



POLITECHNIKA RZESZOWSKA
im. Ignacego Łukasiewicza
WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

Aplikacje bazodanowe
Aplikacja do zarządzania klubem fitness w APEX

Małgorzata Radomska 173203
Adrianna Rapa 173204

Inżynieria i Analiza Danych

Rzeszów 2025

Spis treści

Wstęp	3
Cel projektu	3
Projekt aplikacji	4
Schemat bazy.....	4
Diagram przypadków użycia	4
Pakiety – diagramy i implementacja w PL/SQL.....	6
Pakiet Klient.....	6
Pakiet Klient – Funkcje	7
Pakiet Klient – Procedury	12
Pakiet Administrator	18
Pakiet Administrator – Funkcje	19
Pakiet Klient – Procedury	25
Implementacja aplikacji w APEX	32
Mapa aplikacji.....	32
Wygląd i implementacja poszczególnych stron.....	33
Strona główna – Home	33
Strona główna – Home Klient (Zarządzanie klientami)	34
Strona dodawania nowego klienta.....	34
Strona do wyszukiwania rezerwacji klienta.....	37
Strona do anulowania rezerwacji klienta	39
Strona do wyszukiwania zajęć.....	40
Strona do dodawania rezerwacji na zajęcia	42
Strona do wysyłania przypomnień	45
Strona główna – Home Klub (Zarządzanie klubem)	47
Strona do tworzenia nowej grupy zajęciowej	47
Strona z harmonogramem zajęć	50
Strona generująca raport wypłat instruktorów.....	51
Strona generująca raport efektywności instruktorów.....	53
Strona generująca wykres najpopularniejszych zajęć	54
Eksport i import aplikacji	56
Eksport aplikacji.....	56
Tworzenie nowego Workspace	57
Import aplikacji.....	58
Weryfikacja obiektów bazy danych.....	58
Uruchomienie aplikacji	59
Podsumowanie projektu	60
Wnioski	60

Wstęp

Projekt *Aplikacja do zarządzania klubem fitness w APEX* realizowany w ramach przedmiotu Aplikacje Bazodanowe na kierunku Inżynieria i Analiza Danych, rok III – semestr V.

Cel projektu

Projekt aplikacji do zarządzania klubem fitness w **Oracle APEX** miał na celu stworzenie narzędzia umożliwiającego administratorom i recepcjonistom sprawne zarządzanie rezerwacjami zajęć, harmonogramami oraz danymi klientów i instruktorów. Aplikacja miała usprawnić procesy administracyjne, automatyzować powiadomienia dla klientów oraz zapewnić dostęp do raportów analitycznych.

System wykorzystuje bazę danych Oracle, a logika biznesowa została zaimplementowana w **PL/SQL**, obejmując funkcje i procedury odpowiedzialne za obsługę rezerwacji, zarządzanie grupami zajęciowymi oraz analizę danych. Projekt skupiał się na rozszerzeniu istniejącej bazy danych i integracji nowych funkcjonalności wspierających codzienne zarządzanie klubem fitness.

Projekt aplikacji

Schemat bazy

Poniższy schemat przedstawia strukturę relacyjnej bazy danych wykorzystywanej w aplikacji do zarządzania klubem fitness. Model uwzględnia kluczowe encje oraz ich powiązania, umożliwiające kompleksowe zarządzanie klientami, instruktorami, rezerwacjami oraz harmonogramem zajęć.

Relacje pomiędzy tabelami zapewniają integralność danych i umożliwiają efektywne zarządzanie procesami w klubie fitness, takimi jak rezerwacja miejsc, przypisywanie instruktorów czy monitorowanie dostępności sal.

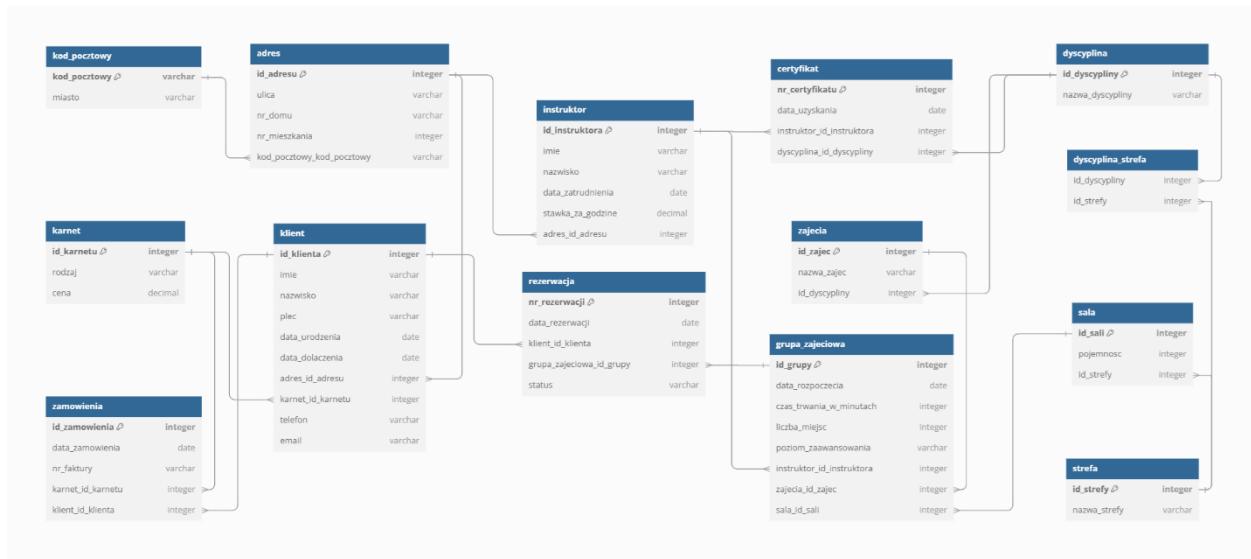


Diagram przypadków użycia

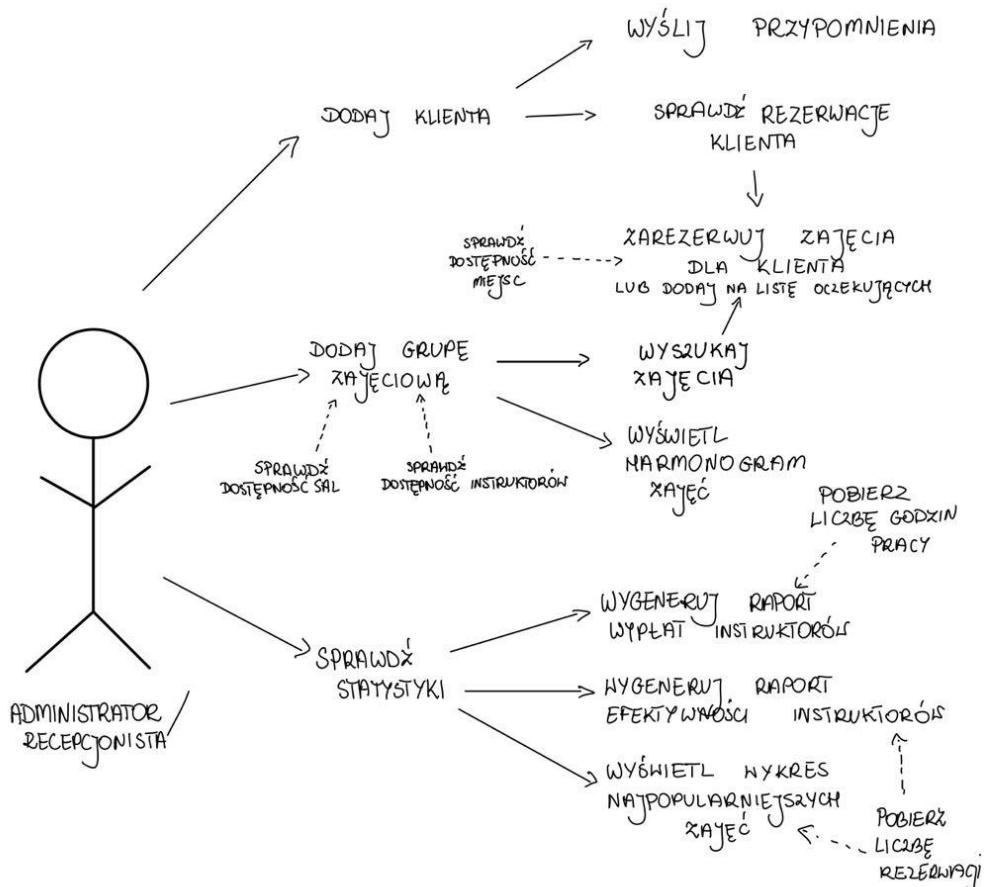
Diagram przypadków użycia przedstawia kluczowe funkcjonalności aplikacji oraz sposób ich wykorzystania przez administratora lub recepcjonistę klubu fitness. Użytkownik systemu może wykonywać operacje związane z zarządzaniem klientami, rezerwacjami, zajęciami oraz analizą statystyk.

Do najważniejszych przypadków użycia należą:

- **Dodawanie klienta** – możliwość rejestracji nowych klientów w systemie;
- **Sprawdzanie rezerwacji klienta** – weryfikacja aktywnych rezerwacji oraz możliwość ich anulowania;
- **Rezerwacja zajęć** – wpisywanie klienta na zajęcia lub dodawanie go na listę oczekujących w przypadku braku miejsc;
- **Dodawanie grupy zajęciowej** – tworzenie nowych grup zajęciowych z uwzględnieniem dostępności sal i instruktorów;
- **Wyszukiwanie zajęć** – filtrowanie zajęć według określonych kryteriów.
- **Wyświetlanie harmonogramu zajęć** – podgląd planu zajęć klubu;

- **Sprawdzanie statystyk** – generowanie raportów dotyczących wypłat instruktorów, ich efektywności oraz popularności zajęć;
- **Wysyłanie powiadomień** – automatyczne wysyłanie e-maili przypominających klientom o nadchodzących zajęciach;
- **Analiza obłożenia zajęć** – pobieranie liczby rezerwacji oraz liczby przepracowanych godzin przez instruktorów;

Diagram ilustruje ścieżki interakcji użytkownika z systemem oraz pokazuje powiązania pomiędzy poszczególnymi funkcjonalnościami, podkreślając zależności pomiędzy operacjami.



Pakiety – diagramy i implementacja w PL/SQL

W celu efektywnego zarządzania funkcjonalnościami systemu klubu fitness, stworzono pakiety PL/SQL, które grupują powiązane procedury i funkcje. Pakiety te odpowiadają za kluczowe operacje biznesowe, takie jak obsługa klientów, rezerwacji, zajęć oraz generowanie raportów. Dzięki zastosowaniu pakietów uzyskano lepszą organizację kodu, zwiększoną modularność oraz możliwość łatwego rozwijania systemu w przyszłości.

Dwa główne pakiety w systemie to:

- **Klient_Pakiet** – obsługuje operacje związane z klientami, rezerwacjami oraz listą oczekujących.
- **Administrator_Pakiet** – umożliwia zarządzanie grupami zajęciowymi, przydzielanie instruktorów i sal, a także analizę statystyk i generowanie raportów.

Pakiet Klient

Stworzono pakiet **Klient_Pakiet**, który zawiera funkcje i procedury umożliwiające zarządzanie klientami oraz ich rezerwacjami w systemie klubu fitness. Pakiet obsługuje operacje związane z wyszukiwaniem dostępnych zajęć, sprawdzaniem miejsc, rezerwacją oraz zarządzaniem listą oczekujących.

Zaimplementowano funkcje pozwalające na pobieranie listy rezerwacji klienta, wyszukiwanie zajęć według określonych kryteriów oraz analizę dostępnych miejsc w grupach zajęciowych. Dodatkowo, procedury umożliwiają rezerwację i anulowanie zajęć, dodawanie klientów do systemu oraz zarządzanie listą oczekujących.

Pakiet stanowi kluczowy element logiki biznesowej systemu, integrując operacje zarządzania rezerwacjami i klientami w jednolitą strukturę PL/SQL.

```
create or replace PACKAGE Klient_Pakiet AS

    -- Deklaracje funkcji
    FUNCTION sprawdz_dostepne_miejsca(
        p_id_grupy IN grupa_zajeciodzialana.id_grupy%TYPE
    ) RETURN INTEGER;

    FUNCTION wyszukaj_zajecia(
        p_data_poczatkowa IN DATE,
        p_data_koncowa IN DATE,
        p_nazwa_zajec IN VARCHAR2 DEFAULT NULL,
        p_dyscyplina IN VARCHAR2 DEFAULT NULL,
        p_imie_instruktora IN VARCHAR2 DEFAULT NULL,
        p_nazwisko_instruktora IN VARCHAR2 DEFAULT NULL,
        p_poziom_zaawansowania IN VARCHAR2 DEFAULT NULL
    ) RETURN TypyZajecia.TabelaZajec PIPELINED;

    FUNCTION pobierz_liste_oczekujacych(
        p_id_grupy IN INTEGER
    ) RETURN ListaOczekujacychTab;

    FUNCTION pobierz_rezerwacje_klienta(
        p_id_klienta IN INTEGER
    ) RETURN RezerwacjaTab PIPELINED;
```

```

-- Deklaracje procedur

PROCEDURE ZarezerwujZajecia(
    p_id_klienta IN INTEGER,
    p_id_grupy   IN INTEGER
);

PROCEDURE AnulujRezerwacje(
    p_nr_rezerwacji  IN INTEGER
);

PROCEDURE DodajNaListeOczekujacych(
    p_id_klienta IN INTEGER,
    p_id_grupy   IN INTEGER,
    p_wiadomosc OUT VARCHAR2
);

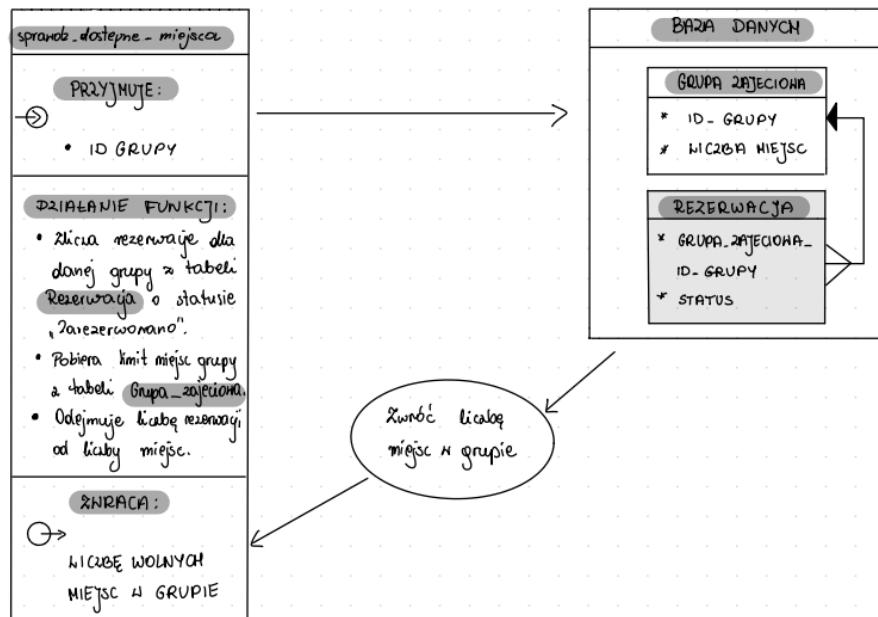
PROCEDURE DodajKlienta(
    p_imie          IN VARCHAR2,
    p_nazwisko     IN VARCHAR2,
    p_plec          IN VARCHAR2,
    p_data_urodzenia IN DATE,
    p_data_dolaczania IN DATE,
    p_telefon       IN VARCHAR2,
    p_email         IN VARCHAR2,
    p_adres         IN TypAdres
);

END Klient_Pakiet;
/

```

Pakiet Klient – Funkcje

- Sprawdz_dostepne_miejsca



Funkcja **sprawdz_dostepne_miejsca** służy do określenia liczby wolnych miejsc w konkretnej grupie zajęciowej, co jest kluczowe dla procesu rezerwacji w systemie zarządzania klubem fitness. Jej działanie opiera się na pobraniu informacji o całkowitej liczbie miejsc w danej grupie oraz zliczeniu aktywnych rezerwacji, które są w statusie „Zarezerwowano”. Na podstawie tych danych funkcja oblicza dostępne miejsca, odejmując liczbę rezerwacji od ogólnej pojemności grupy. Jest to przydatne w momencie dodawania nowych rezerwacji oraz sprawdzania, czy w danej grupie są jeszcze wolne miejsca. Funkcja korzysta z tabel grupa_zajeciona, z której pobiera informację o liczbie miejsc, oraz rezerwacja, gdzie sprawdza liczbę już zajętych miejsc w danej grupie.

Obsługa błędów:

- Sprawdzenie, czy zostało podane ID grupy zajęciowej.

Obsługa błędów wszystkich funkcji i procedur w większości została zaimplementowana w postaci walidacji bezpośrednio w tworzeniu poszczególnych stron w APEX. Przykładowe implementacje zostały przedstawione w rozdziale *Implementacja aplikacji w APEX*.

```
FUNCTION sprawdz_dostepne_miejsca(
    p_id_grupy IN grupa_zajeciova.id_grupy%TYPE
) RETURN INTEGER IS
    v_grupa GrupaInfo;
BEGIN
    -- Pobranie liczby miejsc i zajętych miejsc
    SELECT GrupaInfo(g.liczba_miejsc, NVL(COUNT(r.nr_rezerwacji), 0))
    INTO v_grupa
    FROM grupa_zajeciova g
    LEFT JOIN rezerwacja r
        ON g.id_grupy = r.grupa_zajeciova_id_grupy
        AND r.status = 'Zarezerwowano'
    WHERE g.id_grupy = p_id_grupy
    GROUP BY g.liczba_miejsc;

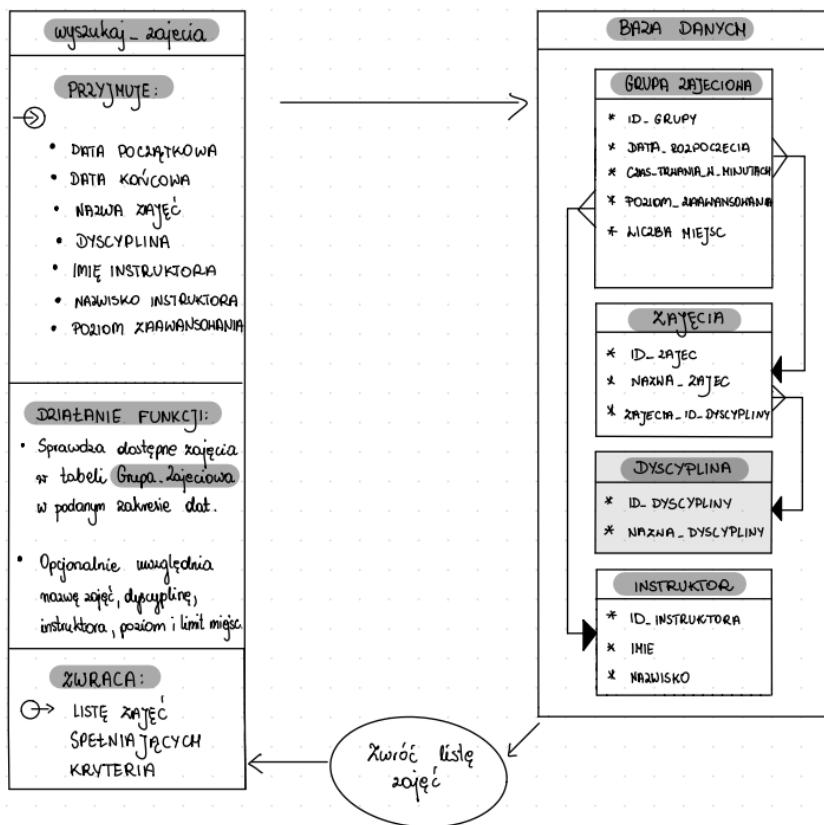
    -- Jeśli liczba miejsc jest NULL, zwracamy NULL zamiast błędu
    IF v_grupa.liczba_miejsc IS NULL OR v_grupa.liczba_miejsc < 0 THEN
        RETURN NULL;
    END IF;

    -- Obliczenie liczby dostępnych miejsc
    RETURN v_grupa.liczba_miejsc - v_grupa.liczba_rezerwacji;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
    WHEN OTHERS THEN
        RETURN NULL;
END sprawdz_dostepne_miejsca;
```

Funkcja jako parametr wejściowy przyjmuje ID grupy, a jej zadaniem jest określenie liczby dostępnych miejsc w danej grupie zajęciowej. Wykorzystuje łączenie tabel grupa_zajeciova oraz rezerwacja, aby policzyć aktywne rezerwacje, uwzględniając tylko te, które mają status „Zarezerwowano”. W sytuacji, gdy dla danej grupy nie ma zarejestrowanych rezerwacji, funkcja zwraca NULL, co pozwala uniknąć błędów związanych z brakiem danych. W przeciwnym razie, oblicza dostępne miejsca, odejmując liczbę aktywnych rezerwacji od maksymalnej liczby miejsc przypisanej do grupy. Wynikiem działania funkcji jest **liczba całkowita**, reprezentująca dostępną liczbę miejsc w danej grupie.

- **Wyszukaj_zajecia**



Funkcja **wyszukaj_zajecia** służy do odnalezienia dostępnych zajęć w określonym przedziale czasowym, uwzględniając opcjonalne kryteria wyszukiwania. Jej głównym zadaniem jest przeszukanie tabeli zawierającej informacje o grupach zajęciowych i zwrócenie listy zajęć spełniających podane kryteria. Użytkownik może określić przedział dat, w którym chce znaleźć zajęcia, a także dodatkowe parametry, takie jak nazwa zajęć, dyscyplina, imię i nazwisko instruktora oraz poziom zaawansowania. Jeśli nie zostaną podane dodatkowe kryteria, funkcja zwróci wszystkie zajęcia z danego zakresu dat. Funkcja bazuje na danych przechowywanych w kilku powiązanych tabelach bazy danych: grupa_zajeciowa, zajecia, dyscyplina i instruktor, które są ze sobą połączone relacjami kluczowymi.

Obsługa błędów:

- Sprawdzenie, czy daty początkowa i końcowa zostały wprowadzone;
- Sprawdzenie, czy data końcowa jest późniejsza niż data początkowa.

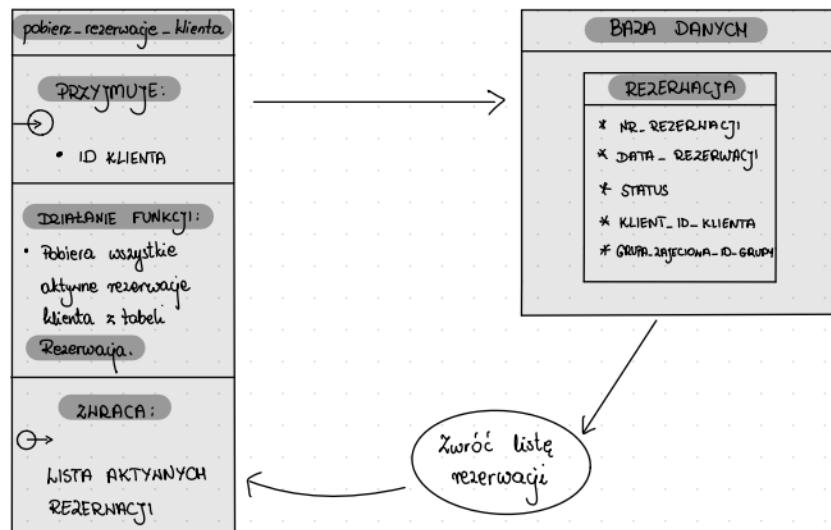
```

FUNCTION wyszukaj_zajecia(
    p_data_poczatkowa IN DATE,
    p_data_koncowa IN DATE,
    p_nazwa_zajec IN VARCHAR2 DEFAULT NULL,
    p_dyscyplina IN VARCHAR2 DEFAULT NULL,
    p_imie_instruktora IN VARCHAR2 DEFAULT NULL,
    p_nazwisko_instruktora IN VARCHAR2 DEFAULT NULL,
    p_poziom_zaawansowania IN VARCHAR2 DEFAULT NULL
) RETURN TypyZajecia.TabelaZajec PIPELINED
IS
BEGIN
    -- Pobieranie wyników i zwracanie rekordów w PIPELINED
    FOR rec IN (
        SELECT g.id_grupy, g.data_rozpoczecia, g.czas_trwania_w_minutach AS czas_trwania,
               z.nazwa_zajec, d.nazwa_dyscypliny, i.imie, i.nazwisko, g.poziom_zaawansowania
        FROM grupa_zajeciowa g
        JOIN zajecia z ON g.zajecia_id_zajec = z.id_zajec
        JOIN dyscyplina d ON z.id_dyscypliny = d.id_dyscypliny
        JOIN instruktor i ON g.instruktor_id_instruktora = i.id_instruktora
        WHERE g.data_rozpoczecia BETWEEN p_data_poczatkowa AND p_data_koncowa
              AND (p_nazwa_zajec IS NULL OR z.nazwa_zajec = p_nazwa_zajec)
              AND (p_dyscyplina IS NULL OR d.nazwa_dyscypliny = p_dyscyplina)
              AND (p_imie_instruktora IS NULL OR i.imie = p_imie_instruktora)
              AND (p_nazwisko_instruktora IS NULL OR i.nazwisko = p_nazwisko_instruktora)
              AND (p_poziom_zaawansowania IS NULL OR g.poziom_zaawansowania = p_poziom_zaawansowania)
    ) LOOP
        PIPE ROW (TypyZajecia.TypZajecia(
            rec.id_grupy, rec.data_rozpoczecia, rec.czas_trwania,
            rec.nazwa_zajec, rec.nazwa_dyscypliny, rec.imie, rec.nazwisko, rec.poziom_zaawansowania
        ));
    END LOOP;
    RETURN;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN;
    WHEN OTHERS THEN
        RETURN;
END wyszukaj_zajecia;

```

Funkcja przyjmuje parametry wejściowe, które określają zakres wyszukiwania i opcjonalne filtry, a następnie pobiera z bazy danych rekordy spełniające podane kryteria. Do realizacji tego zadania wykorzystuje zapytanie SQL z wielokrotnymi JOIN między tabelami, aby połączyć informacje o grupach zajęciowych, ich dyscyplinach oraz instrukturach prowadzących dane zajęcia. Wyszukiwanie odbywa się na podstawie warunku WHERE, który filuluje dane zgodnie z przekazanymi parametrami. Aby umożliwić dynamiczne wyszukiwanie, funkcja stosuje warunki sprawdzające, czy dany parametr jest podany, a jeśli nie – akceptuje dowolne wartości (poprzez warunki IS NULL OR). Funkcja wykorzystuje również mechanizm PIPELINED, co pozwala zwracać wyniki w sposób bardziej efektywny – zamiast zwracać całą tabelę na raz, zwraca kolejne rekordy w strumieniu. Na koniec funkcja zwraca listę zajęć spełniających podane kryteria, a w przypadku braku wyników zwraca pustą listę, unikając błędów związanych z brakiem danych.

- **Pobierz_rezerwacje_klienta**



Funkcja **pobierz_rezerwacje_klienta** ma na celu zwrócenie listy aktywnych rezerwacji dla konkretnego klienta na podstawie jego identyfikatora. Pobiera ona dane dotyczące rezerwacji z tabeli Rezerwacja, uwzględniając status oraz dodatkowe informacje o grupie zajęciowej, do której przypisana jest dana rezerwacja. Wynikiem działania funkcji jest lista rezerwacji zawierająca numer rezerwacji, datę rezerwacji, status, datę rozpoczęcia zajęć, czas ich trwania oraz poziom zaawansowania.

Obsługa błędów:

- Sprawdzenie, czy zostało podane ID klienta.

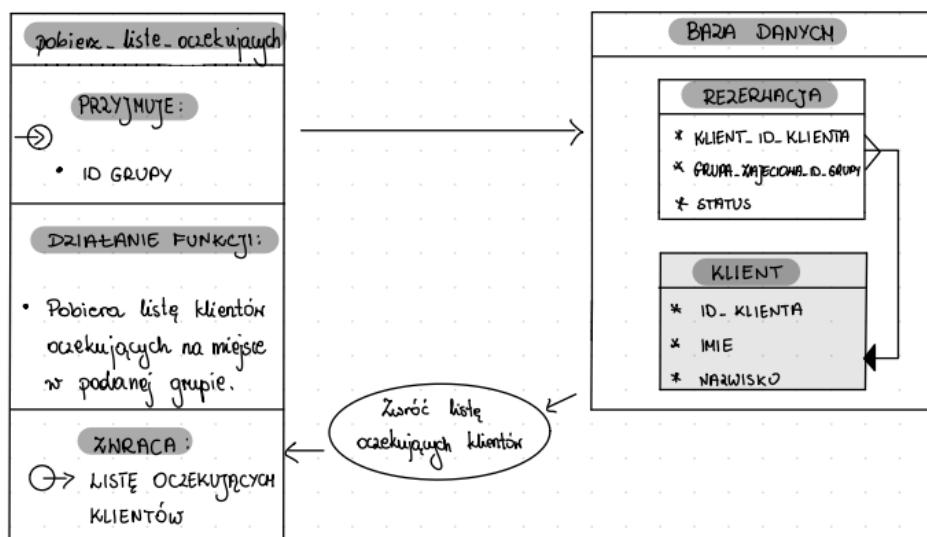
```

FUNCTION pobierz_rezerwacje_klienta(
    p_id_klienta IN INTEGER
) RETURN RezerwacjaTab PIPELINED IS
BEGIN
    FOR rec IN (
        SELECT r.nr_rezerwacji, r.data_rezerwacji, r.status,
               g.data_rozpoczecia, g.czas_trwania_w_minutach,
               z.nazwa_zajec, g.poziom_zaawansowania
        FROM rezerwacja r
        JOIN grupa_zajeciodowa g ON r.grupa_zajeciodowa_id_grupy = g.id_grupy
        JOIN zajecia z ON g.zajecia_id_zajec = z.id_zajec
        WHERE r.klient_id_klienta = p_id_klienta
    ) LOOP
        PIPE ROW (RezerwacjaTyp(
            rec.nr_rezerwacji, rec.data_rezerwacji, rec.status,
            rec.data_rozpoczecia, rec.czas_trwania_w_minutach,
            rec.nazwa_zajec, rec.poziom_zaawansowania
        ));
    END LOOP;
    RETURN;
END pobierz_rezerwacje_klienta;

```

Funkcja przyjmuje jako parametr wejściowy identyfikator klienta (p_id_klienta), a następnie wykonuje zapytanie łączące tabele Rezerwacja, Grupa_zajeciodowa oraz Zajecia, aby uzyskać pełne informacje o zajęciach powiązanych z rezerwacjami. Wyszukuje rezerwacje powiązane z danym klientem i zwraca rekordy w postaci **PIPELINED**, co pozwala na efektywne przetwarzanie wyników w postaci strumienia. Wynikiem działania funkcji jest lista zawierająca numer rezerwacji, datę rezerwacji, status oraz szczegółowe informacje o powiązanych zajęciach, które są zwracane jako wynik zapytania.

- **Pobierz_liste_oczekujacych**



Funkcja **pobierz_liste_oczekujacych** jest odpowiedzialna za zwracanie listy klientów oczekujących na miejsce w danej grupie zajęciowej. Przyjmuje jako parametr identyfikator grupy i wyszukuje w tabeli *Rezerwacja* klientów, którzy mają status "Oczekujący". Aby uzyskać pełne informacje o klientach, łączy się z tabelą *Klient*, pobierając imię i nazwisko. Głównym celem funkcji jest umożliwienie administratorowi łatwego dostępu do listy osób oczekujących, co może być przydatne przy zwalnianiu miejsc lub organizowaniu dodatkowych zajęć.

Obsługa błędów:

- Sprawdzenie, czy zostało podane ID grupy zajęciowej.

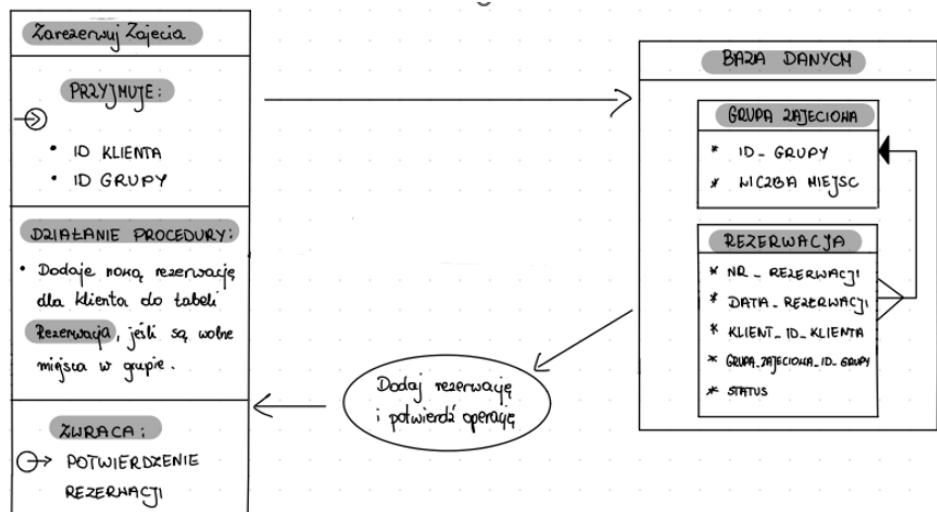
```
FUNCTION pobierz_liste_oczekujacych(
    p_id_grupy IN INTEGER
) RETURN ListaOczekujacychTab IS
    v_lista ListaOczekujacychTab := ListaOczekujacychTab();
BEGIN
    -- Pobranie listy oczekujących
    SELECT ListaOczekujacychObj(k.id_klienta, k.imie, k.nazwisko, r.status)
    BULK COLLECT INTO v_lista
    FROM rezerwacja r
    JOIN klient k ON r.klient_id_klienta = k.id_klienta
    WHERE r.grupa_zajeciod_id_grupy = p_id_grupy AND r.status = 'Oczekujący';

    RETURN v_lista;
EXCEPTION
    WHEN OTHERS THEN
        RETURN v_lista;
END pobierz_liste_oczekujacych;
```

Implementacja wykorzystuje mechanizm BULK COLLECT, który pozwala na pobranie wielu wierszy do kolekcji PL/SQL w jednym zapytaniu, co zwiększa wydajność. Wynik jest przechowywany w **kolekcji** ListaOczekujacychTab, która zawiera **obiekty** przechowujące ID klienta, imię, nazwisko oraz status. Funkcja zwraca tę **kolekcję**, a w przypadku błędów zwraca pustą listę, aby uniknąć niekontrolowanych przerwań działania systemu.

Pakiet Klient – Procedury

- **ZarezerwujZajecia**



Procedura **ZarezerwujZajecia** odpowiada za dodanie nowej rezerwacji dla klienta w tabeli Rezerwacja, ale tylko wtedy, gdy w wybranej grupie zajęciowej są jeszcze wolne miejsca. Procedura przyjmuje dwa parametry: id klienta oraz id grupy, co pozwala jednoznacznie określić, dla kogo i na jakie zajęcia ma zostać dokonana rezerwacja. Rezerwacja zostaje zapisana z aktualną datą oraz statusem

„Zarezerwowano”, co oznacza aktywną rezerwację. Po udanym dodaniu danych transakcja jest zatwierdzana, a w przypadku błędu operacja jest wycofywana i wyświetlany jest komunikat o nieoczekiwanych błędach.

Obsługa błędów:

- Sprawdzenie, czy zostały podane ID grupy zajęciowej i ID klienta;
- Sprawdzenie, czy klient o podanym ID i grupa o podanym ID istnieją;
- Sprawdzenie, czy klient o podanym ID posiada aktywne rezerwacje na dane zajęcia;
- Sprawdzenie, czy klient o podanym ID znajduje się na liście oczekujących na dane zajęcia z wykorzystaniem funkcji **pobierz_liste_oczekujacych**.

```

PROCEDURE ZarezerwujZajecia(
    p_id_klienta IN INTEGER,
    p_id_grupy   IN INTEGER
) IS
    v_status_rezerwacji VARCHAR2(20) := 'Zarezerwowano';

BEGIN
    -- Dodanie rezerwacji do tabeli REZERWACJA
    INSERT INTO rezerwacja(nr_rezerwacji, data_rezerwacji, klient_id_klienta,
grupa_zajeciod_id_grupy, status)
        VALUES (rezerwacja_seq.NEXTVAL, SYSDATE, p_id_klienta, p_id_grupy, v_status_rezerwacji);

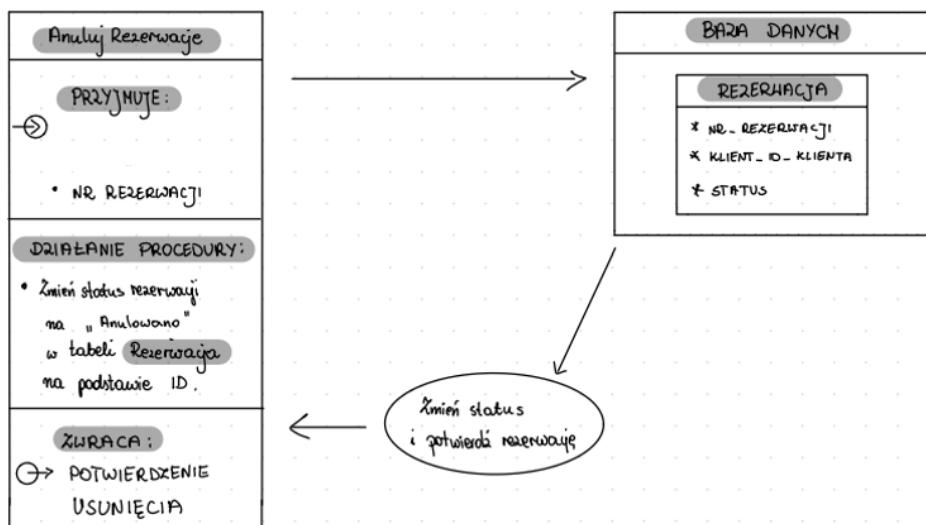
    -- Zatwierdzenie transakcji
    COMMIT;

EXCEPTION
    WHEN OTHERS THEN
        apex_error.add_error(
            p_message => 'Nieoczekiwany błąd: ' || SQLERRM,
            p_display_location => apex_error.c_inline_in_notification
        );
        ROLLBACK;
END ZarezerwujZajecia;

```

Procedura korzysta z instrukcji INSERT INTO, aby dodać nowy rekord do tabeli Rezerwacja, przypisując automatycznie numer rezerwacji z sekwencji rezerwacja_seq.NEXTVAL, a także ustawiając aktualny czas rezerwacji za pomocą SYSDATE. Dodatkowo, każda nowa rezerwacja otrzymuje status „Zarezerwowano”, zapisany w zmiennej v_status_rezerwacji. Procedura kończy się zatwierdzeniem transakcji poprzez COMMIT, co sprawia, że zmiany zostają zapisane na stałe w bazie danych. W razie wystąpienia błędu, mechanizm obsługi wyjątków WHEN OTHERS THEN wywołuje funkcję apex_error.add_error, dzięki czemu użytkownik otrzymuje informację o problemie, a cała operacja zostaje wycofana za pomocą ROLLBACK, co zapewnia integralność danych.

- **AnulujRezerwacje**



Procedura **AnulujRezerwacje** służy do zmiany statusu rezerwacji w systemie. Jej głównym zadaniem jest oznaczenie wybranej rezerwacji jako "Anulowano" na podstawie unikalnego numeru rezerwacji. Dzięki temu administrator lub recepcjonista może efektywnie zarządzać zapisami klientów i anulować ich uczestnictwo w zajęciach. Procedura aktualizuje status rezerwacji w tabeli Rezerwacja, co pozwala na zwolnienie miejsca w grupie zajęciowej, a następnie potwierdza operację poprzez zapis zmian w bazie danych.

Obsługa błędów:

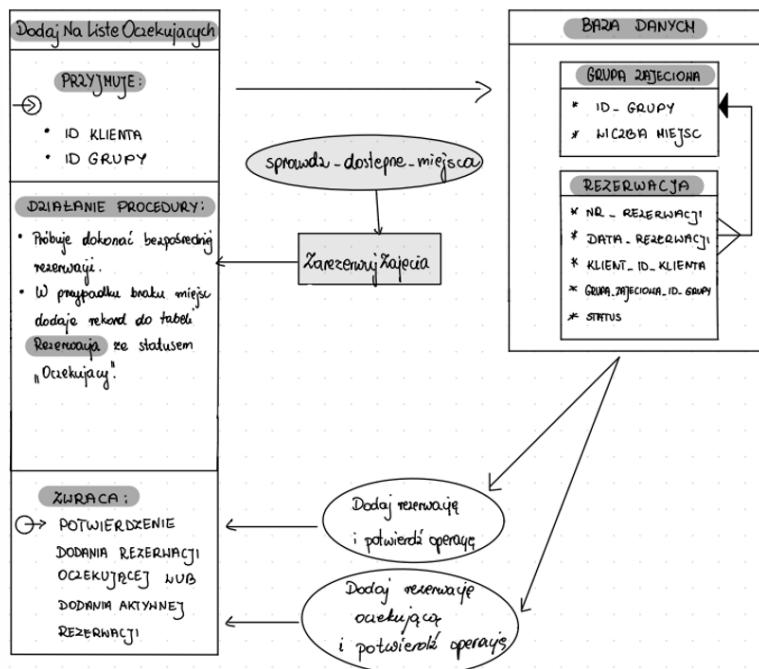
- Sprawdzenie, czy zostały podane ID grupy zajęciowej i ID klienta;
- Sprawdzenie, czy klient o podanym ID i grupa o podanym ID istnieją;
- Sprawdzenie, czy klient o podanym ID posiada aktywne rezerwacje na dane zajęcia;
- Sprawdzenie, czy klient nie posiada już aktywnej rezerwacji na inne zajęcia w terminie zajęć, na które chce dodać rezerwację.

```
PROCEDURE AnulujRezerwacje(
    p_nr_rezerwacji IN INTEGER
) IS
BEGIN
    -- Aktualizacja statusu rezerwacji na 'Anulowano'
    UPDATE rezerwacja
    SET status = 'Anulowano'
    WHERE nr_rezerwacji = p_nr_rezerwacji;

    -- Zatwierdzenie transakcji tylko, jeśli coś zostało zaktualizowane
    IF SQL%ROWCOUNT > 0 THEN
        COMMIT;
    END IF;
END AnulujRezerwacje;
```

Pod względem implementacji, procedura przyjmuje jako parametr nr rezerwacji, który identyfikuje rezerwację w bazie. Następnie wykonuje UPDATE w tabeli Rezerwacja, zmieniając wartość kolumny status na 'Anulowano'. Weryfikuje również, czy jakakolwiek zmiana miała miejsce, korzystając z SQL%ROWCOUNT, a jeśli tak, zatwierdza operację poprzez COMMIT. To zabezpiecza przed przypadkowym zatwierdzeniem transakcji, jeśli podany numer rezerwacji nie istnieje. W ten sposób procedura zapewnia spójność danych i prawidłowe zarządzanie zapisami klientów.

- **DodajNaListeOczekujacych**



Procedura **DodajNaListeOczekujących** służy do obsługi zapisów klientów na zajęcia fitness, zapewniając, że jeśli w danej grupie są wolne miejsca, klient zostanie od razu zapisany, a jeśli miejsc brakuje, zostanie dodany na listę oczekujących. Procedura przyjmuje jako parametry identyfikator klienta oraz identyfikator grupy zajęciowej i zwraca komunikat informujący o wyniku operacji. Najpierw sprawdza, czy w danej grupie są dostępne miejsca, korzystając z funkcji **sprawdz_dostepne_miejsca**. Jeśli są wolne miejsca, klient zostaje zapisany na zajęcia poprzez wywołanie procedury **ZarezerwujZajecia**, a użytkownik otrzymuje informację o pomyślnym zapisie. Jeśli miejsc nie ma, procedura sprawdza, czy klient już znajduje się na liście oczekujących. Jeśli tak, operacja zostaje przerwana z komunikatem błędu, informującym, że klient już oczekuje na miejsce. Jeśli klient nie jest na liście oczekujących, zostaje dodany do tabeli rezerwacji ze statusem „Oczekujący”, a użytkownik otrzymuje informację, że klient został dodany do kolejki oczekujących.

Obsługa błędów:

- Sprawdzenie, czy w danej grupie zajęciowej są wolne miejsca z wykorzystaniem funkcji **sprawdz_dostepne_miejsca**.

```
PROCEDURE DodajNaListeOczekujacych(
    p_id_klienta IN INTEGER,
    p_id_grupy   IN INTEGER,
    p_wiadomosc  OUT VARCHAR2
) IS
    v_dostepne_miejsca INTEGER;
    v_lista_oczek ListaOczekujacychTab;
    v_istnieje INTEGER;
BEGIN
    -- Sprawdzenie dostępnych miejsc
    v_dostepne_miejsca := sprawdz_dostepne_miejsca(p_id_grupy);

    IF v_dostepne_miejsca > 0 THEN
        -- Jeżeli są dostępne miejsca, dokonaj rezerwacji
        ZarezerwujZajecia(p_id_klienta, p_id_grupy);
        p_wiadomosc := 'Rezerwacja zakończona pomyślnie! Klient został zapisany na zajęcia.';
        RETURN;
    END IF;

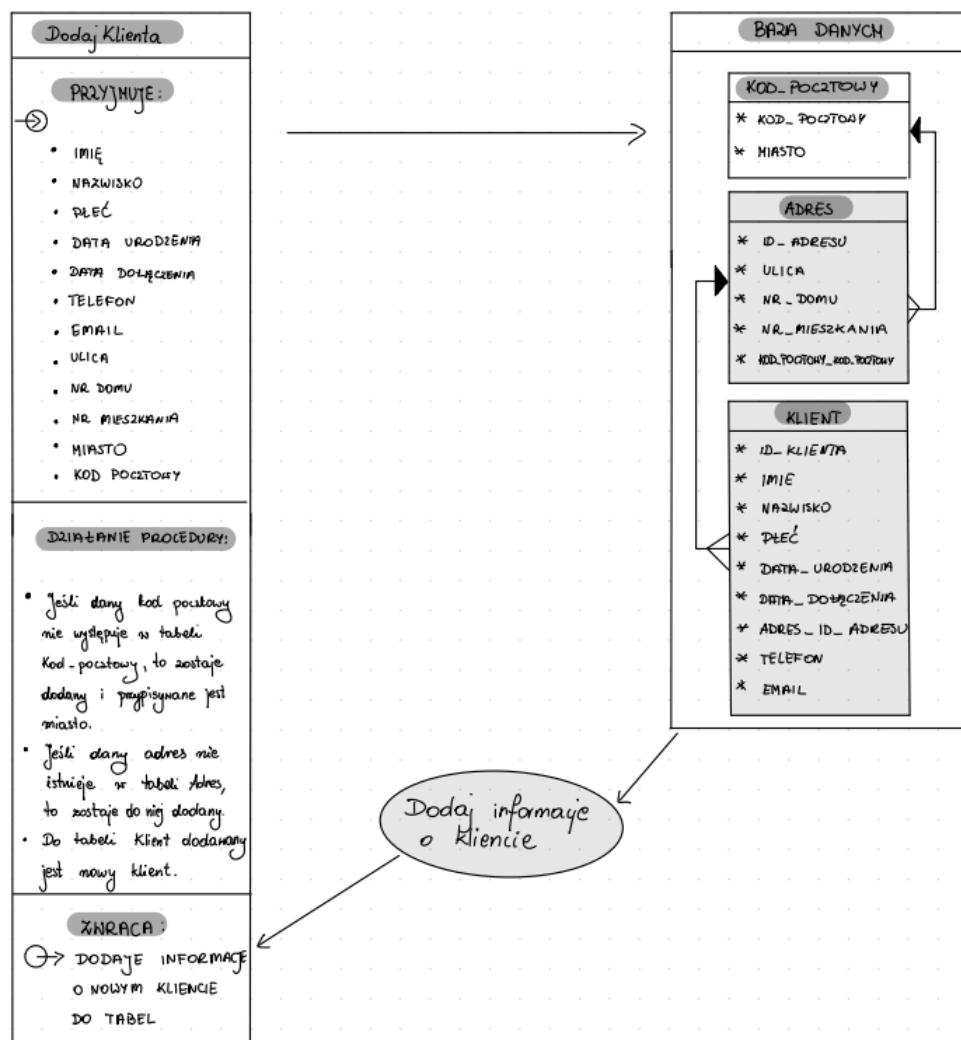
    -- Dodanie klienta na listę oczekujących
    INSERT INTO rezerwacja (nr_rezerwacji, data_rezerwacji, klient_id_klienta,
    grupa_zajeciowa_id_grupy, status)
    VALUES (rezerwacja_seq.NEXTVAL, SYSDATE, p_id_klienta, p_id_grupy, 'Oczekujący');

    p_wiadomosc := 'Brak miejsc w grupie zajęciowej. Klient został dodany na listę oczekujących.';

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        p_wiadomosc := 'Błąd: ' || SQLERRM;
END DodajNaListeOczekujacych;
```

Implementacja procedury w PL/SQL wykorzystuje kilka kluczowych mechanizmów. Funkcja **sprawdz_dostepne_miejsca** pobiera liczbę dostępnych miejsc w grupie zajęciowej, co pozwala określić, czy klient może zostać zapisany od razu, czy powinien trafić na listę oczekujących. Jeśli w grupie są wolne miejsca, procedura **ZarezerwujZajecia** dokonuje bezpośredniego zapisu. Gdy miejsc brakuje, zapytanie SQL sprawdza, czy klient już oczekuje na miejsce, wykorzystując operację COUNT(*) i filtrowanie po statusie „Oczekujący”. Jeśli klient nie jest jeszcze w kolejce, nowy wpis zostaje dodany do tabeli rezerwacji z użyciem instrukcji INSERT INTO, a aktualna data zapisu jest pobierana z SYSDATE. Procedura obsługuje również błędy, stosując mechanizm EXCEPTION, który w przypadku wystąpienia problemu cofa transakcję (ROLLBACK) i zwraca komunikat o błędzie, aby uniknąć nieprawidłowego zapisu w systemie.

- **DodajKlienta**



Procedura **DodajKlienta** umożliwia dodanie nowego klienta do systemu, uwzględniając jego dane osobowe oraz adresowe. Procedura weryfikuje, czy dany kod pocztowy oraz adres klienta już istnieją w bazie danych. Jeśli kod pocztowy nie jest jeszcze zapisany, procedura dodaje go do tabeli wraz z przypisanym miastem. Następnie sprawdza, czy podany adres istnieje – jeśli nie, tworzy nowy rekord w tabeli adresów. Po wykonaniu tych kroków dodaje nowego klienta do tabeli klient, przypisując mu odpowiedni identyfikator adresu.

Obsługa błędów:

- Sprawdzenie, czy wszystkie parametry zostały wprowadzone;
- Sprawdzenie, czy podane dane zostały wprowadzone w odpowiednim formacie z wykorzystaniem wyrażeń regularnych:
 - Imię, nazwisko, ulica i miasto zaczynają się wielką literą,
 - Numer telefonu składa się z 9 cyfr,
 - Adres email zawiera znak @ oraz .
 - Nr domu i nr mieszkania składają się z cyfr i ewentualnej litery,
 - Kod pocztowy ma format XX-XXX, gdzie X to cyfra;
- Sprawdzenie, czy klient o podanym numerze telefonu już istnieje;
- Sprawdzenie, czy podany adres i kod pocztowy już istnieją w bazie danych.

```

PROCEDURE DodajKlienta(
    p_imie          IN VARCHAR2,
    p_nazwisko      IN VARCHAR2,
    p_plec           IN VARCHAR2,
    p_data_urodzenia IN DATE,
    p_data_dolaczenia IN DATE,
    p_telefon        IN VARCHAR2,
    p_email          IN VARCHAR2,
    p_adres          IN TypAdres
) IS
    v_id_adresu INTEGER := NULL;
    v_count INTEGER := 0;
BEGIN
    -- Sprawdzenie, czy kod pocztowy już istnieje, jeśli nie, dodajemy
    BEGIN
        SELECT COUNT(*) INTO v_count
        FROM kod_pocztowy
        WHERE kod_pocztowy = p_adres.kod_pocztowy;

        IF v_count = 0 THEN
            INSERT INTO kod_pocztowy (kod_pocztowy, miasto)
            VALUES (p_adres.kod_pocztowy, p_adres.miasto);
        END IF;
    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
            NULL; -- Ignorujemy błąd, jeśli kod pocztowy już istnieje
    END;

    -- Sprawdzenie, czy adres już istnieje
    BEGIN
        SELECT id_adresu INTO v_id_adresu
        FROM adres
        WHERE ulica = p_adres.ulica
        AND nr_domu = p_adres.nr_domu
        AND NVL(nr_mieszkania, 0) = NVL(p_adres.nr_mieszkania, 0)
        AND kod_pocztowy_kod_pocztowy = p_adres.kod_pocztowy;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                -- Jeżeli adres nie istnieje, dodajemy nowy
                SELECT NVL(MAX(id_adresu), 0) + 1 INTO v_id_adresu FROM adres;
                INSERT INTO adres (id_adresu, ulica, nr_domu, nr_mieszkania, kod_pocztowy_kod_pocztowy)
                VALUES (v_id_adresu, p_adres.ulica, p_adres.nr_domu, p_adres.nr_mieszkania,
                p_adres.kod_pocztowy);
    END;

    -- Dodanie nowego klienta
    INSERT INTO Klient (id_klienta, imie, nazwisko, plec, data_urodzenia, data_dolaczenia,
    adres_id_adresu, telefon, email)
    VALUES (
        klient_seq.NEXTVAL,
        p_imie,
        p_nazwisko,
        p_plec,
        p_data_urodzenia,
        p_data_dolaczenia,
        v_id_adresu,
        p_telefon,
        p_email
    );
    COMMIT; -- Zatwierdzenie transakcji tylko w razie sukcesu
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK; -- Jeżeli wystąpił błąd, cofamy zmiany
        RAISE_APPLICATION_ERROR(-20099, 'Błąd: ' || SQLERRM);
END DodajKlienta;

```

Implementacja procedury wykorzystuje operacje SELECT, INSERT oraz NVL do sprawdzania i dodawania nowych wartości. Obsługuje również błędy, np. DUP_VAL_ON_INDEX, aby uniknąć problemów z dodawaniem duplikatów kodów pocztowych. Jeśli dany adres nie istnieje, dodawany jest nowy identyfikator, który następnie wykorzystywany jest przy zapisie nowego klienta. Cała transakcja jest potwierdzana COMMIT, a w przypadku błędu wykonuje się ROLLBACK i zwracany jest komunikat o błędzie.

Pakiet Administrator

Stworzono pakiet **ADMINISTRATOR_PAKIET**, który zawiera funkcje i procedury umożliwiające administratorom klubu fitness efektywne zarządzanie zasobami oraz analizę danych. Pakiet obsługuje operacje związane z przydzielaniem sal i instruktorów do zajęć, monitorowaniem statystyk oraz generowaniem raportów.

Zaimplementowano funkcje pozwalające na wyszukiwanie dostępnych sal i instruktorów, obliczanie wynagrodzeń instruktorów oraz analizowanie popularności zajęć. Dodatkowo, procedury umożliwiają tworzenie nowych grup zajęciowych, generowanie raportów efektywności i wynagrodzeń oraz wysyłanie powiadomień do klientów.

Pakiet pełni kluczową rolę w zarządzaniu operacyjnym klubu fitness, wspierając procesy organizacyjne i analityczne poprzez optymalizację przydziału zasobów i automatyzację powiadomień.

```
create or replace PACKAGE ADMINISTRATOR_PAKIET AS

    -- Deklaracje funkcji
    FUNCTION pobierz_dostepne_sale(
        p_id_zajec      IN INTEGER,
        p_data_rozpoczecia IN DATE,
        p_czas_trwania   IN INTEGER,
        p_liczba_miejsc  IN INTEGER
    ) RETURN DostepneSaleTab PIPELINED;

    FUNCTION pobierz_dostepnych_instruktorow(
        p_id_sali       IN INTEGER,
        p_data_rozpoczecia IN DATE,
        p_czas_trwania   IN INTEGER
    ) RETURN ListaInstruktorowTab PIPELINED;

    FUNCTION oblicz_pensje_instruktora(
        p_id_instruktora IN INTEGER,
        p_data_poczatkowa IN DATE,
        p_data_koncowa    IN DATE
    ) RETURN NUMBER;

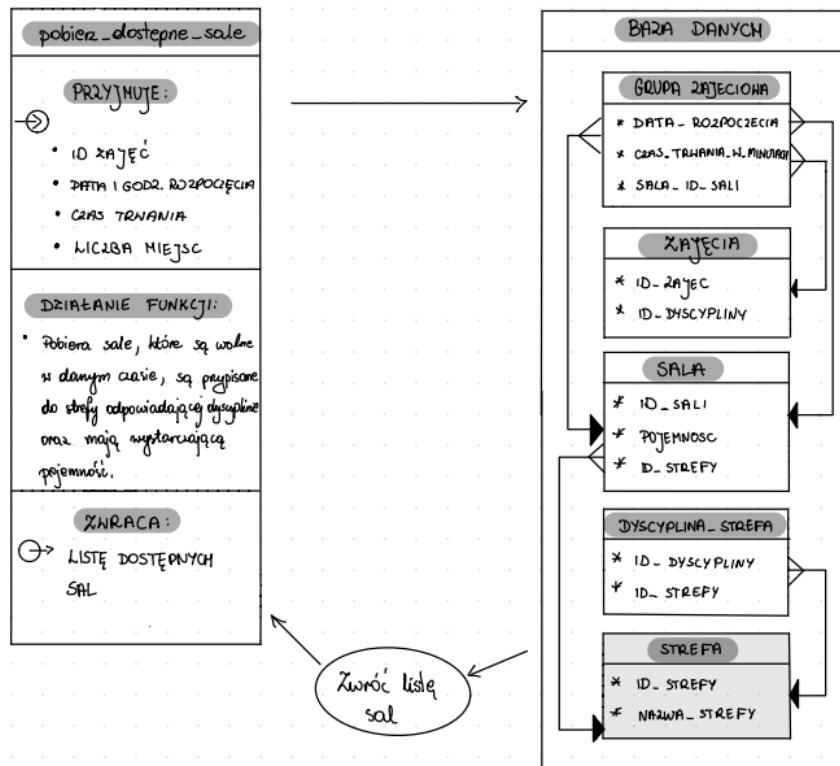
    FUNCTION najpopularniejsze_zajecia(
        p_data_od IN DATE,
        p_data_do IN DATE
    ) RETURN TabelaPopularneZajecia;

    PROCEDURE AnalizujEfektywnoscInstruktora(
        p_data_poczatkowa IN DATE,
        p_data_koncowa    IN DATE
    );
    PROCEDURE GenerujRaportWyplat(
        p_data_poczatkowa IN DATE,
        p_data_koncowa    IN DATE
    );
    PROCEDURE UtworzGrupeZajeciowa(
        p_id_zajec      IN INTEGER,
        p_data_rozpoczecia IN DATE,
        p_czas_trwania   IN INTEGER,
        p_liczba_miejsc  IN INTEGER,
        p_poziom_zaawansowania IN VARCHAR2
    );
    PROCEDURE WyslijPrzypomnienie;

END ADMINISTRATOR_PAKIET;
/
```

Pakiet Administrator – Funkcje

- Pobierz_dostepne_sale



Funkcja **pobierz_dostepne_sale** służy do wyszukiwania dostępnych sal dla zajęć w określonym przedziale czasowym, uwzględniając ich przypisanie do odpowiednich stref oraz minimalną wymaganą pojemność. Jej głównym zadaniem jest zwrócenie listy sal, które spełniają te warunki. Aby to osiągnąć, funkcja wykorzystuje informacje o zajęciach, dyscyplinach, przypisanych strefach oraz wcześniejszych rezerwacjach sal w bazie danych. Kluczowe kryteria filtrowania obejmują zgodność sali z dyscypliną zajęć, brak wcześniejszych rezerwacji w danym czasie oraz wystarczającą pojemność sali. Dzięki temu można znaleźć optymalną salę dla nowych zajęć.

Obsługa błędów:

- Sprawdzenie, czy wszystkie wymagane parametry zostały wprowadzone;
- Sprawdzenie, za pomocą wyrażeń regularnych, czy czas trwania i liczba miejsc są liczbami.

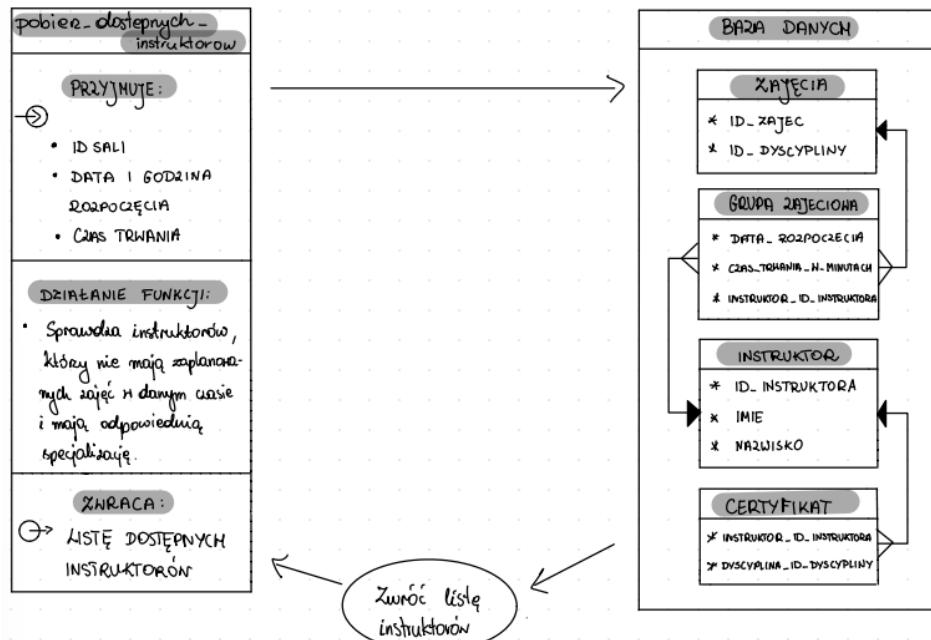
```

FUNCTION pobierz_dostepne_sale(
    p_id_zajec      IN INTEGER,
    p_data_rozpoczecia IN DATE,
    p_czas_trwania   IN INTEGER,
    p_liczba_miejsc   IN INTEGER
) RETURN DostepneSaleTab PIPELINED
IS
BEGIN
    FOR sala_rec IN (
        SELECT s.id_sali, s.pojemnosc, st.nazwa_strefy
        FROM sala s
        JOIN dyscyplina_strefa ds ON s.id_strefy = ds.id_strefy
        JOIN zajecia z ON z.id_dyscypliny = ds.id_dyscypliny
        JOIN strefa st ON s.id_strefy = st.id_strefy
        WHERE z.id_zajec = p_id_zajec
            AND s.pojemnosc >= p_liczba_miejsc
            AND NOT EXISTS (
                SELECT 1
                FROM grupa_zajeciodowa g
                WHERE g.sala_id_sali = s.id_sali
                    AND g.data_rozpoczecia BETWEEN p_data_rozpoczecia
                                                AND p_data_rozpoczecia + (p_czas_trwania / 1440)
            )
    ) LOOP
        PIPE ROW (TypSala(sala_rec.id_sali, sala_rec.pojemnosc, sala_rec.nazwa_strefy));
    END LOOP;
    -- Obsługa błędu, jeśli nie znaleziono żadnych wyników
    IF SQL%ROWCOUNT = 0 THEN
        RETURN;
    END IF;
    RETURN;
EXCEPTION
    WHEN OTHERS THEN
        RETURN;
END pobierz_dostepne_sale;

```

W implementacji funkcja pobiera cztery parametry: identyfikator zajęć, datę rozpoczęcia, czas trwania oraz liczbę miejsc. Wykorzystuje **pipelined table**, co pozwala zwracać listę dostępnych sal w sposób wydajny. Wewnętrznie funkcja łączy tabele sala, dyscyplina_strefa, zajecia oraz strefa, aby sprawdzić zgodność strefy sali z dyscypliną zajęć. Następnie weryfikuje, czy dana sala nie jest już zarezerwowana w podanym przedziale czasowym. Jeśli znajdzie odpowiednie sale, zwraca ich listę z identyfikatorem, pojemnością oraz nazwą przypisanej strefy. Obsługuje również przypadki, gdy brak jest dostępnych sal, zwracając pusty wynik zamiast błędu.

- **Pobierz_dostepnych_instruktorow**



Funkcja **pobierz_dostepnych_instruktorow** służy do wyszukiwania instruktorów, którzy mogą poprowadzić zajęcia w wybranej sali, w określonym przedziale czasowym. Proces rozpoczyna się od sprawdzenia, jaka dyscyplina jest przypisana do danej sali, ponieważ instruktor musi posiadać certyfikat w odpowiedniej dziedzinie. Następnie funkcja przeszukuje bazę danych w poszukiwaniu instruktorów spełniających kryteria dostępności. Instruktorzy, którzy mają już zaplanowane zajęcia w tym samym czasie, są wykluczani z wyniku. Ostatecznie zwracana jest lista instruktorów, którzy mogą poprowadzić zajęcia, spełniając wymagania związane zarówno z dostępnością, jak i kwalifikacjami.

Obsługa błędów:

- Sprawdzenie, czy wszystkie wymagane parametry zostały wprowadzone;
- Sprawdzenie, za pomocą wyrażeń regularnych, czy czas trwania jest liczbą.

```

FUNCTION pobierz_dostepnych_instruktorow(
    p_id_sali           IN INTEGER,
    p_data_rozpoczecia IN DATE,
    p_czas_trwania     IN INTEGER
) RETURN ListaInstruktorowTab PIPELINED
IS
    v_id_dyscypliny INTEGER;
BEGIN
    -- Jeżeli ID sali jest NULL, zwracamy pustą listę instruktorów
    IF p_id_sali IS NULL THEN
        RETURN;
    END IF;

    -- Pobranie ID dyscypliny dla danej sali
    BEGIN
        SELECT DISTINCT z.id_dyscypliny
        INTO v_id_dyscypliny
        FROM sala s
        JOIN dyscyplina_strefa ds ON s.id_strefy = ds.id_strefy
        JOIN zajecia z ON ds.id_dyscypliny = z.id_dyscypliny
        WHERE s.id_sali = p_id_sali
        FETCH FIRST 1 ROW ONLY;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN; -- Zamiast błędu, zwracamy pustą listę instruktorów
    END;

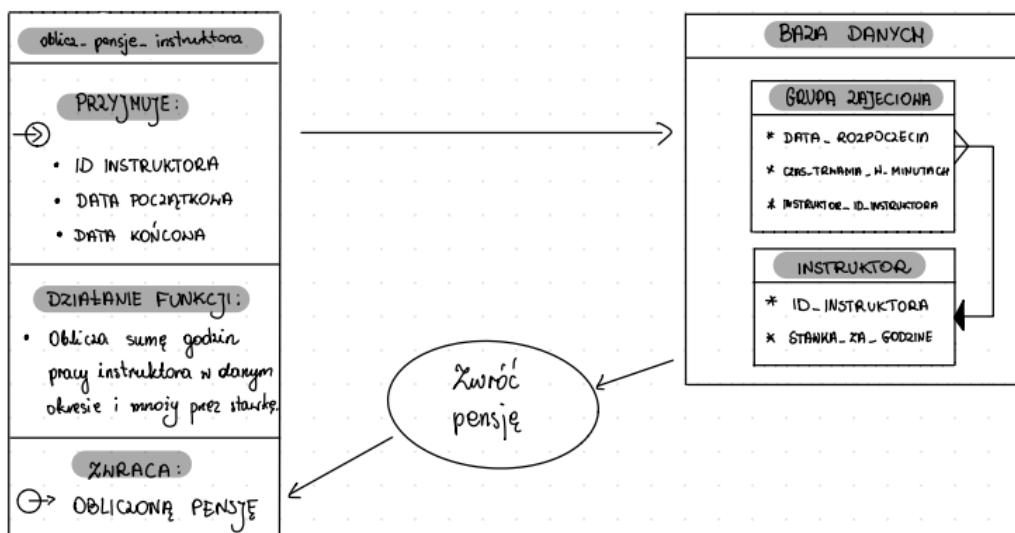
    -- Pobranie dostępnych instruktorów
    FOR instr IN (
        SELECT i.id_instruktora, i.imie, i.nazwisko
        FROM instruktor i
        JOIN certyfikat c ON i.id_instruktora = c.instruktor_id_instruktora
        WHERE c.dyscyplina_id_dyscypliny = v_id_dyscypliny
        AND NOT EXISTS (
            SELECT 1
            FROM grupa_zajeciodawa g
            WHERE g.instruktor_id_instruktora = i.id_instruktora
            AND (
                g.data_rozpoczecia < p_data_rozpoczecia + (p_czas_trwania / 1440)
                AND g.data_rozpoczecia + (g.czas_trwania_w_minutach / 1440) > p_data_rozpoczecia
            )
        )
    ) LOOP
        PIPE ROW (TypInstruktor(instr.id_instruktora, instr.imie, instr.nazwisko));
    END LOOP;

    RETURN;
END pobierz_dostepnych_instruktorow;

```

Implementacja w PL/SQL wykorzystuje mechanizmy zapewniające efektywne wyszukiwanie i filtrowanie danych. Funkcja pobiera ID dyscypliny powiązanej z daną salą poprzez złączenia tabel sala, dyscyplina_strefa oraz zajęcia, a następnie sprawdza dostępnych instruktorów na podstawie ich certyfikacji. Instruktorzy, którzy mają już przypisane zajęcia w danym przedziale czasowym, są odfiltrowywani za pomocą podzapytania NOT EXISTS, co pozwala na szybkie sprawdzenie ich harmonogramu. Zastosowanie **pipelined table** sprawia, że wyniki mogą być zwracane i przetwarzane natychmiastowo, co zwiększa wydajność zapytań SQL. Obsługa błędów zapewnia, że w przypadku braku danych funkcja zwróci pustą listę zamiast generowania błędu, co zwiększa jej stabilność w systemie.

- **Oblicz_pensje_instruktora**



Funkcja **oblicz_pensje_instruktora** służy do wyliczenia wynagrodzenia instruktora za określony przedział czasowy. Przyjmuje jako parametry identyfikator instruktora oraz zakres dat, w którym ma zostać obliczona jego pensja. Działanie funkcji opiera się na pobraniu informacji o zajęciach prowadzonych przez danego instruktora z tabeli grupa_zajeciova, gdzie przechowywane są dane dotyczące czasu trwania zajęć. Następnie liczba minut przeprowadzonych zajęć jest sumowana, przeliczana na godziny i mnożona przez stawkę godzinową instruktora pobraną z tabeli instruktor. Wynikiem działania funkcji jest wartość określająca należną pensję za podany okres.

Obsługa błędów:

- Sprawdzenie, czy data początkowa i końcowa zostały wprowadzone;
- Sprawdzenie, czy data końcowa jest późniejsza niż początkowa;
- Sprawdzenie, czy istnieją instruktory, którzy pracowali w podanym terminie.

```

FUNCTION oblicz_pensje_instruktora(
    p_id_instruktora IN INTEGER,
    p_data_początkowa IN DATE,
    p_data_koncowa IN DATE
) RETURN NUMBER IS
    v_pensja NUMBER := 0;
BEGIN
    -- Obliczenie sumy godzin pracy i pensji
    SELECT ROUND(NVL(SUM(g.czas_trwania_w_minutach / 60 * i.stawka_za_godzine), 0), 2)
    INTO v_pensja
    FROM grupa_zajeciova g
    JOIN instruktor i ON g.instruktor_id_instruktora = i.id_instruktora
    WHERE g.instruktor_id_instruktora = p_id_instruktora
        AND g.data_rozpoczecia BETWEEN p_data_początkowa AND p_data_koncowa;

    RETURN v_pensja;

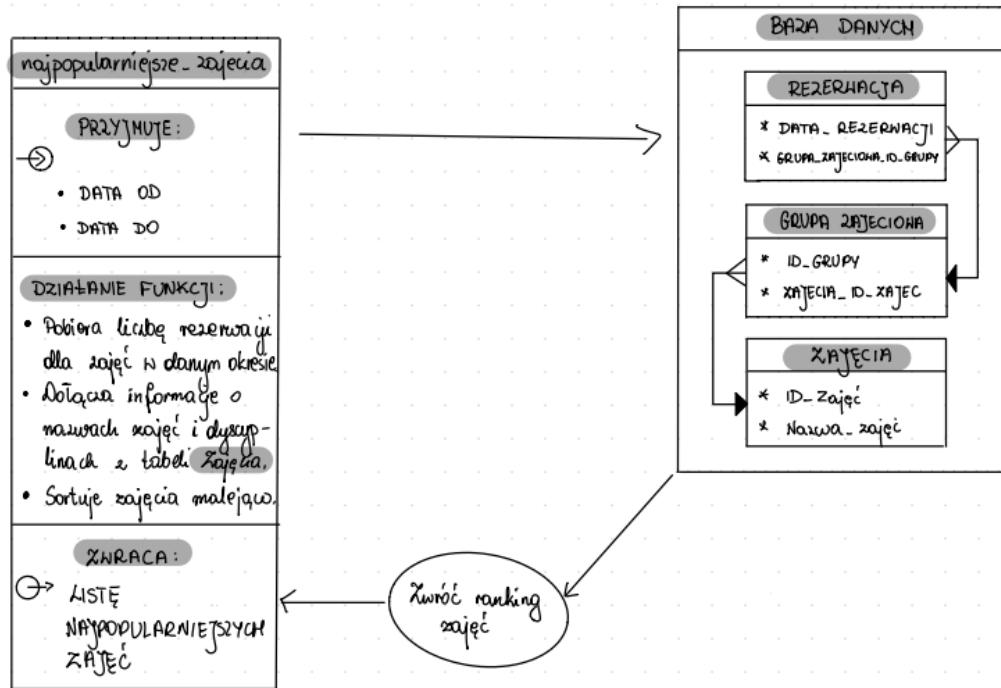
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-21003, 'Nieoczekiwany błąd: ' || SQLERRM);
END oblicz_pensje_instruktora;

```

Implementacja w PL/SQL opiera się na wykonaniu zapytania SQL, które pobiera łączny czas trwania zajęć dla wskazanego instruktora w podanym zakresie dat. Zapytanie wykorzystuje SUM() do agregacji wartości oraz NVL(), aby w przypadku braku danych zwrócić zero. Wynik jest przeliczany na godziny poprzez podzielenie sumy minut przez 60, a następnie mnożony przez stawkę godzinową. Całość jest

zaokrąglana do dwóch miejsc po przecinku przy użyciu funkcji ROUND(). W przypadku błędu funkcja wykorzystuje obsługę wyjątków.

- **Najpopularniejsze_zajecia**



Funkcja **najpopularniejsze_zajecia** pobiera liczbę rezerwacji dla poszczególnych zajęć w podanym przedziale czasowym, co pozwala na określenie, które zajęcia cieszyły się największym zainteresowaniem. Wykorzystuje ona dane z tabel rezerwacja, grupa_zajeciowa i zajecia, gdzie każda rezerwacja jest przypisana do grupy zajęciowej, a następnie do konkretnego rodzaju zajęć. Wynik zwracany przez funkcję to lista zajęć posortowana malejąco według liczby rezerwacji, dzięki czemu administrator może łatwo przeanalizować popularność poszczególnych treningów i dostosować ofertę klubu.

Obsługa błędów:

- Sprawdzenie, czy data początkowa i końcowa zostały wprowadzone;
- Sprawdzenie, czy data końcowa jest późniejsza niż początkowa;
- Sprawdzenie, czy w podanym terminie odbywały się zajęcia.

```

FUNCTION najpopularniejsze_zajecia(
    p_data_od IN DATE,
    p_data_do IN DATE
) RETURN TabelaPopularneZajecia IS
    v_lista_zajec TabelaPopularneZajecia := TabelaPopularneZajecia();

    CURSOR c_zajecia IS
        SELECT z.nazwa_zajec, COUNT(r.nr_rezerwacji) AS ilosc_rezerwacji
        FROM rezerwacja r
        JOIN grupa_zajecia g ON r.grupa_zajecia_id_grupy = g.id_grupy
        JOIN zajecia z ON g.zajecia_id_zajec = z.id_zajec
        WHERE g.data_zajecia BETWEEN p_data_od AND p_data_do
        GROUP BY z.nazwa_zajec
        ORDER BY ilosc_rezerwacji DESC;

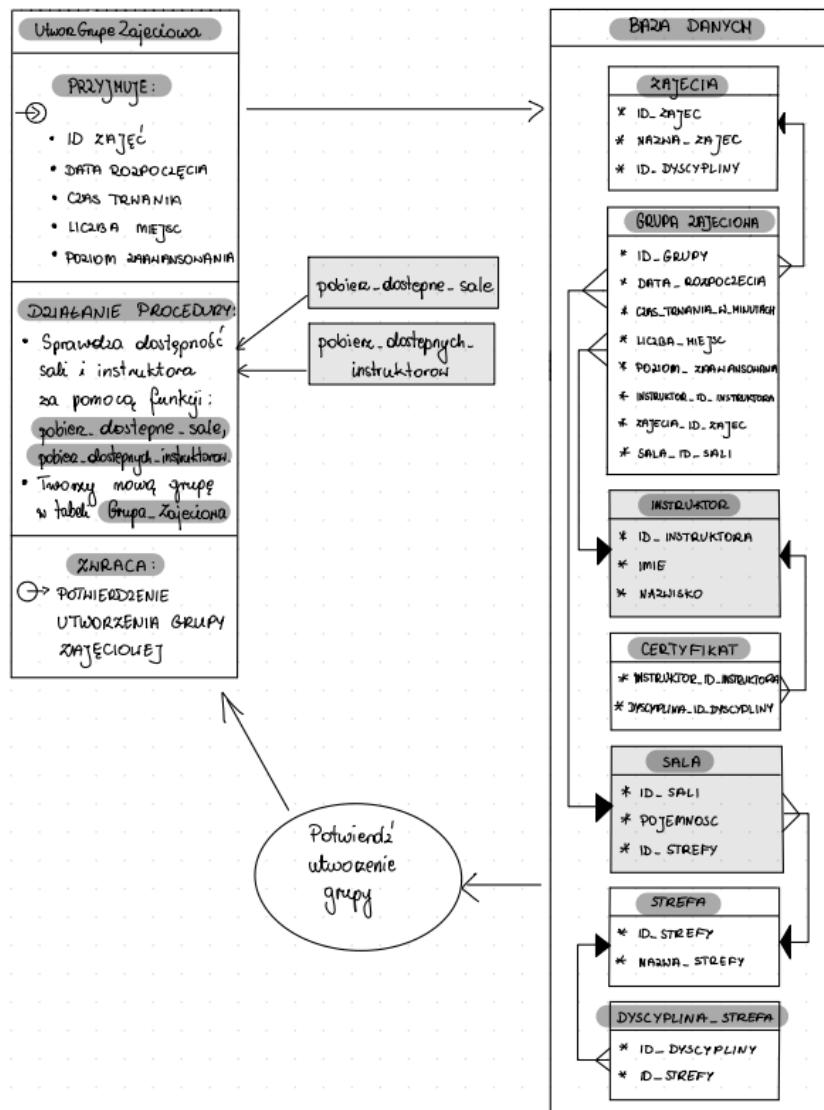
    BEGIN
        FOR v_record IN c_zajecia LOOP
            v_lista_zajec.EXTEND;
            v_lista_zajec(v_lista_zajec.LAST) := TypPopularneZajecia(
                v_record.nazwa_zajec,
                v_record.ilosc_rezerwacji
            );
        END LOOP;
    RETURN v_lista_zajec;
    EXCEPTION
        WHEN OTHERS THEN
            RAISE_APPLICATION_ERROR(-20004, 'Nieoczekiwany błąd: ' || SQLERRM);
    END najpopularniejsze_zajecia;

```

Pod względem implementacji funkcja przyjmuje zakres dat jako parametry wejściowe i wykorzystuje **kursor** do iteracji po wynikach zapytania SQL. W zapytaniu wykorzystywane są połączenia tabel w celu pobrania liczby rezerwacji dla każdego zajęcia oraz zastosowanie grupowania i sortowania, co umożliwia obliczenie rankingu popularności. Dane są następnie przechowywane w **kolekcji** PL/SQL, do której dodawane są kolejne rekordy w pętli, a ostateczny wynik jest zwracany jako **tabela** z nazwami zajęć i odpowiadającą im liczbą rezerwacji. Funkcja uwzględnia również obsługę błędów, co pozwala na odpowiednią reakcję w przypadku wystąpienia nieoczekiwanych problemów.

Pakiet Klient – Procedury

- **UtworzGrupeZajeciowa**



Procedura **UtworzGrupeZajeciowa** odpowiada za tworzenie nowych grup zajęciowych w bazie danych. Jej głównym celem jest weryfikacja dostępności sali oraz instruktora na podany termin zajęć i dopiero po spełnieniu warunków dodanie nowego wpisu do tabeli **grupa_zajeciowa**. Procedura przyjmuje jako parametry identyfikator zajęć, datę rozpoczęcia, czas trwania, liczbę miejsc oraz poziom zaawansowania. W pierwszej kolejności sprawdzana jest dostępność sali oraz instruktora – oba sprawdzenia są realizowane przez wywołanie funkcji pomocniczych **pobierz_dostepne_sale** oraz **pobierz_dostepnych_instruktorow**. Jeśli dla podanych warunków znajdzie się zarówno wolna sala, jak i dostępny instruktor, zostaje utworzona nowa grupa zajęciowa, której dane zapisywane są w tabeli **grupa_zajeciowa**. W przeciwnym razie procedura kończy się bez wprowadzania zmian.

Obsługa błędów:

- Sprawdzenie, czy istnieją dostępne sale dla podanych parametrów z wykorzystaniem funkcji **pobierz_dostepne_sale**;
- Sprawdzenie, czy istnieją dostępni instruktorzy dla podanych parametrów z wykorzystaniem funkcji **pobierz_dostepnych_instruktorow**.

```

PROCEDURE UtworzGrupeZajeciowa(
    p_id_zajec          IN INTEGER,
    p_data_rozpoczecia  IN DATE,
    p_czas_trwania      IN INTEGER,
    p_liczba_miejsc     IN INTEGER,
    p_poziom_zaawansowania IN VARCHAR2
) IS
    -- Zmienne lokalne
    v_id_instruktora INTEGER := NULL;
    v_id_sali         INTEGER := NULL;

BEGIN
    -- Pobranie dostępnej sali (jeśli nie ma, nie tworzymy grupy)
    BEGIN
        SELECT id_sali
        INTO v_id_sali
        FROM TABLE(pobierz_dostepne_sale(p_id_zajec, p_data_rozpoczecia, p_czas_trwania,
        p_liczba_miejsc))
        WHERE ROWNUM = 1;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            v_id_sali := NULL;
    END;

    -- Pobranie dostępnego instruktora (jeśli nie ma, nie tworzymy grupy)
    BEGIN
        SELECT id_instruktora
        INTO v_id_instruktora
        FROM TABLE(pobierz_dostepnych_instruktorow(v_id_sali, p_data_rozpoczecia, p_czas_trwania))
        WHERE ROWNUM = 1;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            v_id_instruktora := NULL;
    END;

    -- Jeśli nie ma dostępnej sali lub instruktora, kończymy procedurę bez zmian w bazie
    IF v_id_sali IS NULL OR v_id_instruktora IS NULL THEN
        RETURN;
    END IF;

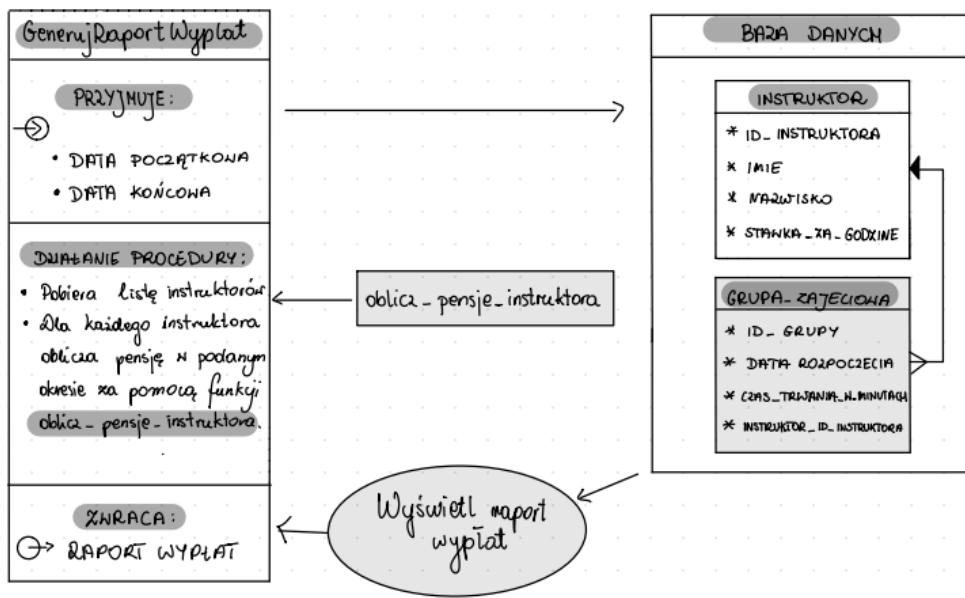
    -- Wstawienie nowej grupy zajęciowej do tabeli
    INSERT INTO grupa_zajeciowa (
        id_grupy, data_rozpoczecia, czas_trwania_w_minutach, liczba_miejsc,
        poziom_zaawansowania, instruktor_id_instruktora, zajecia_id_zajec, sala_id_sali
    ) VALUES (
        grupa_zajeciowa_seq.NEXTVAL,
        p_data_rozpoczecia,
        p_czas_trwania,
        p_liczba_miejsc,
        p_poziom_zaawansowania,
        v_id_instruktora,
        p_id_zajec,
        v_id_sali
    );
    COMMIT; -- Zatwierdzamy tylko w razie sukcesu

    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK; -- Cofnięcie zmian w razie awarii
    END UtworzGrupeZajeciowa;

```

Implementacja w PL/SQL wykorzystuje lokalne zmienne `v_id_sali` i `v_id_instruktora` do przechowywania wartości identyfikatorów znalezionych zasobów. Dane pobierane są z tabel funkcji `pobierz_dostepne_sale` i `pobierz_dostepnych_instruktorow`, a weryfikacja ich dostępności odbywa się poprzez sprawdzenie, czy zmienne te są różne od NULL. Jeśli sala lub instruktor nie zostaną znalezione, procedura kończy działanie przed próbą dodania rekordu do bazy. Operacja dodania grupy zajęciowej realizowana jest przez polecenie `INSERT INTO`, w którym wykorzystywane są wartości pobrane w poprzednich krokach. W przypadku błędów cała transakcja jest cofana przy użyciu `ROLLBACK`, natomiast jeśli operacja zakończy się sukcesem, zmiany są zatwierdzane poleceniem `COMMIT`.

- **GenerujRaportWyplat**



Procedura **GenerujRaportWyplat** służy do generowania raportu wynagrodzeń dla instruktorów w danym przedziale czasowym. Jej głównym zadaniem jest zebranie listy instruktorów i obliczenie dla każdego z nich sumy wynagrodzenia za przepracowane godziny w określonym zakresie dat. Do tego celu wykorzystuje funkcję **oblicz_pensje_instruktora**, która na podstawie danych w tabelach **grupa_zajeciova** i **instruktor** wylicza wynagrodzenie dla pojedynczego instruktora. Raport zawiera ID instruktora, jego imię, nazwisko oraz obliczoną kwotę wynagrodzenia.

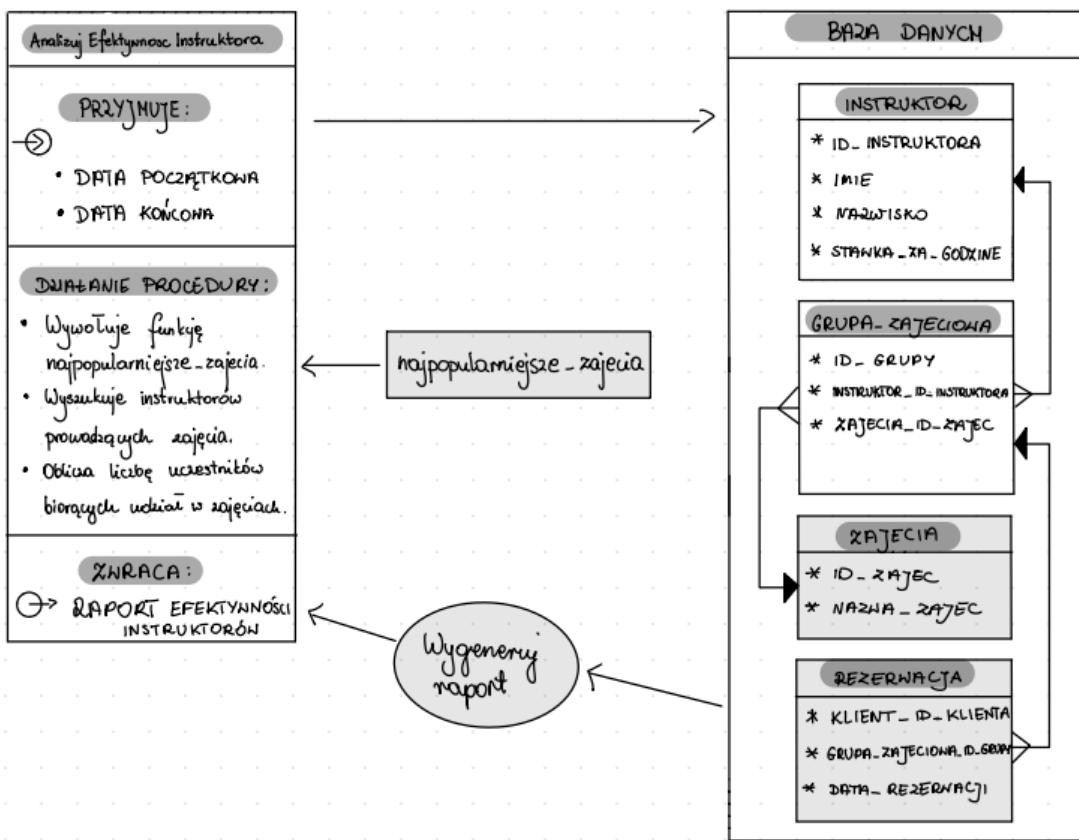
```

PROCEDURE GenerujRaportWyplat(
    p_data_początkowa IN DATE,
    p_data_koncowa     IN DATE
) IS
    v_lista_wyplat WyplatyTab := WyplatyTab();
    CURSOR c_instruktorzy IS
        SELECT id_instruktora, imie, nazwisko
        FROM instruktor;
    v_kwota NUMBER(10,2);
BEGIN
    -- Przetwarzanie instruktorów
    FOR r_instruktor IN c_instruktorzy LOOP
        v_kwota := oblicz_pensje_instruktora(
            r_instruktor.id_instruktora,
            p_data_początkowa,
            p_data_koncowa
        );
        v_lista_wyplat.EXTEND;
        v_lista_wyplat(v_lista_wyplat.LAST) := WyplataInstruktora(
            r_instruktor.id_instruktora,
            r_instruktor.imie,
            r_instruktor.nazwisko,
            v_kwota
        );
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-26005, 'Nieoczekiwany błąd: ' || SQLERRM);
END GenerujRaportWyplat;

```

Procedura została zaimplementowana w PL/SQL i wykorzystuje **kursor** do pobrania listy wszystkich instruktorów z bazy danych. Dla każdego z nich wywoływana jest funkcja **oblicz_pensje_instruktora**, a wynik jej działania jest zapisywany w **kolekcji** (typu tabelarycznego **WyplatyTab**). W ten sposób budowana jest lista instruktorów wraz z ich wynagrodzeniami. Jeśli w trakcie wykonywania procedury wystąpi błąd, generowany jest komunikat o błędzie.

- AnalizujEfektywnoscInstruktora



Procedura **AnalizujEfektywnoscInstruktora** została zaprojektowana w celu oceny skuteczności instruktorów na podstawie liczby zarezerwowanych miejsc na zajęciach, które prowadzili w określonym przedziale czasowym. Przyjmuje dwa parametry wejściowe: **datę początkową i końcową**, określające analizowany okres. Głównym zadaniem procedury jest zebranie danych dotyczących najpopularniejszych zajęć w podanym zakresie, a następnie przypisanie do nich instruktorów prowadzących. Następnie procedura oblicza łączną liczbę miejsc dostępnych na tych zajęciach oraz sumuje liczbę rzeczywiście dokonanych rezerwacji, co pozwala określić stopień obłożenia prowadzonych zajęć. Wynik analizy zostaje zwrocony w postaci raportu, który zawiera identyfikator instruktora, jego imię i nazwisko, liczbę dostępnych miejsc, liczbę rezerwacji oraz procentowe obłożenie prowadzonych zajęć. Dzięki temu raport umożliwia ocenę efektywności poszczególnych instruktorów oraz porównanie ich wyników.

Obsługa błędów:

- Sprawdzenie, czy data początkowa i końcowa zostały wprowadzone;
- Sprawdzenie, czy data końcowa jest późniejsza niż początkowa;
- Sprawdzenie, czy istnieją instruktorzy, którzy pracowali w podanym terminie.

```

PROCEDURE AnalizujEfektywnoscInstruktora(
    p_data_początkowa IN DATE,
    p_data_koncowa     IN DATE
) IS
BEGIN
    -- Usunięcie starej kolekcji, jeśli istnieje
    IF APEX_COLLECTION.COLLECTION_EXISTS('EFEKTYWNOSC_INSTRUKTOROW') THEN
        APEX_COLLECTION.DELETE_COLLECTION('EFEKTYWNOSC_INSTRUKTOROW');
    END IF;

    -- Tworzenie nowej kolekcji
    APEX_COLLECTION.CREATE_COLLECTION('EFEKTYWNOSC_INSTRUKTOROW');

    -- Pobranie danych i zapis do kolekcji
    FOR r IN (
        SELECT i.id_instruktora,
               i.imie,
               i.nazwisko,
               SUM(NVL(g.liczba_miejsc, 0)) AS liczba_miejsc,
               COUNT(rz.nr_rezerwacji) AS liczba_rezerwacji,
               CASE
                   WHEN SUM(NVL(g.liczba_miejsc, 0)) > 0
                   THEN ROUND((COUNT(rz.nr_rezerwacji) / SUM(g.liczba_miejsc)) * 100, 2)
                   ELSE 0
               END AS procent_oblozenia
        FROM instruktor i
        LEFT JOIN grupa_zajeciova g ON i.id_instruktora = g.instruktor_id_instruktora
        LEFT JOIN rezerwacja rz ON g.id_grupy = rz.grupa_zajeciova_id_grupy
                               AND rz.status = 'Zarezerwowano'
        WHERE g.data_rozpoczecia BETWEEN p_data_początkowa AND p_data_koncowa
        GROUP BY i.id_instruktora, i.imie, i.nazwisko
    ) LOOP

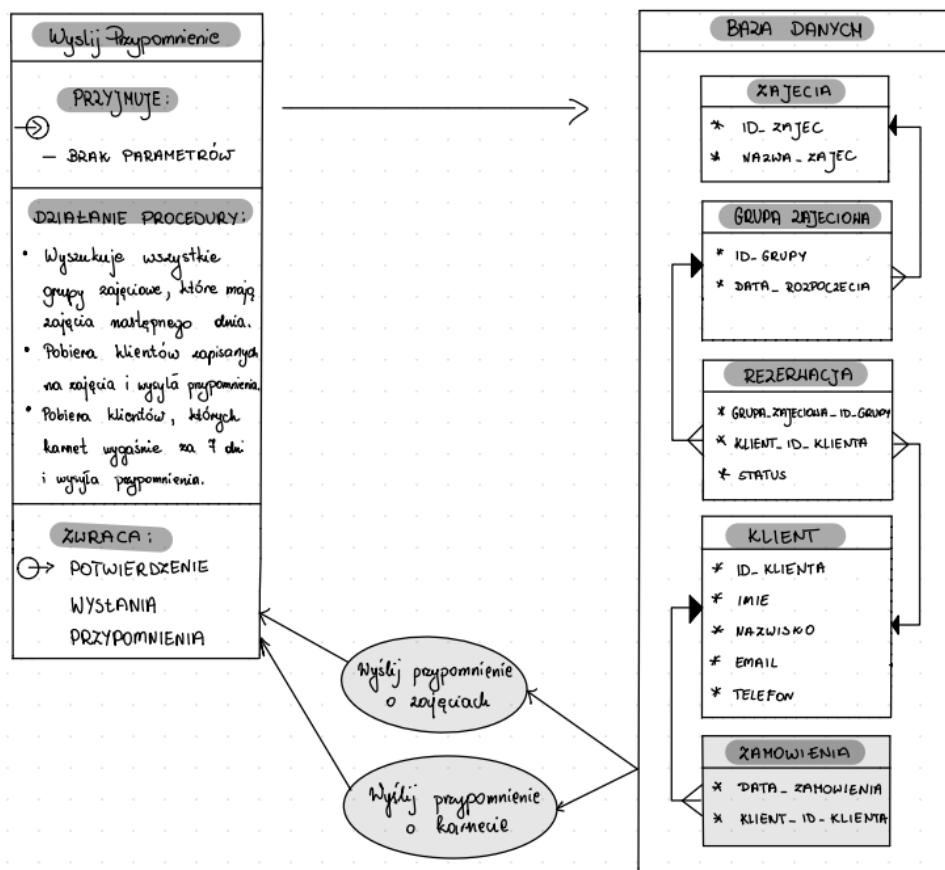
        -- Dodawanie rekordu do kolekcji APEX
        APEX_COLLECTION.ADD_MEMBER(
            p_collection_name => 'EFEKTYWNOSC_INSTRUKTOROW',
            p_c001 => r.id_instruktora,
            p_c002 => r.imie,
            p_c003 => r.nazwisko,
            p_c004 => r.liczba_miejsc,
            p_c005 => r.liczba_rezerwacji,
            p_c006 => r.procent_oblozenia
        );
    END LOOP;

    EXCEPTION
        WHEN OTHERS THEN
            RAISE_APPLICATION_ERROR(-25004, 'Nieoczekiwany błąd: ' || SQLERRM);
    END AnalizujEfektywnoscInstruktora;

```

Procedura wykorzystuje mechanizm **APEX_COLLECTION**, który umożliwia przechowywanie i zarządzanie danymi w kontekście aplikacji APEX. Na początku usuwa ewentualne istniejące dane z kolekcji **EFEKTYWNOSC_INSTRUKTOROW**, aby zapewnić, że analiza opiera się wyłącznie na aktualnych danych. Następnie tworzy nową kolekcję, w której zapisane zostaną wyniki analizy. Pobranie danych realizowane jest poprzez **łączenie tabel instruktorów, grup zajęciowych oraz rezerwacji**. W ramach tej operacji sumowane są dostępne miejsca oraz rezerwacje dla zajęć prowadzonych przez instruktorów w podanym okresie. Obliczany jest również wskaźnik procentowego obłożenia zajęć, który jest określany jako stosunek liczby rezerwacji do liczby miejsc. Każdy wynik jest dodawany do kolekcji APEX za pomocą **APEX_COLLECTION.ADD_MEMBER**. Jeśli podczas działania procedury wystąpi błąd, zwracany jest kod błędu.

- **WyslijPrzypomnienie**



Procedura **WyślijPrzypomnienie** odpowiada za automatyczne wysyłanie wiadomości e-mail do klientów klubu fitness, przypominając im o nadchodzących zajęciach oraz o wygasających karnetach. Procedura nie przyjmuje żadnych parametrów, ponieważ działa w sposób autonomiczny na podstawie aktualnych danych w bazie. W pierwszym kroku identyfikuje wszystkie grupy zajęciowe, których zajęcia odbędą się następnego dnia, a następnie pobiera listę klientów, którzy dokonali rezerwacji na te zajęcia. Dodatkowo wyszukuje klientów, których karnet wygaśnie za siedem dni, analizując historię zamówień. Dla każdego klienta generowana jest spersonalizowana wiadomość e-mail zawierająca informacje o rezerwacji lub wygasającym członkostwie. Na końcu proces wysyła wiadomości i zwraca potwierdzenie wykonania operacji.

Obsługa błędów:

- Sprawdzenie, czy istnieją klienci, którzy posiadają aktywne rezerwacje na kolejny dzień;
- Sprawdzenie, czy istnieją klienci, których karnet wygaśnie za 7 dni.

```

PROCEDURE WyslijPrzypomnienie IS
    -- Kolekcja na przypomnienia
    v_lista_przypomnien KlientPrzypomnienieTab := KlientPrzypomnienieTab();

    -- Zmienne na treść wiadomości
    v_tresc_email CLOB;
    v_tytu_email VARCHAR2(200);

    -- Kursor do pobrania klientów z rezerwacjami na jutro
    CURSOR c_rezerwacje_jutro IS
        SELECT k.id_klienta, k.imie, k.nazwisko, k.email, k.telefon,
               z.nazwa_zajec, g.data_rozpoczecia
        FROM klient k
        JOIN rezerwacja r ON k.id_klienta = r.klient_id_klienta
        JOIN grupa_zajeciova g ON r.grupa_zajeciova_id_grupy = g.id_grupy
        JOIN zajecia z ON g.zajecia_id_zajec = z.id_zajec
        WHERE TRUNC(g.data_rozpoczecia) = TRUNC(SYSDATE) + 1;

    -- Kursor do pobrania klientów, których karnet wygasza za 7 dni
    CURSOR c_karnety_wygasajace IS
        SELECT k.id_klienta, k.imie, k.nazwisko, k.email, k.telefon, z.data_zamowienia
        FROM klient k
        JOIN zamowienia z ON k.id_klienta = z.klient_id_klienta
        WHERE TRUNC(z.data_zamowienia) + 30 = TRUNC(SYSDATE) + 7;

BEGIN
    -- Pobranie klientów z rezerwacjami na jutro
    FOR r IN c_rezerwacje_jutro LOOP
        -- Treść przypomnienia o zajęciach
        v_tytu_email := 'Przypomnienie o jutrzyszych zajęciach: ' || r.nazwa_zajec;
        v_tresc_email := 'Czesc ' || r.imie || ' ' || r.nazwisko || '!,<br><br>' ||
                         'Przypominamy, że masz rezerwację na zajęcia "<b>' || r.nazwa_zajec || '</b>" , które odbędą się w dniu ' || TO_CHAR(r.data_rozpoczecia, 'YYYY-MM-
DD HH24:MI') ||
                         '.<br><br>Zapraszamy!';

        -- Wysywanie e-maila
        APEX_MAIL.SEND(
            p_to => r.email,
            p_from => 'noreply@fitnessclub.com',
            p_subj => v_tytu_email,
            p_body => v_tresc_email
        );
    END LOOP;

    -- Pobranie klientów z wygasającym karnetem za 7 dni
    FOR r IN c_karnety_wygasajace LOOP
        -- Treść przypomnienia o wygasającym karnecie
        v_tytu_email := 'Twój karnet wkrótce wygaśnie!';
        v_tresc_email := 'Czesc ' || r.imie || ' ' || r.nazwisko || '!,<br><br>' ||
                         'Twój karnet wygasza w dniu ' || TO_CHAR(r.data_zamowienia + 30, 'YYYY-MM-
DD') ||
                         '.<br><br>Zapraszamy do odnowienia członkostwa, aby nadal korzystać z
zajęć!';

        -- Wysyłanie e-maila
        APEX_MAIL.SEND(
            p_to => r.email,
            p_from => 'noreply@fitnessclub.com',
            p_subj => v_tytu_email,
            p_body => v_tresc_email
        );
    END LOOP;

    -- Wynagane, aby e-maile zostały wysłane w APEX
    APEX_MAIL.PUSH_QUEUE;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-28002, 'Nieoczekiwany błąd: ' || SQLERRM);
END WyslijPrzypomnienie;

```

Implementacja w PL/SQL wykorzystuje **kursory** do iteracyjnego pobierania danych klientów na podstawie warunków określonych w zapytaniach SQL. Do przechowywania informacji o klientach używana jest **kolekcja** PL/SQL, a treści wiadomości są dynamicznie generowane w zmiennych CLOB i VARCHAR2. Proces wysyłania e-maili realizowany jest za pomocą pakietu **APEX_MAIL**, który umożliwia automatyczne wysyłanie wiadomości e-mail w systemie Oracle APEX. Wiadomości są kolejno wysyłane dla klientów z nadchodzącymi rezerwacjami oraz tych, których karnet wkrótce wygaśnie. Na końcu procedura wykorzystuje **APEX_MAIL.PUSH_QUEUE**, aby zapewnić, że wszystkie wiadomości zostaną przesłane. Obsługa błędów zapewnia, że w razie problemów zgłoszony zostanie wyjątek aplikacyjny, co ułatwia diagnostykę problemów.

Implementacja aplikacji w APEX

Mapa aplikacji

Aplikacja **Fitness Club App** podzielona jest na dwa główne moduły: **Klient** i **Klub**, dostępne z poziomu strony głównej.

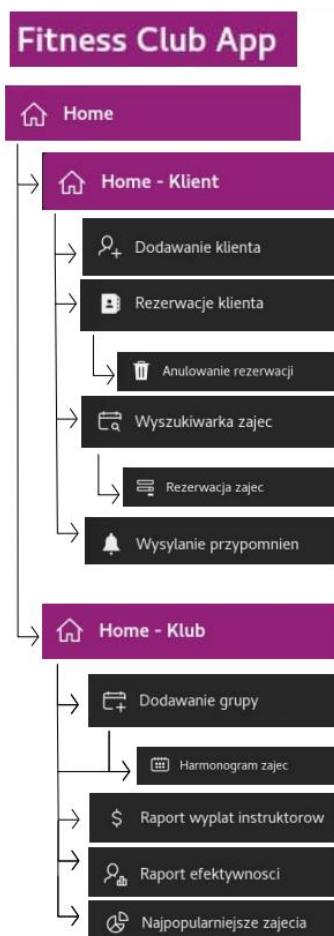
Moduł Klienta - Umożliwia zarządzanie klientami i ich rezerwacjami:

- Dodawanie klienta
- Rezerwacje klienta (z możliwością anulowania)
- Wyszukiwarka zajęć (z opcją rezerwacji)
- Wysyłanie przypomnień o zajęciach

Moduł Klubu - Obsługuje organizację zajęć i analizę statystyk:

- Dodawanie grupy i harmonogram zajęć
- Raport wypłat instruktorów
- Raport efektywności instruktorów
- Najpopularniejsze zajęcia

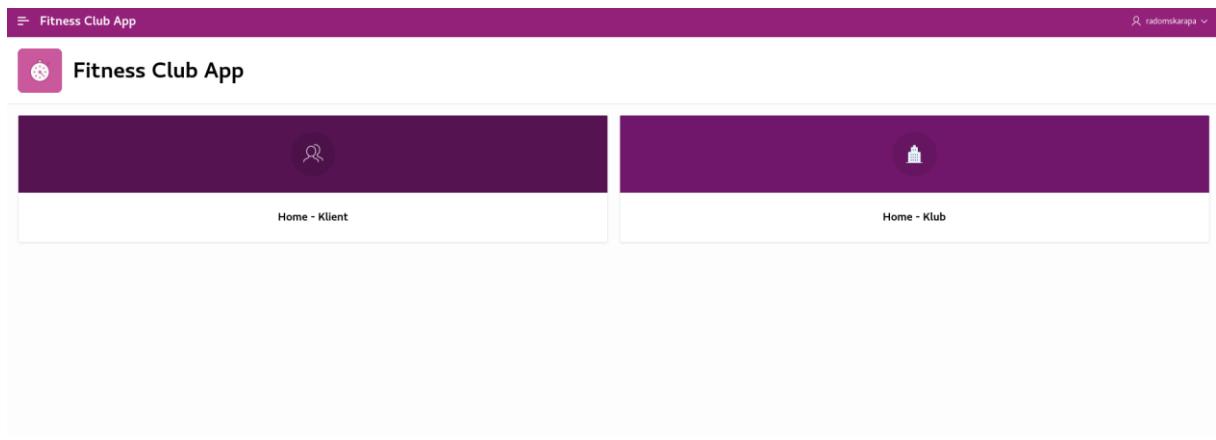
Podział ten zapewnia przejrzystość i łatwy dostęp do kluczowych funkcji systemu.



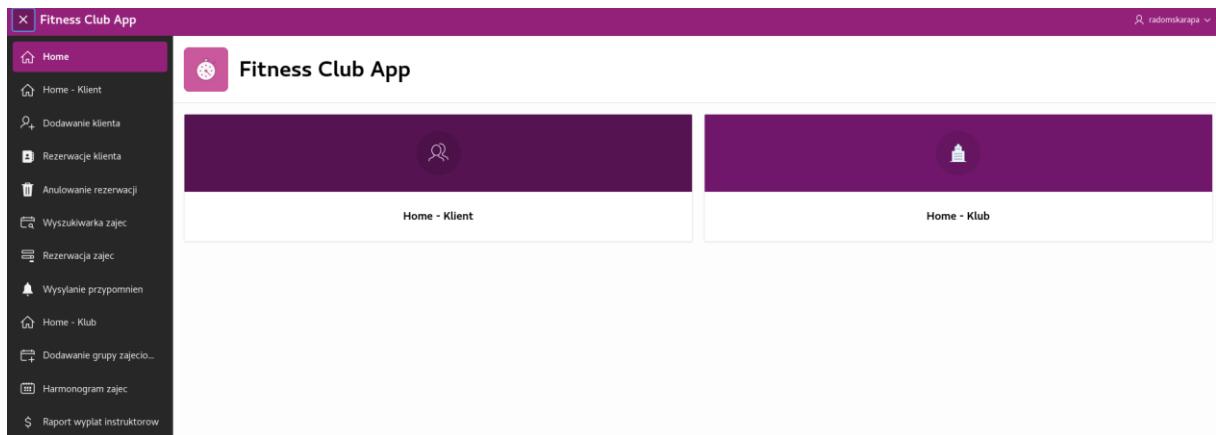
Wgląd i implementacja poszczególnych stron

Strona główna – Home

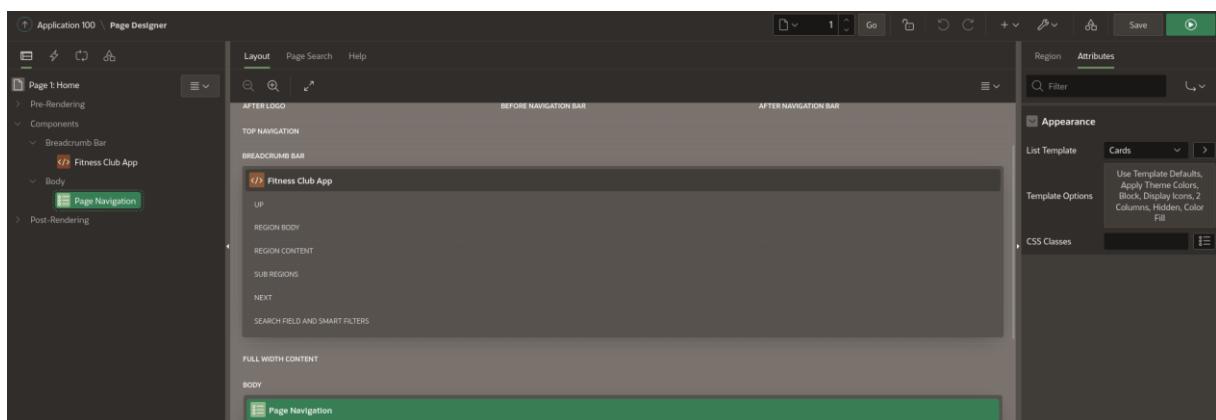
Strona główna aplikacji **Fitness Club App** składa się z dwóch głównych kart: **Home - Klient** i **Home - Klub**, które prowadzą do odpowiednich sekcji.



Po lewej stronie znajduje się nawigacja boczna z odnośnikami do kluczowych funkcji, takich jak dodawanie klientów, rezerwacje, wyszukiwarka zajęć i raporty.

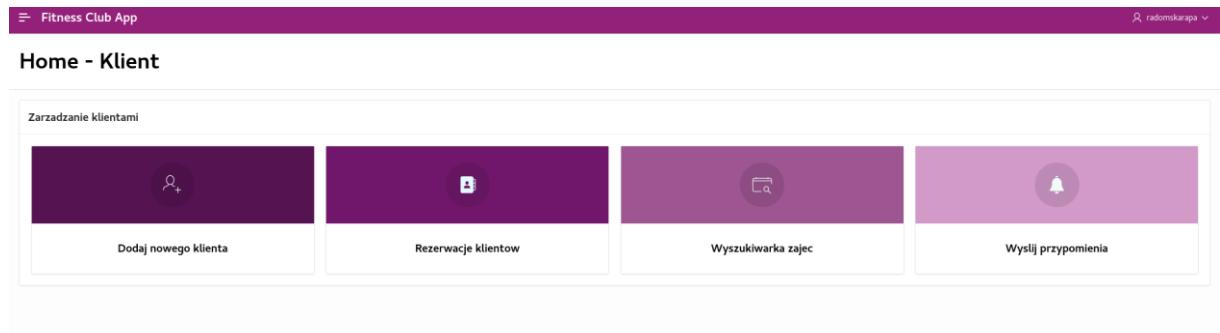


Strona została zaprojektowana w APEX z wykorzystaniem szablonu kart (Cards Template) i listy Page navigation oraz paska nawigacyjnego (Breadcrumb Bar), co zapewnia przejrzystość i intuicyjność.



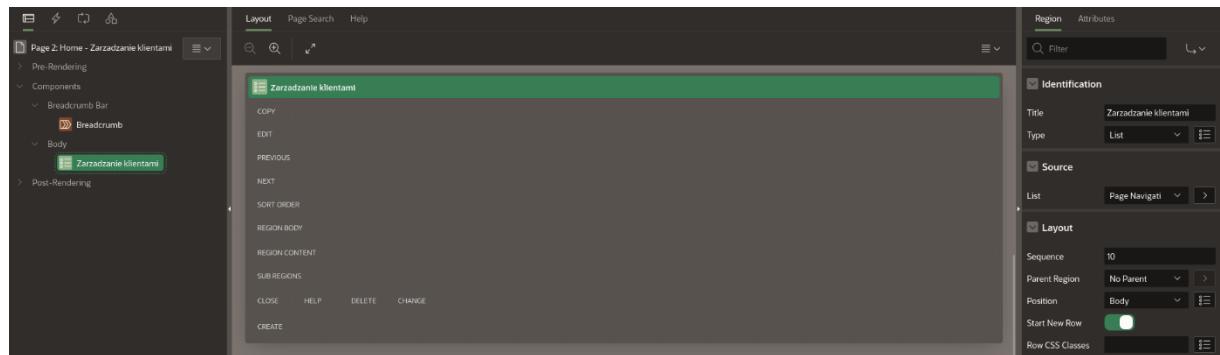
Strona główna – Home Klient (Zarządzanie klientami)

Strona **Home - Klient** prezentuje panel zarządzania klientami w aplikacji. Zawiera cztery interaktywne kafelki umożliwiające szybki dostęp do funkcji takich jak dodawanie nowego klienta, przegląd rezerwacji, wyszukiwanie zajęć oraz wysyłanie przypomnień. Strona jest zorganizowana w sekcje, a jej layout oparty jest na responsywnych kartach.



The screenshot shows the main client management page. At the top, there's a header bar with the application name 'Fitness Club App' and a search bar. Below the header, the title 'Home - Klient' is displayed. The main content area is titled 'Zarządzanie klientami'. It features four large, rounded rectangular cards arranged horizontally. From left to right, the cards are: 'Dodaj nowego klienta' (with a plus sign icon), 'Rezerwacje klientów' (with a person icon), 'Wyszukiwarka zajęć' (with a magnifying glass icon), and 'Wyslij przypomnienia' (with a bell icon). Each card has a small description below it.

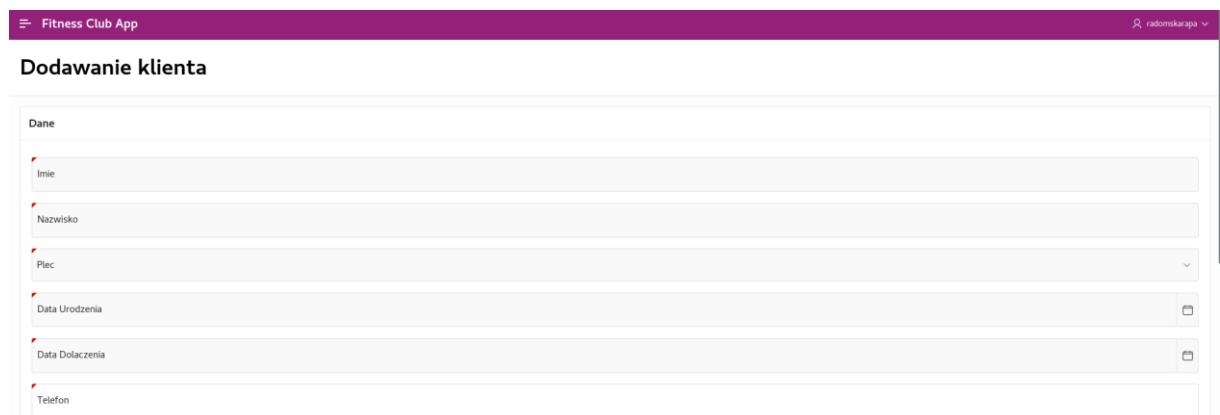
W Page Designer widoczna jest struktura regionów, w której „Zarządzanie klientami” jest listą nawigacyjną, a każda opcja kieruje użytkownika do odpowiedniej funkcjonalności aplikacji.



The screenshot shows the Page Designer interface. On the left, there's a sidebar with navigation sections like 'Pre-Rendering', 'Components', 'Breadcrumb', 'Body', and 'Post-Rendering'. The main area shows a page titled 'Zarządzanie klientami'. This page has a green header bar with the title 'Zarządzanie klientami'. Below the header, there are several buttons: 'COPY', 'EDIT', 'PREVIOUS', 'NEXT', 'SORT ORDER', 'REGION BODY', 'REGION CONTENT', 'SUB REGIONS', 'CLOSE', 'HELP', 'DELETE', 'CHANGE', and 'CREATE'. To the right of the page content, there's a 'Region' panel with tabs for 'Region' and 'Attributes'. Under the 'Region' tab, there's a 'Filter' input field and a 'List' section. The 'List' section contains fields for 'Title' (set to 'Zarządzanie klientami'), 'Type' (set to 'List'), and 'Source' (set to 'Page Navigation'). Below the 'List' section, there's a 'Layout' section with fields for 'Sequence' (set to '10'), 'Parent Region' (set to 'No Parent'), 'Position' (set to 'Body'), and 'Start New Row' (which is checked). There's also a 'Row CSS Classes' field.

Strona dodawania nowego klienta

Na tej stronie użytkownik może dodać nowego klienta do systemu. Formularz zawiera pola do wprowadzenia podstawowych danych, takich jak imię, nazwisko, płeć, data urodzenia, data dołączenia, telefon i e-mail, a także sekcję adresową obejmującą ulicę, numer domu, numer mieszkania, kod pocztowy i miasto.



The screenshot shows the 'Dodawanie klienta' (Adding Client) form. At the top, there's a header bar with the application name 'Fitness Club App' and a search bar. Below the header, the title 'Dodawanie klienta' is displayed. The main form area is titled 'Dane'. It contains several input fields, each with a red asterisk indicating it's required. The fields are: 'Imię' (First Name), 'Nazwisko' (Last Name), 'Płeć' (Gender), 'Data Urodzenia' (Date of Birth), 'Data Dołączenia' (Join Date), and 'Telefon' (Phone). Each field has a placeholder text and a small info icon to its right.

Fitness Club App

Dodawanie klienta

Email

Adres

Ulica

Nr Domu

Nr Mieszkania

Kod Pocztowy

Miasto

Powrót

Dodaj Klienta

Wprowadzane dane są walidowane pod kątem poprawności – sprawdzana jest m.in. obecność wymaganych pól, format numeru telefonu (musi zawierać dokładnie 9 cyfr), format kodu pocztowego (XX-XXX) oraz poprawność adresu e-mail, a także wielkość liter w polach: imię, nazwisko, ulica czy miasto.

Fitness Club App

Dodawanie klienta

Dane

Imię
Należy podać imię

Nazwisko
Należy podać nazwisko

Płeć
Należy podać płeć

Data Urodzenia
Należy podać datę urodzenia

Data Dolaczenia
Należy podać datę dołączenia

Dodawanie klienta

Joanna

Nazwisko
ptak
Nazwisko musi zaczynać się wielką literą i zawierać tylko litery

Płeć
Kobieta

Data Urodzenia
2004-02-19

Data Dolaczenia
2025-01-25

Telefon
90063409
Numer telefonu musi składać się dokładnie z 9 cyfr

Email
jptakemail.pl
Niepoprawny format adresu e-mail

Po wypełnieniu formularza i kliknięciu przycisku "Dodaj Klienta" uruchamiana jest procedura **KLIENT_PAKIET.DodajKlienta**, która zapisuje dane w bazie. Jeśli walidacja zakończy się sukcesem, użytkownik zostaje przekierowany do strony głównej z komunikatem potwierdzającym dodanie nowego klienta. W przypadku błędów formularz wyświetla stosowne komunikaty, np. brak wymaganych pól lub niepoprawny format wprowadzonych danych.

The screenshot shows the 'Fitness Club App' interface. At the top, there's a green notification bar with a checkmark icon and the text 'Klient został dodany!'. Below it, the title 'Home - Klient' is displayed. Underneath, there's a section titled 'Zarządzanie klientami' with four cards: 'Dodaj nowego klienta', 'Rezerwacje klientów', 'Wyszukiwarka zajęć', and 'Wyslij przypomnienia'. The main area is labeled 'Application 100 \ Page Designer'. It shows the 'Dodaj nowego klienta' page structure with regions like 'BANNER', 'AFTER LOGO', 'TOP NAVIGATION', 'REGION BODY', 'REGION CONTENT', and 'SUB REGIONS'. The 'REGION BODY' contains fields for 'P8_IMIE', 'P8_NAZWISKO', 'P8_PLEC', 'P8_DATA_URODZENIA', and 'P8_TELEFON'. The 'Page' panel on the right shows details for the page, including its name ('Dodaj nowego klienta'), alias ('dodawanie-klienta1'), title ('Dodaj nowego Klienta'), and page mode ('Normal'). The 'Process' panel on the right shows a process named 'Dodawanie klienta' with a PL/SQL code block:

```

-- Wywołanie procedury dodawania
Klient_Pakiet.DodaKlienta(
    p_imie      => :P8_IMIE,
    p_nazwisko  => :P8_NAZWISKO,
    p_plec      => :P8_PLEC,
    p_data_urodenia => :P8_DATA_URODZENIA,
    p_data_dolaczania => :P8_DATA_DOLACZENIA
);

```

Przykłady walidacji:

- Sprawdzenie poprawnego formatu wprowadzanych danych

The screenshot shows the 'Validation' configuration dialog. It includes sections for 'Editable Region' (set to '- Select -'), 'Type' (set to 'Item matches'), 'Item' (set to 'P8_EMAIL'), 'Regular Expression' (set to '^([A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,})\$'), and 'Always Execute' (checkbox is checked). In the 'Error' section, the 'Error Message' is set to 'Niepoprawny format adresu e-mail'. The 'Display Location' is set to 'Inline with Field', and the 'Associated Item' is set to 'P8_EMAIL'.

- Sprawdzenie, czy klient już istnieje w bazie danych

The screenshot shows a configuration interface for validation and error handling. In the Validation section, there is an Editable Region dropdown set to "Select", a Type dropdown set to "No Rows returned", and an SQL Query input field containing:

```
SELECT 1
FROM Klient
WHERE telefon = :P4_TELEFON
```

The "Always Execute" toggle switch is turned on. In the Error section, there is an Error Message input field containing "Klient o podanych danych już istnieje". The Display Location dropdown is set to "Inline in Notification", and the Associated Item dropdown is set to "P8_TELEFON".

Strona do wyszukiwania rezerwacji klienta

Strona **Rezerwacje klienta** umożliwia użytkownikowi przeglądanie i zarządzanie rezerwacjami danego klienta. U góry znajduje się sekcja z listą rozwijaną, w której należy wybrać klienta.

This screenshot shows a modal dialog titled "Wybierz klienta" (Select Client). It contains a dropdown menu with one item labeled "Klient". Below the dropdown are two buttons: "Powrót" (Return) and "Pobierz Rezerwacje" (Get Reservations). The background of the main application window is visible, showing the title bar "Fitness Club App" and a user profile "radomskarapa".

This screenshot shows a modal dialog titled "Rezerwacje klienta" (Client Reservations). It displays a table with one row, which is highlighted with a red border. The table has columns for "Numer rezerwacji" (Reservation number), "Status" (Status), "Data rezerwacji" (Reservation date), "Czas trwania zajęć" (Duration of activities), "Nazwa zajęć" (Activity name), and "Poziom zaawansowania" (Advanced level). Below the table is a "Pobierz Rezerwacje" (Get Reservations) button. The background of the main application window is visible, showing the title bar "Fitness Club App" and a user profile "radomskarapa".

Po wybraniu klienta i kliknięciu przycisku „Pobierz Rezerwacje” wyświetlana jest tabela interaktywna z danymi o rezerwacjach tego klienta, w tym numer rezerwacji, status, datę rezerwacji, czas trwania zajęć, nazwę zajęć i poziom zaawansowania. W tabeli umieszczono także przycisk do anulowania rezerwacji.

	Nr Rezerwacji	Data Rezerwacji	Status	Data Rozpoczecia	Czas Trwania W Minutach	Nazwa Zajęć	Poziom Zaawansowania
1	1	2024-01-10	Zarezerwowano	2024-01-10 10:00	60	Aerobik	Początkujący
11	11	2025-01-20	Anulowano	2024-01-20 11:30	75	Boks	Początkujący
24	24	2025-01-20	Oczekujący	2024-01-20 11:30	75	Boks	Początkujący
25	25	2025-01-20	Zarezerwowano	2024-01-15 12:45	90	Gimnastyka	Zawansowany
38	38	2025-01-24	Zarezerwowano	2025-01-31 15:30	45	Zdrowy kregutup	Dla wszystkich

Walidacje sprawdzają, czy wybrano klienta, a proces „Pobierz rezerwacje” wykorzystuje procedurę **Klient_Pakiет.pobierz_rezerwacje_klienta** z pakietu Klient, która zwraca informacje o rezerwacjach wybranego klienta. Na stronie zastosowano interaktywny raport do prezentacji danych oraz walidacje zapewniające poprawność wprowadzonych informacji.

```

-- Tworzenie linku do aktualizacji rezerwacji
<#> if not exists (select * from user_objects where object_name = 'P6_ANULUJ') then
  create or replace procedure P6_ANULUJ
  as
    begin
      update Klient_Pakiет.pobierz_rezerwacje_klienta
      set status = 'Anuluj'
      where id_rezerwacji = :P6_ANULUJ;
    end;
  end if;

```

Przykład walidacji:

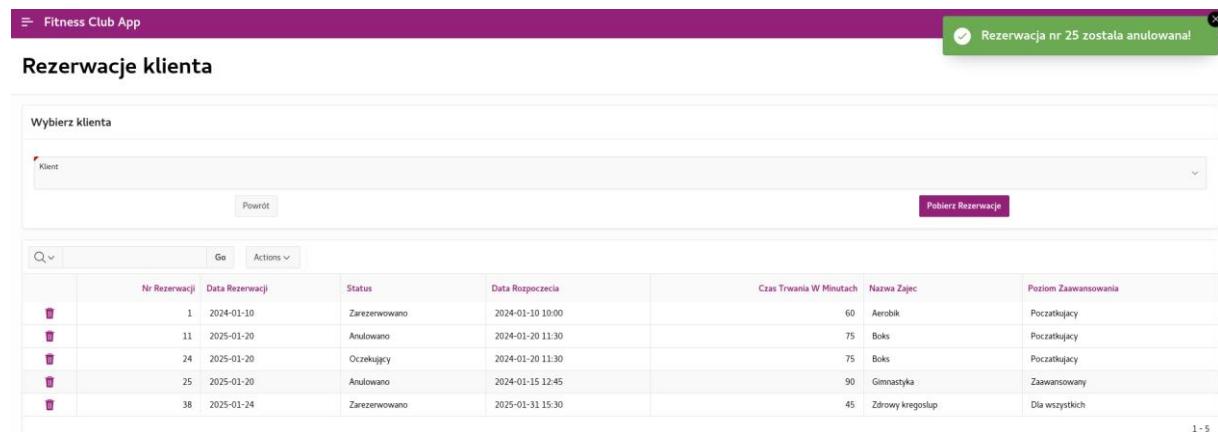
- Sprawdzenie, czy wprowadzono wymagany parametr

Strona do anulowania rezerwacji klienta

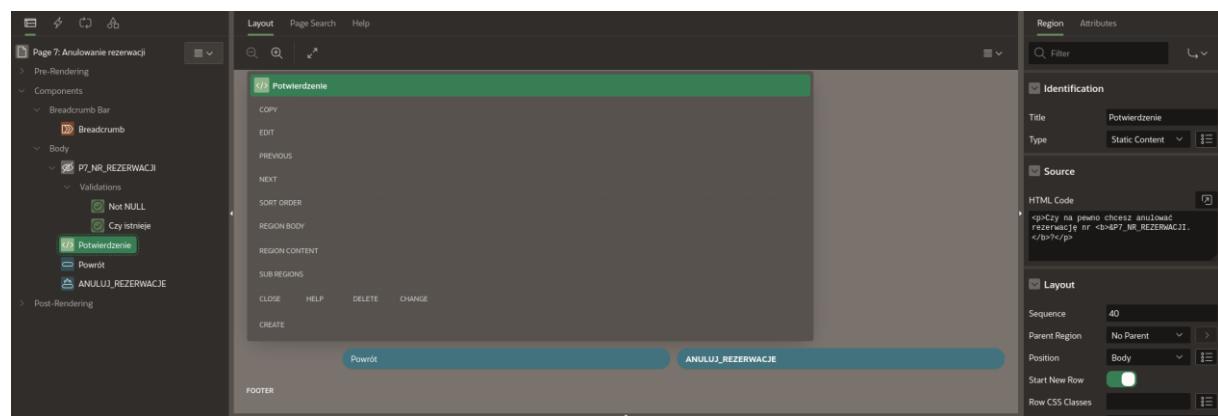
Strona **Anulowanie rezerwacji** w aplikacji APEX służy do potwierdzenia i wykonania anulowania wybranej rezerwacji. Po wejściu na stronę użytkownik widzi komunikat z numerem rezerwacji, którą zamierza anulować, oraz przycisk do potwierdzenia operacji.



Po wykonaniu procedury użytkownik jest przekierowywany z powrotem na stronę „Rezerwacje klienta”, gdzie aktualizowana jest lista rezerwacji, a anulowana pozycja zmienia swój status.



Zastosowane walidacje sprawdzają, czy numer rezerwacji został przekazany i czy istnieje w bazie danych. Proces anulowania jest realizowany poprzez wywołanie procedury **Klient_Pakiет.AnulujRezerwacje**, która zmienia status rezerwacji w bazie danych.



The screenshot shows the Oracle APEX application builder interface. On the left, there's a tree view of application components under 'After Processing'. One item, 'Anulowanie rezerwacji', is highlighted. The main panel displays a confirmation message 'Potwierdzenie' with options like 'COPY', 'EDIT', 'PREVIOUS', 'NEXT', 'REGION BODY', 'REGION CONTENT', and 'SUB REGIONS'. Below this is a toolbar with 'CLOSE', 'HELP', 'DELETE', and 'CHANGE'. To the right, the 'Process' tab is selected in the top navigation bar. Under 'Identification', the process is named 'Anulowanie rezerwacji', has type 'Execute Code', and no execution chain. It's set to an 'Editable Region'. In the 'Source' section, the location is 'Local Database', language is 'PL/SQL', and the code is:

```
CLIENT_PAKIET.AnulujRezerwacje(
    p_nr_rezerwacji => :P7_NR_REZERWACJI
);
```

Przykład walidacji:

- Sprawdzenie, czy rezerwacja o danym numerze istnieje

This screenshot shows the 'Validation' configuration screen. It includes fields for 'Editable Region' (set to '- Select -'), 'Type' (set to 'Rows returned'), and an 'SQL Query' containing the following code:

```
SELECT 1 FROM rezerwacja WHERE nr_rezerwacji = :P7_NR_REZERWACJI
```

Below the query is a toggle switch for 'Always Execute'. The 'Error' section contains an 'Error Message' field with the text: 'Rezerwacja o podanym numerze nie istnieje'.

Strona do wyszukiwania zajęć

Strona **Wyszukiwarka zajęć** umożliwia użytkownikowi filtrowanie dostępnych zajęć według określonych kryteriów, takich jak data początkowa i końcowa, nazwa zajęć, dyscyplina, imię i nazwisko instruktora oraz poziom zaawansowania. Formularz zawiera pola wyboru, a wyniki są prezentowane w interaktywnym raporcie, który dynamicznie wyświetla listę dostępnych zajęć po kliknięciu „Szukaj”.

The screenshot shows the 'Wyszukiwarka zajęć' search form. At the top, there's a header bar with the app name 'Fitness Club App' and a user icon. Below it, the form title is 'Wyszukiwarka zajęć'. The form itself has several input fields grouped under 'Wybierz kryteria':

- Data Początkowa: 2025-01-01
- Data Koncowa: 2025-01-31
- Nazwa Zajęć
- Dyscyplina
- Imię Instruktora
- Nazwisko Instruktora
- Poziom Zaawansowania

At the bottom of the form is a toolbar with buttons for 'Powrót' (Back), navigation icons (Home, App 100, Page 3, Session, Debug, Quick Edit, Customize), and a prominent 'Szukaj' (Search) button.

Wyszukiwarka zajęć

Nazwa Zajęć
Aerobik

Dyscyplina
Fitness

Imię Instruktora
Anna

Nazwisko Instruktora
Kowalska

Poziom Zaawansowania
Średniozaawansowany

Powrót Szukaj

Id Grupy	Data Rozpoczęcia	Czas Trwania	Nazwa Zajęć	Nazwa Dyscypliny	Imię	Nazwisko	Poziom Zaawansowania	Rezerwacja
18	2025-01-29 18:00	60	Aerobik	Fitness	Anna	Kowalska	Średniozaawansowany	ZAREZERWUJ

Zastosowano validację dla pól daty początkowej i końcowej, sprawdzając ich poprawność i zgodność logiczną.

Wyszukiwarka zajęć

1 error has occurred
- Data początkowa nie może być późniejsza niż końcowa ⓘ

Wybierz kryteria

Data Początkowa
2025-01-01

Data Koncowa
2024-12-03

Dane pobierane są z bazy przy użyciu funkcji **KLIENT_PAKIET.wyszukaj_zajecia**, która zwraca listę zajęć spełniających podane kryteria. Wyniki zapytania są filtrowane według statusu zajęć, dostępności oraz przypisanych instruktorów.

Page 3: Wyszukiwarka zajęć

BANNER

BEFORE NAVIGATION BAR

AFTER NAVIGATION BAR

BREADCRUMB BAR

Breadcrumb

UP

REGION BODY

P3_DATA_POCZĄTKOWA

Validations

- Not NULL
- Poprawna data

P3_DATA_KONCOWA

Validations

- Not NULL

P3_NAZWA_ZAJEC

P3_DYSCYPLINA

P3_IMIE_INSTRUKTORA

P3_NAZWISKO_INSTRUKTORA

Identification

Name: Wyszukiwarka zajęć
Alias: wyszukiwarka-zajec
Title: Wyszukiwarka zajęć
Page Group: - Select -

Appearance

Page Mode: Normal
Page Template: Theme Default
Template Options: Use Template Defaults
CSS Classes:
Media Type:

Navigation Menu

Override User Interface Level:

Page 3: Wyszukiwarka zajęć

Layout: Page Search Help

Wyszukiwarka zajęć

BANNER

BEFORE NAVIGATION BAR

AFTER NAVIGATION BAR

BREADCRUMB BAR

Breadcrumb

UP

REGION BODY

P3_DATA_POCZÄTKOWA

P3_NAZWA_ZAJEC

P3_DYSCYPLINA

P3_IMIE_INSTRUKTORA

P3_NAZWISKO_INSTRUKTORA

P3_POZIOM_ZAAWANSOWANIA

Powrót

SZUKAJ

Columns

- ID.GRUPY
- DATA.RZPOCZECIA
- CZAS.TRWANIA
- NAZWA.ZAJEC
- NAZWA.DYSCYPLINY
- IMIE
- NAZWISKO
- POLZOM.ZAAWANSOWANIA
- REZERWACJA

Sort Order

Previous

Region Body

Region Content

Sub Regions

Next

Search Field and Smart Filters

RIGHT OF INTERACTIVE REPORT SEARCH BAR

Region: Attributes: Printing

Type: Interactive Re

Source

Location: Local Database

Type: SQL Query

SQL Query

```
SELECT * FROM KLIENT_PAKIET.wyszukaj_zajecia
WHERE DATE([P3_DATA_POCZÄTKOWA], 'YYYY-MM-DD') <= DATE([P3_DATA_KONCOWA], 'YYYY-MM-DD')
AND DATE([P3_DATA_KONCOWA], 'YYYY-MM-DD') <= DATE([P3_DATA_POCZÄTKOWA], 'YYYY-MM-DD')
```

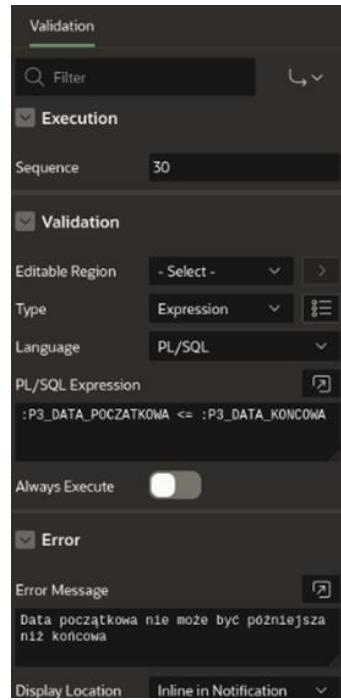
Page Items to Submit: P3_DATA_POCZÄTKOWA

Optimizer Hint:

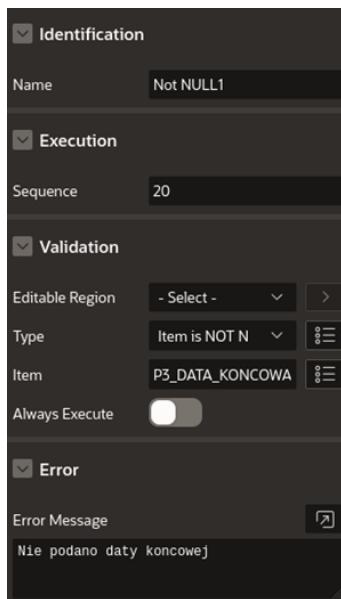
Layout

Przykłady walidacji:

- Sprawdzenie, czy data początkowa jest wcześniejsza niż końcowa



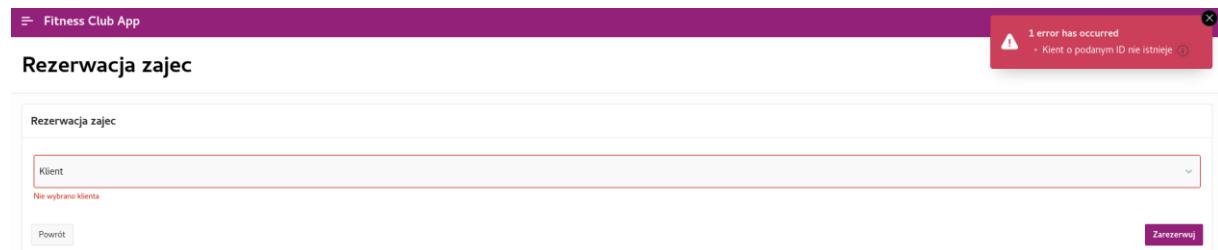
- Sprawdzenie, czy parametr został podany



Strona do dodawania rezerwacji na zajęcia

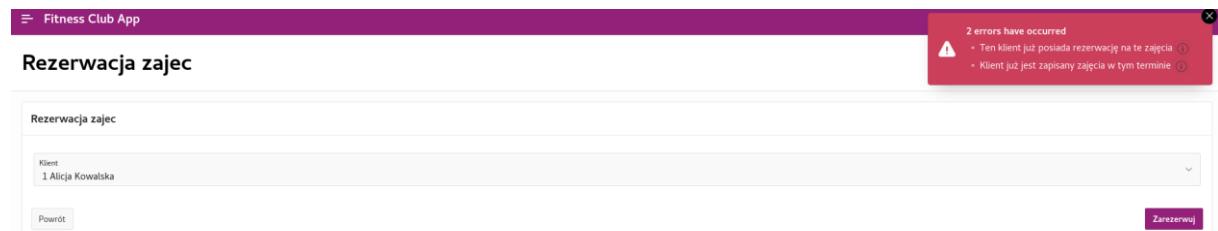
Strona **Rezerwacja zajęć** umożliwia użytkownikowi dokonanie rezerwacji na wybrane zajęcia poprzez wybór klienta z listy.

Po naciśnięciu przycisku „Zarezerwuj” wykonywane są walidacje sprawdzające, czy klient istnieje, czy już posiada aktywną rezerwację na te zajęcia, czy nie jest zapisany na inne zajęcia w tym samym terminie oraz czy nie znajduje się na liście oczekujących.

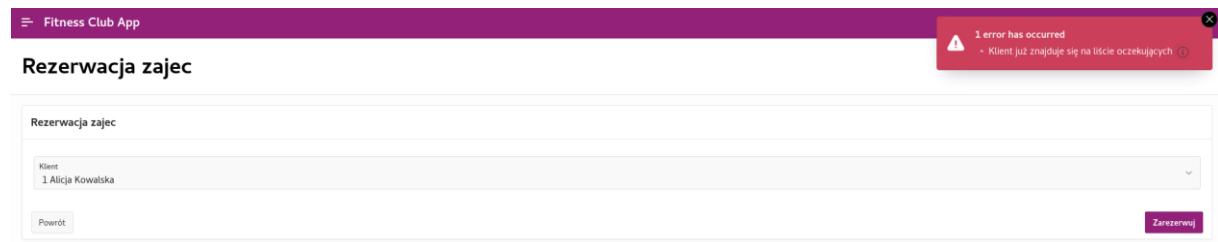


The screenshot shows a user interface for booking a class. At the top, there's a navigation bar with the app's logo and name. Below it is a section titled "Rezerwacja zajec". A dropdown menu labeled "Klient" is open, showing a single option: "1 Alicja Kowalska". There's also a "Powrot" (Return) button. On the right side of the screen, there's a prominent purple "Zarezerwuj" (Reserve) button. In the top right corner of the main area, a red warning box displays an error message: "1 error has occurred" followed by a bullet point: "Klient o podanym ID nie istnieje".

Jeśli walidacje zakończą się pomyślnie, wywoływana jest procedura **KLIENT_PAKIET.dodajKlientaDoZajec**, która zapisuje klienta do grupy zajęciowej. W przypadku błędów system wyświetla stosowne komunikaty o problemach, takich jak brak klienta w bazie lub podwójna rezerwacja.

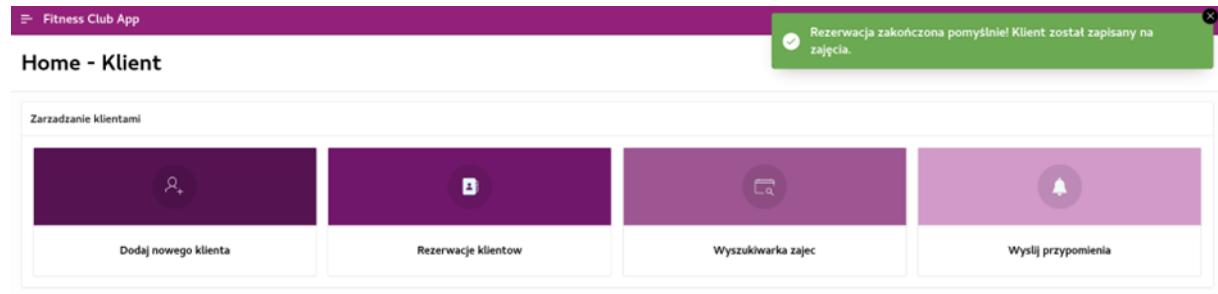


This screenshot shows the same reservation form as the previous one, but with a different selection in the "Klient" dropdown: "1 Alicja Kowalska". The "Zarezerwuj" button is still present. A red error message box at the top right indicates two errors: "Ten klient już posiada rezerwację na te zajęcia" and "Klient już jest zapisany zajęcia w tym terminie".



This screenshot shows the same reservation form again, with the "Klient" dropdown set to "1 Alicja Kowalska". The error message now states: "Klient już znajduje się na liście oczekujących".

Po pomyślnym zapisaniu użytkownik zostaje przekierowany na stronę główną klienta z komunikatem o zakończonej rezerwacji.



The screenshot shows the "Home - Klient" page. At the top, there's a green success message: "Rezerwacja zakończona pomyślnie! Klient został zapisany na zajęcia.". Below this, there are four cards with icons and labels: "Dodaj nowego klienta" (client icon), "Rezerwacje klientów" (calendar icon), "Wyszukiwarka zajęć" (magnifying glass icon), and "Wyślij przypomnienia" (bell icon).

The screenshot displays the Oracle Application Express (APEX) page builder interface. It shows two pages side-by-side:

- Page 1: Rezerwacja zajęć**
 - Regions:** P4_ID_Klienta, P4_ID_GRUPY, P4_KOMUNIKAT.
 - Validations:** Czy klient istnieje, Czy grupa istnieje, Aktywna rezerwacja, Lista oczekujących, Zły termin.
 - Buttons:** Close, Powrót, Change, SUBMIT.
- Page 2: Proces rezervacji**
 - Regions:** After Submit, Validating, Processing, Processes, Proces rezervacji, After Processing, Branches, Go To Page 2, Ajax Callback.
 - Validations:** Not NULL, Not NULL 2, Czy klient istnieje, Czy grupa istnieje, Aktywna rezerwacja, Lista oczekujących, Zły termin.
 - Process:**
 - Name:** Proces rezervacji
 - Type:** Execute Code
 - Execution Chain:** None
 - Editable Region:** - Select -
 - Source:**
 - Location:** Local Database
 - Language:** PL/SQL
 - PL/SQL Code:**

```
DECLARE
    v_wiadomosc VARCHAR2(4000);
BEGIN
    -- Wywołanie procedury z parametrem
    OUT
    Klient_Pakiet.GodzinyZajecieZwrocenie(
        p_id_klienta => :P4_ID_Klienta,
        p_id_grupy => :P4_ID_GRUPY,
        p_wiadomosc => v_wiadomosc
    );
END;
```

Przykłady zastosowania walidacji:

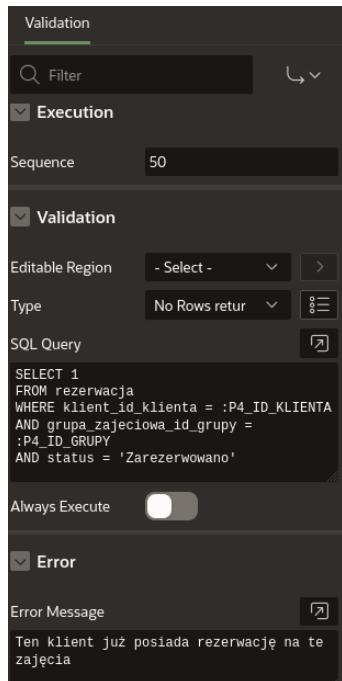
- Sprawdzenie, czy istnieje klient o podanym ID

The screenshot shows the validation configuration dialog for the 'Czy klient istnieje' rule:

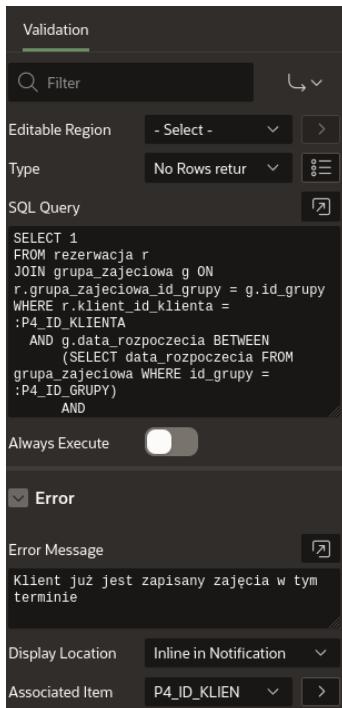
- Execution:** Sequence 30
- Validation:**
 - Editable Region:** - Select -
 - Type:** Rows returned
 - SQL Query:**

```
SELECT 1 FROM klient WHERE id_klienta = :P4_ID_Klienta
```
- Error:** Error Message: Klient o podanym ID nie istnieje

- Sprawdzenie, czy podany klient posiada już rezerwację na dane zajęcia



- Sprawdzenie, czy dany klient posiada już aktywną rezerwację na zajęcia w danym terminie



Strona do wysyłania przypomnień

Strona **Wysyłanie przypomnień** w aplikacji Fitness Club App umożliwia użytkownikowi wysyłanie automatycznych powiadomień do klientów o nadchodzących zajęciach oraz wygasających karnetach. Interfejs zawiera sekcję informacyjną oraz przycisk „Wyślij”, który inicjuje proces wysyłki.

Wysyłanie przypomnien

Informacja

Kliknij przycisk poniżej, aby wysłać przypomnienia do klientów:
 - Przypomnienia o jutrzajnych zajęciach dla osób z aktywną rezerwacją.
 - Przypomnienia o wygasających karnetach (7 dni przed końcem ważności).

Powrót Wyslij

W procesie weryfikowana jest obecność klientów spełniających kryteria – jeśli brak klientów z aktywnymi rezerwacjami lub karnetami bliskimi wygaśnięcia, wyświetlany jest komunikat błędu. Wykorzystano validację SQL do sprawdzenia, czy istnieją klienci z rezerwacjami na następny dzień lub z karnetami wygasającymi w ciągu 7 dni.

Wysyłanie przypomnien

Informacja

Kliknij przycisk poniżej, aby wysłać przypomnienia do klientów:
 - Przypomnienia o jutrzajnych zajęciach dla osób z aktywną rezerwacją.
 - Przypomnienia o wygasających karnetach (7 dni przed końcem ważności).

Wróć Wyslij

Proces wysyłki jest realizowany przez procedurę **PAKET_ADMINISTRATOR.WyslijPrzypomnienie**, która pobiera odpowiednie dane i generuje przypomnienia.

The screenshot shows two panels of the Oracle Application Designer:

- Page Designer:** Shows the structure of the 'Wyslij przypomnienia' page with regions like BANNER, AFTER LOGO, TOP NAVIGATION, and BODY. The BODY region contains a Breadcrumb component.
- Process Designer:** Shows the process flow for the page. It includes validation rules (e.g., 'Czy są klienci') and a process step named 'Wyslij przypomnienia'. This step is configured as an Execute Code process with the following PL/SQL code:

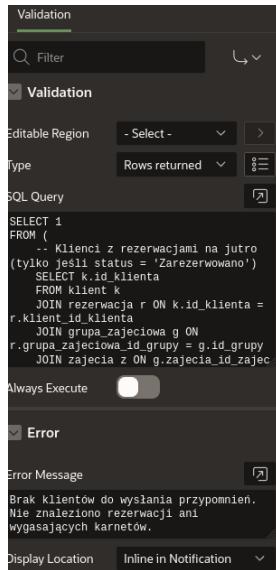
```

BEGIN
    WyslijPrzypomnienie;
END;

```

Przykład walidacji:

- Sprawdzenie, czy są klienci, którzy spełniają kryteria do otrzymania powiadomienia z przypomnieniem



Strona główna – Home Klub (Zarządzanie klubem)

Strona **Home - Klub** w aplikacji APEX służy do zarządzania funkcjami związanymi z klubem. Główna sekcja zawiera kafelki nawigacyjne umożliwiające szybki dostęp do kluczowych modułów: dodawania grupy zajęciowej, harmonogramu zajęć, wypłat instruktorów, efektywności instruktorów oraz analizy najpopularniejszych zajęć.

Strona korzysta z regionu "Zarządzanie klubem", który jest listą nawigacyjną kierującą użytkownika do odpowiednich podstron aplikacji. Struktura wizualna jest zgodna z resztą aplikacji, bazując na układzie kart z ikonami i etykietami ułatwiającymi intuicyjną nawigację.

Strona do tworzenia nowej grupy zajęciowej

Strona **Dodawanie grupy zajęciowej** umożliwia administratorowi klubu tworzenie nowych grup zajęciowych poprzez wprowadzenie niezbędnych parametrów, takich jak rodzaj zajęć, data rozpoczęcia, czas trwania, liczba miejsc i poziom zaawansowania.

Fitness Club App

Dodawanie grupy zajęciowej

Wprowadź parametry

- Rodzaj zajęć
- Data Rozpoczęcia
- Czas Trwania
- Liczba Miejsc
- Poziom Zaawansowania

[Powrót](#) [Dodaj](#)

Weryfikacje zapewniają poprawność wprowadzonych danych, sprawdzając, czy podane wartości są niepuste, czy liczba miejsc jest liczbą oraz czy wybrano odpowiednią salę i instruktora.

Fitness Club App

Dodawanie grupy zajęciowej

Wprowadź parametry

- Rodzaj zajęć Aerobik
- Data Rozpoczęcia 2025-01-29 12:00
- Czas Trwania 50
- Liczba Miejsc 15
- Poziom Zaawansowania Dla wszystkich

[Powrót](#) [Dodaj](#)

2 errors have occurred

- Brak dostępnych sal dla nowej grupy zajęciowej ⓘ
- Brak dostępnych instruktorów dla nowej grupy zajęciowej ⓘ

Dodawanie grupy zajęciowej

Wprowadź parametry

- Rodzaj zajęć Należy wybrać rodzaj zajęć
- Data Rozpoczęcia 2025-01-29 12:00
- Czas Trwania
- Liczba Miejsc 600 Uczać miejsc musi być liczba max. 2-cyfrowa
- Poziom Zaawansowania

[Powrót](#) [Dodaj](#)

Fitness Club App

Dodawanie grupy zajęciowej

Wprowadź parametry

- Rodzaj zajęć Gimnastyka
- Data Rozpoczęcia 2025-02-01 20:45
- Czas Trwania 50
- Liczba Miejsc 8
- Poziom Zaawansowania Zaawansowany

[Powrót](#) [Dodaj](#)

Proces dodawania grupy zajęciowej opiera się na procedurze **ADMINISTRATOR_PAKIET.dodaj_grupe_zajeciova**, która zapisuje nową grupę w bazie danych. Dodatkowe walidacje wykorzystują funkcje **ADMINISTRATOR_PAKIET.pobierz_dostepne_sale** i **ADMINISTRATOR_PAKIET.pobierz_dostepnych_instruktorow**, sprawdzając dostępność zasobów dla nowej grupy. Jeśli nie ma dostępnej sali lub instruktora, wyświetlany jest stosowny komunikat błędu. Po pomyślnym dodaniu grupy użytkownik zostaje przekierowany na stronę podsumowania.

The screenshot shows two windows from Oracle Application Designer:

- Page Designer:** Displays the "Dodawanie grupy zajęciowej" page template. It includes sections for BANNER, AFTER LOGO, TOP NAVIGATION, and a BREADCRUMB BAR labeled "Breadcrumb". The page has buttons for PREVIOUS, CLOSE, DELETE, HELP, CHANGE, COPY, and CREATE. A "NEXT" button is visible at the bottom. The right panel shows the "Identification" section with the name "Dodawanie grupy zajęciowej", alias "dodawanie-grupy-zajeciove", and title "Dodawanie grupy zajęciowej".
- Application Designer:** Shows the "Run Stored Procedure" process. The "Source" tab displays PL/SQL code:

```

#OMEWAH "#WYSZUKAJEPEZDOSTEPNA"
: "P_ID_ZAJEC" => :P9_ID_ZAJEC,
: "P_DATA_ROZPOCZECIA" => :P9_DATA_ROZPOCZECIA,
TO_DATE(:P9_DATA_ROZPOCZECIA, 'YYYY-MM-
DD HH24:MI'),
: "P_CZAS_TRWANIA",
: "P_LICZBA_MIEJSC" =>
:P9_LICZBA_MIEJSC,
: "P_POTOM_ZAMANOWANIA" =>
:P9_POTOM_ZAMANOWANIA

```

Przykłady walidacji:

- Sprawdzenie, czy istnieją dostępne sale dla podanych parametrów

The screenshot shows the validation configuration window:

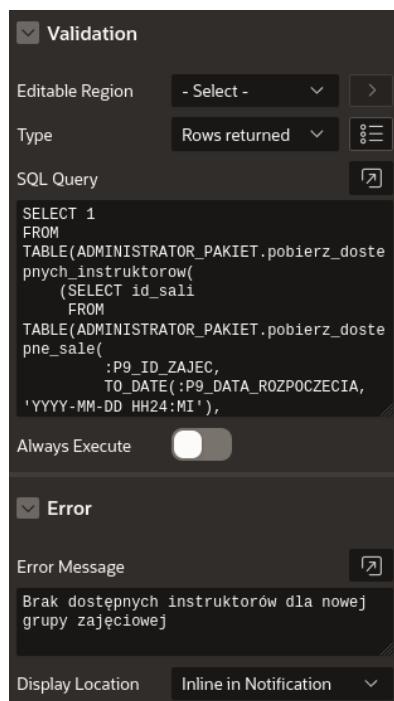
- Validation:** Set to "Rows returned". The SQL Query is:

```

SELECT 1
FROM
TABLE(ADMINISTRATOR_PAKIET.pobierz_dostepne_sale(
:P9_ID_ZAJEC,
TO_DATE(:P9_DATA_ROZPOCZECIA, 'YYYY-MM-
DD HH24:MI'),
:P9_CZAS_TRWANIA,
:P9_LICZBA_MIEJSC
))

```
- Error:** The error message is "Brak dostępnych sal dla nowej grupy zajęciowej".

- Sprawdzenie, czy istnieją dostępni instruktorzy dla podanych parametrów

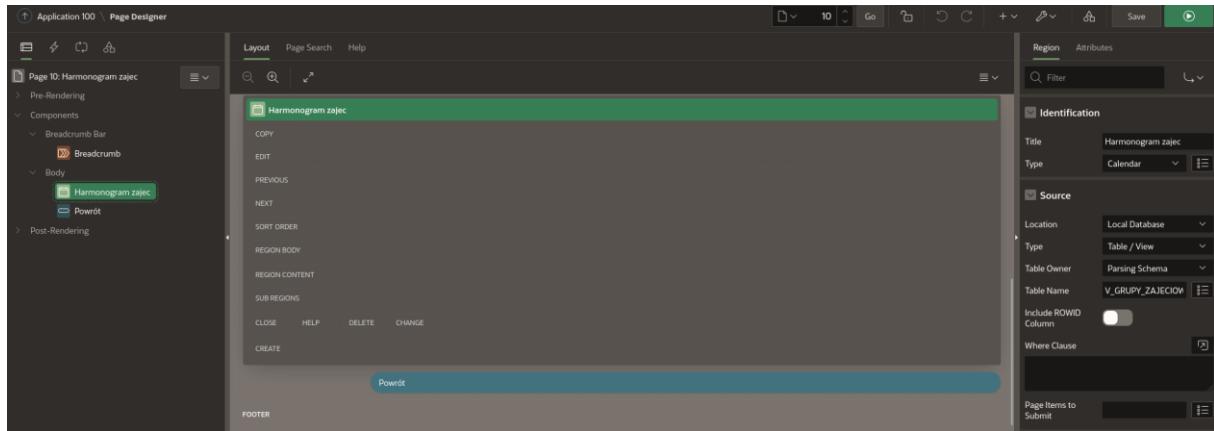


Strona z harmonogramem zajęć

Strona **Harmonogram zajęć** prezentuje kalendarz zajęć w formie interaktywnego widoku, umożliwiającego przeglądanie planu w ujęciu miesięcznym, tygodniowym i dziennym. Po dodaniu nowej grupy zajęciowej harmonogram jest aktualizowany, co potwierdza komunikat o pomyślnym utworzeniu grupy.

The screenshot shows the Fitness Club App interface. At the top, there's a purple header bar with the app name and a green notification bubble saying 'Grupa zajeciowa została utworzona!'. Below the header, the title 'Harmonogram zajec' is displayed. The main area is titled 'Harmonogram zajec' and shows a monthly calendar for February 2025. The days of the week are labeled: pon., wt., śr., czw., pt., sob., niedz. The dates 27, 28, 29, 30, 31, 1, 2 are shown below the days. Blue horizontal bars represent scheduled classes for each day. A legend at the bottom right indicates: Miesiąc, Tydzień, Dzień, Plan dnia. The legend buttons are: Miesiąc (selected), Tydzień, Dzień, Plan dnia.

Widok jest oparty na komponentie kalendarza w APEX, a źródłem danych jest widok „V_GRUPY_ZAJECIOWE”, który przechowuje informacje o zaplanowanych zajęciach. System dynamicznie pobiera dane, zapewniając bieżący wgląd w dostępne zajęcia.



Strona generująca raport wypłat instruktorów

Na tej stronie użytkownik może wygenerować raport wypłat instruktorów na podstawie wybranego zakresu dat. Formularz zawiera dwa wymagane pola: Data Początkowa i Data Końcowa, które użytkownik musi wypełnić, aby przejść dalej.

This screenshot shows the 'Raport wyplat instruktorow' form. It has two input fields: 'Data Początkowa' (2025-01-01) and 'Data Koncowa' (2024-12-31). Below the fields are 'Powrót' and 'Generuj Raport' buttons. A red error message box is visible, stating: '3 errors have occurred' with three items: 'Data początkowa nie może być późniejsza niż data końcowa', 'Brak instruktorów prowadzących zajęcia w podanym okresie', and 'Brak danych do wygenerowania raportu wypłat'.

Jeśli daty są niepoprawne, np. data początkowa jest późniejsza niż końcowa, użytkownik otrzyma odpowiednie komunikaty błędów. Po poprawnym wprowadzeniu danych aplikacja sprawdza, czy istnieją instruktorzy prowadzący zajęcia w podanym okresie oraz czy są dostępne dane do wygenerowania raportu.

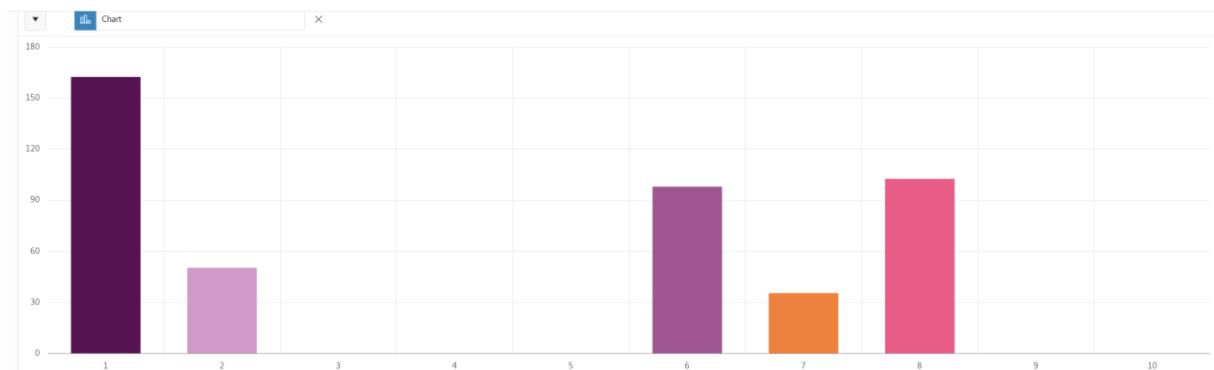
This screenshot shows the same form after entering valid dates. A green success message box appears: 'Raport został wygenerowany pomyslnie!'. The table below shows the generated report data.

Jeśli warunki są spełnione, aplikacja wyświetla raport w formie tabeli z listą instruktorów oraz kwotami ich wynagrodzeń, a także wizualizację w postaci wykresu słupkowego.

This screenshot shows the generated report table. The columns are 'Id Instruktora', 'Imię', 'Nazwisko', and 'Kwota'. The data is as follows:

Id Instruktora	Imię	Nazwisko	Kwota
1	Anna	Kowalska	162,50
2	Jan	Nowak	50,50
3	Magdalena	Kaczor	0,00
4	Piotr	Lis	0,00
5	Anna	Koziara	0,00
6	Krzysztof	Zajac	98,08
7	Tomasz	Kowalski	35,63

Raport wyplat instruktorow



Zapytanie SQL wykorzystuje funkcję **ADMINISTRATOR_PAKIET.oblicz_pensje_instruktora** do pobrania danych o wynagrodzeniach instruktorów w zadanym przedziale czasowym oraz procedurę **ADMINISTRATOR_PAKIET.GenerujRaportWyplat** do wygenerowania raportu zawierającego odpowiednie dane.

Przykład walidacji:

- Sprawdzenie, czy w danym okresie odbywały się zajęcia

```

Validation
Editable Region - Select -
Type Rows returned
SQL Query
SELECT 1
FROM grupa_zajeciodawa g
JOIN instruktor i ON
g.instruktor_id_instruktora =
i.id_instruktora
WHERE g.data_rozpoczecia BETWEEN
:P12_DATA_POCZATKOWA AND
:P12_DATA_KONCOWA;

Always Execute [ ] Error
Error Message
Brak danych do wygenerowania raportu wyplat
Display Location Inline in Notification

```

Strona generująca raport efektywności instruktorów

Na stronie **Raport efektywności instruktorów** użytkownik może wygenerować raport dotyczący zajętości i efektywności pracy instruktorów w wybranym zakresie dat. Formularz wymaga podania daty początkowej i końcowej, co jest walidowane pod kątem poprawności (np. brak pustych wartości).

The screenshot shows a web application interface for generating a report. At the top, there's a purple header bar with the text "Fitness Club App". Below it, the main title is "Raport efektywnosci instruktorow". There are two input fields: "Data Początkowa" (initial date) and "Data Koncowa" (final date). Both fields have validation messages: "Należy podać datę początkową" and "Należy podać datę końcową". Below these fields are two buttons: "Powrót" (back) and "Generuj Raport" (generate report). A small status message "Raport został wygenerowany!" is visible in the top right corner.

Po wygenerowaniu raportu użytkownik otrzymuje listę instruktorów, liczbę dostępnych miejsc na ich zajęciach, liczbę dokonanych rezerwacji oraz procent obłożenia zajęć.

The screenshot displays the generated report as a table. The columns are labeled: "Id Instruktora", "Imię", "Nazwisko", "Liczba Miejsc", "Liczba Rezerwacji", and "Procent Obłożenia". The data is as follows:

Id Instruktora	Imię	Nazwisko	Liczba Miejsc	Liczba Rezerwacji	Procent Obłożenia
1	Anna	Kowalska	90	5	5,56
2	Jan	Nowak	64	8	12,5
6	Krzysztof	Zająć	34	2	5,88
7	Tomasz	Kowalski	5	1	20
8	Karolina	Wojcik	42	0	0

W przypadku błędnych danych lub braku instruktorów w danym okresie system wyświetla stosowny komunikat o błędzie.

The screenshot shows the same report page but with an error message. A red alert box at the top right says "1 error has occurred" with the message "Brak instruktorów prowadzących zajęcia w wybranym okresie". The rest of the interface is identical to the previous screenshot.

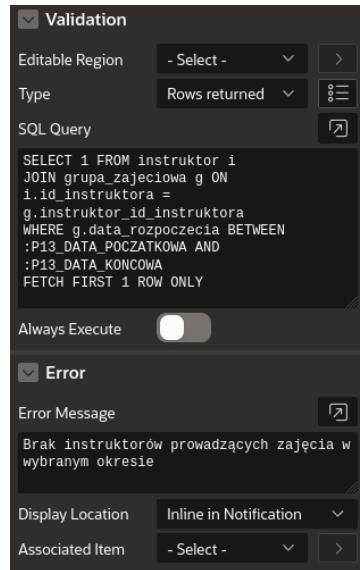
Dane są pobierane z bazy za pomocą zapytania SQL z wykorzystaniem procedury **ADMINISTRATOR_PAKIET.AnalizujEfektywnoscInstruktora**, a wyniki są prezentowane w formie interaktywnego raportu.

This screenshot shows the Oracle Apex Report Builder interface. On the left, the "Components" tree view includes sections for "Pre-Rendering", "Components", "Breadcrumb Bar", "Body", "Validations", "GENERUJ_RAPORT", and "Report efektywnosci instruktorow". The "Body" section contains regions "P13_DATA_POCZATKOWA" and "P13_DATA_KONCOWA". The "Report efektywnosci instruktorow" section has a "GENERUJ_RAPORT" button. On the right, the "Source" panel shows the SQL query used to generate the report:

```
SELECT
    C001 AS id_instruktora,
    C002 AS imie,
    C003 AS nazwisko,
    C004 AS liczba_miejsc,
    C005 AS liczba_rezerwacji,
    C006 AS procent_oblozenia
FROM APEX_COLLECTIONS
WHERE COLLECTION_NAME =
    'EFEKTYWNOSC_INSTRUKTOROW';
```

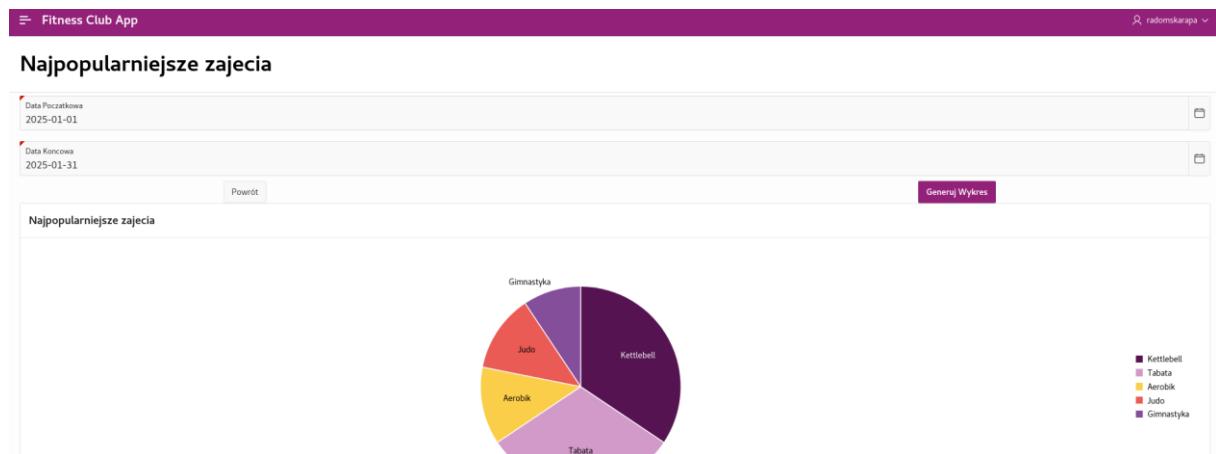
Przykład walidacji:

- Sprawdzenie, czy istnieją instruktorzy prowadzący zajęcia w danym okresie



Strona generująca wykres najpopularniejszych zajęć

Strona **Najpopularniejsze zajęcia** pozwala na wygenerowanie wykresu najczęściej wybieranych zajęć w zadanym przedziale czasowym. Użytkownik podaje daty i uruchamia raport, który pobiera dane z funkcji **ADMINISTRATOR_PAKIET.najpopularniejsze_zajecia**.



Walidacje sprawdzają poprawność wprowadzonych dat i dostępność danych. W przypadku błędów wyświetlane są komunikaty, a jeśli raport się powiedzie, generowany jest wykres kołowy z podziałem na liczbę rezerwacji dla poszczególnych zajęć.

Fitness Club App

Najpopularniejsze zajęcia

Data Początkowa
2025-01-01

Data Koncowa
2024-12-11

Powrót Generuj Wykres

Najpopularniejsze zajęcia

Brak danych do wyświetlenia

2 errors have occurred

- Data początkowa nie może być późniejsza niż data koncowa.
- Brak zajęć w podanym okresie.

Page 19: Najpopularniejsze zajęcia

Layout Page Search Help

Na popularniejsze zajęcia

COPY EDIT PREVIOUS NEXT SORT ORDER REGION BODY REGION CONTENT SUB REGIONS CLOSE HELP DELETE CHANGE CREATE FOOTER

DIALOGS DRAWERS AND POPUPS

Identification Name Series 1 Execution Sequence 10 Source Location Local Database Type SQL Query

```
SELECT nazwa_zajec, ilosc_rezerwacji
FROM (SELECT * FROM PAKIET.najpopularniejsze_zajecia
      WHERE data_rezerwacji BETWEEN :P19_DATA_POCZATKOWA AND :P19_DATA_KONCOWA
      ORDER BY ilosc_rezerwacji DESC
      LIMIT 10)
) WHERE ilosc_rezerwacji > 0;
```

After Submit

Validating

Not NULL Not NULL 2 Czy poprawne data Czy zajęcia

Processing

Ajax Callback

Layout Page Search Help

Na popularniejsze zajęcia

COPY EDIT PREVIOUS NEXT SORT ORDER REGION BODY REGION CONTENT SUB REGIONS CLOSE HELP DELETE CHANGE CREATE FOOTER

DIALOGS DRAWERS AND POPUPS

Validation Identification Name Czy zajęcia Execution Sequence 40 Validation Editable Region -Select- Type Rows returned

```
SELECT 1 FROM zajecia z
JOIN grupa_zajeciodowa g ON z.id_zajec = g.zajecia_id_zajec
WHERE g.data_rozpoczecia BETWEEN
:P19_DATA_POCZATKOWA AND
:P19_DATA_KONCOWA
FETCH FIRST 1 ROW ONLY
```

Always Execute

Przykład walidacji:

- Sprawdzenie, czy istnieją instruktorzy prowadzący zajęcia w danym okresie

Validation

Editable Region - Select - Type Rows returned SQL Query

```
SELECT 1 FROM zajecia z
JOIN grupa_zajeciodowa g ON z.id_zajec = g.zajecia_id_zajec
WHERE g.data_rozpoczecia BETWEEN
:P19_DATA_POCZATKOWA AND
:P19_DATA_KONCOWA
FETCH FIRST 1 ROW ONLY
```

Always Execute

Error Error Message Brak zajęć w podanym okresie

Display Location Inline in Notification Associated Item - Select -

Eksport i import aplikacji

Podjęta została próba eksportu aplikacji na nowy workspace wraz obiektami bazy danych.

Eksport aplikacji

W pierwszym kroku wykonano eksport aplikacji Fitness Club App z Oracle APEX. Wybrano odpowiednie opcje eksportu, m.in. eksport definicji obiektów wspierających, co umożliwia automatyczne odtworzenie struktury bazy danych podczas importu. Po zakończeniu procesu pobrano plik f100(8).sql, zawierający skrypt SQL do odtworzenia aplikacji w innym środowisku.

The screenshot shows the Oracle APEX application export interface. At the top, there is a table listing the objects being exported:

Lock	Name	Sequence	Script	Status	Updated	Updated By	Source
	Tabele	10	CREATE TABLE "ADRES" ("ID_ADRESU" NUMBER(*,0) NOT NULL ENABLE, "ULICA" VARCHAR2(50 CHAR) COLLATE "USING_NLS_COMP" NOT NULL ENABLE, "NR_DOMU" VARCHAR2(10 CHAR) COLLATE "USING_NLS_COMP" NO...)	9 minutes ago	radomskarapa	Database Object	
	Widok	20	CREATE OR REPLACE FORCE EDITABLE VIEW "V_GRUPY_ZAJECIOWE_NAZWY" ("ID_GRUPY", "DATA_ROZPOCZECIA", "CZAS_TRWANIA_W_MINUTACH", "LICZBA_MIEJSC", "POZIOM_ZAWANGOWANIA", "NAZWA_ZAJEC") DEFAULT COLLAT...	9 minutes ago	radomskarapa	Database Object	
	Typy	30	create or replace TYPE DostepneSaleTab AS TABLE OF TypSala; / create or replace TYPE EfektywnoscInstruktorowTab AS TABLE OF TypEfektywnoscInstruktora; / create or replace TYPE GrupaInfo AS OBJECT...	8 minutes ago	radomskarapa	Database Object	
	Pakietы	40	create or replace PACKAGE ADMINISTRATOR_PAKIET AS -- Deklaracje funkcji FUNCTION pobierz_dostepne_sale(p_id_zajec IN INTEGER, p_data_rozpoczenia IN DATE, ...	68 seconds ago	radomskarapa	Database Object	
	Data	50	begin --STREFA: 5/10000 rows exported. APEX\$DATA\$PKG/STREFA\$391022 apex_data_install.load_supporting_object_data(p_table_name => 'STREFA', p_delete_after_install => true);--ZAJECIA:...	Now	radomskarapa	Manual	

Below the table, the export configuration is shown:

- * Application: 100 Fitness Club App
- Selected Application: Fitness Club App
- Page Count: 18
- Owner: KLUB_FITNESS
- Split into Multiple Files: Enabled
- Owner Override: [empty]
- Build Status Override: Run and Build Application
- Debugging: Enabled
- As of: [empty] minutes ago (~ 5 min delay)
- File Character Set: Unicode UTF-8

The "Export Preferences" section includes the option "Export Supporting Object Definitions" set to "Yes and Install on Import Automatically".

The bottom of the interface shows the file "f100(8).sql" has been completed at 1.1 MB.

Tworzenie nowego Workspace

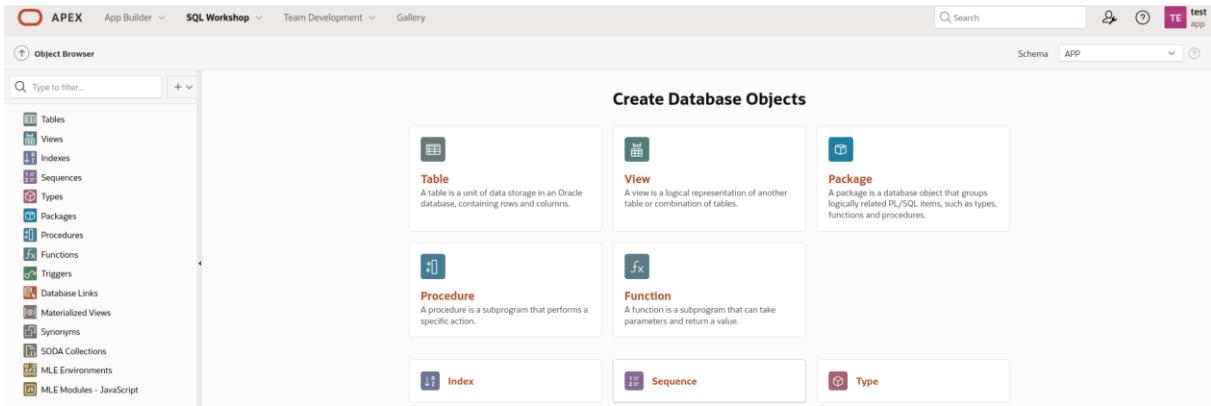
Aby przygotować środowisko do importu, utworzono nowy Workspace o nazwie App. Następnie zalogowano się na konto użytkownika - TEST, które będzie wykorzystywane do zarządzania aplikacją.

The screenshot illustrates the workflow for creating a new workspace and logging in to Oracle APEX.

Create Workspace: This step shows the "Identify Workspace" screen. It includes fields for "Workspace Name" (set to "App"), "Workspace ID" (empty), and "Workspace Description" (empty). A progress bar at the top indicates the first step is completed.

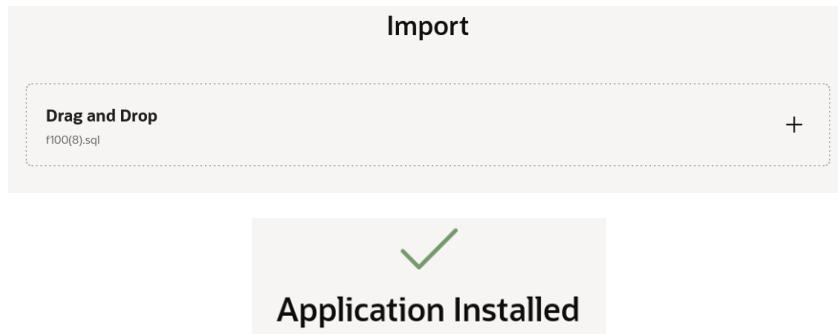
Oracle APEX: This step shows the sign-in screen. It displays the workspace name "App" and user "test" both with green checkmarks. Below these, a password field contains a masked password ("••••") with a green checkmark. A checkbox for "Remember Workspace and Username" is unchecked. A large green "Sign In" button is prominent at the bottom.

Oracle APEX Home Page: This is the final view after signing in. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user profile "test app" are also present. The main content area features four buttons: "Create", "Import", "Dashboard", and "Workspace Utilities". A "Get Started Now" section offers options to "Create a New App" or "Install a Starter or Sample App". On the right side, there are sections for "About", "Recent" (empty), and "Tasks" (links to "Manage Backups" and "Browse by Facets").



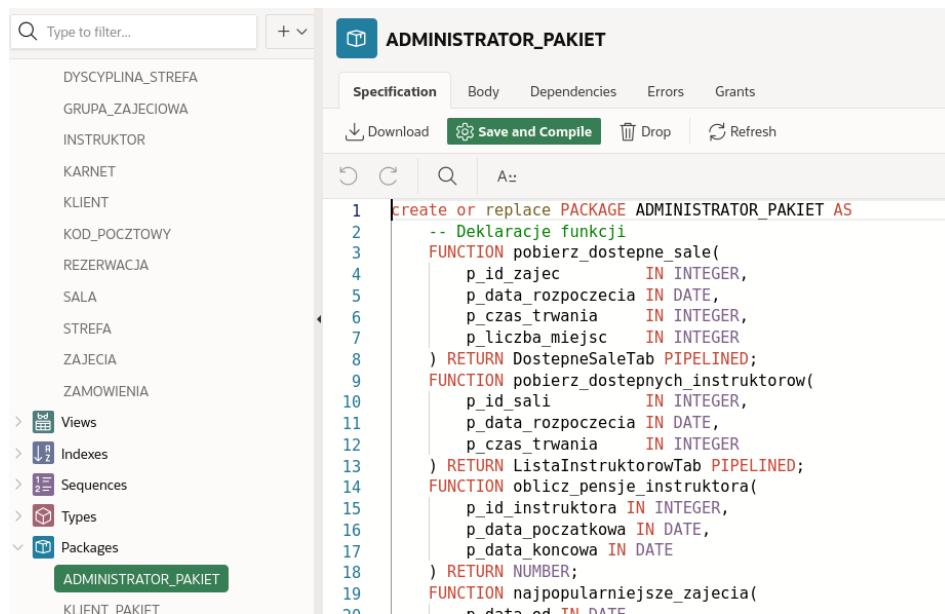
Import aplikacji

Po zalogowaniu przeprowadzono proces importu aplikacji do nowego środowiska. W tym celu przesłano wcześniej pobrany plik SQL do sekcji Import w SQL Workshop. Po załadowaniu pliku wykonano skrypt, który utworzył w bazie wszystkie niezbędne obiekty i struktury aplikacji.



Weryfikacja obiektów bazy danych

Po zakończeniu importu sprawdzono poprawność odtworzonych obiektów bazy danych w SQL Workshop - Object Browser. Zweryfikowano istnienie kluczowych tabel, widoków, sekwencji oraz pakietów PL/SQL. Zauważono brak potrzebnych sekwencji oraz danych w tabelach, przez co uzupełniono bazę o brakujące obiekty ręcznie z wykorzystaniem dodatkowych skryptów.



Uruchomienie aplikacji

Na końcu sprawdzono, czy aplikacja została poprawnie zainstalowana. Na liście aplikacji w nowym Workspace pojawiła się **Fitness Club App**, co potwierdziło poprawność procesu importu. Aplikację uruchomiono w celu zweryfikowania jej działania. Zalogowano się na konto użytkownika TEST oraz sprawdzono działanie funkcjonalności.

The screenshot displays the 'Fitness Club App' interface. At the top, there is a navigation bar with a search icon, a 'Go' button, and other action icons. Below this is a list of applications, where 'Fitness Club App' is highlighted with a pink background and the number '103' next to it. The main area shows a login screen for 'TEST' user, featuring a placeholder for a password, a 'Remember username' checkbox, and a large purple 'Sign In' button. Below the login screen is a dark header bar with the application name 'Fitness Club App' and a user dropdown. The main content area is divided into two sections: 'Home - Klient' (Client Home) and 'Home - Klub' (Club Home), both featuring a search icon. At the bottom is a search results table titled 'Wyszukiwarka zajęć' (Search for activities). The table includes columns for Id Grupy, Data Rozpoczęcia, Czas Trwania, Nazwa Zajęć, Nazwa Dyscypliny, Imię, Nazwisko, Poziom Zaawansowania, and Rezerwacja. The data is as follows:

Id Grupy	Data Rozpoczęcia	Czas Trwania	Nazwa Zajęć	Nazwa Dyscypliny	Imię	Nazwisko	Poziom Zaawansowania	Rezerwacja
1	2024-01-10 10:00	60	Aerobik	Fitness	Anna	Kowalska	Poczatkujący	ZAREZERWUJ
2	2024-01-12 15:30	75	Zumba	Taniec	Jan	Nowak	Sredniozaawansowany	ZAREZERWUJ
3	2024-01-15 12:45	90	Gimnastyka	Gimnastyka	Magdalena	Kaczor	Zaawansowany	ZAREZERWUJ
4	2024-01-18 18:00	60	Joga	Stretching	Piotr	Lis	Zaawansowany	ZAREZERWUJ
5	2024-01-20 11:30	75	Boks	Sztuki walki	Anna	Kozyra	Poczatkujący	ZAREZERWUJ
6	2024-01-22 17:15	90	Judo	Sztuki walki	Krzysztof	Zajac	Zaawansowany	ZAREZERWUJ
7	2024-01-25 14:45	60	Zdrowy kregosłup	Stretching	Tomasz	Kowalski	Dla wszystkich	ZAREZERWUJ
8	2024-01-27 09:30	45	Tabata	Fitness	Karolina	Wojcik	Sredniozaawansowany	ZAREZERWUJ
9	2024-01-30 16:00	90	Kettlebell	Trening siłowy	Jakub	Szymanik	Zaawansowany	ZAREZERWUJ

Podsumowanie projektu

Projekt aplikacji do zarządzania klubem fitness w Oracle APEX pozwolił na stworzenie systemu usprawniającego pracę administratorów i recepcjonistów. Dzięki wykorzystaniu bazy danych Oracle oraz PL/SQL, system umożliwia efektywne zarządzanie rezerwacjami, klientami, harmonogramami zajęć i raportami analitycznymi.

W ramach projektu zrealizowano:

- Implementację pakietów PL/SQL, które porządkują logikę biznesową i ułatwiają rozwój aplikacji,
- Stworzenie raportów analitycznych, pozwalających na monitorowanie popularności zajęć i efektywności instruktorów,
- Wdrożenie automatycznych powiadomień e-mail, informujących klientów o nadchodzących zajęciach,
- Możliwość eksportu i importu aplikacji, co ułatwia wdrożenie systemu w różnych środowiskach.

Wnioski

- Oracle APEX okazał się efektywnym narzędziem do budowy aplikacji bazodanowych, umożliwiającym szybkie wdrożenie i łatwe zarządzanie danymi.
- Struktura pakietów PL/SQL ułatwia organizację kodu oraz pozwala na jego elastyczną rozbudowę.
- Obsługa błędów i walidacja danych były kluczowe dla zapewnienia stabilności i poprawnego działania systemu.
- Eksport aplikacji wymaga szczególnej uwagi w zakresie poprawności obiektów bazy danych.
- Aplikacja może zostać rozszerzona o moduł obsługi karnetów i zamówień, umożliwiający automatyczne zarządzanie subskrypcjami klientów, monitorowanie ważności karnetów oraz generowanie powiadomień o konieczności ich odnowienia.