Dokumentacja Wombat Grylls sp. z.o.o.

Maria Wilgosz, Oliwia Marut, Jan Sobkowiak, Kuba Klimek, Michał Pluciński

1. Spis użytych technologii

W celu optymalizacji naszej pracy podczas procesu tworzenia projektu korzystaliśmy z wielu dostępnych technologii.

1.1 Do tworzenia i wypełniania bazy danych zdecydowaliśmy się wybrać język programowania **Python** wraz z niezbędnymi bibliotekami:

- mysql.connector
- random
- datetime

1.2 Z uwagi na możliwość automatycznego generowania raportów za pomocą silnika oprogramowania do dynamicznego generowania raportów Knitr. Do analizy danych wykorzystaliśmy język programowania **R** wraz z bibliotekami:

- DBI
- dplyr
- lubridate
- ggplot2
- RMariaDB

2. Lista plików i opis ich zawartości

Pliki do bazy danych

- generate_trips.py – skrypt do generowania zrealizowanych wycieczek i wstawiania ich do bazy danych.
- zapłata.py - skrypt do obliczania kosztów wycieczek zrealizowanych przez klientów i wstawiania do bazy**.**
- Projekt bazy danych.sql – plik do tworzenia tabel w bazie oraz wypełnienie bazy
- Uzupełnienie_tabel.R – plik służący do uzupełniania tabel danymi z plików o rozszerzeniu .csv

Pliki z danymi

- adresy_data.csv – plik z wygenerowanymi danymi dotyczącymi adresów klientów oraz pracowników przez aplikacje Mackaroo
- klienci_data.csv – plik z wygenerowanymi danymi klientów przez aplikacje Mockaroo
- pracownicy_data.csv – plik z wygenerowanymi danymi pracowników przez aplikacje Mockaroo

Pliki z analizą danych

- raport.rmd - Plik do generowania raportu.
- raport.pdf - Plik z gotowym raportem.

3. Instrukcja uruchomienia projektu

3.1 Instalacja wymaganych pakietów

1. Upewnij się, że masz zainstalowanego Pythona (zalecana wersja 3.8 lub wyższa)

2. Zainstaluj potrzebne pakiety
3. Należy posiadać język oprogramowania R wraz ze środowiskiem Rstudio.

3.2 Jeśli baza danych nie jest pusta

1. Usuń wszystkie tabele i przejdź do punktu 3.3.

3.3 Jeśli baza danych jest pusta

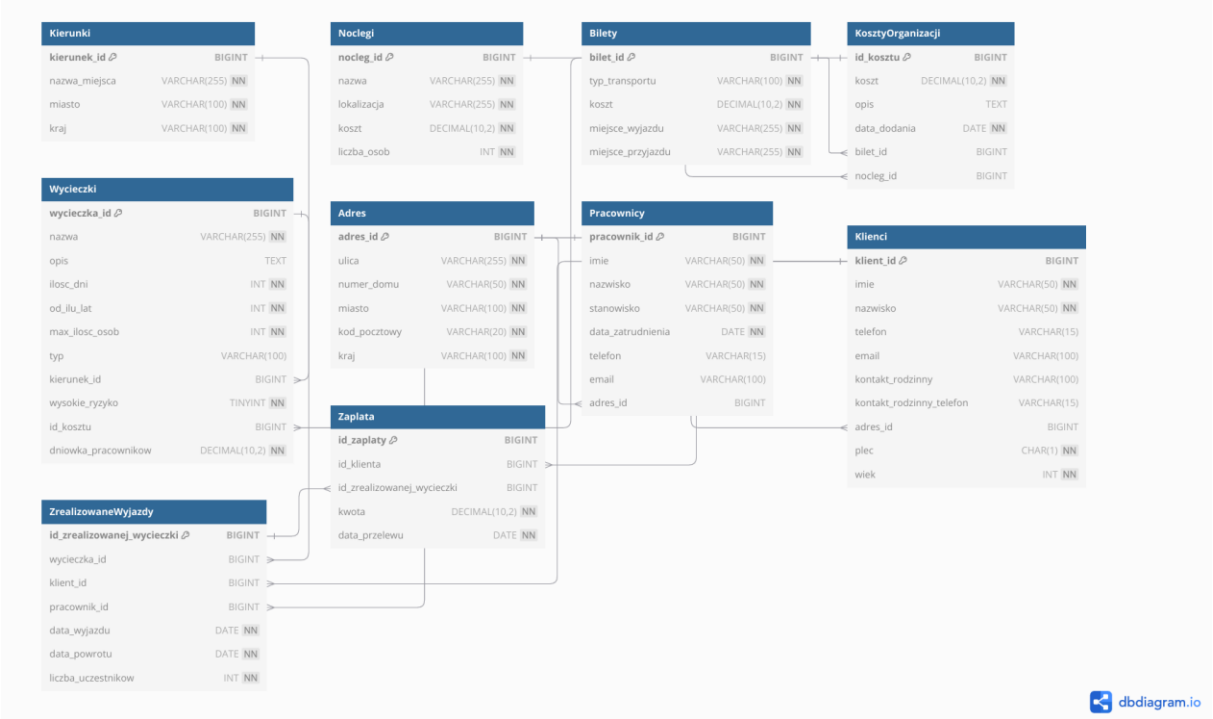
1. Uruchom polecenia do stworzenia tabel, zawarte w pliku 'Projekt bazy danych.sql'.

2. Wygeneruj dane do bazy danych, otwórz i uruchom skrypty 'generate_trips.py' i 'zapłata.py'.

3.4 Generowanie raportu

1. Aby wygenerować raport należy uruchomić plik raport.Rmd i go przekompliować.

4. Schemat projektu bazy danych



5. Lista zależności funkcyjnych

Poniżej prezentujemy tabele wraz z ich listami zależności funkcyjnych.

5.1 Adres

Kolumna	**Typ**	**Opis**
adres_id	BIGINT	Klucz główny tabeli, jednoznacznie identyfikuje każdy adres
ulica	VARCHAR(255)	Nazwa ulicy
numer_domu	VARCHAR(50)	Numer domu
miasto	VARCHAR(100)	Miasto

kod_pocztowy	VARCHAR(20)	Kod pocztowy
kraj	VARCHAR(100)	Kraj

Tabela 1: Struktura tabeli Adres

Zależności funkcyjne dla tabeli klientów:

Σ = {adres_id → ulica, adres_id → numer_domu, adres_id → miasto, adres_id → kod_pocztowy, adres_id → kraj}

Kolumna adres_id jest kluczem głównym tabeli, jednoznacznie identyfikującym każdy adres. Każdy adres składa się z ulicy, numeru domu, miasta, kodu pocztowego oraz kraju, które są zależne od adres_id.

5.2 Bilety

Kolumna	**Typ**	**Opis**
bilet_id	BIGINT	Klucz główny tabeli, unikalny identyfikator biletu
typ_transportu	VARCHAR(100)	Środek transportu używany w podróży
koszt	DECIMAL	Cena biletu dla jednej osoby
miejsce_wyjazdu	VARCHAR(255)	Miejsce, z którego rozpoczyna się podróż
miejsce_przyjadu	VARCHAR(255)	Docelowe miejsce podróży

Tabela 2: Struktura tabeli Bilety

Zależności funkcyjne dla tabeli klientów:

Σ = {bilet_id → typ_transportu, bilet_id → koszt, bilet_id → miejsce_wyjazdu, bilet_id → miejsce_przyjazdu}

Kolumna bilet_id jest kluczem głównym i jednoznacznie identyfikuje każdy bilet. typ_transportu, koszt, miejsce_wyjazdu oraz miejsce_przyjazdu są atrybutami zależnymi od bilet_id, ponieważ opisują charakterystykę konkretnego biletu.

5.3 Kierunki

Kolumna	**Typ**	**Opis**
kierunek_id	BIGINT	Klucz główny tabeli, jednoznacznie identyfikuje dany kierunek podróży
nazwa_miejsca	VARCHAR(255)	Nazwa konkretnego miejsca docelowego
miasto	VARCHAR(100)	Miasto, w którym znajduje się cel podróży
kraj	VARCHAR(100)	Państwo, w którym znajduje się cel podróży

Tabela 3: Struktura tabeli Kierunki

Zależności funkcyjne dla tabeli klientów:

Σ = {kierunek_id → nazwa_miejsca, kierunek_id → miasto, kierunek_id → kraj}

Kolumna kierunek_id jednoznacznie identyfikuje każdy kierunek podróży. nazwa_miejsca, miasto i kraj zależą bezpośrednio od kierunek_id, ponieważ opisują konkretne miejsce podróży.

5.4 Klienci

Kolumna	**Typ**	**Opis**
kilent_id	BIGINT	Klucz główny tabeli, jednoznacznie identyfikuje każdego klienta
imie	VARCHAR(50)	Imię klienta
nazwisko	VARCHAR(50)	Nazwisko klienta
telefon	VARCHAR(15)	Numer telefonu klienta, unikalny dla każdego wpisu
email	VARCHAR(100)	Adres e-mail klienta, unikalny dla każdego wpisu
kontakt_rodzinny	VARCHAR(100)	osoba kontaktowa w razie nagłej sytuacji
kontakt_rodzinny_telefon	VARCHAR(15)	Numer telefonu osoby kontaktowej
adres_id	BEGINT	Klucz obcy do tabeli Adres, wskazujący na adres zamieszkania klienta
Płeć	CHAR(1)	Płeć klienta (M – mężczyzna, K – kobieta)
wiek	INT	Wiek klienta

Tabela 4: Struktura tabeli Klienci

Zależności funkcyjne dla tabeli klientów:

Σ = {klient_id → imie, klient_id → nazwisko, klient_id → telefon, klient_id → email, klient_id → kontakt_rodzinny, klient_id → kontakt_rodzinny_telefon, klient_id → adres_id, klient_id → plec, klient_id → wiek, telefon → klient_id, email → klient_id}

Kolumna klient_id jest kluczem głównym tabeli Klienci i jednoznacznie identyfikuje każdego klienta. Kolumny telefon i email są unikalne i mogą być użyte do identyfikacji klienta. Kolumny imie, nazwisko, kontakt_rodzinny, kontakt_rodzinny_telefon, adres_id, plec i wiek są zależne od klient_id, ponieważ opisują konkretnego klienta.

5.5 KosztyOrganizacji

Kolumna	**Typ**	**Opis**
id_kosztu	BIGINT	Klucz główny tabeli, jednoznacznie identyfikuje dany koszt organizacyjny
koszt	DECIMAL	Całkowity koszt organizacji wycieczki dla jednej osoby
data_dodania	DATE	Data zapisania kosztu do systemu

bilet_id	BEGINT	Klucz obcy do tabeli Bilety, wskazujący na koszt biletu
nocleg_id	BEGINT	Klucz obcy do tabeli Noclegi, wskazujący na koszt noclegu

Tabela 5: Struktura tabeli KosztyOrganizacji

Zależności funkcyjne dla tabeli klientów:

Σ = {id_kosztu → koszt, id_kosztu → data_dodania, id_kosztu → bilet_id, id_kosztu → nocleg_id}

Kolumna id_kosztu jednoznacznie identyfikuje każdy koszt organizacyjny. koszt zależy od ceny biletu (bilet_id) oraz kosztu noclegu (nocleg_id).

5.6 Noclegi

Kolumna	**Typ**	**Opis**
nocleg_id	BIGINT	Klucz główny tabeli, jednoznacznie identyfikuje miejsce noclegu
nazwa	VARCHAR(255)	Nazwa obiektu noclegowego
lokalizacja	VARCHAR(255)	Lokalizacja noclegu
koszt	DECIMAL	Koszt noclegu za cały wyjazd za jedną osobę
liczba_osob	INT	Liczba osób w noclegu

Tabela 6: Struktura tabeli Noclegi

Zależności funkcyjne dla tabeli klientów:

Σ = {nocleg_id → nazwa, nocleg_id → lokalizacja, nocleg_id → koszt, nocleg_id → liczba_osob}

Kolumna nocleg_id jest kluczem głównym tabeli, jednoznacznie identyfikuje dany obiekt noclegowy. nazwa, lokalizacja, koszt oraz liczba_osob zależą bezpośrednio od nocleg_id, ponieważ określają cechy konkretnego miejsca noclegowego.

5.7 Pracownicy

Kolumna	**Typ**	**Opis**
parcownik_id	BIGINT	Klucz główny tabeli, jednoznacznie identyfikuje pracownika
imie	VARCHAR(50)	Imię pracownika
nazwisko	VARCHAR(50)	Nazwisko pracownika
stanowisko	VARCHAR(50)	Stanowisko pracownika
data_zatrudnienia	DATE	Data zatrudnienia
telefon	VARCHAR(15)	Telefon pracownika

email	VARCHAR(100)	Adres e-mail
adres_id	BEGINT	Klucz obcy do tabeli Adres, wskazujący na miejsce zamieszkania pracownika

Tabela 7: Struktura tabeli Pracownicy

Zależności funkcyjne dla tabeli klientów:

Σ = {pracownik_id → imie, pracownik_id → nazwisko, pracownik_id → stanowisko, pracownik_id → data_zatrudnienia, pracownik_id → telefon, pracownik_id → email, pracownik_id → adres_id, telefon → pracownik_id, email → pracownik_id}

Kolumna pracownik_id jednoznacznie identyfikuje każdego pracownika. telefon i email są unikalne, więc mogą identyfikować pracownika niezależnie. Kolumny imie, nazwisko, stanowisko, telefon, email oraz adres_id są zależne od pracownik_id, ponieważ opisują konkretnego pracownika.

5.8 Wycieczki

Kolumna	**Typ**	**Opis**
wycieczka_id	BIGINT	Klucz główny tabeli, jednoznacznie identyfikuje daną wycieczkę
nazwa	VARCHAR(255)	Nazwa wycieczki
opis	TEXT	Opis wycieczki
ilosc_dni	INT	Liczba dni wycieczki
od_ilu_lat	INT	Minimalny wiek uczestników
max_ilosc_osob	INT	Maksymalna liczba uczestników
typ	VARCHAR(100)	Typ wycieczki
kierunek_id	BEGINT	Klucz obcy do tabeli Kierunki, wskazujący na miejsce docelowe wycieczki

wysokie_ryzyko	TINYINT	Wskaźnik określający poziom ryzyka (0 – niskie, 1 – wysokie)
id_kosztu	BEGINT	Klucz obcy do tabeli KosztyOrganizacji, wskazujący na koszt wycieczki
dniowka_pracownikow	DECIMAL	Stawka dzienna wynagrodzenia dla pracownika obsługującego wycieczkę

Tabela 8: Struktura tabeli Wycieczki

Zależności funkcyjne dla tabeli klientów:

Σ = {wycieczka_id → nazwa, wycieczka_id → opis, wycieczka_id → ilosc_dni, wycieczka_id → od_ilu_lat, wycieczka_id → max_ilosc_osob, wycieczka_id → typ, wycieczka_id → kierunek_id, wycieczka_id → wysokie_ryzyko, wycieczka_id → id_kosztu, wycieczka_id → dniowka_pracownikow}

Tabela Wycieczki zawiera informacje o wszystkich dostępnych wycieczkach oferowanych przez firmę. Każdy wpis w tabeli reprezentuje jedną wycieczkę, określając jej szczegóły, takie jak nazwa, liczba dni, maksymalna liczba uczestników, rodzaj, a także miejsce docelowe. Dodatkowo przechowuje informacje o poziomie ryzyka oraz powiązanym koszcie organizacji.

5.9 Zapłata

Kolumna	**Typ**	**Opis**
id_zapłaty	BIGINT	Klucz główny tabeli, jednoznacznie identyfikuje każdą transakcję
id_klienta	BEGINT	Klucz obcy do tabeli Klienci, wskazujący na osobę płacącą
id_zrealizowanej_wycieczki	BEGINT	Klucz obcy do tabeli ZrealizowaneWyjazdy, wskazujący na wycieczkę, za którą dokonano płatności
kwota	DECIMAL	Kwota wpłaty klienta
data_przelewu	DATE	Data wykonania przelewu

Tabela 9: Struktura tabeli Zapłata

Zależności funkcyjne dla tabeli klientów:

Σ = {id_zrealizowanej_wycieczki → wycieczka_id, id_zrealizowanej_wycieczki → klient_id, id_zrealizowanej_wycieczki → pracownik_id, id_zrealizowanej_wycieczki → data_wyjazdu, id_zrealizowanej_wycieczki → data_powrotu, id_zrealizowanej_wycieczki → liczba_uczestnikow}

Tabela Zapłata przechowuje informacje o płatnościach dokonywanych przez klientów za udział w zrealizowanych wycieczkach. Każdy wpis w tabeli oznacza jedną płatność, przypisaną do konkretnego klienta i zrealizowanej wycieczki.

5.10 ZrealizowaneWyjazdy

Kolumna	**Typ**	**Opis**
id_zrealizowanej_wycieczki	BIGINT	Klucz główny tabeli, jednoznacznie identyfikuje każdą realizację wycieczki
wycieczka_id	BEGINT	Klucz obcy do tabeli Wycieczki, wskazujący na rodzaj zrealizowanej wycieczki
klient_id	BEGINT	Klucz obcy do tabeli Klienci, wskazujący na uczestnika danej wycieczki
pracownik_id	BEGINT	Klucz obcy do tabeli Pracownicy, wskazujący na osobę prowadzącą wycieczkę
data_wyjazdu	BEGINT	Data rozpoczęcia wycieczki
data_powrotu	DATE	Data zakończenia wycieczki
liczba_uczestnikow	INT	Liczba uczestników tej konkretnej wycieczki

Tabela 10: Struktura tabeli ZrealizowaneWyjazdy

Zależności funkcyjne dla tabeli klientów:

Σ = {id_zrealizowanej_wycieczki → wycieczka_id, klient_id, pracownik_id, data_wyjazdu, data_powrotu, liczba_uczestnikow, wycieczka_id → liczba_uczestnikow, data_wyjazdu, data_powrotu, pracownik_id → wycieczka_id, data_wyjazdu, data_powrotu}

Każda realizacja wycieczki (id_zrealizowanej_wycieczki) ma przypisane konkretne wartości dla wycieczka_id, klient_id, pracownik_id, data_wyjazdu, data_powrotu i liczba_uczestnikow, ponieważ jeden wpis reprezentuje jedną unikalną realizację wycieczki. wycieczka_id określa termin (data_wyjazdu, data_powrotu) oraz liczbę uczestników, ponieważ każda wycieczka ma określony czas trwania i maksymalną liczbę uczestników. pracownik_id wskazuje pracownika prowadzącego wycieczkę i definiuje, w jakim czasie ta osoba była zajęta, ponieważ jeden przewodnik może obsługiwać tylko jedną wycieczkę naraz.

6. Baza typu EKNF

Normalizacja baz danych to proces organizowania danych w bazie w taki sposób, aby zminimalizować redundancję i poprawić integralność danych. Proces ten składa się z kilku poziomów normalizacji, z których każdy wprowadza coraz bardziej restrykcyjne zasady. Poniżej znajdują się definicje najważniejszych form normalizacji: 1NF, 2NF, 3NF oraz EKNF.

1NF (First Normal Form)

Pierwsza postać normalna (1NF) wprowadza podstawowe zasady organizacji danych:

• Każda tabela ma swój własny klucz główny, który jednoznacznie identyfikuje każdy rekord.

• Każda komórka w tabeli zawiera tylko jedną wartość, co oznacza, że komórki nie mogą zawierać zbiorów wartości, ani list.

• Każda komórka opisuje tylko jeden atrybut danego obiektu.

• Kolejność wierszy w tabeli nie ma znaczenia, a dane mogą być w dowolnym porządku

2NF (Second Normal Form)

Druga postać normalna (2NF) rozszerza zasady 1NF i dodaje nowe warunki:

• Baza danych spełnia wymagania 1NF.

• Wszystkie atrybuty, które nie są kluczami głównymi, muszą być w pełni zależne od całego klucza głównego, a nie tylko od jego części. Oznacza to, że w tabelach złożonych z kluczy złożonych, atrybuty niekluczowe muszą zależeć od wszystkich części tego klucza

3NF (Third Normal Form)

Trzecia postać normalna (3NF) wprowadza dodatkowe ograniczenia do 2NF:

• Baza danych spełnia wymagania 2NF.

• Żaden atrybut nie będący kluczem nie może zależeć od innego atrybutu nie będącego kluczem w tej samej tabeli. Oznacza to, że atrybuty niekluczowe powinny zależeć bezpośrednio od klucza głównego, a nie od siebie nawzajem.

EKNF (Elementary Key Normal Form)

Elementary Key Normal Form (EKNF) wprowadza jeszcze bardziej zaawansowane zasady:

• Baza danych spełnia wymagania 3NF.

• Nie istnieje żadna relacja, która jest niezgodna z definicją EKNF. Oznacza to, że każda relacja w bazie danych musi być zgodna z zasadami kluczy elementarnych, co eliminuje niepożądane zależności pomiędzy atrybutami.

Na podstawie pokazanych wyżej zależności funkcyjnych widzimy, że nasza baza spełnia powyższą definicje.

7. Wyzwania podczas realizacji projektu

Największym wyzwaniem dla nas było stworzenie bazy danych tak, aby jak najlepiej odwzorowywała ona rzeczywistość. Dodatkowo napotkaliśmy trudności podczas optymalnego doboru liczby danych w poszczególnych tabelach tak, żeby było ich wystarczająco dużo do przeprowadzenia analizy i wyciągnięcia wniosków, a jednocześnie na tyle mało, żeby generowały się one względnie szybko. W trakcie projektu, napotkaliśmy sytuację, w której pracownik jednocześnie był na dwóch wycieczkach. Udało nam się zlikwidować tę usterkę za pomocą dodania w poleceniu warunku, że jeden pracownik nie może być w tym samym czasie na dwóch wycieczkach. Dużym wyzwaniem dla nas okazało się kodowanie w języku R, z którym nie mieliśmy wcześniej do czynienia, a także generowanie PDF bez kodów.