Sprawozdanie

Wydanie 1.0.0

Szymon Piskorz

03 lip 2025

Spis treści:

ROZDZIAŁ 1

1. Wprowadzenie

Autor: Szymon Piskorz Prowadzący: Piotr Czaja

Niniejsze sprawozdanie stanowi kompleksową dokumentację projektu bazodanowego, którego realizacja przyświecała dwóm głównym celom: dydaktycznemu oraz praktycznemu.

1.1 Cel dydaktyczny

Nadrzędnym celem było pogłębienie wiedzy i zdobycie praktycznych umiejętności z zakresu zaawansowanej administracji systemami baz danych. Proces ten obejmował badanie i aplikację następujących zagadnień:

- **Infrastruktura sprzętowa i konfiguracja:** Analiza wymagań sprzętowych oraz optymalna konfiguracja parametrów serwera bazy danych pod kątem wydajności i stabilności.
- **Kontrola, konserwacja i diagnostyka:** Poznanie narzędzi do monitorowania stanu bazy, diagnozowania wąskich gardeł oraz wdrażanie procedur konserwacyjnych, takich jak aktualizacja statystyk czy reindeksacja.
- **Wydajność, skalowanie i replikacja:** Techniki optymalizacji zapytań, strategie skalowania (pionowego i poziomego) oraz konfiguracja mechanizmów replikacji w celu zapewnienia wysokiej dostępności i rozłożenia obciążenia.
- **Partycjonowanie danych:** Zrozumienie i zastosowanie metod partycjonowania tabel w celu poprawy zarządzania dużymi zbiorami danych i zwiększenia wydajności zapytań.
- **Bezpieczeństwo:** Implementacja polityk bezpieczeństwa, zarządzanie użytkownikami i uprawnieniami, a także ochrona przed nieautoryzowanym dostępem.
- **Kopie zapasowe i odzyskiwanie danych:** Projektowanie i automatyzacja strategii tworzenia kopii zapasowych (pełnych, różnicowych, przyrostowych) oraz procedur odtwarzania danych po awarii.

1.2 Cel praktyczny i zakres projektu

Drugim celem była realizacja kompletnego projektu bazy danych o nazwie „Karnety na siłowni”. Projekt ten ilustruje pełen cykl życia produktu bazodanowego, od analizy wymagań, przez modelowanie, aż po wdrożenie i analizę.

Problem biznesowy: Współczesne siłownie i kluby fitness obsługują setki, a nawet tysiące klientów. Ręczne zarządzanie karnetami, datami ich ważności oraz rejestracja wejść są nieefektywne, podatne na błędy i uniemożliwiają prowadzenie analiz biznesowych. Projektowana baza danych ma na celu rozwiązanie tych problemów poprzez automatyzację kluczowych procesów.

Zakres projektu obejmuje: * Zarządzanie kartoteką klientów. * Obsługę sprzedaży i cyklu życia karnetów o różnym okresie ważności. * Rejestrację i monitorowanie wejść klientów. * Podstawową analitykę i raportowanie.

Poza zakresem projektu pozostają: * Zarządzanie grafikami zajęć i rezerwacjami. * Systemy księgowe i fakturowanie. * Zarządzanie personelem.

1.3 Wykorzystane technologie

- **System Bazy Danych:** PostgreSQL 16
- **Język skryptowy:** Python 3.11 (z biblioteką *psycopg2*)
- **System dokumentacji:** Sphinx
- **System kontroli wersji:** Git / GitHub

1.4 Struktura sprawozdania

Dokument został podzielony na pięć rozdziałów. Po niniejszym wprowadzeniu, rozdział drugi odnosi się do zrealizowanego projektu grupowego. Rozdział trzeci szczegółowo opisuje projekt bazy danych „Karnety na siłowni", prezentując jego modele i procesy. Rozdział czwarty skupia się na analizie normalizacji, wydajności i bezpieczeństwa. Całość zamyka rozdział piąty, zawierający podsumowanie, wnioski oraz listę wykorzystanych repozytoriów.

ROZDZIAŁ 2

2. Projekt grupowy

Równolegle do prac nad projektem indywidualnym, zrealizowano projekt grupowy, którego celem było praktyczne zbadanie i wdrożenie strategii tworzenia kopii zapasowych oraz odzyskiwania danych w systemie PostgreSQL.

2.1 Podsumowanie projektu grupowego

Projekt koncentrował się na zapewnieniu ciągłości działania hipotetycznego systemu bazodanowego. Główne zadania obejmowały:

- **Analizę wymagań biznesowych:** Zdefiniowano kluczowe parametry, takie jak RPO (Recovery Point Objective) i RTO (Recovery Time Objective), które determinowały częstotliwość i rodzaj wykonywanych kopii zapasowych.
- **Implementację skryptów:** Stworzono zautomatyzowane skrypty powłoki (bash) wykorzystujące natywne narzędzia PostgreSQL (*pg_dump*, *pg_restore*) do wykonywania pełnych i różnicowych kopii zapasowych.
- **Automatyzację:** Skrypty zostały zintegrowane z systemowym harmonogramem zadań (*cron*) w celu zapewnienia regularnego i bezobsługowego działania.
- **Testowanie procedur odtwarzania:** Przeprowadzono symulacje różnych scenariuszy awarii (usunięcie danych, uszkodzenie bazy), aby zweryfikować poprawność i skuteczność procedur odtwarzania danych z backupu.

W ramach projektu autor niniejszego sprawozdania był odpowiedzialny za przygotowanie skryptów automatyzujących proces tworzenia kopii zapasowych oraz za przeprowadzenie testów odtwarzania danych w scenariuszu „point-in-time recovery".

2.2 Repozytorium projektu

Pełna dokumentacja, kod źródłowy skryptów oraz wyniki testów projektu grupowego dostępne są w publicznym repozytorium na platformie GitHub:

- Repozytorium: Kopie zapasowe i odzyskiwanie danych

ROZDZIAŁ 3

3. Projekt bazy danych „Karnety na siłowni”

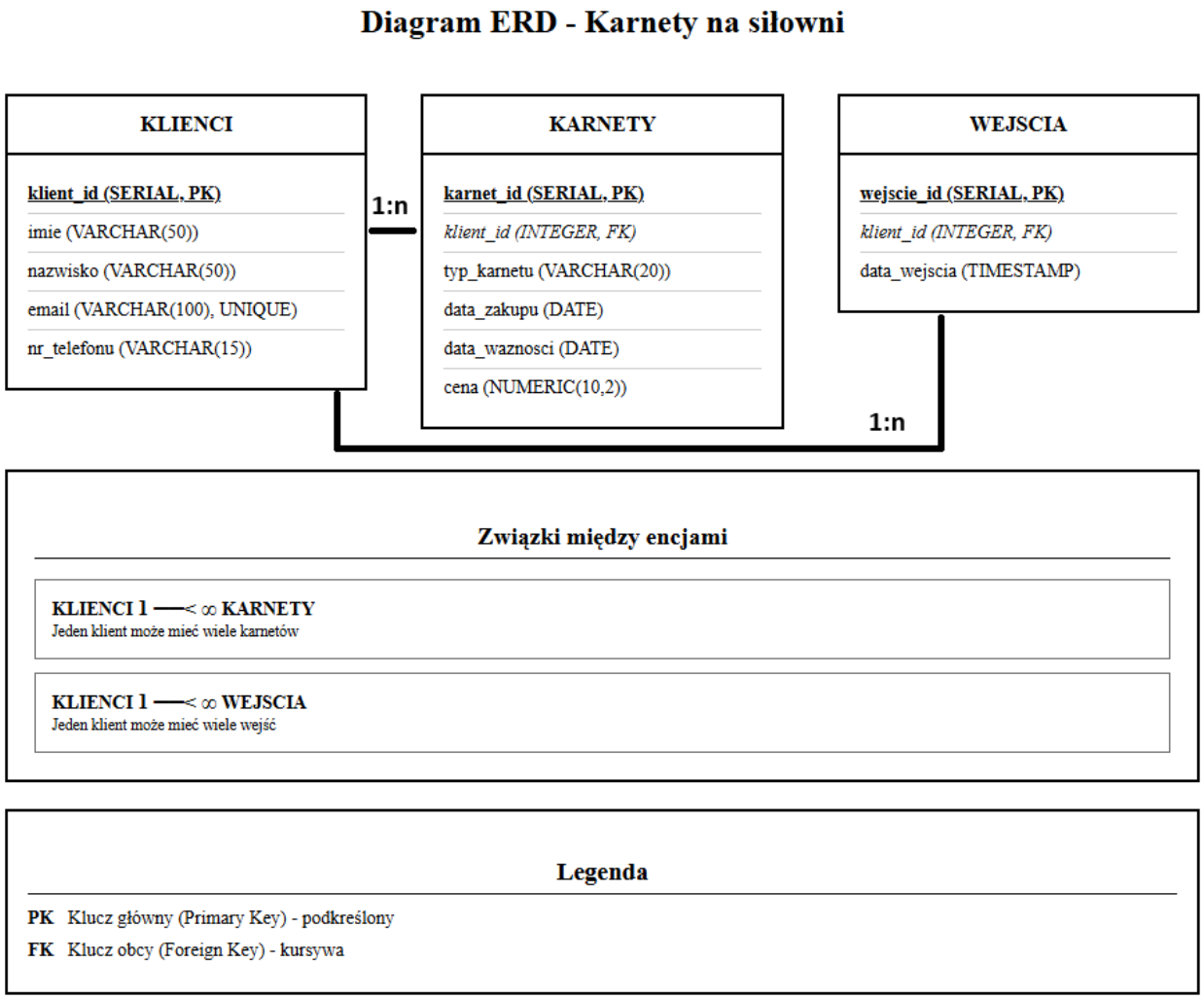
W tym rozdziale szczegółowo przedstawiono proces projektowania i implementacji bazy danych, od modelu koncepcyjnego po fizyczną realizację.

3.1 Opis procesów biznesowych

System bazodanowy został zaprojektowany w celu wsparcia trzech fundamentalnych procesów biznesowych siłowni:

1. **Rejestracja nowego klienta:** Proces polega na zebraniu podstawowych danych klienta (imię, nazwisko, dane kontaktowe) i zapisaniu ich w systemie. Każdy klient otrzymuje unikalny identyfikator. Proces ten jest realizowany poprzez operację *INSERT* na tabeli *Klienci*.
2. **Sprzedaż i aktywacja karnetu:** Klient może zakupić jeden z dostępnych karnetów. System rejestruje typ karnetu, datę zakupu, oblicza datę ważności i zapisuje cenę transakcji. Karnet jest jednoznacznie powiązany z klientem, który go zakupił. Proces ten obsługuje operacja *INSERT* na tabeli *Karnety*, z kluczem obcym wskazującym na klienta.
3. **Rejestracja wejścia na siłownię:** Przy każdej wizycie klienta, system weryfikuje, czy posiada on aktywny (ważny) karnet. Weryfikacja polega na wyszukaniu w tabeli *Karnety* rekordu powiązanego z danym klientem, którego *data_waznosci* jest późniejsza lub równa bieżącej dacie. Jeśli weryfikacja przebiegnie pomyślnie, system rejestruje wejście, zapisując identyfikator klienta i dokładny czas w tabeli *Wejscia*.

3.2 Model Koncepcyjny (ERD)



3.3 Model Logiczny

Model logiczny przekłada koncepcje na konkretną strukturę tabel, kolumn, typów danych i więzów integralności.

- **Tabela: Klienci**
  - *klient_id* (SERIAL, PK): Unikalny, automatycznie inkrementowany identyfikator klienta. Klucz główny.
  - *imie* (VARCHAR(50), NOT NULL): Imię klienta.
  - *nazwisko* (VARCHAR(50), NOT NULL): Nazwisko klienta.
  - *email* (VARCHAR(100), UNIQUE, NOT NULL): Adres e-mail, musi być unikalny, służy jako login lub do komunikacji.
  - *nr_telefonu* (VARCHAR(15)): Numer telefonu, opcjonalny.
- **Tabela: Karnety**
  - *karnet_id* (SERIAL, PK): Unikalny identyfikator transakcji zakupu karnetu.

  - *klient_id* (INTEGER, FK, NOT NULL): Klucz obcy wskazujący na klienta, który zakupił karnet.
  - *typ_karnetu* (VARCHAR(20), NOT NULL): Typ karnetu (np. «miesieczny»). Ograniczony więzem CHECK.
  - *data_zakupu* (DATE, NOT NULL): Data, w której karnet został sprzedany.
  - *data_waznosci* (DATE, NOT NULL): Data, do której karnet jest ważny.
  - *cena* (NUMERIC(10, 2), NOT NULL): Cena zapłacona za karnet. Użycie typu *NUMERIC* zapobiega błędom zaokrągleń typowym dla typów zmiennoprzecinkowych.

- **Tabela: Wejscia**
  - *wejscie_id* (SERIAL, PK): Unikalny identyfikator zdarzenia wejścia.
  - *klient_id* (INTEGER, FK, NOT NULL): Klucz obcy wskazujący na wchodzącego klienta.
  - *data_wejscia* (TIMESTAMP, NOT NULL): Dokładna data i godzina wejścia. Wartość domyślna to *CURRENT_TIMESTAMP*.

3.4 Model Fizyczny (Kod SQL)

Poniższy kod DDL (Data Definition Language) dla PostgreSQL tworzy opisaną strukturę bazy danych.

Listing 1: Skrypt tworzący strukturę bazy danych

```
-- Tabela przechowująca dane klientów siłowni.
-- Każdy klient jest unikalnie identyfikowany przez email.
CREATE TABLE Klienci (
    klient_id SERIAL PRIMARY KEY,
    imie VARCHAR(50) NOT NULL,
    nazwisko VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    nr_telefonu VARCHAR(15)
);

-- Tabela przechowująca informacje o zakupionych karnetach.
-- Więzy integralności (FOREIGN KEY z ON DELETE CASCADE) zapewniają,
-- że usunięcie klienta spowoduje usunięcie jego karnetów.
CREATE TABLE Karnety (
    karnet_id SERIAL PRIMARY KEY,
    klient_id INTEGER NOT NULL,
    typ_karnetu VARCHAR(20) NOT NULL,
    data_zakupu DATE NOT NULL,
    data_waznosci DATE NOT NULL,
    cena NUMERIC(10, 2) NOT NULL,

    CONSTRAINT fk_klient
        FOREIGN KEY(klient_id)
        REFERENCES Klienci(klient_id)
        ON DELETE CASCADE,

    CONSTRAINT chk_typ_karnetu
        CHECK (typ_karnetu IN ('miesieczny', 'trzymiesieczny', 'polroczny'))
);
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
-- Tabela rejestrująca wejścia klientów.
-- Każde wejście jest powiązane z istniejącym klientem.
CREATE TABLE Wejscia (
    wejscie_id SERIAL PRIMARY KEY,
    klient_id INTEGER NOT NULL,
    data_wejscia TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT fk_klient
        FOREIGN KEY(klient_id)
        REFERENCES Klienci(klient_id)
        ON DELETE CASCADE
);
```

ROZDZIAŁ 4

4. Normalizacja, Wydajność i Bezpieczeństwo

W tym rozdziale przeprowadzono analizę kluczowych niefunkcjonalnych aspektów zaprojektowanej bazy danych.

4.1 Analiza normalizacji

Normalizacja jest procesem projektowania schematu bazy danych w celu zminimalizowania redundancji danych i wyeliminowania niepożądanych charakterystyk, takich jak anomalie wstawiania, aktualizacji i usuwania. Zaproponowany schemat jest zgodny z **trzecią postacią normalną (3NF)**.

- **Zgodność z 1NF:** Każda tabela posiada klucz główny, a wszystkie atrybuty zawierają wartości atomowe (niepodzielne).
- **Zgodność z 2NF:** Schemat spełnia drugą postać normalną, ponieważ wszystkie klucze główne są kluczami prostymi (jednokolumnowymi), co automatycznie eliminuje problem częściowych zależności funkcyjnych.
- **Zgodność z 3NF:** W schemacie nie występują zależności przechodnie. Żaden atrybut niekluczowy nie jest funkcyjnie zależny od innego atrybutu niekluczowego. Na przykład, w tabeli *Karnety*, atrybut *cena* jest bezpośrednio zależny od klucza *karnet_id*. Nie ma zależności *karnet_id -> typ_karnetu -> cena*, ponieważ cena jest zapisywana w momencie transakcji, co jest celowym zabiegiem denormalizacyjnym w celu archiwizacji danych historycznych i uodpornienia na zmiany cennika. Gdyby cena była zależna od typu karnetu, należałoby stworzyć osobną tabelę *Cennik*, a w tabeli *Karnety* przechowywać jedynie klucz obcy do niej.

Wnioski: Osiągnięty poziom normalizacji zapewnia wysoką integralność danych i elastyczność schematu, jednocześnie zachowując prostotę i zrozumiałość modelu.

4.2 Analiza wydajności i indeksowanie

Wydajność zapytań jest kluczowa dla responsywności systemu. Podstawową techniką optymalizacji jest strategiczne stosowanie indeksów.

Identyfikacja kandydatów do indeksowania: * **Klucze obce:** Kolumny używane jako klucze obce (*Karnety.klient_id*, *Wejscia.klient_id*) są głównymi kandydatami do indeksowania. Indeksy te drastycznie przyspieszają operacje *JOIN* oraz wyszukiwanie rekordów powiązanych z danym klientem. PostgreSQL automatycznie nie tworzy indeksów na kluczach obcych, więc należy je dodać ręcznie. * **Często filtrowane kolumny:** Kolumna *Karnety.data_waznosci* będzie

często używana w klauzuli *WHERE* do sprawdzania aktywnych karnetów. Dodanie na niej indeksu przyspieszy ten krytyczny proces biznesowy.

Przykładowa implementacja indeksów:

```
-- Indeks na kluczu obcym w tabeli Karnety
CREATE INDEX idx_karnety_klient_id ON Karnety(klient_id);

-- Indeks na kluczu obcym w tabeli Wejscia
CREATE INDEX idx_wejscia_klient_id ON Wejscia(klient_id);

-- Indeks wspomagający wyszukiwanie aktywnych karnetów
CREATE INDEX idx_karnety_data_waznosci ON Karnety(data_waznosci);
```

Analiza planu zapytania (`EXPLAIN ANALYZE`): Przed dodaniem indeksu *idx_karnety_klient_id*, zapytanie o wszystkie karnety danego klienta skutkowałoby pełnym skanowaniem tabeli (*Seq Scan*). Po jego dodaniu, planer zapytań PostgreSQL wykorzysta znacznie szybszy *Index Scan*, co przy dużej liczbie rekordów może skrócić czas wykonania zapytania z sekund do milisekund.

4.3 Zarządzanie bezpieczeństwem

Bezpieczeństwo danych osobowych i operacyjnych jest priorytetem. Zastosowano model bezpieczeństwa oparty na rolach (Role-Based Access Control).

Definicja ról: * **`rola_admin`**: Superużytkownik z pełnymi uprawnieniami do wszystkich tabel (CRUD - Create, Read, Update, Delete). Przeznaczona dla administratorów bazy danych. * **`rola_recepcja`**: Rola dla pracowników recepcji. Powinna mieć uprawnienia do:

- *SELECT* na *Klienci*.
- *INSERT* do *Klienci*.
- *SELECT*, *INSERT* na *Karnety*.
- *SELECT*, *INSERT* na *Wejscia*.
- Brak uprawnień *DELETE* i *UPDATE* na większości danych w celu ochrony przed przypadkowym usunięciem.
- **`rola_analityk`**: Rola tylko do odczytu (*SELECT*) na wszystkich tabelach. Przeznaczona dla analityków biznesowych generujących raporty.

Przykładowa implementacja ról i uprawnień:

```
-- Tworzenie ról
CREATE ROLE rola_recepcja;
CREATE ROLE rola_analityk;

-- Nadawanie uprawnień dla recepcji
GRANT SELECT, INSERT ON Klienci, Karnety, Wejscia TO rola_recepcja;
GRANT USAGE, SELECT ON SEQUENCE klienci_klient_id_seq, karnety_karnet_id_seq, wejscia_
↪wejscie_id_seq TO rola_recepcja;


-- Nadawanie uprawnień dla analityka
GRANT SELECT ON ALL TABLES IN SCHEMA public TO rola_analityk;

-- Tworzenie użytkowników i przypisywanie im ról
```

(ciąg dalszy na następnej stronie)

(kontynuacja poprzedniej strony)

```
CREATE USER pracownik_recepcji WITH PASSWORD 'bezpieczne_haslo';
GRANT rola_recepcja TO pracownik_recepcji;
```

4.4 Skrypty wspomagające

Poniższy skrypt w Pythonie został rozszerzony o dodatkową funkcjonalność - wyszukiwanie klientów, którym wkrótce wygasa karnet, co może być użyteczne dla działu marketingu.

Listing 1: Zaawansowany skrypt do generowania raportów

```
import psycopg2
from datetime import date, timedelta

# ... (konfiguracja połączenia DB_CONFIG) ...

def generuj_raport_wygasajacych_karnetow(dni_do_konca=7):
    """
    Znajduje klientów, których karnety wygasają
    w ciągu najbliższych 'dni_do_konca' dni.
    """
    # ... (logika połączenia z bazą) ...
    query = """
    SELECT k.imie, k.nazwisko, k.email, kr.data_waznosci
    FROM Klienci k
    JOIN Karnety kr ON k.klient_id = kr.klient_id
    WHERE kr.data_waznosci BETWEEN %s AND %s
    ORDER BY kr.data_waznosci ASC;
    """
    dzis = date.today()
    data_koncowa = dzis + timedelta(days=dni_do_konca)
    cur.execute(query, (dzis, data_koncowa))
    # ... (logika wyświetlania raportu) ...
```

ROZDZIAŁ 5

5. Podsumowanie, Wnioski i Repozytoria

5.1 Podsumowanie projektu

Niniejszy projekt stanowił pełne ćwiczenie inżynierskie, obejmujące cały cykl życia systemu bazodanowego. Rozpoczynając od analizy potrzeb biznesowych fikcyjnej siłowni, poprzez staranne modelowanie danych, aż po fizyczną implementację i analizę aspektów niefunkcjonalnych, projekt ten z powodzeniem przełożył wymagania na działające, bezpieczne i wydajne rozwiązanie. Zastosowanie normalizacji zapewniło integralność danych, podczas gdy świadome planowanie indeksów i polityk bezpieczeństwa przygotowało system do działania w rzeczywistym środowisku.

5.2 Wnioski

Realizacja projektu pozwoliła na sformułowanie następujących wniosków:

1. **Modelowanie jest kluczowe:** Czas poświęcony na staranne stworzenie modelu koncepcyjnego i logicznego procentuje na etapie implementacji i późniejszej konserwacji. Dobrze zaprojektowany schemat jest intuicyjny i łatwy do rozbudowy.
2. **Normalizacja to kompromis:** Chociaż dążenie do wyższych postaci normalnych jest teoretycznie pożądane, w praktyce należy uwzględniać również wydajność i logikę biznesową. Celowe, udokumentowane odstępstwa (jak przechowywanie ceny w momencie transakcji) są często uzasadnione.
3. **Wydajność i bezpieczeństwo nie są opcjonalne:** Aspekty te muszą być uwzględniane od samego początku procesu projektowego, a nie dodawane jako „łatki” na końcu. Strategiczne indeksowanie i model bezpieczeństwa oparty na rolach to fundamenty stabilnego systemu.
4. **Praktyka utrwala teorię:** Projekt ten był nieocenionym doświadczeniem, które pozwoliło na praktyczne zastosowanie i głębsze zrozumienie teoretycznych koncepcji omawianych na zajęciach, takich jak replikacja, partycjonowanie czy zaawansowane strategie backupu.

5.3 Możliwe kierunki dalszego rozwoju

Zaprojektowana baza danych stanowi solidny fundament, który można rozwijać w wielu kierunkach, aby zwiększyć jej wartość biznesową:

- **Moduł rezerwacji zajęć:** Dodanie tabel *Zajecia*, *Instruktorzy* oraz *Rezerwacje* w celu umożliwienia klientom rezerwacji miejsc na zajęciach grupowych.
- **Integracja z systemem płatności:** Połączenie z bramką płatniczą w celu automatyzacji sprzedaży karnetów online.
- **Aplikacja kliencka:** Stworzenie aplikacji mobilnej lub webowej dla klientów, gdzie mogliby sprawdzać ważność swojego karnetu, historię wejść i rezerwować zajęcia.
- **Zaawansowana analityka:** Budowa hurtowni danych i wykorzystanie narzędzi BI (Business Intelligence) do analizy trendów, np. godzin największego obłożenia siłowni, najpopularniejszych typów karnetów czy segmentacji klientów.

5.4 Spis repozytoriów

1. **Repozytorium niniejszego sprawozdania:** * *https://github.com/HoszeQ/karnety_silownia_sprawozdanie*
2. **Repozytorium projektu grupowego:** * *https://github.com/m-smieja/Kopie_zapasowe_i_odzyskiwanie_danych*