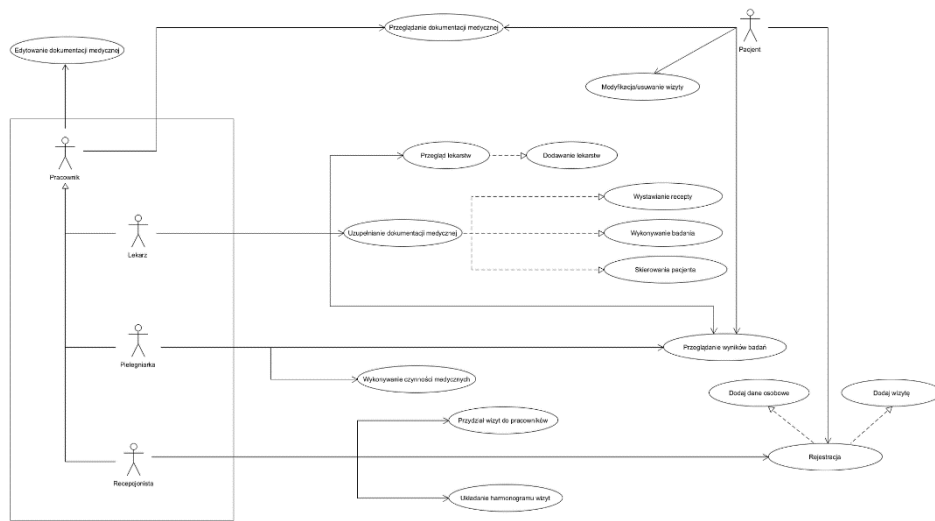


Projekt Kliniki w Oracle APEX

Spis treści

1.	Use cases	3
2.	Pakiety i procedury.....	4
3.	Utworzenie pakietów	18
4.	Webowy interfejs aplikacji w APEX	34
5.	Autoryzacja użytkowników	44
6.	Podsumowanie	45

1. Use cases



2. Pakiety i procedury

PAKIET I

Procedury

- Procedura Dodaj_Pacjenta:

Opis: Ta procedura pozwala na dodanie nowego pacjenta do bazy danych. Przyjmuje dane osobowe pacjenta takie jak imię, nazwisko, data urodzenia, PESEL, numer telefonu, identyfikator uprawnień pacjenta oraz adres pacjenta.

Wynik: Procedura nie zwraca wartości.

```
CREATE OR REPLACE PROCEDURE Dodaj_Pacjenta (  
    p_imie VARCHAR2,  
    p_nazwisko VARCHAR2,  
    p_data_urodzenia DATE,  
    p_pesel VARCHAR2,  
    p_numer_telefonu VARCHAR2,  
    p_uprawnienia_id NUMBER,  
    p_adres_id NUMBER  
)  
IS  
    v_id_pacjenta NUMBER;  
BEGIN  
    -- Pobierz nowy identyfikator pacjenta  
    SELECT MAX(id_pacjenta) + 1 INTO v_id_pacjenta FROM pacjenci;  
  
    -- Wstaw nowego pacjenta  
    INSERT INTO pacjenci (  
        id_pacjenta,  
        imie,  
        nazwisko,  
        data_urodzenia,  
        pesel,  
        numer_telefonu,  
        uprawnienia_id_uprawnienia,  
        adresy_id_adresu  
    ) VALUES (  
        v_id_pacjenta,  
        p_imie,  
        p_nazwisko,  
        p_data_urodzenia,  
        p_pesel,  
        p_numer_telefonu,  
        p_uprawnienia_id,  
        p_adres_id  
    );  
  
    COMMIT;  
  
    DBMS_OUTPUT.PUT_LINE('Pacjent dodany do bazy danych. ID pacjenta: ' || v_id_pacjenta);  
EXCEPTION  
    WHEN OTHERS THEN
```

```

ROLLBACK;
DBMS_OUTPUT.PUT_LINE('Błąd podczas dodawania pacjenta: ' || SQLERRM);
END Dodaj_Pacjenta;

```

- **Procedura Aktualizuj_Pacjenta:**

Opis: Ta procedura umożliwia aktualizację danych pacjenta w bazie danych na podstawie jego identyfikatora. Można zmienić dane osobowe, numer telefonu oraz adres pacjenta.

Wynik: Procedura nie zwraca wartości.

```

CREATE OR REPLACE PROCEDURE Aktualizuj_Pacjenta (
    p_id_pacjenta      INTEGER,
    p_nowe_imie        VARCHAR2,
    p_nowe_nazwisko    VARCHAR2,
    p_nowa_data_urodzenia DATE,
    p_nowy_pesel       VARCHAR2,
    p_nowy_numer_telefonu VARCHAR2,
    p_nowe_id_adresu    INTEGER
)
AS
BEGIN
    UPDATE pacjenci
    SET
        imie = p_nowe_imie,
        nazwisko = p_nowe_nazwisko,
        data_urodzenia = p_nowa_data_urodzenia,
        pesel = p_nowy_pesel,
        numer_telefonu = p_nowy_numer_telefonu,
        adresy_id_adresu = p_nowe_id_adresu
    WHERE
        id_pacjenta = p_id_pacjenta;

    COMMIT;
END Aktualizuj_Pacjenta;

```

- **Procedura Usun_Pacjenta:**

Opis: Ta procedura służy do usuwania pacjenta z bazy danych na podstawie jego identyfikatora. Usuwa wszystkie związane z nim dane, takie jak dokumentacja medyczna, wizyty, recepty, itp.

Wynik: Procedura nie zwraca wartości.

```

CREATE OR REPLACE PROCEDURE Usun_Pacjenta (
    p_id_pacjenta IN INTEGER
)
IS
BEGIN
    -- Usuwanie z tabeli "dokumentacja_medyczna"
    DELETE FROM dokumentacja_medyczna WHERE pacjenci_id_pacjenta = p_id_pacjenta;

```

```

-- Usuwanie z tabeli "rejestracja"
DELETE FROM rejestracja WHERE pacjenci_id_pacjenta = p_id_pacjenta;

-- Usuwanie z tabeli "recepty"
DELETE FROM recepty WHERE pracownicy_id_pracownika IN (SELECT id_pracownika FROM
pracownicy WHERE adresy_id_adresu IN (SELECT id_adresu FROM adresy WHERE id_adresu =
p_id_pacjenta));

-- Usuwanie z tabeli "uprawnienia_pracownicy"
DELETE FROM uprawnienia_pracownicy WHERE pracownicy_id_pracownika IN (SELECT
id_pracownika FROM pracownicy WHERE adresy_id_adresu IN (SELECT id_adresu FROM adresy
WHERE id_adresu = p_id_pacjenta));

-- Usuwanie z tabeli "pacjenci"
DELETE FROM pacjenci WHERE id_pacjenta = p_id_pacjenta;

COMMIT;

END;

```

- **Procedura Anuluj wizytę:**

Opis: Ta procedura służy do usuwania wizyty pacjenta z bazy danych na podstawie identyfikatora wizyty. Usuwa wszystkie związane z nim dane, takie jak data wizyty, godzina rozpoczęcia, godzina zakończenia, opis wizyty, status wizyty, itp.

Wynik: Procedura nie zwraca wartości.

```

CREATE OR REPLACE PROCEDURE AnulujWizyte(
    p_wizyta_id INTEGER
) AS
BEGIN
    DELETE FROM rejestracja WHERE id_wizyty = p_wizyta_id;
    DBMS_OUTPUT.PUT_LINE('Wizyta o ID: ' || p_wizyta_id || ' została anulowana.');
```

EXCEPTION

```

    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Wystąpił błąd podczas anulowania wizyty.');
```

END AnulujWizyte;

Funkcje

- **Funkcja Pobierz_Informacje_Pacjenta:**

Opis: Ta funkcja pozwala na pobranie informacji o pacjencie na podstawie jego identyfikatora. Zwraca dane osobowe pacjenta oraz adres.

Wynik: Zwraca rekord zawierający imię, nazwisko, data urodzenia, PESEL, numer telefonu, identyfikator uprawnień oraz adres pacjenta.

```
CREATE OR REPLACE FUNCTION Pobierz_Informacje_Pacjent(p_pacjent_id INTEGER)
RETURN SYS_REFCURSOR AS pacjent_cursor SYS_REFCURSOR;
```

```
BEGIN
```

```
    OPEN pacjent_cursor FOR
```

```
    SELECT
```

```
        p.imie,
```

```
        p.nazwisko,
```

```
        p.data_urodzenia,
```

```
        p.pesel,
```

```
        p.numer_telefonu,
```

```
        p.uprawnienia_id_uprawnienia,
```

```
        a.ulica,
```

```
        a.nr_lokalu,
```

```
        ka.miasto,
```

```
        ka.wojewodztwo
```

```
    FROM
```

```
        pacjenci p
```

```
    JOIN adresy a ON p.adresy_id_adresu = a.id_adresu
```

```
    JOIN kod_adres ka ON a.kod_adres_kod_pocztowy = ka.kod_pocztowy
```

```
    WHERE
```

```
        p.id_pacjenta = p_pacjent_id;
```

```
    RETURN pacjent_cursor;
```

```
END Pobierz_Informacje_Pacjent;
```

- Funkcja Sprawdz_Pacjenta:

Opis: Ta funkcja pozwala sprawdzić, czy pacjent o podanym PESEL już istnieje w bazie danych. Zwraca true, jeśli istnieje lub false w przeciwnym razie.

Wynik: Zwraca BOOLEAN.

```
CREATE OR REPLACE FUNCTION Sprawdz_Pacjenta(p_pesel VARCHAR2)
```

```
RETURN BOOLEAN
```

```
IS
```

```

    v_count NUMBER;
BEGIN
    -- Sprawdzenie, czy pacjent o podanym PESEL istnieje
    SELECT COUNT(*)
    INTO v_count
    FROM pacjenci
    WHERE pesel = p_pesel;
    -- Jeśli liczba rekordów jest większa niż 0, to pacjent istnieje
    IF v_count > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        -- Obsługa wyjątków (możesz dostosować obsługę błędów według potrzeb)
        RETURN FALSE;
END Sprawdz_Pacjenta;

```

- **Funkcja Pobierz_Liste_Pacjentow:**

Opis: Ta funkcja zwraca listę wszystkich pacjentów w bazie danych w postaci kursora.

Wynik: Zwraca kursor zawierający dane pacjentów.

```

CREATE OR REPLACE FUNCTION Pobierz_Liste_Pacjentow
RETURN SYS_REFCURSOR IS
    pacjenci_cursor SYS_REFCURSOR;
BEGIN
    OPEN pacjenci_cursor FOR
        SELECT *
        FROM pacjenci;

    RETURN pacjenci_cursor;
END Pobierz_Liste_Pacjentow;

```


- **Funkcja Pobierz_Pacjenta_Z_Uprawnieniem:**

Opis: Ta funkcja pobiera listę pacjentów posiadających określone uprawnienie na podstawie identyfikatora uprawnienia.

Wynik: Zwraca kursor zawierający dane pacjentów.

```
CREATE OR REPLACE FUNCTION Pobierz_Pacjenta_Z_Uprawnieniem(p_uprawnienie_id
INTEGER)
```

```
RETURN SYS_REFCURSOR
```

```
AS
```

```
    pacjenci_cursor SYS_REFCURSOR;
```

```
BEGIN
```

```
    OPEN pacjenci_cursor FOR
```

```
        SELECT p.id_pacjenta, p.imie, p.nazwisko, p.data_urodzenia, p.pesel, p.numer_telefonu
```

```
        FROM pacjenci p
```

```
        JOIN uprawnienia_pracownicy up ON p.uprawnienia_id_uprawnienia =
up.uprawnienia_id_uprawnienia
```

```
        WHERE up.uprawnienia_id_uprawnienia = p_uprawnienie_id;
```

```
    RETURN pacjenci_cursor;
```

```
END Pobierz_Pacjenta_Z_Uprawnieniem;
```

Pakiet II

Procedury

- **Procedura Przydziel_Wizyte:**

Opis: Ta procedura pozwala przydzielenie wizyty do danego pracownika Przyjmuje dane osobowe pracownika. Pozwala na przydzielenie pracownika do danej wizyty.

Wynik: Procedura nie zwraca wartości.

```
CREATE OR REPLACE PROCEDURE Przydziel_Wizyte(
```

```
    p_id_wizyty          IN INTEGER,
```

```
    p_id_pracownika      IN INTEGER,
```

```
    p_data_wizyty        IN DATE,
```

```
    p_godzina_rozpoczecia IN TIMESTAMP,
```

```
p_godzina_zakonczenia    IN TIMESTAMP,  
p_opis_wizyty            IN VARCHAR2,  
p_status_wizyty          IN VARCHAR2,  
p_skierowania_id_skierowania IN INTEGER,  
p_recepty_id_recepty     IN INTEGER,  
p_badania_id_badania     IN INTEGER,  
p_pacjenci_id_pacjenta   IN INTEGER  
)
```

IS

BEGIN

```
INSERT INTO rejestracja (  
    id_wizyty,  
    data_wizyty,  
    godzina_rozpoczecia,  
    godzina_zakonczenia,  
    opis_wizyty,  
    status_wizyty,  
    skierowania_id_skierowania,  
    recepty_id_recepty,  
    badania_id_badania,  
    pracownicy_id_pracownika,  
    pacjenci_id_pacjenta  
)
```

```
VALUES (  
    p_id_wizyty,  
    p_data_wizyty,  
    p_godzina_rozpoczecia,  
    p_godzina_zakonczenia,  
    p_opis_wizyty,  
    p_status_wizyty,  
    p_skierowania_id_skierowania,  
    p_recepty_id_recepty,
```

```

        p_badiana_id_badiana,
        p_id_pracownika,
        p_pacjenci_id_pacjenta
    );
END Przydziel_Wizyte;

```

- **Procedura Dodaj_Pracownika:**

Opis: Ta procedura pozwala na dodanie nowego pracownika do bazy danych. Przyjmuje dane osobowe pracownika, takie jak imię, nazwisko, numer telefonu, rola pracownika oraz adres.

Wynik: Procedura nie zwraca wartości.

```

CREATE OR REPLACE PROCEDURE Dodaj_Pracownika (
    p_imie VARCHAR2,
    p_nazwisko VARCHAR2,
    p_telefon VARCHAR2,
    p_rola_pracownika_id NUMBER,
    p_adres_id NUMBER
)
IS
    v_id_pracownika NUMBER;
BEGIN
    SELECT MAX(id_pracownika) + 1 INTO v_id_pracownika FROM pracownicy;

    INSERT INTO pracownicy (id_pracownika, imie, nazwisko, telefon, rola_pracownika_id_rola,
adresy_id_adresu)
    VALUES (v_id_pracownika, p_imie, p_nazwisko, p_telefon, p_rola_pracownika_id, p_adres_id);

    DBMS_OUTPUT.PUT_LINE('Dodano pracownika o identyfikatorze ' || v_id_pracownika);

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
        ROLLBACK;
END Dodaj_Pracownika;

```

- Procedura Aktualizuj_Pracownika:

Opis: Ta procedura pozwala na aktualizację danych pracownika w bazie danych na podstawie jego identyfikatora. Można zmienić dane osobowe, numer telefonu oraz adres pracownika.

Wynik: Procedura nie zwraca wartości.

```
CREATE OR REPLACE PROCEDURE Aktualizuj_Pracownika(  
    p_id_pracownika      IN INTEGER,  
    p_imie               IN VARCHAR2,  
    p_nazwisko           IN VARCHAR2,  
    p_telefon            IN VARCHAR2,  
    p_rola_pracownika_id_rola IN INTEGER,  
    p_adresy_id_adresu IN INTEGER  
)  
IS  
BEGIN  
    UPDATE pracownicy  
    SET  
        imie = p_imie,  
        nazwisko = p_nazwisko,  
        telefon = p_telefon,  
        rola_pracownika_id_rola = p_rola_pracownika_id_rola,  
        adresy_id_adresu = p_adresy_id_adresu  
    WHERE  
        id_pracownika = p_id_pracownika;  
  
    COMMIT;  
END Aktualizuj_Pracownika;
```

- Procedura Usun_Pracownika:

Opis: Ta procedura służy do usuwania pracownika z bazy danych na podstawie jego identyfikatora. Usuwa wszystkie związane z nim dane, takie jak wizyty, recepty, skierowania, itp.

Wynik: Procedura nie zwraca wartości.

```
CREATE OR REPLACE PROCEDURE Usun_Pracownika(p_id_pracownika IN INTEGER) AS  
BEGIN
```

```

DELETE FROM rejestracja WHERE pracownicy_id_pracownika = p_id_pracownika;

DELETE FROM recepty WHERE pracownicy_id_pracownika = p_id_pracownika;

DELETE FROM skierowania WHERE id_skierowania IN (SELECT skierowania_id_skierowania
FROM rejestracja WHERE pracownicy_id_pracownika = p_id_pracownika);

DELETE FROM dokumentacja_medyczna WHERE pacjenci_id_pacjenta IN (SELECT
pacjenci_id_pacjenta FROM rejestracja WHERE pracownicy_id_pracownika = p_id_pracownika);

DELETE FROM uprawnienia_pracownicy WHERE pracownicy_id_pracownika =
p_id_pracownika;

DELETE FROM pracownicy WHERE id_pracownika = p_id_pracownika;

COMMIT;

END;

```

Funkcje

- Funkcja Pobierz_Informacje_Pracownika:

Opis: Ta funkcja pozwala na pobranie informacji o pracowniku na podstawie jego identyfikatora. Zwraca dane osobowe pracownika oraz adres.

Wynik: Zwraca rekord zawierający imię, nazwisko, numer telefonu, rola pracownika oraz adres pracownika.

```

CREATE OR REPLACE FUNCTION Pobierz_Informacje_Pracownika(p_id_pracownika IN
INTEGER)

```

```

RETURN VARCHAR2

```

```

IS

```

```

    v_imie VARCHAR2(50);
    v_nazwisko VARCHAR2(70);
    v_numer_telefonu VARCHAR2(40);
    v_rola_pracownika VARCHAR2(50);
    v_ulica VARCHAR2(50);
    v_nr_lokalu VARCHAR2(50);
    v_kod_pocztowy VARCHAR2(50);
    v_miasto VARCHAR2(50);
    v_województwo VARCHAR2(50);
    v_informacje VARCHAR2(400);

```

```

BEGIN

```

```

    SELECT

```

pr.imie,
pr.nazwisko,
pr.telefon,
rp.stanowisko,
adr.ulica,
adr.nr_lokalu,
ka.kod_pocztowy,
ka.miasto,
ka.wojewodztwo

INTO

v_imie,
v_nazwisko,
v_numer_telefonu,
v_rola_pracownika,
v_ulica,
v_nr_lokalu,
v_kod_pocztowy,
v_miasto,
v_wojewodztwo

FROM pracownicy pr

JOIN rola_pracownika rp ON pr.rola_pracownika_id_rola = rp.id_rola_pracownika

JOIN adresy adr ON pr.adresy_id_adresu = adr.id_adresu

JOIN kod_adres ka ON adr.kod_adres_kod_pocztowy = ka.kod_pocztowy

WHERE pr.id_pracownika = p_id_pracownika;

v_informacje := 'Imię: ' || v_imie || CHR(10) ||

'Nazwisko: ' || v_nazwisko || CHR(10) ||

'Numer telefonu: ' || v_numer_telefonu || CHR(10) ||

'Rola pracownika: ' || v_rola_pracownika || CHR(10) ||

'Adres: ' || CHR(10) ||

' Ulica: ' || v_ulica || CHR(10) ||

' Nr lokalu: ' || v_nr_lokalu || CHR(10) ||

```

        ' Kod pocztowy: ' || v_kod_pocztowy || CHR(10) ||
        ' Miasto: ' || v_miasto || CHR(10) ||
        ' Województwo: ' || v_wojewodztwo;

RETURN v_informacje;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Brak danych dla podanego identyfikatora pracownika.';
    WHEN OTHERS THEN
        RETURN 'Wystąpił błąd: ' || SQLERRM;
END Pobierz_Informacje_Pracownika;

```

- Funkcja WizytyPracownika:

Opis: Ma na celu sprawdzenie istnienia pracownika o określonym numerze identyfikacyjnym w bazie danych. Funkcja ta zwraca kolekcję rekordów typu WizytaTypList, która zawiera informacje o wizytach przypisanych do danego pracownika. Wynik: Zwraca BOOLEAN.

Wynik: Funkcja zwraca kolekcję rekordów WizytaTypList.

```

FUNCTION WizytyPracownika(p_id_pracownika INTEGER) RETURN WizytaTypList PIPELINED
IS

```

```

    v_wizyta WIZYTATYP;
    TYPE WizytaTypListRec IS TABLE OF rejestracja%ROWTYPE;
    v_wizyty_list WizytaTypListRec;

```

```

BEGIN

```

```

    SELECT * BULK COLLECT INTO v_wizyty_list
    FROM rejestracja
    WHERE pracownicy_id_pracownika = p_id_pracownika;

```

```

    FOR i IN 1 .. v_wizyty_list.COUNT LOOP

```

```

        v_wizyta := WIZYTATYP(
            v_wizyty_list(i).id_wizyty,
            v_wizyty_list(i).data_wizyty,

```

```

        v_wizyty_list(i).godzina_rozpoczecia,
        v_wizyty_list(i).godzina_zakonczenia,
        v_wizyty_list(i).opis_wizyty,
        v_wizyty_list(i).status_wizyty,
        v_wizyty_list(i).skierowania_id_skierowania,
        v_wizyty_list(i).recepty_id_recepty,
        v_wizyty_list(i).badania_id_badania,
        v_wizyty_list(i).pracownicy_id_pracownika,
        v_wizyty_list(i).pacjenci_id_pacjenta
    );

    PIPE ROW(v_wizyta);
END LOOP;

RETURN;
END WizytyPracownika;

```

- **Funkcja Pobierz_Liste_Pracownikow:**

Opis: Ta funkcja zwraca listę wszystkich pracowników w bazie danych w postaci kursora.

Wynik: Zwraca kursor zawierający dane pracowników.

```

CREATE OR REPLACE FUNCTION Pobierz_Liste_Pracownikow RETURN SYS_REFCURSOR IS
    pracownicy_cursor SYS_REFCURSOR;
BEGIN
    OPEN pracownicy_cursor FOR
        SELECT id_pracownika, imie, nazwisko, telefon, rola_pracownika_id_rola, adresy_id_adresu
        FROM pracownicy;

    RETURN pracownicy_cursor;
END Pobierz_Liste_Pracownikow;

```

- **Funkcja Pobierz_Pracownikow_Z_Rola:**

Opis: Ta funkcja pobiera listę pracowników posiadających określoną rolę na podstawie identyfikatora roli.

Wynik: Zwraca kursor zawierający dane pracowników.

```
CREATE OR REPLACE FUNCTION Pobierz_Pracownikow_Z_Rola(p_id_rola IN INTEGER)
RETURN SYS_REFCURSOR IS
    v_cursor SYS_REFCURSOR;
BEGIN
    OPEN v_cursor FOR
        SELECT p.id_pracownika, p.imie, p.nazwisko, p.telefon, r.stanowisko, r.specjalizacja
        FROM pracownicy p
        JOIN rola_pracownika r ON p.rola_pracownika_id_rola = r.id_rola_pracownika
        WHERE r.id_rola_pracownika = p_id_rola;
    RETURN v_cursor;
END Pobierz_Pracownikow_Z_Rola;
```

3. Utworzenie pakietów

PAKIET I

create or replace PACKAGE Pakiet_1 AS

PROCEDURE Dodaj_Pacjenta (

 p_imie VARCHAR2,

 p_nazwisko VARCHAR2,

 p_data_urodzenia DATE,

 p_pesel VARCHAR2,

 p_numer_telefonu VARCHAR2,

 p_uprawnienia_id NUMBER,

 p_adres_id NUMBER

);

PROCEDURE Aktualizuj_Pacjenta (

 p_id_pacjenta INTEGER,

 p_nowe_imie VARCHAR2,

 p_nowe_nazwisko VARCHAR2,

 p_nowa_data_urodzenia DATE,

 p_nowy_pesel VARCHAR2,

 p_nowy_numer_telefonu VARCHAR2,

 p_nowe_id_adresu INTEGER

);

PROCEDURE Usun_Pacjenta (

 p_id_pacjenta IN INTEGER

);

```
PROCEDURE AnulujWizyte(  
    p_wizyta_id INTEGER  
);
```

```
FUNCTION Pobierz_Informacje_Pacjent(p_pacjent_id INTEGER) RETURN SYS_REFCURSOR;
```

```
FUNCTION Sprawdz_Pacjenta(p_pesel VARCHAR2) RETURN BOOLEAN;
```

```
FUNCTION Pobierz_Liste_Pacjentow RETURN SYS_REFCURSOR;
```

```
FUNCTION Pobierz_Pacjenta_Z_Uprawnieniem(p_uprawnienie_id INTEGER) RETURN  
SYS_REFCURSOR;
```

```
END Pakiet_1;
```

```
/
```

```
create or replace PACKAGE BODY Pakiet_1 AS
```

```
PROCEDURE Dodaj_Pacjenta (  
    p_imie VARCHAR2,  
    p_nazwisko VARCHAR2,  
    p_data_urodzenia DATE,  
    p_pesel VARCHAR2,  
    p_numer_telefonu VARCHAR2,  
    p_uprawnienia_id NUMBER,  
    p_adres_id NUMBER  
)  
IS  
    v_id_pacjenta NUMBER;
```

BEGIN

SELECT MAX(id_pacjenta) + 1 INTO v_id_pacjenta FROM pacjenci;

INSERT INTO pacjenci (

id_pacjenta,

imie,

nazwisko,

data_urodzenia,

pesel,

numer_telefonu,

uprawnienia_id_uprawnienia,

adresy_id_adresu

) VALUES (

v_id_pacjenta,

p_imie,

p_nazwisko,

p_data_urodzenia,

p_pesel,

p_numer_telefonu,

p_uprawnienia_id,

p_adres_id

);

COMMIT;

DBMS_OUTPUT.PUT_LINE('Pacjent dodany do bazy danych. ID pacjenta: ' || v_id_pacjenta);

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS_OUTPUT.PUT_LINE('Błąd podczas dodawania pacjenta: ' || SQLERRM);

```
END Dodaj_Pacjenta;
```

```
PROCEDURE Aktualizuj_Pacjenta (
```

```
    p_id_pacjenta      INTEGER,  
    p_nowe_imie        VARCHAR2,  
    p_nowe_nazwisko    VARCHAR2,  
    p_nowa_data_urodzenia  DATE,  
    p_nowy_pesel       VARCHAR2,  
    p_nowy_numer_telefonu  VARCHAR2,  
    p_nowe_id_adresu    INTEGER
```

```
)
```

```
AS
```

```
BEGIN
```

```
    UPDATE pacjenci
```

```
    SET
```

```
        imie = p_nowe_imie,  
        nazwisko = p_nowe_nazwisko,  
        data_urodzenia = p_nowa_data_urodzenia,  
        pesel = p_nowy_pesel,  
        numer_telefonu = p_nowy_numer_telefonu,  
        adresy_id_adresu = p_nowe_id_adresu
```

```
    WHERE
```

```
        id_pacjenta = p_id_pacjenta;
```

```
    COMMIT;
```

```
END Aktualizuj_Pacjenta;
```

```
PROCEDURE Usun_Pacjenta (
```

```
    p_id_pacjenta IN INTEGER
```

```
)
```

```
IS
```

```
BEGIN
```

```

DELETE FROM dokumentacja_medyczna WHERE pacjenci_id_pacjenta = p_id_pacjenta;

DELETE FROM rejestracja WHERE pacjenci_id_pacjenta = p_id_pacjenta;

DELETE FROM recepty WHERE pracownicy_id_pracownika IN (SELECT id_pracownika
FROM pracownicy WHERE adresy_id_adresu IN (SELECT id_adresu FROM adresy WHERE
id_adresu = p_id_pacjenta));

DELETE FROM uprawnienia_pracownicy WHERE pracownicy_id_pracownika IN (SELECT
id_pracownika FROM pracownicy WHERE adresy_id_adresu IN (SELECT id_adresu FROM adresy
WHERE id_adresu = p_id_pacjenta));

DELETE FROM pacjenci WHERE id_pacjenta = p_id_pacjenta;

COMMIT;

END;

```

```

PROCEDURE AnulujWizyte(
    p_wizyta_id INTEGER
) AS
BEGIN
    DELETE FROM rejestracja WHERE id_wizyty = p_wizyta_id;

    DBMS_OUTPUT.PUT_LINE('Wizyta o ID: ' || p_wizyta_id || ' została anulowana.');
```

EXCEPTION

```

    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE('Wystąpił błąd podczas anulowania wizyty.');
```

END AnulujWizyte;

```

FUNCTION Pobierz_Informacje_Pacjent(p_pacjent_id INTEGER) RETURN SYS_REFCURSOR
AS pacjent_cursor SYS_REFCURSOR;

BEGIN

    OPEN pacjent_cursor FOR

        SELECT

            p.imie,

            p.nazwisko,

```

```

        p.data_urodzenia,
        p.pesel,
        p.numer_telefonu,
        p.uprawnienia_id_uprawnienia,
        a.ulica,
        a.nr_lokalu,
        ka.miasto,
        ka.wojewodztwo
FROM
    pacjenci p
JOIN adresy a ON p.adresy_id_adresu = a.id_adresu
JOIN kod_adres ka ON a.kod_adres_kod_pocztowy = ka.kod_pocztowy
WHERE
    p.id_pacjenta = p_pacjent_id;
RETURN pacjent_cursor;
END Pobierz_Informacje_Pacjent;

```

```

FUNCTION Sprawdz_Pacjenta(p_pesel VARCHAR2) RETURN BOOLEAN
IS
    v_count NUMBER;
BEGIN

    SELECT COUNT(*)
    INTO v_count
    FROM pacjenci
    WHERE pesel = p_pesel;

    IF v_count > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;

```

EXCEPTION

WHEN OTHERS THEN

RETURN FALSE;

END Sprawdz_Pacjenta;

FUNCTION Pobierz_Liste_Pacjentow

RETURN SYS_REFCURSOR AS

pacjenci_cursor SYS_REFCURSOR;

BEGIN

OPEN pacjenci_cursor FOR

SELECT *

FROM pacjenci;

RETURN pacjenci_cursor;

END Pobierz_Liste_Pacjentow;

FUNCTION Pobierz_Pacjenta_Z_Uprawnieniem(p_uprawnienie_id INTEGER)

RETURN SYS_REFCURSOR

AS

pacjenci_cursor SYS_REFCURSOR;

BEGIN

OPEN pacjenci_cursor FOR

SELECT p.id_pacjenta, p.imie, p.nazwisko, p.data_urodzenia, p.pesel, p.numer_telefonu

FROM pacjenci p

JOIN uprawnienia_pracownicy up ON p.uprawnienia_id_uprawnienia =
up.uprawnienia_id_uprawnienia

WHERE up.uprawnienia_id_uprawnienia = p_uprawnienie_id;

RETURN pacjenci_cursor;

END Pobierz_Pacjenta_Z_Uprawnieniem;

END Pakiet_1;

PAKIET II

create or replace PACKAGE Pakiet_2 AS

```
PROCEDURE Przydziel_Wizyte(  
    p_id_pracownika      IN INTEGER,  
    p_data_wizyty        IN DATE,  
    p_godzina_rozpoczecia  IN VARCHAR2,  
    p_godzina_zakonczenia  IN VARCHAR2,  
    p_opis_wizyty         IN VARCHAR2,  
    p_status_wizyty       IN VARCHAR2,  
    p_skierowania_id_skierowania IN INTEGER,  
    p_recepty_id_recepty   IN INTEGER,  
    p_badania_id_badania   IN INTEGER,  
    p_pacjenci_id_pacjenta IN INTEGER  
);
```

```
PROCEDURE Dodaj_Pracownika (  
    p_imie VARCHAR2,  
    p_nazwisko VARCHAR2,  
    p_telefon VARCHAR2,  
    p_rola_pracownika_id NUMBER,  
    p_adres_id NUMBER  
);
```

```
PROCEDURE Aktualizuj_Pracownika(  
    p_id_pracownika      IN INTEGER,  
    p_imie                IN VARCHAR2,  
    p_nazwisko            IN VARCHAR2,  
    p_telefon             IN VARCHAR2,
```

```
p_rola_pracownika_id_rola IN INTEGER,  
p_adresy_id_adresu IN INTEGER  
);
```

```
PROCEDURE Usun_Pracownika(p_id_pracownika IN INTEGER);
```

```
FUNCTION Pobierz_Informacje_Pracownika(p_id_pracownika IN INTEGER) RETURN  
VARCHAR2;
```

```
FUNCTION Pobierz_Liste_Pracownikow RETURN SYS_REFCURSOR;
```

```
FUNCTION Pobierz_Pracownikow_Z_Rola(p_id_rola IN INTEGER) RETURN  
SYS_REFCURSOR;
```

```
TYPE WizytaTyp IS RECORD (  
    id_wizyty INTEGER,  
    data_wizyty DATE,  
    godzina_rozpoczecia TIMESTAMP,  
    godzina_zakonczenia TIMESTAMP,  
    opis_wizyty VARCHAR2(150 CHAR),  
    status_wizyty VARCHAR2(50 CHAR),  
    skierowania_id_skierowania INTEGER,  
    recepty_id_recepty INTEGER,  
    badania_id_badania INTEGER,  
    pracownicy_id_pracownika INTEGER,  
    pacjenci_id_pacjenta INTEGER  
);
```

```
TYPE WizytaTypList IS TABLE OF WizytaTyp;
```

```
FUNCTION WizytyPracownika(p_id_pracownika IN INTEGER) RETURN WizytaTypList  
PIPELINED;
```

END Pakiet_2;

/

create or replace PACKAGE BODY Pakiet_2 AS

PROCEDURE Przydziel_Wizyte(

 p_id_pracownika IN INTEGER,
 p_data_wizyty IN DATE,
 p_godzina_rozpoczecia IN VARCHAR2,
 p_godzina_zakonczenia IN VARCHAR2,
 p_opis_wizyty IN VARCHAR2,
 p_status_wizyty IN VARCHAR2,
 p_skierowania_id_skierowania IN INTEGER,
 p_recepty_id_recepty IN INTEGER,
 p_badania_id_badania IN INTEGER,
 p_pacjenci_id_pacjenta IN INTEGER

)

IS

 v_id_wizyty rejestracja.id_wizyty%TYPE;

BEGIN

 SELECT NVL(MAX(id_wizyty), 0) + 1 INTO v_id_wizyty FROM rejestracja;

 INSERT INTO rejestracja (

 id_wizyty,
 data_wizyty,
 godzina_rozpoczecia,
 godzina_zakonczenia,
 opis_wizyty,
 status_wizyty,
 skierowania_id_skierowania,

```

    recepty_id_recepty,
    badania_id_badania,
    pracownicy_id_pracownika,
    pacjenci_id_pacjenta
)
VALUES (
    v_id_wizyty,
    p_data_wizyty,
    (TO_TIMESTAMP(p_godzina_rozpoczecia, 'DD-MON-YYYY HH24:MI')),
    (TO_TIMESTAMP(p_godzina_zakonczenia, 'DD-MON-YYYY HH24:MI')),
    p_opis_wizyty,
    p_status_wizyty,
    p_skierowania_id_skierowania,
    p_recepty_id_recepty,
    p_badania_id_badania,
    p_id_pracownika,
    p_pacjenci_id_pacjenta
);
END Przydziel_Wizyte;

```

```

PROCEDURE Dodaj_Pracownika (
    p_imie VARCHAR2,
    p_nazwisko VARCHAR2,
    p_telefon VARCHAR2,
    p_rola_pracownika_id NUMBER,
    p_adres_id NUMBER
)
IS
    v_id_pracownika NUMBER;
BEGIN
    SELECT MAX(id_pracownika) + 1 INTO v_id_pracownika FROM pracownicy;
    INSERT INTO pracownicy (id_pracownika, imie, nazwisko, telefon, rola_pracownika_id_rola,
adresy_id_adresu)

```

```
VALUES (v_id_pracownika, p_imie, p_nazwisko, p_telefon, p_rola_pracownika_id, p_adres_id);
DBMS_OUTPUT.PUT_LINE('Dodano pracownika o identyfikatorze ' || v_id_pracownika);
COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
        ROLLBACK;
END Dodaj_Pracownika;
```

```
PROCEDURE Aktualizuj_Pracownika(
    p_id_pracownika      IN INTEGER,
    p_imie                IN VARCHAR2,
    p_nazwisko            IN VARCHAR2,
    p_telefon              IN VARCHAR2,
    p_rola_pracownika_id_rola IN INTEGER,
    p_adresy_id_adresu IN INTEGER
)
IS
BEGIN
    UPDATE pracownicy
    SET
        imie = p_imie,
        nazwisko = p_nazwisko,
        telefon = p_telefon,
        rola_pracownika_id_rola = p_rola_pracownika_id_rola,
        adresy_id_adresu = p_adresy_id_adresu
    WHERE
        id_pracownika = p_id_pracownika;

    COMMIT;
END Aktualizuj_Pracownika;
```

```

PROCEDURE Usun_Pracownika(p_id_pracownika IN INTEGER) AS
BEGIN
    DELETE FROM rejestracja WHERE pracownicy_id_pracownika = p_id_pracownika;
    DELETE FROM recepty WHERE pracownicy_id_pracownika = p_id_pracownika;
    DELETE FROM skierowania WHERE id_skierowania IN (SELECT skierowania_id_skierowania
FROM rejestracja WHERE pracownicy_id_pracownika = p_id_pracownika);
    DELETE FROM dokumentacja_medyczna WHERE pacjenci_id_pacjenta IN (SELECT
pacjenci_id_pacjenta FROM rejestracja WHERE pracownicy_id_pracownika = p_id_pracownika);
    DELETE FROM uprawnienia_pracownicy WHERE pracownicy_id_pracownika =
p_id_pracownika;
    DELETE FROM pracownicy WHERE id_pracownika = p_id_pracownika;
    COMMIT;
END Usun_Pracownika;

```

```

FUNCTION Pobierz_Informacje_Pracownika(p_id_pracownika IN INTEGER) RETURN
VARCHAR2

```

```

IS

```

```

    v_imie VARCHAR2(50);
    v_nazwisko VARCHAR2(70);
    v_numer_telefonu VARCHAR2(40);
    v_rola_pracownika VARCHAR2(50);
    v_ulica VARCHAR2(50);
    v_nr_lokalu VARCHAR2(50);
    v_kod_pocztowy VARCHAR2(50);
    v_miasto VARCHAR2(50);
    v_wojewodztwo VARCHAR2(50);
    v_informacje VARCHAR2(400);

```

```

BEGIN

```

```

    SELECT

```

```

        pr.imie,
        pr.nazwisko,
        pr.telefon,
        rp.stanowisko,

```

```

    adr.ulica,
    adr.nr_lokalu,
    ka.kod_pocztowy,
    ka.miasto,
    ka.województwo
INTO
    v_imie,
    v_nazwisko,
    v_numer_telefonu,
    v_rola_pracownika,
    v_ulica,
    v_nr_lokalu,
    v_kod_pocztowy,
    v_miasto,
    v_województwo
FROM pracownicy pr
JOIN rola_pracownika rp ON pr.rola_pracownika_id_rola = rp.id_rola_pracownika
JOIN adresy adr ON pr.adresy_id_adresu = adr.id_adresu
JOIN kod_adres ka ON adr.kod_adres_kod_pocztowy = ka.kod_pocztowy
WHERE pr.id_pracownika = p_id_pracownika;

v_informacje := 'Imię: ' || v_imie || CHR(10) ||
    'Nazwisko: ' || v_nazwisko || CHR(10) ||
    'Numer telefonu: ' || v_numer_telefonu || CHR(10) ||
    'Rola pracownika: ' || v_rola_pracownika || CHR(10) ||
    'Adres: ' || CHR(10) ||
    ' Ulica: ' || v_ulica || CHR(10) ||
    ' Nr lokalu: ' || v_nr_lokalu || CHR(10) ||
    ' Kod pocztowy: ' || v_kod_pocztowy || CHR(10) ||
    ' Miasto: ' || v_miasto || CHR(10) ||
    ' Województwo: ' || v_województwo;

```

```
    RETURN v_informacje;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Brak danych dla podanego identyfikatora pracownika.';
    WHEN OTHERS THEN
        RETURN 'Wystąpił błąd: ' || SQLERRM;
END Pobierz_Informacje_Pracownika;
```

```
FUNCTION Pobierz_Liste_Pracownikow RETURN SYS_REFCURSOR
IS
    pracownicy_cursor SYS_REFCURSOR;
BEGIN
    OPEN pracownicy_cursor FOR
        SELECT id_pracownika, imie, nazwisko, telefon, rola_pracownika_id_rola, adresy_id_adresu
        FROM pracownicy;

    RETURN pracownicy_cursor;
END Pobierz_Liste_Pracownikow;
```

```
FUNCTION Pobierz_Pracownikow_Z_Rola(p_id_rola IN INTEGER) RETURN
SYS_REFCURSOR
IS
    v_cursor SYS_REFCURSOR;
BEGIN
    OPEN v_cursor FOR
        SELECT p.id_pracownika, p.imie, p.nazwisko, p.telefon, r.stanowisko, r.specjalizacja
        FROM pracownicy p
        JOIN rola_pracownika r ON p.rola_pracownika_id_rola = r.id_rola_pracownika
        WHERE r.id_rola_pracownika = p_id_rola;

    RETURN v_cursor;
END Pobierz_Pracownikow_Z_Rola;
```



```
FUNCTION WizytyPracownika(p_id_pracownika INTEGER) RETURN WizytyTypList  
PIPELINED IS
```

```
    v_wizyta WIZYTATYP;
```

```
    TYPE WizytyTypListRec IS TABLE OF rejestracja%ROWTYPE;
```

```
    v_wizyty_list WizytyTypListRec;
```

```
BEGIN
```

```
    SELECT * BULK COLLECT INTO v_wizyty_list
```

```
    FROM rejestracja
```

```
    WHERE pracownicy_id_pracownika = p_id_pracownika;
```

```
    FOR i IN 1 .. v_wizyty_list.COUNT LOOP
```

```
        v_wizyta := WIZYTATYP(
```

```
            v_wizyty_list(i).id_wizyty,
```

```
            v_wizyty_list(i).data_wizyty,
```

```
            v_wizyty_list(i).godzina_rozpoczecia,
```

```
            v_wizyty_list(i).godzina_zakonczenia,
```

```
            v_wizyty_list(i).opis_wizyty,
```

```
            v_wizyty_list(i).status_wizyty,
```

```
            v_wizyty_list(i).skierowania_id_skierowania,
```

```
            v_wizyty_list(i).recepty_id_recepty,
```

```
            v_wizyty_list(i).badania_id_badania,
```

```
            v_wizyty_list(i).pracownicy_id_pracownika,
```

```
            v_wizyty_list(i).pacjenci_id_pacjenta
```

```
        );
```

```
        PIPE ROW(v_wizyta);
```

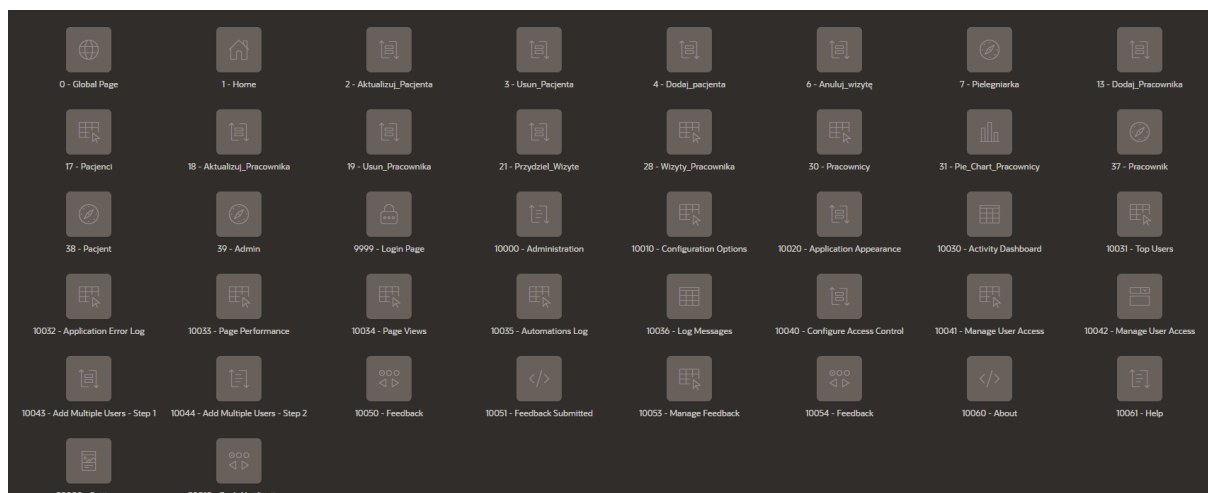
```
    END LOOP;
```

```
    RETURN;
```

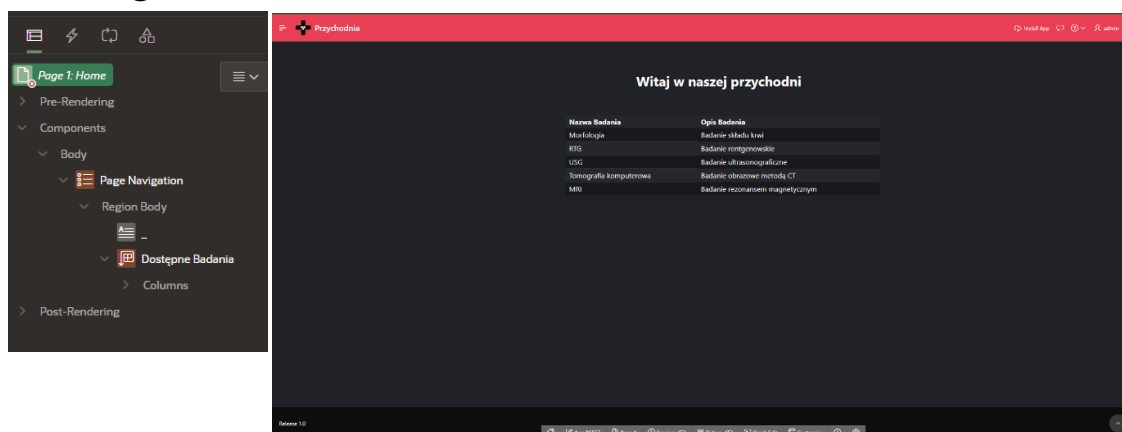
```
END WizytyPracownika;
```

```
END Pakiet_2;
```

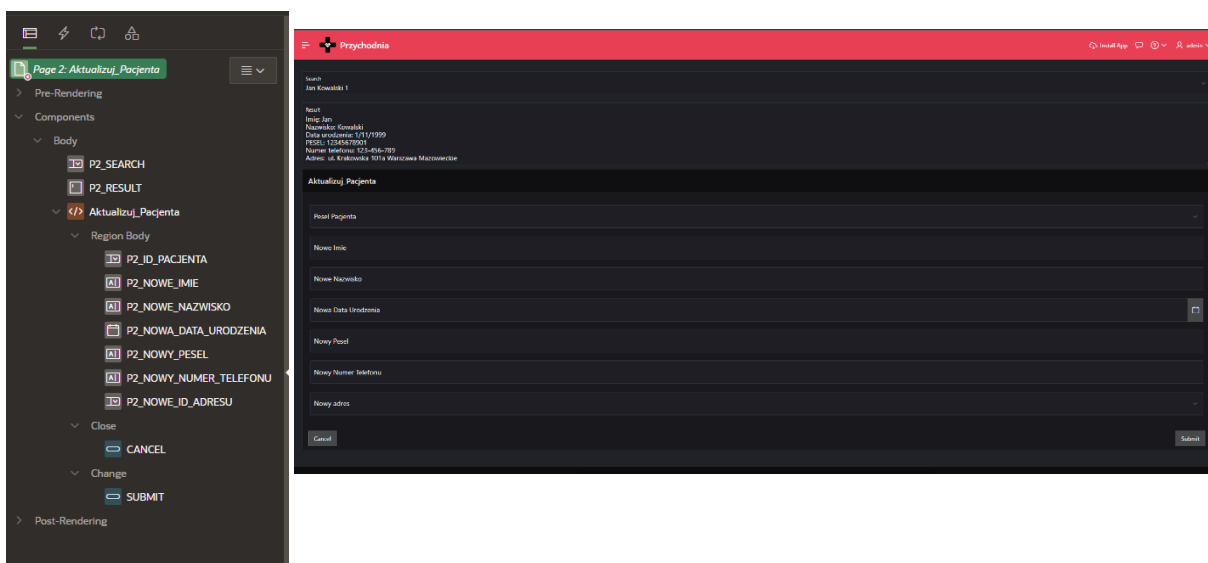
4. Webowy interfejs aplikacji w APEX



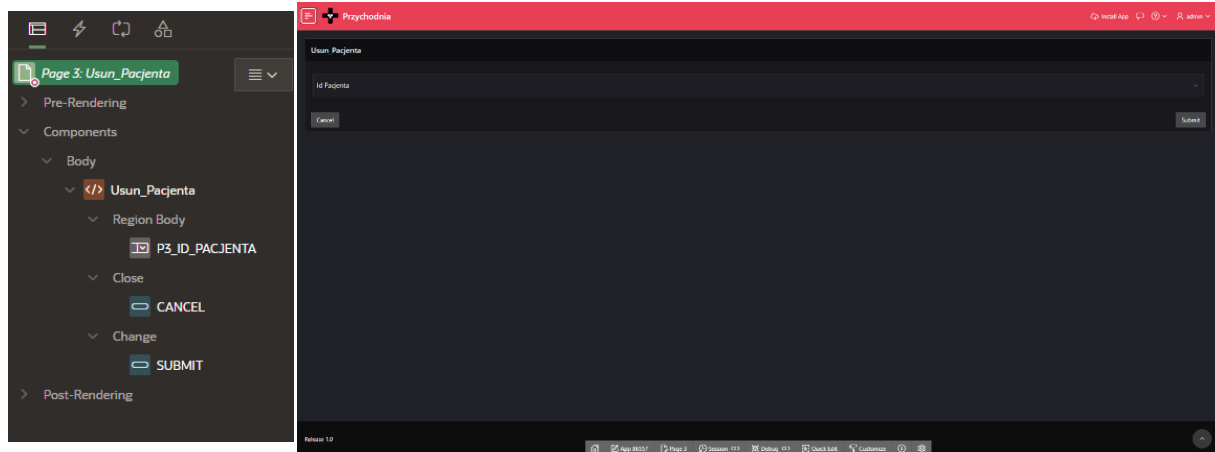
Strona główna



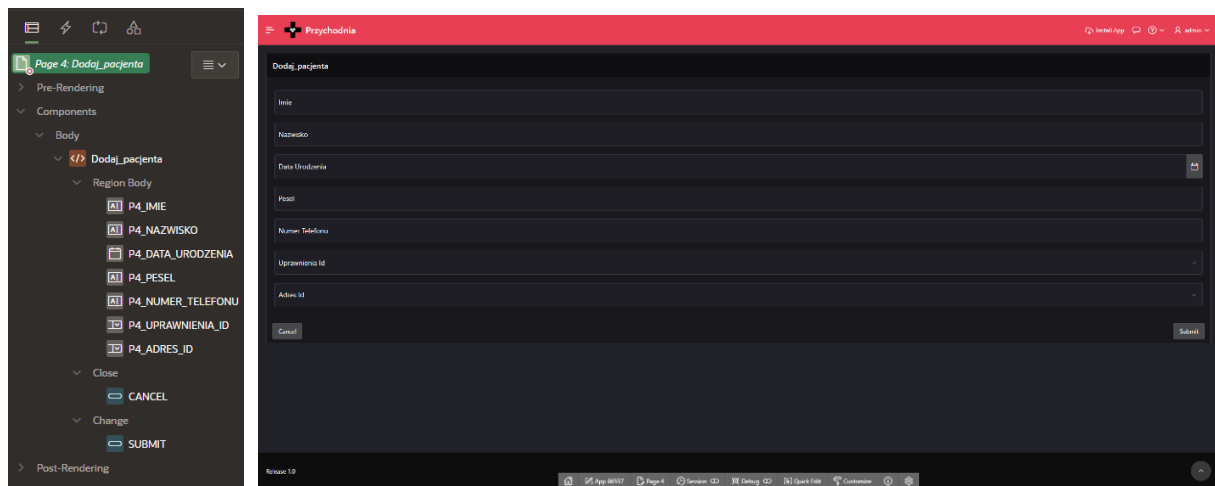
Panel do aktualizacji pacjenta



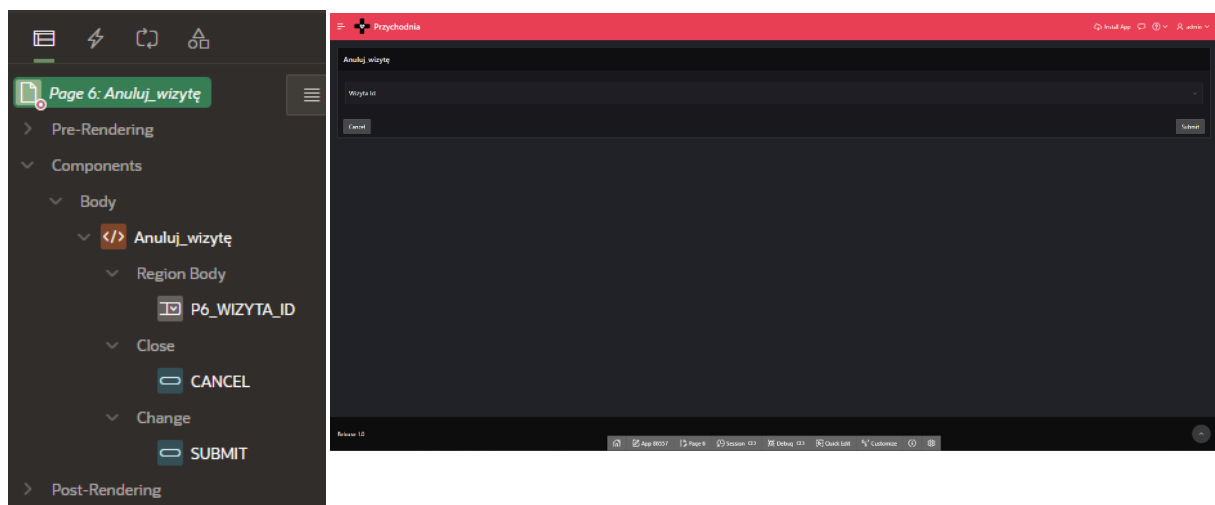
Panel do usuwania pacjenta



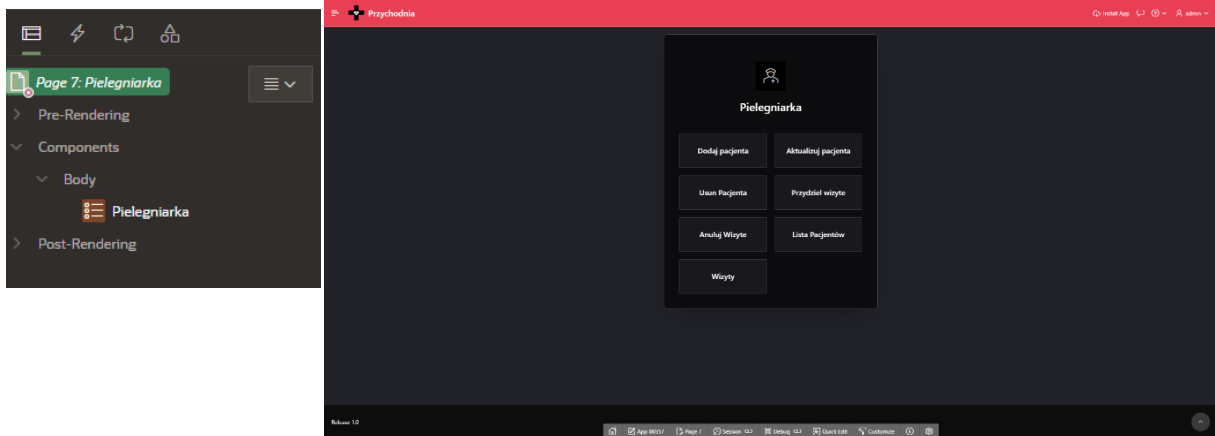
Panel do dodawania pacjenta



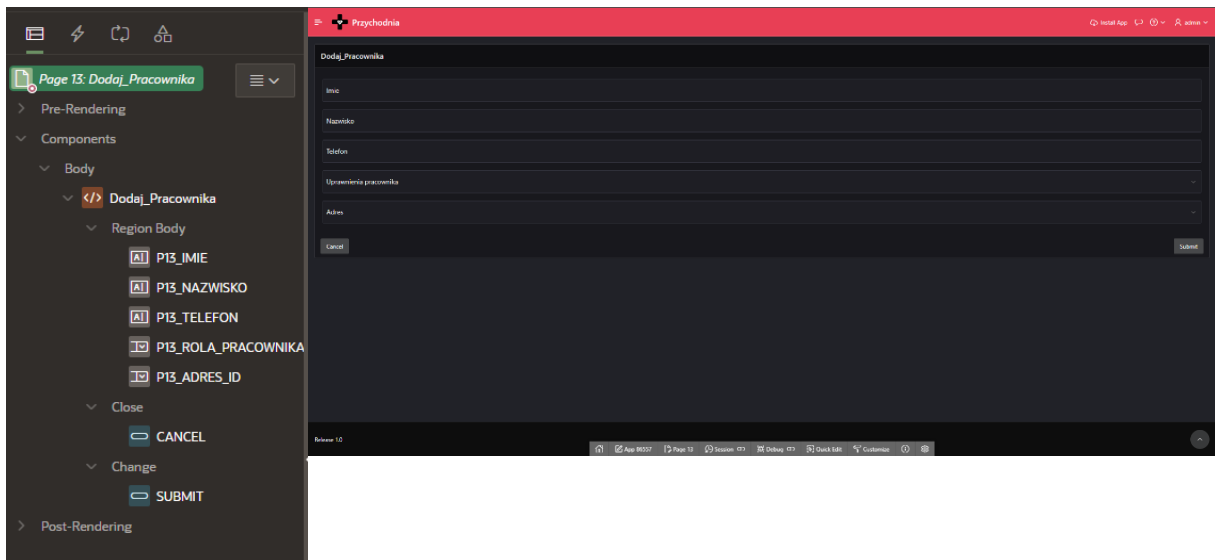
Panel do anulowania wizyty



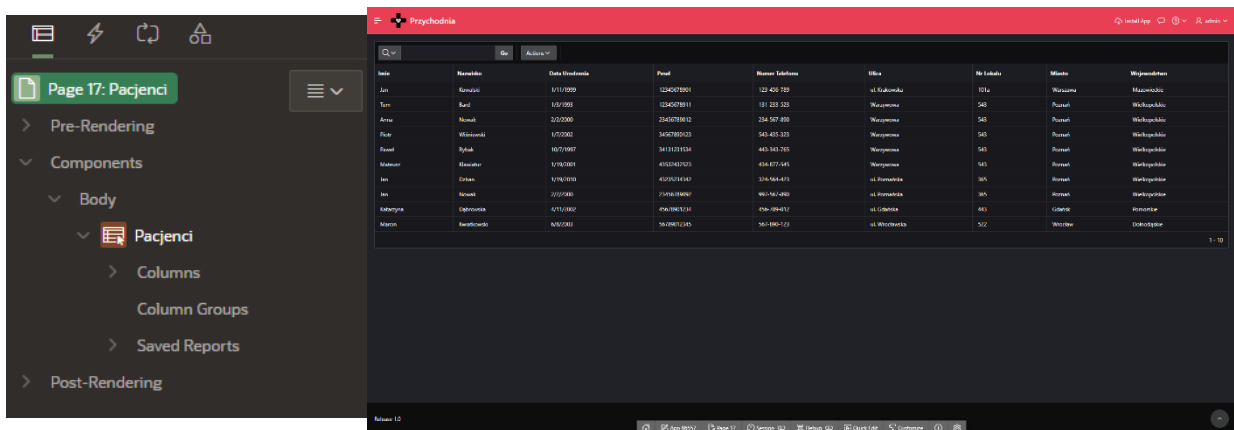
Panel pielęgniarce



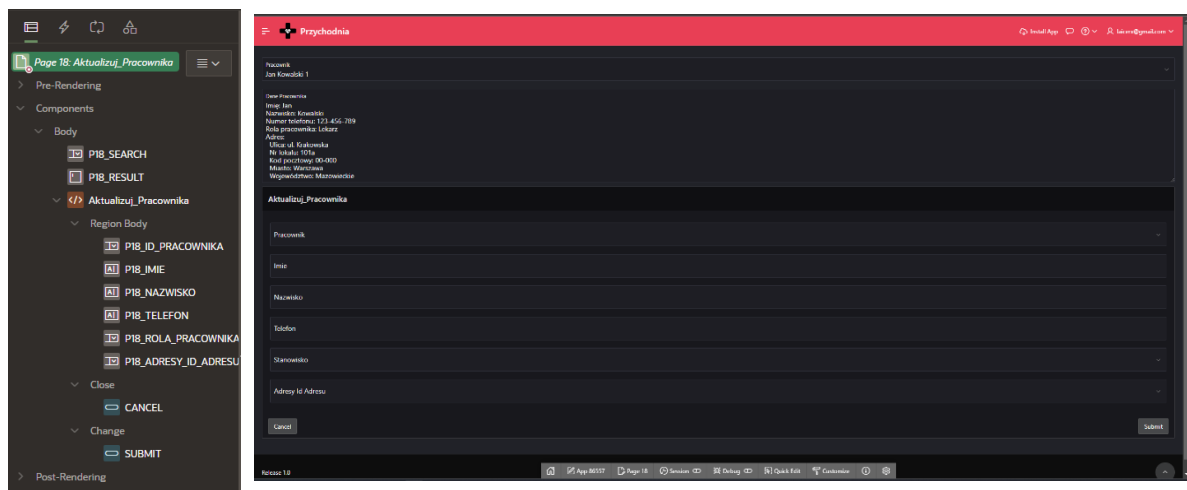
Panel dodaj pracownika



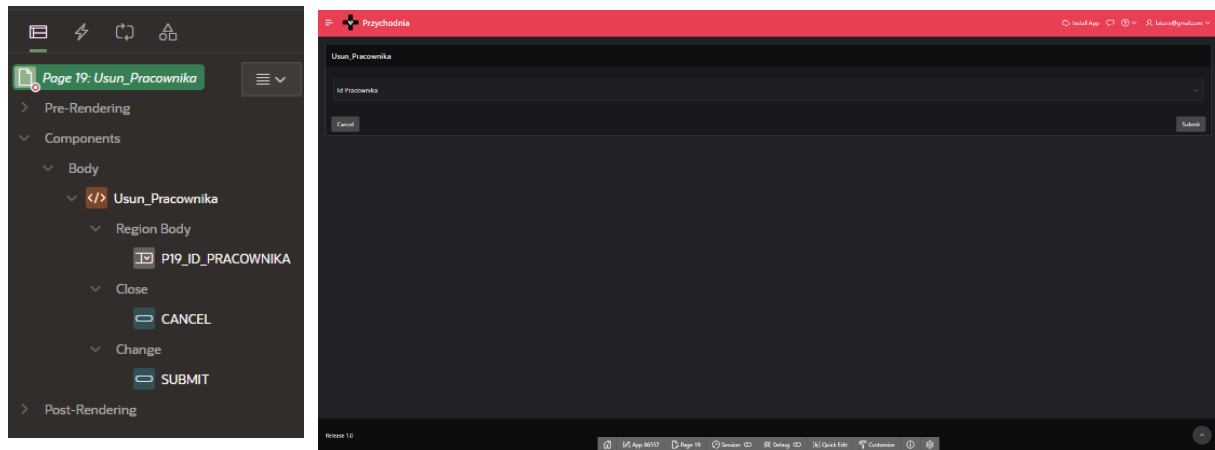
Strona z lista pacjentów



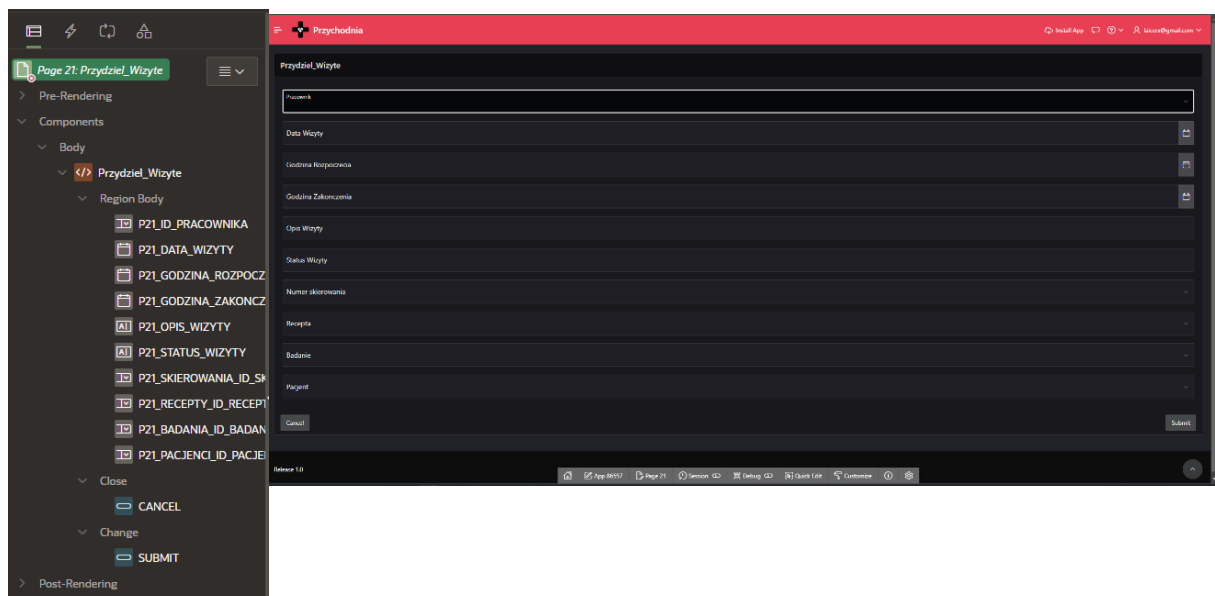
Panel aktualizuj dane pracownika



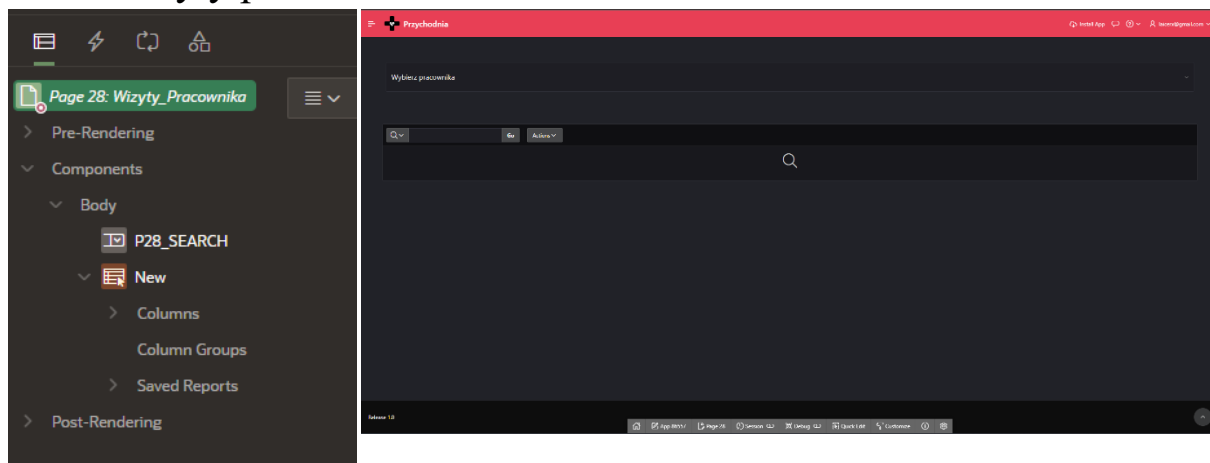
Panel usuń pracownika



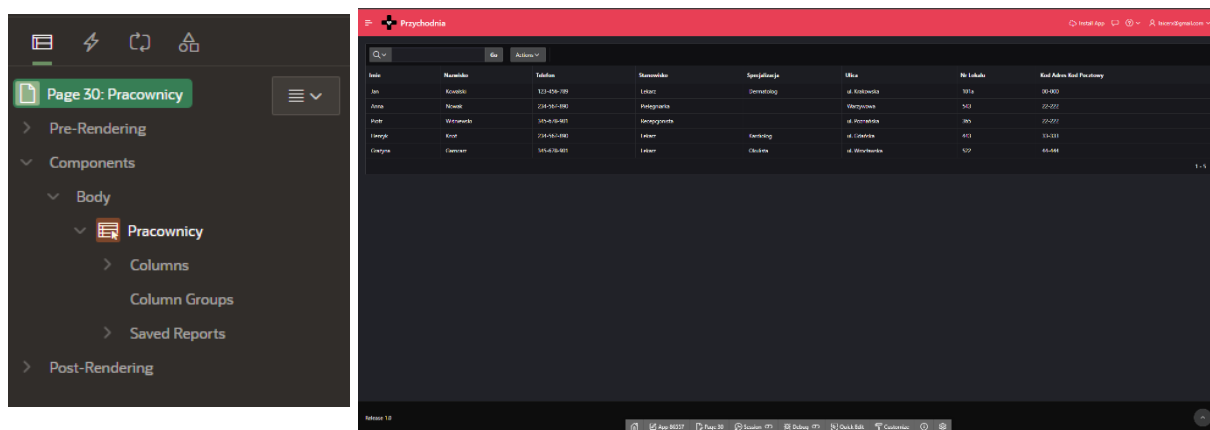
Panel przydziel wizytę



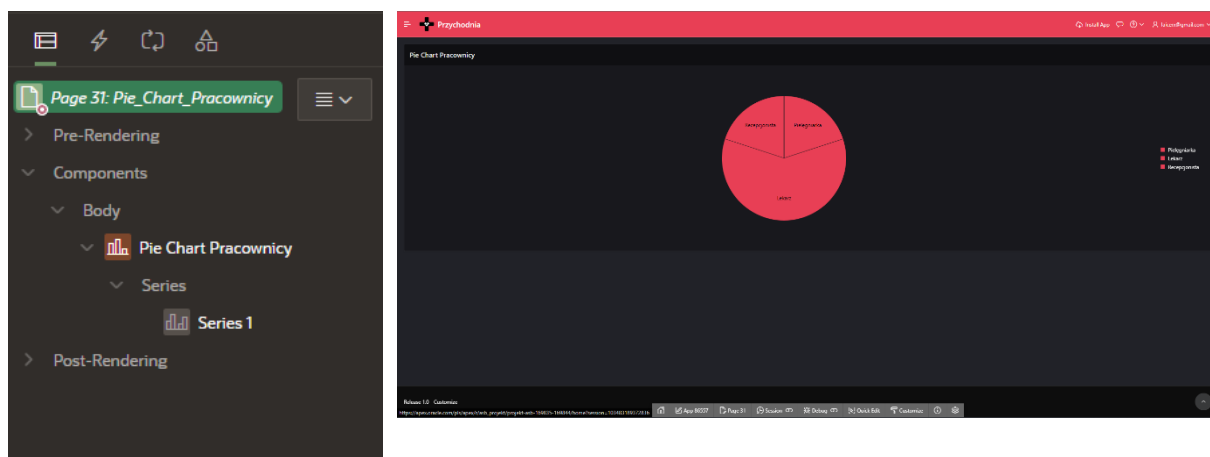
Panel wizyty pracownika



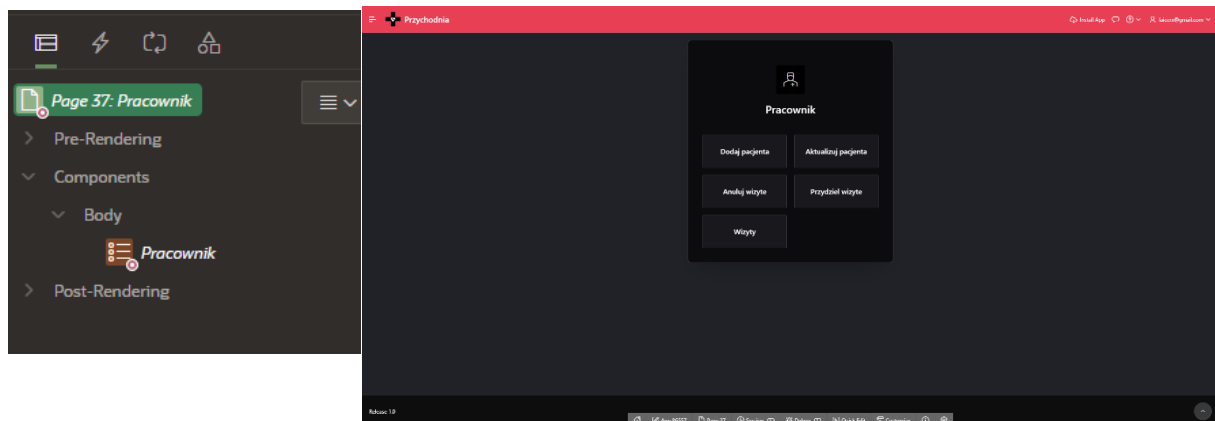
Strona z lista pracowników



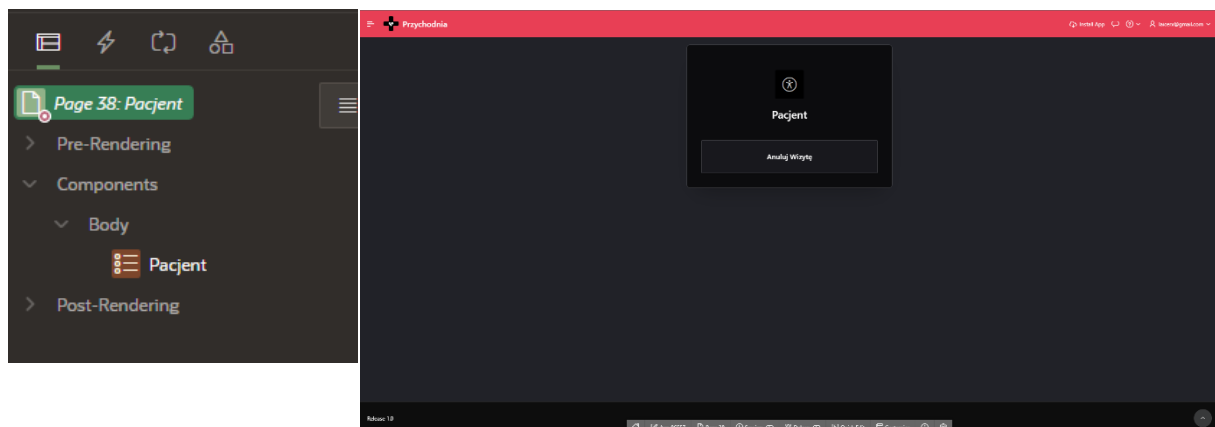
Strona z wykresem pracowników



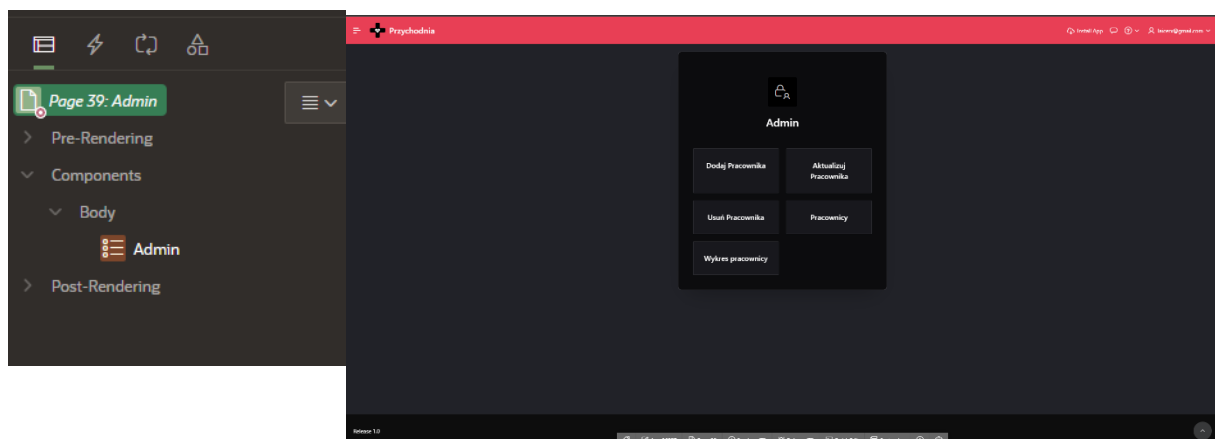
Panel pracownika



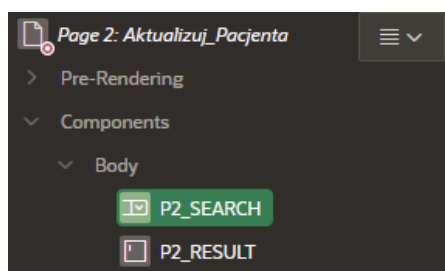
Panel pacjenta



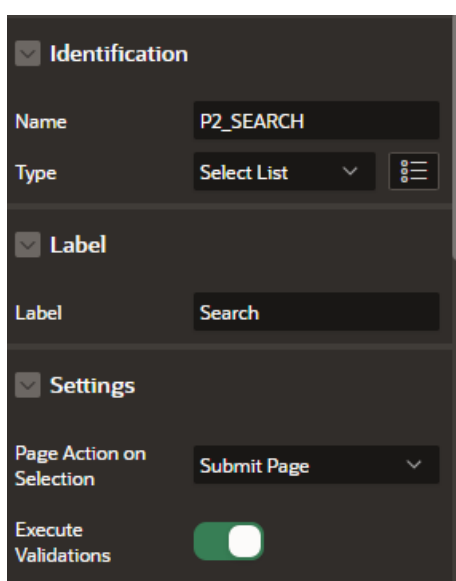
Panel Admina



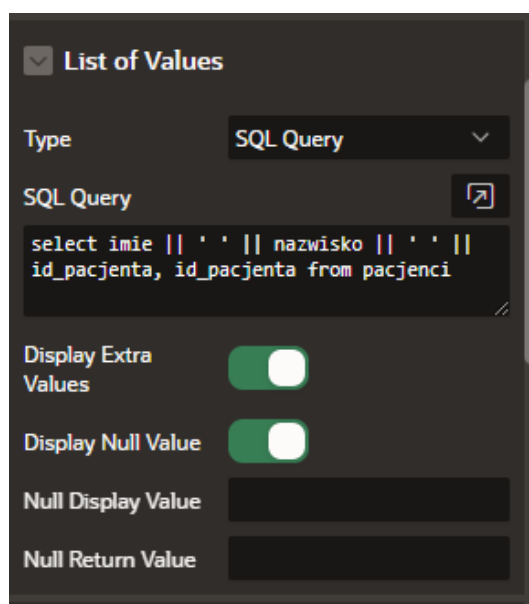
Utworzenie listy rozwijanej



Rozpoczynamy od wybrania z sekcji Items dwóch pól tekstowych text field. Jedno pozwoli na wprowadzenie danych, a drugie na ich wyświetlenie.

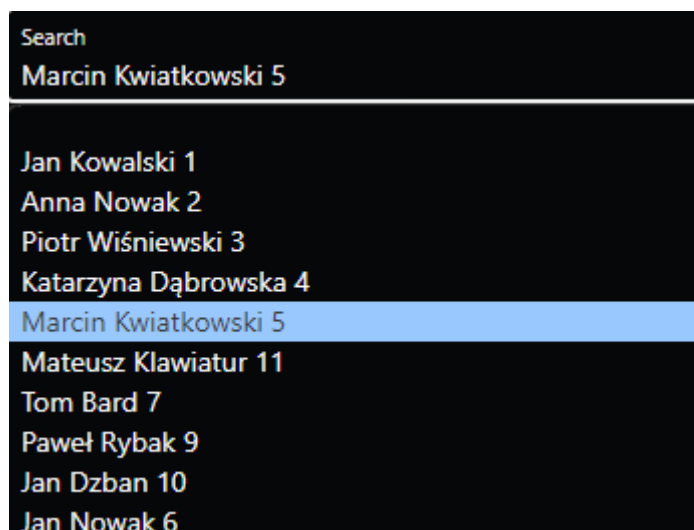


Jako typ ustawiamy select list a akcje po zmianie opcji na liście na submit page.

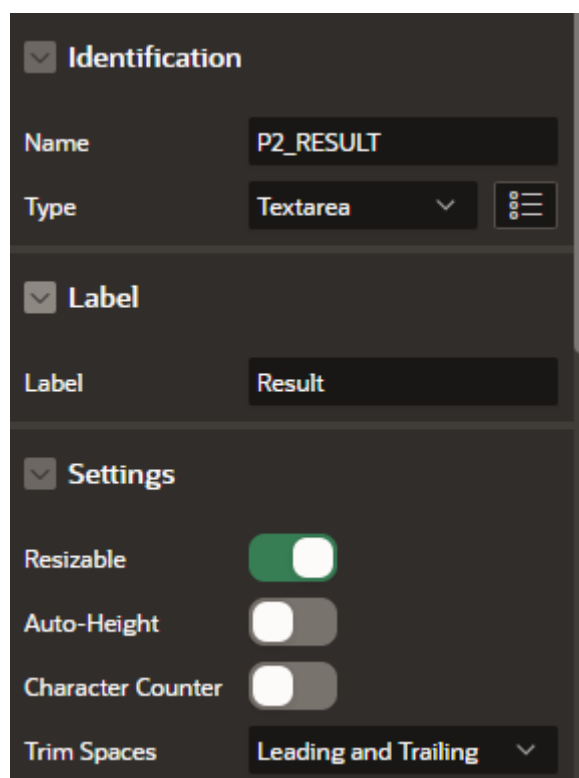


Typ wartości listy ustawiamy na SQL Query i za pomocą || ' ' || łączymy informacje które chcemy wyświetlić jako tekst rozwijanej listy. Druga wartość po przecinku to wartość, która zostanie przesłana po wybraniu opcji z listy. W tym przypadku jest to id pacjenta.

Uzyskany efekt:



A screenshot of a search dropdown menu. At the top, there is a search bar with the text "Search" and "Marcin Kwiatkowski 5". Below the search bar, a list of names and IDs is displayed. The names are in a light blue font, and the IDs are in a light orange font. The list includes: Jan Kowalski 1, Anna Nowak 2, Piotr Wiśniewski 3, Katarzyna Dąbrowska 4, Marcin Kwiatkowski 5 (highlighted with a blue background), Mateusz Klawiatur 11, Tom Bard 7, Paweł Rybak 9, Jan Dzban 10, and Jan Nowak 6.



A screenshot of a configuration panel for a text area. The panel is divided into three sections: Identification, Label, and Settings. In the Identification section, the Name is set to "P2_RESULT" and the Type is set to "Textarea". In the Label section, the Label is set to "Result". In the Settings section, the Resizable toggle is turned on, and the Auto-Height and Character Counter toggles are turned off. The Trim Spaces dropdown is set to "Leading and Trailing".

Drugi element typu textarea odpowiada za wyświetlenie wyniku funkcji

```

pacjent_info := Pakiet_1.Pobierz_Informacje_Pacjent(:P2_SEARCH);

FETCH pacjent_info INTO
    v_imie,
    v_nazwisko,
    v_data_urodzenia,
    v_pesel,
    v_numer_telefonu,
    v_uprawnienia_id,
    v_ulica,
    v_nr_lokalu,
    v_miasto,
    v_wojewodztwo;

```

```

IF pacjent_info%FOUND THEN
    l_result := 'Imię: ' || v_imie || CHR(10) ||
        'Nazwisko: ' || v_nazwisko || CHR(10) ||
        'Data urodzenia: ' || v_data_urodzenia || CHR(10) ||
        'PESEL: ' || v_pesel || CHR(10) ||
        'Numer telefonu: ' || v_numer_telefonu || CHR(10) ||
        'Adres: ' || ' ' || v_ulica || ' ' || v_nr_lokalu || ' ' || v_miasto || ' ' || v_wojewodztwo;
ELSE
    l_result := 'Brak danych dla podanego identyfikatora.';
END IF;

:P2_RESULT := l_result;

CLOSE pacjent_info;
EXCEPTION
    WHEN OTHERS THEN
        :P2_RESULT := 'Błąd: ' || SQLERRM;
        CLOSE pacjent_info;

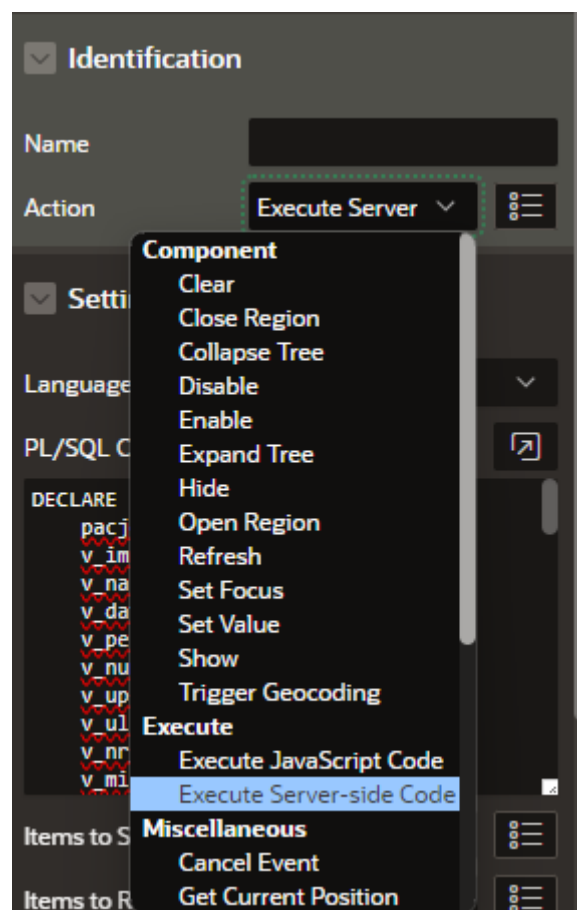
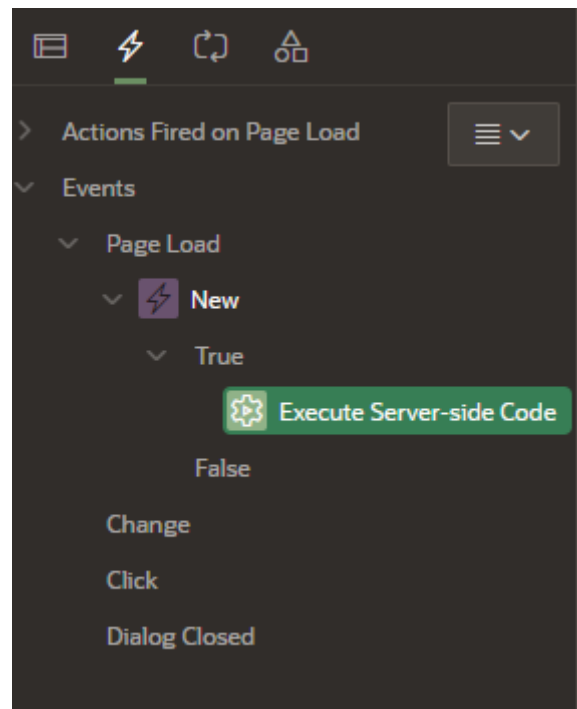
```

```

:P2_RESULT := l_result;

```

Wynik przypisujemy do elementu i wyświetlamy go po załadowaniu strony za pomocą utworzonej akcji dynamicznej.



Dzięki temu po wybraniu elementu z listy rozwijanej wykonywany jest kod po stronie serwera i wyświetlany jest wynik jego działania

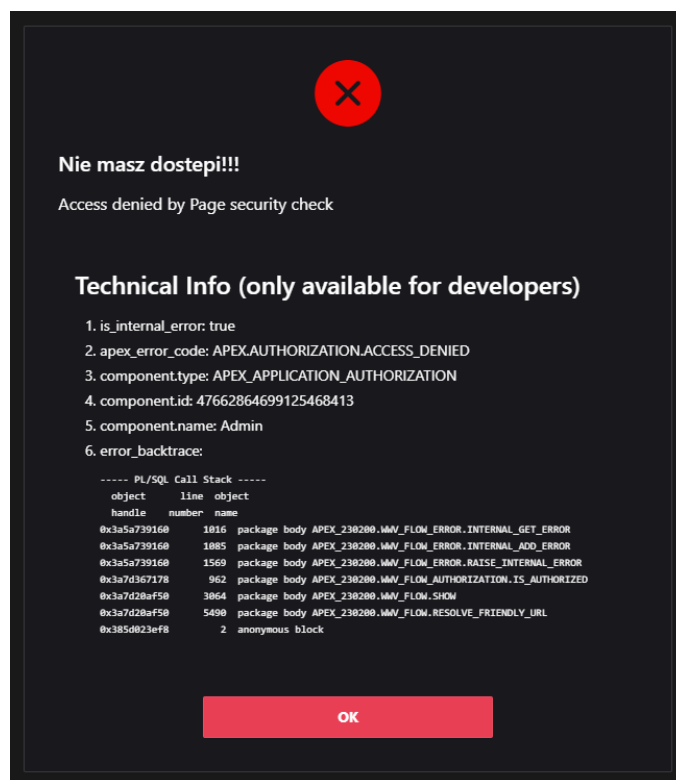
5. Autoryzacja użytkowników

Authorization Schemes						
Subscriptions						
Utilization						
History						
<div>Q</div> <div>Go</div> <div></div> <div></div> <div>Actions</div> <div>Copy</div> <div>Reset</div> <div>Create</div>						
Name	Type	Caching	Subscribed From	Subscribers	Updated	Copy
Admin	Is In Role or Group	Once per session			25 hours ago	
Administration Rights	Is In Role or Group	Once per page view			5 days ago	
Contribution Rights	Is In Role or Group	Once per page view			5 days ago	
Doktor	Is In Role or Group	Once per session			26 hours ago	
Pacjent	Is In Role or Group	Once per session			26 hours ago	
Pielęgniarka	Is In Role or Group	Once per session			24 hours ago	
Reader Rights	PL/SQL Function Returning Boolean	Once per session			5 days ago	
wspolna	Is In Role or Group	Once per session			24 hours ago	

Stworzyliśmy nowe schematy autoryzacji na podstawie roli dla każdej roli oraz jedna wspólna role dla lekarza oraz pielęgniarki. Authorization Schemes pozwalają na definiowanie reguł i warunków, które decydują, czy użytkownik ma uprawnienia do wykonania określonych działań w ramach aplikacji.

Application Access Control							
Show All							
Roles							
User Role Assignments							
Role	Static Identifier	Users	Description	Subscribed From	Subscription Status	Subscribers	Copy
Admin	ADMIN	3	-	-	-	-	
Administrator	ADMINISTRATOR	3	Role assigned to application administrators.	-	-	-	
Contributor	CONTRIBUTOR	3	Role assigned to application contributors.	-	-	-	
Doktor	DOKTOR	4	-	-	-	-	
Pacjent	PACIENT	4	-	-	-	-	
Pielęgniarka	PIELĘGNIARKA	4	-	-	-	-	
Reader	READER	3	Role assigned to application readers.	-	-	-	
wspolna	WSPOLNA	5	-	-	-	-	
							1 - 8
User Role Assignments							
<div>Q</div> <div>Go</div> <div>Actions</div> <div>Add User Role Assignment</div>							
User Name	Roles						
ADMIN	Admin, Administrator, Contributor, Doktor, Pacjent, Pielęgniarka, Reader, wspolna						
DOKTOR	Doktor, wspolna						
LAICERX@GMAIL.COM	Admin, Administrator, Contributor, Doktor, Pacjent, Pielęgniarka, Reader, wspolna						
PACIENT	Pacjent						
PIELĘGNIARKA	Pielęgniarka, wspolna						
PS	Admin, Administrator, Contributor, Doktor, Pacjent, Pielęgniarka, Reader, wspolna						
							1 - 6

A następnie przypisaliśmy każdemu użytkownikowi odpowiednie role. Role są zbiorami uprawnień, które określają, jakie czynności i zasoby są dostępne dla użytkowników należących do danej roli.



Dzięki utworzonej autoryzacji nikt nie powołany nie ma dostępu do stron do których nie powinien mieć dostępu.

6.Podsumowanie

Utworzona przez nas w Oracle Apex aplikacja to system zarządzania bazą danych dla przychodni lekarskiej. Składa się z dwóch pakietów: Pakiet_1 i Pakiet_2, które obsługują odpowiednio operacje związane z pacjentami jak i operacje związane z personelem medycznym. W Pakiecie_1 zaimplementowane zostały procedury dodawania, aktualizacji i usuwania pacjentów, odwoływania wizyt oraz funkcje pobierania informacji o pacjentach, sprawdzania istnienia pacjenta o określonym PESEL-u i pobierania list pacjentów na podstawie różnych kryteriów. Pakiet_2 dotyczy funkcji personelu medycznego, w tym dodawania, aktualizowania i usuwania personelu, przypisywania wizyt oraz pobierania informacji o personelu i list personelu. Strony podzielone są na panele do zarządzania pacjentami, personelem i administracją. Zawierają listy rozwijane i panele do manipulacji danymi. Uwierzytelnianie użytkowników jest realizowane za pomocą ról wbudowanych w Oracle Apex zapewniając, że dostęp do różnych funkcji jest kontrolowany i ograniczony do odpowiednich użytkowników.