobranie ID dokumentacji medycznej na podstawie wizyty
    BEGIN
        SELECT ID INTO V_DOCUMENTATION_ID FROM DOCUMENTATIONS WHERE VISIT_ID = P_VISIT_ID;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            V_DOCUMENTATION_ID := NULL;
            DBMS_OUTPUT.PUT_LINE('Nie znaleziono dokumentacji medycznej dla tej wizyty.');
    END;

    -- Jeśli dokumentacja medyczna istnieje, usuwamy ją
    IF V_DOCUMENTATION_ID IS NOT NULL THEN
        -- Usuwanie dokumentacji medycznej
        DELETE_DOCUMENTATION(V_DOCUMENTATION_ID);

        -- Debug: Wyświetlenie informacji o usunięciu dokumentacji medycznej
        DBMS_OUTPUT.PUT_LINE('Dokumentacja medyczna została usunięta.');
    END IF;

    BEGIN
        SELECT ID INTO V_PRESCRIPTION_ID FROM PRESCRIPTIONS WHERE VISIT_ID = P_VISIT_ID;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            V_PRESCRIPTION_ID := NULL;
            DBMS_OUTPUT.PUT_LINE('Nie znaleziono recepty dla tej wizyty.');
    END;

    -- Jeśli recepta istnieje, usuwamy leki przypisane do niej oraz samą receptę
    IF V_PRESCRIPTION_ID IS NOT NULL THEN
        -- Usuwanie leków przypisanych do recepty
        DELETE_PRESCRIPTION_MEDICINE_BY_PRESCRIPTION_ID(V_PRESCRIPTION_ID);

        -- Debug: Wyświetlenie informacji o usunięciu leków
        DBMS_OUTPUT.PUT_LINE('Leki przypisane do recepty zostały usunięte.');

        -- Usuwanie recepty
        DELETE_PRESCRIPTION(V_PRESCRIPTION_ID);

        -- Debug: Wyświetlenie informacji o usunięciu recepty
        DBMS_OUTPUT.PUT_LINE('Recepta została usunięta.');
    END IF;

    -- Usuwanie wizyty
    DELETE_VISIT(P_VISIT_ID);

    -- Debug: Wyświetlenie informacji o usunięciu wizyty
    DBMS_OUTPUT.PUT_LINE('Wizyta została usunięta.');

    COMMIT;

    -- Debug: Wyświetlenie informacji o sukcesie
    DBMS_OUTPUT.PUT_LINE('Operacje usuwania zostały zakończone sukcesem.');

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;

        -- Debug: Wyświetlenie informacji o błędzie
        DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
END DELETE_VISIT_AND_ASSOCIATED_DATA;
```

Zrzut ekranu 15 - Implementacja procedury

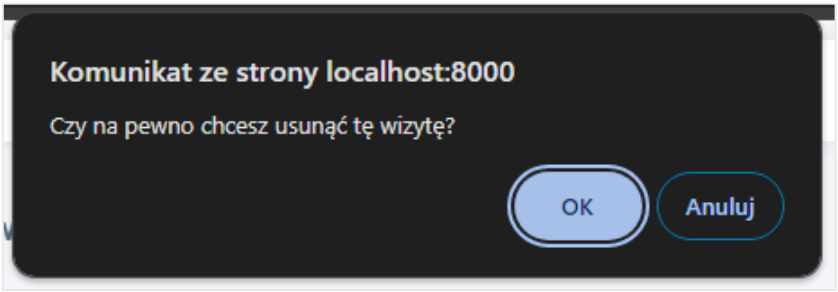
Procedura służy do usuwania danych związanych z wizytą w systemie medycznym na podstawie przekazanego identyfikatora wizyty.

Najpierw procedura próbuje pobrać identyfikator dokumentacji medycznej i recepty przypisanej do wizyty. Jeśli takie dane istnieją, są one usuwane zgodnie z kolejnością: najpierw leki przypisane do recepty, następnie sama recepta oraz dokumentacja medyczna. Po usunięciu wszystkich powiązanych danych, procedura usuwa informacje o samej wizycie.

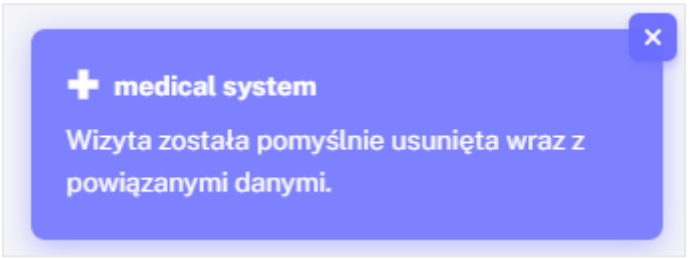
Po pomyślnym zakończeniu operacji, zmiany są zatwierdzane przez COMMIT. W przypadku wystąpienia błędu, procedura wykonuje rollback transakcji i wyświetla informacje o błędzie.

```php
public function deleteVisit($visitId)
{
    try {
        DB::transaction(function () use ($visitId) {
            DB::statement("CALL DELETE_VISIT_AND_ASSOCIATED_DATA($visitId)");
        });
    } catch (Exception $e) {
        DB::rollBack();

        return redirect()
            ->back()
            ->with('error', $e->getMessage());
    }

    return redirect()
        ->back()
        ->with('success', 'Wizyta została pomyślnie usunięta wraz z powiązanymi danymi.');
}
```

Zrzut ekranu 16 - Wykorzystanie procedury w kodzie PHP



Zrzut ekranu 17 - Potwierdzenie usunięcia wizyty



Zrzut ekranu 18 – Powiadomienie

5.2. Opis funkcji do analizy danych

- **SEARCH_EXPENSIVE_MEDICINES**

```
create or replace FUNCTION                    "SEARCH_EXPENSIVE_MEDICINES" RETURN SYS_REFCURSOR
IS
    AVG_PRICE NUMBER;
    CUR_MEDICINES SYS_REFCURSOR;
BEGIN
    OPEN CUR_MEDICINES FOR SELECT * FROM MEDICINES WHERE PRICE > (SELECT AVG(PRICE) FROM MEDICINES);
    RETURN CUR_MEDICINES;
END SEARCH_EXPENSIVE_MEDICINES;
```

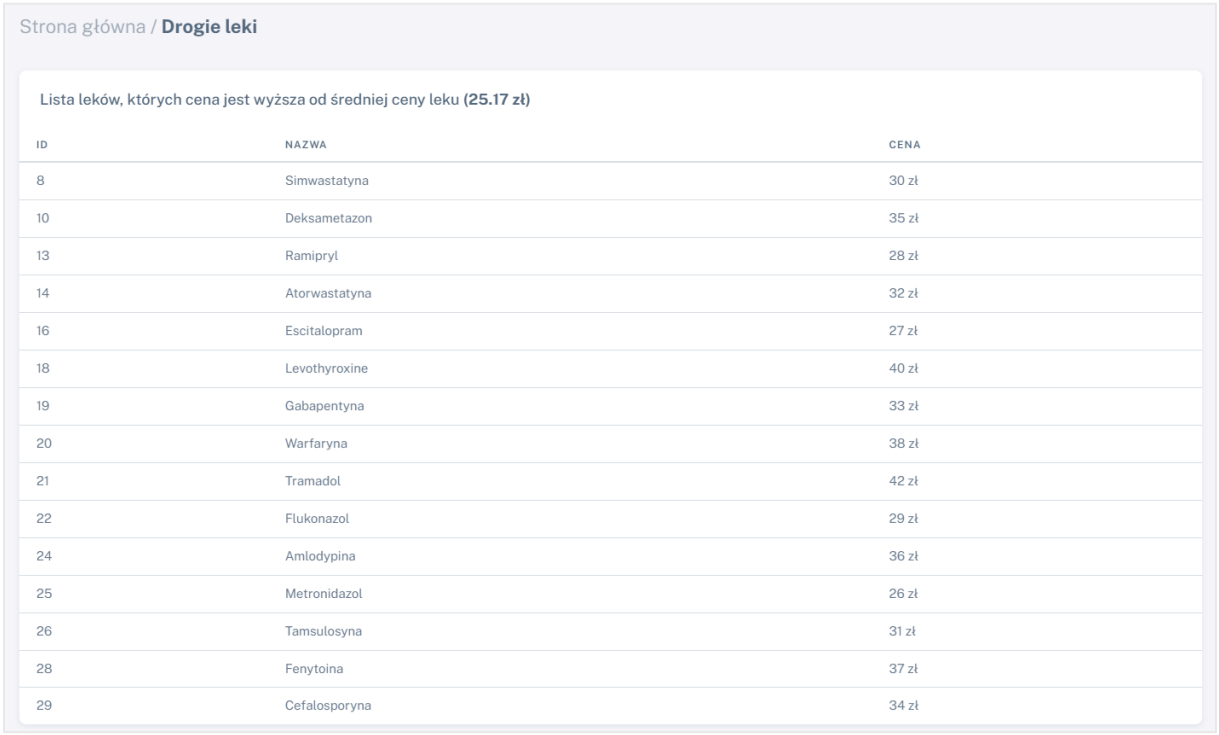
Zrzut ekranu 19 – Implementacja procedury

Funkcja służy do wyszukiwania drogich leków w systemie medycznym. Funkcja nie przyjmuje żadnych parametrów i zwraca kursor zawierający dane leków, których cena przekracza średnią cenę wszystkich leków w systemie.

Po wywołaniu funkcji, najpierw obliczana jest średnia cena wszystkich leków. Następnie otwierany jest kursor, który zawiera informacje o lekach, których cena jest wyższa niż obliczona średnia cena. Kursor ten jest następnie zwracany jako wynik funkcji.

```
public function expensiveMedicines()
{
    $medicines = executeFunctionWithCursor('SEARCH_EXPENSIVE_MEDICINES');
    $averageMedicinePrice = self::calculateAverageMedicinePrice();

    return view('doctor.components.expensive-medicines', compact('medicines', 'averageMedicinePrice'));
}
```

Zrzut ekranu 20 - Wykorzystanie procedury w kodzie PHP



Zrzut ekranu 21 - Wyświetlanie leków

- **CALCULATE_AVERAGE_MEDICINE_PRICE**

```
create or replace FUNCTION                    "CALCULATE_AVERAGE_MEDICINE_PRICE"
RETURN NUMBER
AS
    AVERAGE_PRICE NUMBER;
BEGIN
    SELECT AVG(PRICE) INTO AVERAGE_PRICE FROM MEDICINES;
    RETURN AVERAGE_PRICE;
END CALCULATE_AVERAGE_MEDICINE_PRICE;
```

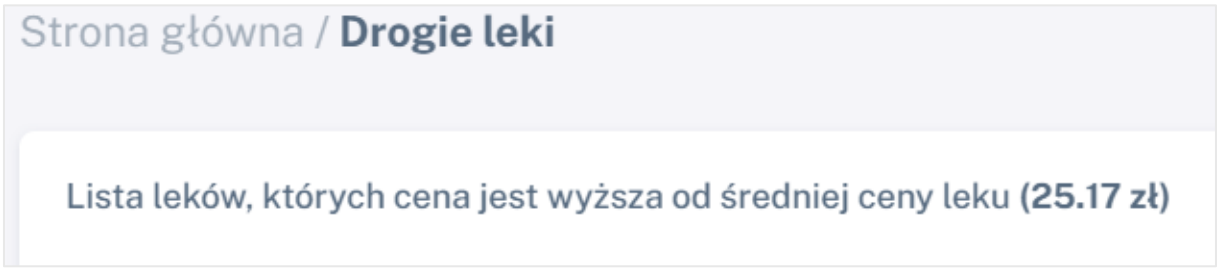
Zrzut ekranu 22 - Implementacja procedury

Procedura służy do obliczania średniej ceny leków w systemie medycznym. Funkcja nie przyjmuje żadnych parametrów i zwraca wartość typu NUMBER, która reprezentuje średnią cenę wszystkich leków zapisanych w bazie danych.

Po wywołaniu funkcji, wykonywane jest zapytanie SQL, które oblicza średnią cenę leków. Wynik tego zapytania jest przypisywany do zmiennej AVERAGE_PRICE, a następnie ta zmienna jest zwracana jako wynik funkcji.

```
function calculateAverageMedicinePrice()
{
    $result = DB::selectOne("SELECT CALCULATE_AVERAGE_MEDICINE_PRICE() AS average_price FROM DUAL");
    $averagePrice = $result->average_price;

    return $averagePrice;
}
```

Zrzut ekranu 23 - Wykorzystanie procedury w kodzie PHP



Zrzut ekranu 24 - Wyświetlanie średniej ceny leków

- **GENERATE_DOCTOR_PATIENT_COUNT_REPORT**

```
create or replace FUNCTION                    "GENERATE_DOCTOR_PATIENT_COUNT_REPORT"
RETURN SYS_REFCURSOR
AS
    doctor_patient_count_cursor SYS_REFCURSOR;
BEGIN
    OPEN doctor_patient_count_cursor FOR
        SELECT D.*, COUNT(V.PATIENT_ID) AS PATIENT_COUNT
        FROM DOCTORS D
        LEFT JOIN VISITS V ON D.ID = V.DOCTOR_ID
        GROUP BY D.ID, D.NAME, D.LAST_NAME, D.SPECIALIZATION, D.PHONE_NUMBER
        ORDER BY PATIENT_COUNT DESC;
    RETURN doctor_patient_count_cursor;
END GENERATE_DOCTOR_PATIENT_COUNT_REPORT;
```

Zrzut ekranu 25 - Implementacja procedury

Funkcja służy do generowania raportu zawierającego liczbę pacjentów przypisanych do każdego lekarza w systemie medycznym. Funkcja zwraca kursor SYS_REFCURSOR, który zawiera zestaw wyników z informacjami o lekarzach i liczbie ich pacjentów.

Po wywołaniu funkcji, wykonywane jest zapytanie SQL, które łączy tabelę DOCTORS z tabelą VISITS, aby zliczyć liczbę pacjentów przypisanych do każdego lekarza. Wynik tego zapytania jest sortowany w kolejności malejącej według liczby pacjentów i zwracany jako kursor.

```
public function doctorPatientCountReport()
{
    $report = executeFunctionWithCursor('GENERATE_DOCTOR_PATIENT_COUNT_REPORT');
    return view('doctor.components.doctor-patient-count-report', compact('report'));
}
```

Zrzut ekranu 26 - Wykorzystanie procedury w kodzie PHP

Zrzut ekranu 27 - Wyświetlanie ilości wizyt dla lekarzy

- **GENERATE_TOP_DIAGNOSIS_REPORT**

```
create or replace FUNCTION                    "GENERATE_TOP_DIAGNOSIS_REPORT" (
    P_TOP_COUNT IN NUMBER,
    P_DIAGNOSIS_FILTER IN VARCHAR2
)
RETURN SYS_REFCURSOR
AS
    top_diagnosis_cursor SYS_REFCURSOR;
BEGIN
    OPEN top_diagnosis_cursor FOR
        SELECT DIAGNOSIS, COUNT(*) AS DIAGNOSIS_COUNT
        FROM DOCUMENTATIONS
        WHERE UPPER(DIAGNOSIS) LIKE '%' || UPPER(P_DIAGNOSIS_FILTER) || '%'
        GROUP BY DIAGNOSIS
        ORDER BY DIAGNOSIS_COUNT DESC
        FETCH FIRST P_TOP_COUNT ROWS ONLY;
    RETURN top_diagnosis_cursor;
END GENERATE_TOP_DIAGNOSIS_REPORT;
```

Zrzut ekranu 28 - Implementacja procedury

Funkcja służy do generowania raportu zawierającego najczęściej występujące diagnozy w systemie medycznym. Funkcja przyjmuje dwa parametry: P_TOP_COUNT określający liczbę najczęściej występujących diagnoz do zwrócenia oraz P_DIAGNOSIS_FILTER jako opcjonalny filtr tekstowy, który pozwala na zawężenie wyników do diagnoz zawierających określony ciąg znaków.

Po wywołaniu funkcji, wykonywane jest zapytanie SQL, które zlicza wystąpienia każdej diagnozy w tabeli DOCUMENTATIONS, filtruje wyniki na podstawie przekazanego parametru P_DIAGNOSIS_FILTER, a następnie zwraca najczęściej występujące diagnozy według liczby wystąpień, w kolejności malejącej. Wyniki są zwracane jako kursor SYS_REFCURSOR.

```
public function topDiagnoses(Request $request)
{
    $topCount = $request->input('top_count', 10);
    $searchTerm = $request->input('search_term', '');

    $diagnoses = executeFunctionWithCursor('GENERATE_TOP_DIAGNOSIS_REPORT', [$topCount, $searchTerm]);

    return view('doctor.components.top-diagnoses', compact('diagnoses', 'topCount', 'searchTerm'));
}
```

Zrzut ekranu 29 - Wykorzystanie procedury w kodzie PHP

Zrzut ekranu 30 - Widok najczęstszych diagnoz pacjentów

- **GENERATE_VISIT_COUNT_BY_SPECIALIZATION_REPORT**

```
create or replace FUNCTION                    "GENERATE_VISIT_COUNT_BY_SPECIALIZATION_REPORT"
RETURN SYS_REFCURSOR
AS
    visit_count_by_specialization_cursor SYS_REFCURSOR;
BEGIN
    OPEN visit_count_by_specialization_cursor FOR
        SELECT D.SPECIALIZATION, COUNT(V.ID) AS VISIT_COUNT
        FROM DOCTORS D
        LEFT JOIN VISITS V ON D.ID = V.DOCTOR_ID
        GROUP BY D.SPECIALIZATION
        ORDER BY VISIT_COUNT DESC;
    RETURN visit_count_by_specialization_cursor;
END GENERATE_VISIT_COUNT_BY_SPECIALIZATION_REPORT;
```

Zrzut ekranu 31 - Implementacja procedury

Funkcja służy do generowania raportu zliczającego wizyty pacjentów w systemie medycznym, pogrupowane według specjalizacji lekarzy. Funkcja nie przyjmuje żadnych parametrów i zwraca kursor SYS_REFCURSOR, który zawiera wyniki zapytania SQL.

Po wywołaniu funkcji, wykonywane jest zapytanie SQL, które zlicza liczbę wizyt (VISIT_COUNT) dla każdej specjalizacji lekarza. Zapytanie wykonuje lewy zewnętrzny JOIN tabel DOCTORS i VISITS, grupuje wyniki według specjalizacji (SPECIALIZATION) i sortuje je w kolejności malejącej według liczby wizyt. Wyniki są zwracane jako kursor SYS_REFCURSOR.

```php
public function specializationPopularity()
{
    $popularityData = executeFunctionWithCursor('GENERATE_VISIT_COUNT_BY_SPECIALIZATION_REPORT');
    return view('doctor.components.specialization-popularity', compact('popularityData'));
}
```

Zrzut ekranu 32 - Wykorzystanie procedury w kodzie PHP

LP.	NAZWA SPECJALIZACJI	CAŁKOWITA LICZBA WIZYT
1	Fizjoterapia	3
2	Dermatologia	2
3	Ginekologia	2
4	Alergologia	1
5	Urologia	1
6	Psychologia	1
7	Neurologia	0
8	Onkologia dziecięca	0
9	Kardiologia	0

Zrzut ekranu 33 - Wyświetlanie najbardziej popularnych specjalizacji

- **SEARCH_TOP_PRESCRIBED_MEDICINES_BY_DOCTOR**

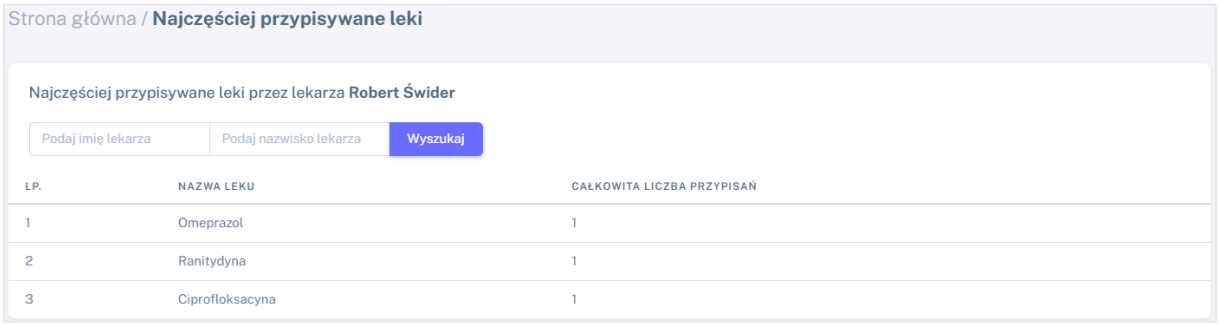
```sql
create or replace FUNCTION                    "SEARCH_TOP_PRESCRIBED_MEDICINES_BY_DOCTOR" (
    P_DOCTOR_ID IN NUMBER
)
RETURN SYS_REFCURSOR
AS
    top_prescribed_medicines_cursor SYS_REFCURSOR;
BEGIN
    OPEN top_prescribed_medicines_cursor FOR
        SELECT *
        FROM (
            SELECT M.NAME AS MEDICINE_NAME, COUNT(PM.PRESCRIPTION_ID) AS PRESCRIPTION_COUNT
            FROM MEDICINES M
            INNER JOIN PRESCRIPTION_MEDICINES PM ON M.ID = PM.MEDICINE_ID
            INNER JOIN PRESCRIPTIONS P ON PM.PRESCRIPTION_ID = P.ID
            INNER JOIN VISITS V ON P.VISIT_ID = V.ID
            WHERE V.DOCTOR_ID = P_DOCTOR_ID
            GROUP BY M.ID, M.NAME
            ORDER BY PRESCRIPTION_COUNT DESC
        );
    RETURN top_prescribed_medicines_cursor;
END SEARCH_TOP_PRESCRIBED_MEDICINES_BY_DOCTOR;
```

Zrzut ekranu 34 - Implementacja procedury

Funkcja służy do wyszukiwania najczęściej przepisywanych leków przez konkretnego lekarza na podstawie przekazanego identyfikatora lekarza. Funkcja przyjmuje jeden parametr wejściowy P_DOCTOR_ID, który jest identyfikatorem lekarza, i zwraca kursor SYS_REFCURSOR, zawierający wyniki zapytania SQL.

```php
public function topPrescribedMedicines(Request $request)
{
    $name = $request->input('name');
    $lastName = $request->input('last_name');

    if ($name && $lastName) {
        $doctorId = getDoctorTableId($name, $lastName);

        if (!$doctorId) {
            return redirect()
                ->back()
                ->with('error', 'Lekarz o takim imieniu i nazwisku nie istnieje!');
        }
    } else {
        $user = Auth::user();
        $doctorId = $user->table_id;

        $name = $user->name;
        $lastName = $user->last_name;
    }
    $medicines = executeFunctionWithCursor('SEARCH_TOP_PRESCRIBED_MEDICINES_BY_DOCTOR', [$doctorId]);

    return view('doctor.components.top-prescribed-medicines', compact('medicines', 'name', 'lastName'));
}
```

Zrzut ekranu 35 - Wykorzystanie procedury w kodzie PHP



Zrzut ekranu 36 - Wyświetlanie najczęściej przypisywanych leków

- **CALCULATE_AVERAGE_VISIT_TIME**

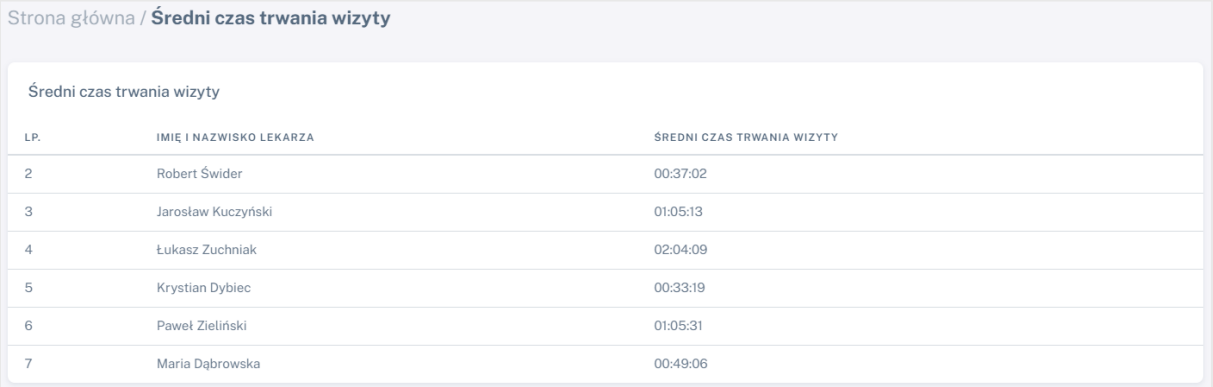
```sql
create or replace FUNCTION "CALCULATE_AVERAGE_VISIT_TIME" RETURN SYS_REFCURSOR IS
    p_result SYS_REFCURSOR;
BEGIN
    OPEN p_result FOR
        SELECT d.ID,
               d.NAME,
               d.LAST_NAME,
               AVG(EXTRACT(SECOND FROM (v.END_DATE - v.START_DATE)) +
                   EXTRACT(MINUTE FROM (v.END_DATE - v.START_DATE)) * 60 +
                   EXTRACT(HOUR FROM (v.END_DATE - v.START_DATE)) * 3600) AS AVERAGE_TIME
        FROM MEDICAL_SYSTEM.DOCTORS d
        LEFT JOIN MEDICAL_SYSTEM.VISITS v ON d.ID = v.DOCTOR_ID
        GROUP BY d.ID, d.NAME, d.LAST_NAME;

    RETURN p_result;
END;
```

Zrzut ekranu 37 - Implementacja procedury

Funkcja służy do obliczania średniego czasu trwania wizyt lekarskich. Funkcja zwraca kursor SYS_REFCURSOR, który zawiera wyniki zapytania SQL, prezentujące identyfikatory lekarzy, ich imiona, nazwiska oraz średni czas trwania wizyt dla każdego z lekarzy.

```php
public function visitsDuration()
{
    $visitsData = executeFunctionWithCursor('CALCULATE_AVERAGE_VISIT_TIME');
    return view('doctor.components.visits-duration', compact('visitsData'));
}
```

Zrzut ekranu 38 - Wykorzystanie procedury w kodzie PHP



Zrzut ekranu 39 - Wyświetlanie średniego czasu trwania wizyty

- **CALCULATE_AVERAGE_VISITS_BY_AGE**

```sql
create or replace FUNCTION                    "CALCULATE_AVERAGE_VISITS_BY_AGE"
RETURN SYS_REFCURSOR
AS
    visits_by_age_cursor SYS_REFCURSOR;
BEGIN
    OPEN visits_by_age_cursor FOR
        SELECT
            TRUNC(MONTHS_BETWEEN(SYSDATE, P.DATE_OF_BIRTH) / 12) AS AGE,
            COUNT(V.ID) AS VISITS_COUNT,
            COUNT(V.ID) / COUNT(DISTINCT P.ID) AS AVERAGE_VISITS
        FROM
            VISITS V
        JOIN
            PATIENTS P ON V.PATIENT_ID = P.ID
        GROUP BY
            TRUNC(MONTHS_BETWEEN(SYSDATE, P.DATE_OF_BIRTH) / 12)
        ORDER BY
            AGE;
    RETURN visits_by_age_cursor;
END CALCULATE_AVERAGE_VISITS_BY_AGE;
```

Zrzut ekranu 40 - Implementacja procedury

Procedura służy do obliczania średniej liczby wizyt w podziale na wiek pacjentów. Funkcja zwraca kursor SYS_REFCURSOR, który zawiera wyniki zapytania SQL, prezentujące wiek pacjentów, liczbę wizyt oraz średnią liczbę wizyt na pacjenta w każdej grupie wiekowej.

```php
public function averageVisitsByAge()
{
    $visitsByAge = executeFunctionWithCursor('CALCULATE_AVERAGE_VISITS_BY_AGE');

    return view('doctor.components.average-visits-by-age', compact('visitsByAge'));
}
```

Zrzut ekranu 41 - Wykorzystanie procedury w kodzie PHP

Zrzut ekranu 42 - Wyświetlanie średniej wizyt według wieku

6. Konfiguracja

Pierwszym krokiem w uruchomieniu aplikacji będzie przygotowanie bazy danych.

Po zalogowaniu się jako administrator bazy danych Oracle, należy utworzyć użytkownika odpowiedzialnego za placówkę medyczną:

```
CREATE USER MEDICAL_SYSTEM IDENTIFIED BY MEDICAL_SYSTEM;
```

Następnie trzeba przyznać mu odpowiednie uprawnienia:

```
GRANT ALL PRIVILEGES TO MEDICAL_SYSTEM;
```

Po zalogowaniu się na MEDICAL_SYSTEM, wymagana jest zmiana kodowania bazy danych na UTF-8 zaimportowanie struktury bazy danych. Aby to zrobić, uruchom Worksheet i wklej całą zawartość pliku `medical_system.sql`, który znajduje się w głównym folderze repozytorium.

Następnym krokiem w konfiguracji projektu będzie upewnienie się, że plik .env znajdujący się w folderze Laravel zawiera odpowiednią konfigurację połączenia do bazy danych.

```
DB_CONNECTION=oracle
DB_HOST=127.0.0.1
DB_PORT=1521
DB_DATABASE=xe
DB_USERNAME=MEDICAL_SYSTEM
DB_PASSWORD=MEDICAL_SYSTEM
DB_SERVICE_NAME=
DB_CHARSET=utf8
```

W celu konfiguracji aplikacji Laravel, wymagane będzie wykonanie następujących poleceń:

```
composer install
php artisan tinker
php artisan migrate
php artisan serve
```

7. Podsumowanie

W ramach projektu stworzona została nowoczesna aplikację webową dla systemu medycznego, która opiera się na bazie danych Oracle i wykorzystuje funkcje oraz procedury napisane w PL/SQL. Celem było nie tylko zapewnienie podstawowych operacji CRUD, ale także umożliwienie zaawansowanej analizy danych i generowania raportów. Dzięki integracji z frameworkiem Laravel, aplikacja posiada intuicyjny interfejs, który ułatwia zarządzanie danymi pacjentów, lekarzy, wizyt i recept. Cały system działa sprawnie i bezpiecznie, zapewniając szybki dostęp do kluczowych informacji.