Obraz zawierający tekst, Czcionka, Grafika, zrzut ekranu

Opis wygenerowany automatycznie

**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.

1. **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.

1. **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.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie

Zrzut ekranu 1 - Diagram ERD

1. **Wybór technologii**

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

* 1. **Bazy danych**
* **Oracle Database:**
* Wersja: 18c
* Link do dokumentacji: <https://docs.oracle.com/en/database/>
  1. **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>
  1. **Języki znaczników**
* **HTML:**
* Wersja: HTML 5
* Link do dokumentacji: <https://developer.mozilla.org/en-US/docs/Web/HTML>
  1. **Języki arkuszy stylów**
* **CSS:**
* Wersja: CSS 5
* Link do dokumentacji: <https://developer.mozilla.org/en-US/docs/Web/CSS>
  1. **Narzędzia**
* **Docker:**
* Wersja: 26.1.3
* Link: <https://docs.docker.com/>
* **Composer:**
* Wersja: 2.7.6
* Link do dokumentacji: <https://getcomposer.org/doc/>
  1. **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/>
  1. **Środowiska programistyczne**
* **Visual Studio Code:**
* Wersja: 1.89.1
* Link: <https://code.visualstudio.com/>

1. **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 |

* 1. **Opis przykładowych funkcji i procedur z zakresu funkcjonalności typu CRUD**
* **ADD\_VISIT**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Zrzut ekranu 3 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Zrzut ekranu 4 - Formularz tworzenia wizyty

Obraz zawierający tekst, Jaskrawoniebieski, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Zrzut ekranu 5 - Potwierdzenie dodania wizyty

* **GET\_PATIENT\_PRESCRIPTIONS**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Zrzut ekranu 7 - Wykorzystanie funkcji w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, Czcionka, dokument

Opis wygenerowany automatycznie

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

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Zrzut ekranu 9 - Lista wszystkich recept dla danego pacjenta

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Zrzut ekranu 10 - Szczegóły danej recepty

* **UPDATE\_PATIENT**

Obraz zawierający tekst, zrzut ekranu, Czcionka, dokument

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Zrzut ekranu 12 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Zrzut ekranu 13 - Ustawienia konta pacjenta

Obraz zawierający tekst, Jaskrawoniebieski, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Zrzut ekranu 14 - Powiadomienie o pomyślnym zaktualizowaniu ustawień

* **DELETE\_VISIT\_AND\_ASSOCIATED\_DATA**

Obraz zawierający tekst, zrzut ekranu, Czcionka, dokument

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, dokument

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Zrzut ekranu 16 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, multimedia, gadżet

Opis wygenerowany automatycznie

Zrzut ekranu 17 - Potwierdzenie usunięcia wizyty

Obraz zawierający tekst, Jaskrawoniebieski, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Zrzut ekranu 18 – Powiadomienie

* 1. **Opis funkcji do analizy danych**
* **SEARCH\_EXPENSIVE\_MEDICINES**

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Zrzut ekranu 20 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Zrzut ekranu 21 - Wyświetlanie leków

* **CALCULATE\_AVERAGE\_MEDICINE\_PRICE**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Zrzut ekranu 23 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, Czcionka, biały

Opis wygenerowany automatycznie

Zrzut ekranu 24 - Wyświetlanie średniej ceny leków

* **GENERATE\_DOCTOR\_PATIENT\_COUNT\_REPORT**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Zrzut ekranu 26 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Zrzut ekranu 27 - Wyświetlanie ilości wizyt dla lekarzy

* **GENERATE\_TOP\_DIAGNOSIS\_REPORT**

**Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie**

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.

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Zrzut ekranu 29 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, numer, linia

Opis wygenerowany automatycznie

Zrzut ekranu 30 - Widok najczęstszych diagnoz pacjentów

* **GENERATE\_VISIT\_COUNT\_BY\_SPECIALIZATION\_REPORT**

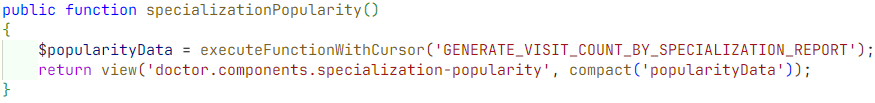
**Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie**

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.



Zrzut ekranu 32 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, numer, paragon

Opis wygenerowany automatycznie

Zrzut ekranu 33 - Wyświetlanie najbardziej popularnych specjalizacji

* **SEARCH\_TOP\_PRESCRIBED\_MEDICINES\_BY\_DOCTOR**

**Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie**

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.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Zrzut ekranu 35 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Zrzut ekranu 36 - Wyświetlanie najczęściej przypisywanych leków

* **CALCULATE\_AVERAGE\_VISIT\_TIME**

**Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie**

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.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Zrzut ekranu 38 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Zrzut ekranu 39 - Wyświetlanie średniego czasu trwania wizyty

* **CALCULATE\_AVERAGE\_VISITS\_BY\_AGE**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Zrzut ekranu 41 - Wykorzystanie procedury w kodzie PHP

Obraz zawierający tekst, zrzut ekranu, linia, Czcionka

Opis wygenerowany automatycznie

Zrzut ekranu 42 - Wyświetlanie średniej wizyt według wieku

1. **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

1. **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.