

Przedmiot	Bazy danych
Zadanie	Dokumentacja
Imię i nazwisko	Krystian Smagacz
Nr albumu	178756

Dokumentacja

Tworzenie bazy danych dla platformy zarządzania relacjami influencerów, umożliwiającej śledzenie kampanii, wynagrodzeń i efektów działań marketingowych

Tematyka i zakres projektu

Projekt dotyczy stworzenia relacyjnej bazy danych wspierającej platformę do zarządzania relacjami z influencerami, która odpowiada na potrzeby nowoczesnego marketingu internetowego. W dobie dominacji mediów społecznościowych, firmy coraz częściej współpracują z influencerami przy promocji swoich produktów i usług. Skuteczne zarządzanie tymi współpracami wymaga jednak gromadzenia i analizowania dużej ilości danych – od podstawowych informacji o influencerach, przez szczegóły kampanii, aż po konkretne wyniki i efekty działań reklamowych. Projektowana baza danych ma umożliwić scentralizowane zarządzanie wszystkimi tymi informacjami w sposób uporządkowany i przejrzysty.

Zakres projektu obejmuje kompleksowe podejście do problemu: od analizy wymagań i zaprojektowania modelu danych, przez stworzenie struktury bazy w systemie MySQL, aż po wdrożenie aplikacji internetowej zbudowanej za pomocą Node.js + Express (backend) oraz React (frontend). Kluczowe elementy to m.in. rejestrowanie relacji między influencerami a kampaniami, przypisywanie budżetów, śledzenie efektów (takich jak zasięgi czy konwersje), rozliczanie wynagrodzeń oraz generowanie raportów. Projekt uwzględnia również aspekty związane z użytkownikami platformy, jak administratorzy i menedżerowie kampanii.

System ma być skalowalny i przygotowany do rozbudowy, np. o możliwość integracji z API platform społecznościowych (takich jak Instagram, TikTok czy YouTube), co w przyszłości pozwoli na automatyczne pobieranie statystyk. Chociaż w obecnej wersji prezentacyjnej funkcje te będą przedstawione w formie prototypu, struktura bazy danych zostanie zaprojektowana w taki sposób, by wspierać przyszłe rozszerzenia. Dzięki temu platforma może stać się realnym narzędziem wspierającym działania marketingowe w profesjonalnym środowisku agencji reklamowych lub działów marketingu firm.

Zakres projektu obejmuje:

- zaprojektowanie modelu bazy danych w MySQL,
- stworzenie struktury tabel uwzględniającej relacje między encjami,
- stworzenie backendu w Node.js z użyciem Express do obsługi REST API,

- budowę frontendowej aplikacji w React, która przez HTTP (fetch) komunikuje się z serwerem,
- testowe wdrożenie lokalne na serwer Express - port 3001, aplikacja React port - 3000 (następnie jest planowane zastosowanie hostingu np. Render, Railway, Vercel)

Zagadnienia związane z tematem

Projekt zahacza o następujące zagadnienia:

- modelowanie danych i tworzenie schematu ERD,
- zarządzanie danymi marketingowymi (kampanie, zasięgi, konwersje),
- rejestrowanie współprac z influencerami,
- rozliczenia i raportowanie efektów działań,
- bezpieczeństwo danych i autoryzacja dostępu,
- podstawy analityki i raportowania w marketingu.

Funkcje bazy danych i ich priorytety

Funkcja	Opis	Priorytet
Zarządzanie influencerami	Rejestracja danych osobowych, kanałów social media, statystyk	Wysoki
Zarządzanie kampaniami	Tworzenie kampanii, przypisywanie influencerów, określanie budżetu i terminów	Wysoki
Śledzenie efektów	Rejestracja zasięgów, zaangażowania, kliknięć, konwersji	Średni
System wynagrodzeń	Obliczanie i rejestrowanie wypłat za kampanie	Wysoki
Historia współprac	Przechowywanie informacji o zakończonych kampaniach	Średni
Raportowanie	Generowanie zestawień dla kampanii i influencerów	Średni
Panel administratora	Zarządzanie użytkownikami i kampaniami	Wysoki

Wybór technologii i rodzaju bazy danych

Rodzaj bazy danych:
Relacyjna baza danych

Silnik bazy danych:
MySQL

Powody wyboru:

- szerokie wsparcie społeczności,
- integracja,
- odpowiednia dla relacyjnych danych z dużą ilością połączeń między tabelami,
- podstawowa/średnia znajomość poniższych technologii.

Narzędzie	Zastosowanie
Node.js	Środowisko uruchomieniowe JavaScript (backend)
Express	Lekki framework HTTP na Node.js (REST API)
React	Frontendowa biblioteka JavaScript (interfejs użytkownika)
MySQL	System zarządzania relacyjną bazą danych
phpMyAdmin	Graficzny interfejs do zarządzania bazą MySQL
Git & GitHub	Kontrola wersji

Repozytorium projektu

Repozytorium znajduje się na platformie GitHub i zawiera:

- pliki SQL z definicjami tabel i przykładowymi danymi,
- kod serwera Node.js/Express (` backend`),
- kod aplikacji React (` frontend`),

- endpointy,
- dokumentację techniczną i instrukcje uruchomienia (README.md).

Link do repozytorium:

<https://github.com/Dziad573/Influencer-Campaign-Manager>

Środowisko prezentacyjne

System zostanie zaprezentowany lokalnie przy użyciu:

Backend

- Node.js + Express na porcie 3001,
- Połączenie z lokalnym MySQL (domyślnie port 3306)

Frontend

- React na porcie 3000

Baza danych i narzędzia admina

- MySQL uruchomiony np. przez XAMPP,
- phpMyAdmin do podglądu i edycji tabel (np. <http://localhost/phpmyadmin>)

Diagram bazy danych:

Campaign_influencers – encja pomocnicza realizująca relację wiele do wielu między kampaniami a influencerami.

Klucz główny: **id**,

Klucze obce: *campaign_id*, *influencer_id*.

Campaign_effects – zawiera dane o efektach kampanii (np. liczba lajków, wyświetleń).

Klucz główny: **effect_id**

Klucze obce: *campaign_id*.

Payments – przechowuje informacje o płatnościach dla influencerów.

Klucz główny: **payment_id**

Klucze obce: *influencer_id*, *campaign_id*.

Platforms – przechowuje dane o platformach społecznościowych.

Klucz główny: **platform_id**,

Influencer_platforms – zawiera dane o kontach influencerów na poszczególnych platformach.

Złożony klucz główny: *influencer_id*, *platform_id*.

Wykonanie bazy danych (bez wartości):

```
CREATE TABLE `users` (  
  `user_id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  `username` varchar(100) NOT NULL,  
  `password_hash` varchar(255) NOT NULL,  
  `email` varchar(320) NOT NULL,  
  `role` enum('admin','manager','influencer','viewer') NOT NULL,  
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),  
  `icon` varchar(255) DEFAULT NULL
```

```
);  
  
CREATE TABLE `campaigns` (  
  `campaign_id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  `name` varchar(150) NOT NULL,  
  `client` varchar(150) NOT NULL,  
  `start_date` date NOT NULL,  
  `end_date` date NOT NULL,  
  `budget` decimal(12,2) NOT NULL,  
  `goal_description` text DEFAULT NULL,  
  `status` enum('Planned','Ongoing','Completed','Cancelled') NOT NULL,  
  `description` text NOT NULL  
);
```

```
CREATE TABLE `influencers` (  
  `influencer_id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  `first_name` varchar(100) NOT NULL,  
  `last_name` varchar(100) NOT NULL,  
  `email` varchar(320) NOT NULL,  
  `phone` varchar(50) NOT NULL,  
  `platform` enum('Instagram','YouTube','TikTok','Twitch','Kick','X') DEFAULT NULL,  
  `followers_count` int(11) DEFAULT NULL,
```



```

`engagement_rate` decimal(5,2) DEFAULT NULL,
`notes` text DEFAULT NULL,
`user_id` int(11) NOT NULL

FOREIGN KEY (user_id) REFERENCES users(user_id)

);

CREATE TABLE `campaign_effects` (
  `effect_id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `campaign_id` int(11) NOT NULL,
  `influencer_id` int(11) NOT NULL,
  `views` int(11) NOT NULL,
  `likes` int(11) NOT NULL,
  `comments` int(11) NOT NULL,
  `shares` int(11) NOT NULL,
  `report_date` date DEFAULT NULL,

  FOREIGN KEY (campaign_id) REFERENCES campaigns(campaign_id),

);

```

```

CREATE TABLE `campaign_influencers` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `campaign_id` int(11) NOT NULL,
  `influencer_id` int(11) NOT NULL,
  `agreed_fee` decimal(10,2) NOT NULL,
  `contract_signed` tinyint(1) NOT NULL,
  `notes` text DEFAULT NULL,

  FOREIGN KEY (campaign_id) REFERENCES campaigns(campaign_id),

```

```

FOREIGN KEY (influencer_id) REFERENCES influencers(influencer_id)

);

CREATE TABLE `payments` (
  `payment_id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `influencer_id` int(11) NOT NULL,
  `campaign_id` int(11) NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `payment_date` date NOT NULL,
  `status` enum('Pending','Paid','Failed') NOT NULL,
  FOREIGN KEY (campaign_id) REFERENCES campaigns(campaign_id),
  FOREIGN KEY (influencer_id) REFERENCES influencers(influencer_id)
);

CREATE TABLE `platforms` (
  `platform_id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `name` varchar(50) NOT NULL,
  `url_template` varchar(255) NOT NULL,
  `description` text NOT NULL,
  UNIQUE KEY ux_platform_name (name)
);

CREATE TABLE `influencer_platforms` (
  `influencer_id` int(11) NOT NULL
  `platform_id` int(11) NOT NULL,
  `account_url` varchar(255) NOT NULL,
  `platform_username` varchar(255) NOT NULL,
  `joined_date` date NOT NULL,
  `follows` bigint unsigned,
  PRIMARY KEY (influencer_id, platform_id)

```




);

Proste zapytania SQL (dodanie danych, selekcja).

-Dodanie użytkownika

```
INSERT INTO users (username, password_hash, email, role)
VALUES ('admin', 'admin@gmail.com', '123456789', 'admin');
```




+ Opcje

	user_id	username	password_hash	email	role	created_at	icon
<input type="checkbox"/>    1		admin	admin@gmail.com	123456789	admin	2025-05-21 21:31:09	NULL

-Dodanie influencera

```
INSERT INTO influencers (first_name, last_name, email, phone, platform,
followers_count, user_id)
VALUES ('Anna', 'Kowalska', 'annak@gmail.com', '678456123', 'Instagram', 150000, 5);
```

+ Opcje

	influencer_id	first_name	last_name	email	phone	platform	followers_count	engagement_rate	notes	user_id
<input type="checkbox"/>    2		Anna	Kowalska	annak@gmail.com	678456123	Instagram	150000	NULL	NULL	5

-Wyświetlenie wszystkich kampanii:

```
SELECT * FROM campaigns;
```

-Wyświetlenie trzech najpopularniejszych kampanii:

```
SELECT campaigns.name, campaign_effects.views
FROM campaigns
```

```
JOIN campaign_effects ON campaigns.campaign_id = campaign_effects.campaign_id  
  
ORDER BY campaign_effects.views DESC LIMIT 3;
```

Zaawansowane zapytania SQL (dodawanie, aktualizacja, selekcja danych)

-Połączenie influencera z kampanią:

```
INSERT INTO campaign_influencers(campaign_id, influencer_id, agreed_fee, contract_signed)  
VALUES (6, 6, 3300, 1);
```

-Dodanie raportu z przeprowadzonej kampanii (bezpośrednia relacja pomiędzy tabelami influencers i campaign_effects została usunięta, lecz kolumna influencer_id jest potrzebna by znać raport poszczególnych influencerów):

```
INSERT INTO campaign_effects(campaign_id, influencer_id, views, likes, comments, shares,  
report_date)  
  
VALUES (6, 6, 150000, 12000, 1200, 400, '2025-07-02');
```

-Zmiana imienia:

```
UPDATE influencers SET first_name = 'Magdalena' WHERE influencer_id = 3;
```

-Zmiana danych raportu

```
UPDATE campaign_effects SET views = 23500, likes = 1237, comments = 79, shares = 45  
WHERE effect_id = 10;
```

-Pełny raport z przeprowadzonej kampanii:

```
SELECT campaign_influencers.*, campaign_effects.*, influencers.*, campaigns.*,  
users.username, users.email, users.created_at  
FROM campaign_influencers  
    JOIN campaign_effects  
        ON campaign_influencers.campaign_id = campaign_effects.campaign_id  
        AND campaign_influencers.influencer_id = campaign_effects.influencer_id  
    JOIN influencers  
        ON campaign_influencers.influencer_id = influencers.influencer_id  
    JOIN campaigns  
        ON campaign_influencers.campaign_id = campaigns.campaign_id
```

```
JOIN users
      ON influencers.user_id = users.user_id WHERE campaigns.status = '
Completed';
```

Prezentacja użytkowników BD i ich ról

```
CREATE USER 'manager_influencer_campaign_manager_db'@'localhost' IDENTIFIED BY
'manager_influencer_campaign_manager_db';
```

```
CREATE USER 'admin_influencer_campaign_manager_db'@'localhost' IDENTIFIED BY
'admin_influencer_campaign_manager_db';
```

<input type="checkbox"/>	admin_influencer_campaign_manager_db	localhost	Tak	USAGE	Nie		
<input type="checkbox"/>	manager_influencer_campaign_manager_db	localhost	Tak	USAGE	Nie		

```
GRANT ALL PRIVILEGES ON influencer_campaign_manager_db.* TO
'admin_influencer_campaign_manager_db'@'localhost';
```

Edytuj uprawnienia: Konto użytkownika 'admin_influencer_campaign_manager_db'@'localhost'

Uprawnienia specyficzne dla baz danych						
Baza danych	Uprawnienia	Nadawanie	Uprawnienia specyficzne dla tabel	Działanie		
influencer_campaign_manager_db	ALL PRIVILEGES	Nie	Nie			

```
GRANT SELECT ON influencer_campaign_manager_db.* TO
'manager_influencer_campaign_manager_db'@'localhost';
```

Edytuj uprawnienia: Konto użytkownika 'manager_influencer_campaign_manager_db'@'localhost'

Uprawnienia specyficzne dla baz danych						
Baza danych	Uprawnienia	Nadawanie	Uprawnienia specyficzne dla tabel	Działanie		
influencer_campaign_manager_db	SELECT	Nie	Nie			

Funkcje i procedury obsługujące BD

Funkcja wyświetlająca imię i nazwisko influencera o podanym id:

```
CREATE FUNCTION getInfluencerFullName(id INT)
RETURNS VARCHAR(255)
DETERMINISTIC
BEGIN
    DECLARE fullName VARCHAR(255);
    SELECT CONCAT(first_name, ' ', last_name) INTO fullName
    FROM influencers
    WHERE influencer_id = id; RETURN fullName;
END;
```

Użycie:

✓ Pokazano wiersze 0 - 0 (1 ogółem, Wykonanie zapytania trwało 0,0004 sekund(y).)

```
SELECT getInfluencerFullName(4);
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edytuj](#)] [[Wyjaśnij SQL](#)] [[Utwórz kod PHP](#)] [[Odśwież](#)]

☐ Pokaż wszystko

Liczba wierszy: 25 ▼

Filtrowanie wierszy:

+ Opcje

getInfluencerFullName(4)

Bartek Nowak

Procedura zmieniająca wartość statusu kampanii o podanym id:

```
CREATE PROCEDURE completeCampaign(IN id INT)
BEGIN
    UPDATE campaigns SET status = 'Completed' WHERE campaign_id = id;
END;
```

Użycie:

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0006 sekund(y).)

```
CALL completeCampaign(1);
```

[[Edytuj w linii](#)] [[Edytuj](#)] [[Utwórz kod PHP](#)]

Procedura dodająca użytkownika

AddUser

Szczegóły

Nazwa procedury

AddUser

Typ

PROCEDURE

Parametry

	Kierunek	Nazwa	Typ	Długość/Wartości	Opcje
‡	IN	p_username	VARCHAR	100	Ko Usun
‡	IN	p_password	VARCHAR	255	Ko Usun
‡	IN	p_email	VARCHAR	320	Ko Usun
‡	IN	p_role	ENUM	'admin','manager'	Ko Usun
‡	IN	p_icon	VARCHAR	255	Ko Usun

Dodaj parametr

Określenie

```
1 BEGIN
2   INSERT INTO users(username,password_hash,email,role,created_at,icon)
3   VALUES(p_username,p_password_hash,p_email,p_role,NOW(),p_icon);
4 END
```

Jest deterministyczna

☐

Dostosuj uprawnienia

☒

Definiujący

`root`@`localhost`

Rodzaj zabezpieczenia

DEFINER

Dostęp SQL do danych

CONTAINS SQL

Komentarz

Wykonaj

Zamknij

Procedura dodająca Platformę

AddPlatform

Szczegóły

Nazwa procedury

AddPlatform

Typ

PROCEDURE

Parametry

	Kierunek	Nazwa	Typ	Długość/Wartości	Opcje
‡	IN	p_name	VARC	50	Ko Usun
‡	IN	p_url_tem	VARC	255	Ko Usun
‡	IN	p_descrip	TEXT	---	Ko Usun

Dodaj parametr

1 BEGIN

2 INSERT INTO platforms(name,url_template,description)

3 VALUES(p_name,p_url_template,p_description);

4 END

Określenie

Jest deterministyczna

☐

Dostosuj uprawnienia

☒

Definiujący

`root` @ `localhost`

Rodzaj zabezpieczenia

DEFINER

Dostęp SQL do danych

CONTAINS SQL

Komentarz

Wykonaj

Zamknij

Procedura dodająca Płatność

AddPayment

Szczegóły

Nazwa procedury

AddPayment

Typ

PROCEDURE

Parametry

	Kierunek	Nazwa	Typ	Długość/Wartości	Opcje
‡	IN	p_influencer_	INT		Usuń
‡	IN	p_campaign_	INT		Usuń
‡	IN	p_amount	DECIMAL	10,2	Usuń
‡	IN	p_payment_	DATE	---	Usuń
‡	IN	p_status	ENUM	'Pending','Paid','Fai	Ko Usuń

Dodaj parametr

Określenie

```
1 BEGIN
2   INSERT INTO payments(influencer_id,campaign_id,amount,payment_date,status)
3   VALUES(p_influencer_id,p_campaign_id,p_amount,p_payment_date,p_status);
4 END
```

Jest deterministyczna

☐

Dostosuj uprawnienia

☒

Definiujący

`root`@`localhost`

Rodzaj zabezpieczenia

DEFINER

Dostęp SQL do danych

CONTAINS SQL

Komentarz

Wykonaj

Zamknij

Procedura łącząca influencer z platformami

AddInfluencerPlatform

Szczegóły

Nazwa procedury

AddInfluencerPlatform

Typ

PROCEDURE

Parametry

	Kierunek	Nazwa	Typ	Długość/Wartości	Opcje
‡	IN	p_influencer_id	INT		Usun
‡	IN	p_platform_id	INT		Usun
‡	IN	p_account_url	VARCHAR	255	Ko Usun
‡	IN	p_platform_userr	VARCHAR	255	Ko Usun
‡	IN	p_joined_date	DATE	---	Usun
‡	IN	p_follows	BIGINT		Usun

Dodaj parametr

```
1 BEGIN
2   INSERT INTO
3   influencer_platforms (influencer_id,platform_id,account_url,platform_username,joined_date,follows)
4   VALUES (p_influencer_id,p_platform_id,p_account_url,p_platform_username,p_joined_date,p_follows);
END
```

Określenie

Jest deterministyczna

☐

Dostosuj uprawnienia

☒

Definiujący

`root`@`localhost`

Rodzaj zabezpieczenia

DEFINER

Dostęp SQL do danych

CONTAINS SQL

Komentarz

Wykonaj

Zamknij

Procedura dodająca influencera

AddInfluencer

Szczegóły

Nazwa procedury
AddInfluencer

Typ
PROCEDURE

	Kierunek	Nazwa	Typ	Długość/Wartości	Opcje
‡	IN	p_first_name	VARCHAR	100	Ko Usuń
‡	IN	p_last_name	VARCHAR	100	Ko Usuń
‡	IN	p_email	VARCHAR	320	Ko Usuń
‡	IN	p_phone	VARCHAR	50	Ko Usuń
‡	IN	p_engagement	DECIMAL	5,2	Ko Usuń
‡	IN	p_notes	TEXT	---	Ko Usuń
‡	IN	p_user_id	INT		Ko Usuń

Dodaj parametr

```

1 BEGIN
2   INSERT INTO
3     influencers(first_name,last_name,email,phone,engagement_rate,notes,user_id)
4     VALUES(p_first_name,p_last_name,p_email,p_phone,p_engagement_rate,p_notes,p_user_id);
5 END

```

Określenie

Jest deterministyczna
☐

Dostosuj uprawnienia
☒

Definiujący
`root`@`localhost`

Rodzaj zabezpieczenia
DEFINER

Dostęp SQL do danych
CONTAINS SQL

Wykonaj
Zamknij

Procedura łącząca influencera z kampanią

AddCampaignInfluencer

Szczegóły

Nazwa procedury

AddCampaignInfluencer

Typ

PROCEDURE

	Kierunek	Nazwa	Typ	Długość/Wartości	Opcje
‡	IN	p_campaign_id	INT		Usun
‡	IN	p_influencer_id	INT		Usun
‡	IN	p_agreed_fee	DECIMAL	10,2	Usun
‡	IN	p_contract_signed	TINYINT	1	Usun
‡	IN	p_notes	TEXT	---	Ko Usun

Dodaj parametr

Określenie

```
1 BEGIN
2   INSERT INTO
  campaign_influencers(campaign_id,influencer_id,agreed_fee,contract_signed,notes)
3   VALUES(p_campaign_id,p_influencer_id,p_agreed_fee,p_contract_signed,p_notes);
4 END
```

Jest deterministyczna

☐

Dostosuj uprawnienia

☒

Definiujący

`root`@`localhost`

Rodzaj zabezpieczenia

DEFINER

Dostęp SQL do danych

CONTAINS SQL

Komentarz

Wykonaj

Zamknij

Procedura dodająca raport efektów kampanii

AddCampaignEffect

Szczegóły

Nazwa procedury

AddCampaignEffect

Typ

PROCEDURE

Parametry

	Kierunek	Nazwa	Typ	Długość/Wartości	Opcje
‡	IN	p_campaign_id	INT		Usun
‡	IN	p_influencer_id	INT		Usun
‡	IN	p_views	INT		Usun
‡	IN	p_likes	INT		Usun
‡	IN	p_comments	INT		Usun
‡	IN	p_shares	INT		Usun
‡	IN	p_report_date	DATE	---	Usun

Dodaj parametr

Określenie

```
1 BEGIN
2   INSERT INTO
  campaign_effects(campaign_id,influencer_id,views,likes,comments,shares,report_date)
3   VALUES(p_campaign_id,p_influencer_id,p_views,p_likes,p_comments,p_shares,p_report_date);
4 END
```

Jest deterministyczna

☐

Dostosuj uprawnienia

☒

Definiujący

`root` @ `localhost`

Rodzaj zabezpieczenia

DEFINER

Dostęp SQL do danych

CONTAINS SQL

Wykonaj

Zamknij

Procedura dodająca kampanię

AddCampaign

Szczegóły

Nazwa procedury

AddCampaign

Typ

PROCEDURE

Parametry

	Kierunek	Nazwa	Typ	Długość/Wartości	Opcje
‡	IN	p_name	VARCHAR	150	Ko Usuń
‡	IN	p_client	VARCHAR	150	Ko Usuń
‡	IN	p_start_date	DATE	---	--- Usuń
‡	IN	p_end_date	DATE	---	--- Usuń
‡	IN	p_budget	DECIMAL	12,2	Usuń
‡	IN	p_goal_description	TEXT	---	Ko Usuń
‡	IN	p_status	ENUM	'Planned','Ongoing'	Ko Usuń
‡	IN	p_description	TEXT	---	Ko Usuń

Dodaj parametr

1 BEGIN

2 INSERT INTO campaigns(name,client,start_date,end_date,budget,goal_description,status,description)

3 VALUES(p_name,p_client,p_start_date,p_end_date,p_budget,p_goal_description,p_status,p_description);

4 END

Określenie

Jest deterministyczna

☐

Dostosuj uprawnienia

☒

Definiujący

`root`@`localhost`

Rodzaj zabezpieczenia

DEFINER

Wykonaj

Zamknij

Zarządzanie bazą danych (backup, restore)

Eksport danych:

Eksportowanie tabeli z bazy "influencer_campaign_manager_db"

Metoda eksportu:

- ☒ Szybko - wyświetlane są tylko minimalne opcje
- ☐ Dostosuj - wyświetli wszystkie możliwe opcje

Format:

SQL

Wykonaj

Import Danych:

Importowanie do bazy danych "influencer_campaign_manager_db"

Plik do importu:

Plik może być skompresowany (gzip, bzip2, zip) bądź nie.

Nazwa skompresowanego pliku musi kończyć się na `[format].[compression]`. Przykład: `.sql.zip`

Wyszukaj w komputerze: influencer_c...er_db (1).sql (Maksymalny rozmiar: 40MB)

Możesz także przeciągnąć i upuścić plik na dowolnej stronie.

Kodowanie znaków pliku: utf-8

Częściowy import:

- ☒ Zezwalaj na przerwanie importu w przypadku, gdy skrypt wykryje, że zbliża się limit czasu PHP. (Może to być dobry sposób importowania dużych plików, jednak może zepsuć transakcje.)

Pomnij tę liczbę zapytań (dla SQL), zaczynając od pierwszej: 0

Inne opcje:

- ☒ Włącz sprawdzanie kluczy obcych

Format:

SQL

Specyficzne opcje formatu:

Tryb zgodności SQL:

NONE

- ☒ Nie używaj AUTO_INCREMENT dla wartości zerowych

Wykonaj

Wariant testowy i na produkcji

W wariancie testowym dane są wymyślane by sprawdzić czy baza danych działa poprawnie. W tym wariancie można bezpiecznie testować zapytania, funkcje, procedury i relacje. W wariancie produkcyjnym baza zawiera już prawdziwe dane które są używane przez użytkowników danego systemu. Musi ona być zabezpieczona tak aby dane nie wyciekły na zewnątrz.

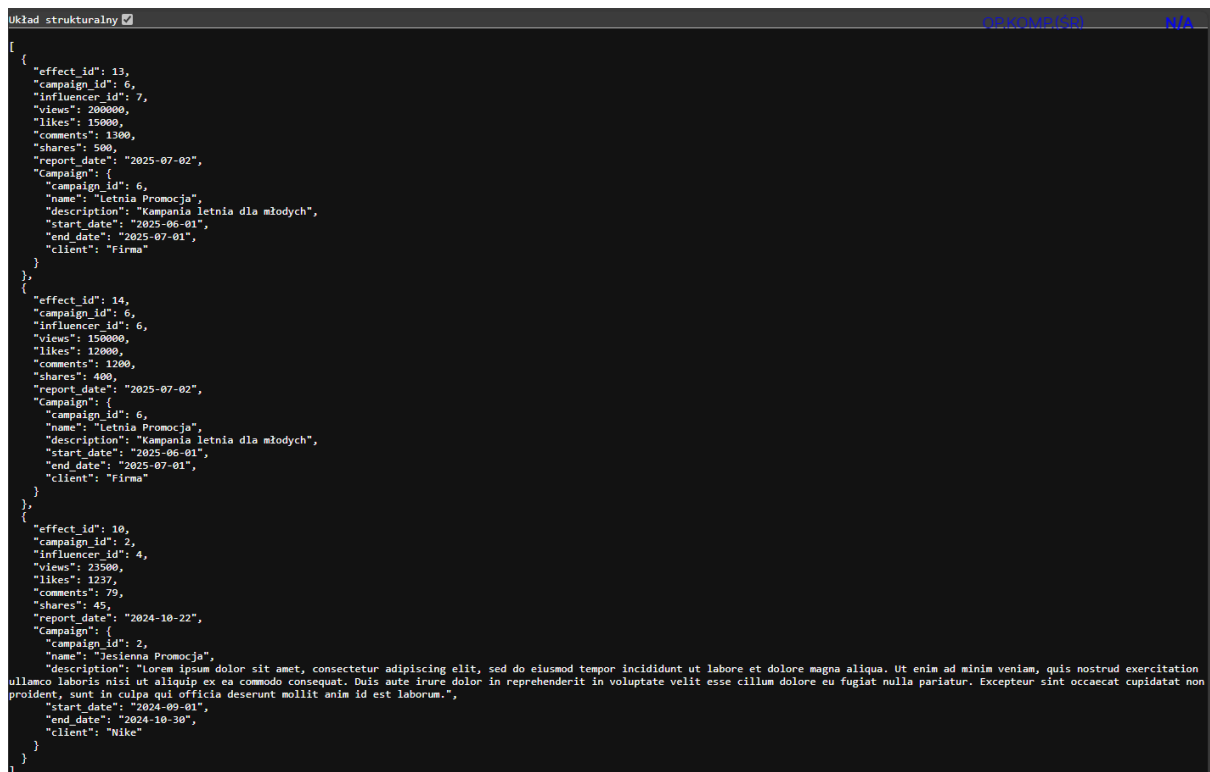
Komunikacja z językiem programowania (kod)

Stworzenie endpointu

```
app.get('/campaigns/top_campaigns', async (req, res) => {
  try {
    const topCampaigns = await CampaignEffect.findAll({
      include: [{
        model: Campaign,
        attributes: ['campaign_id', 'name', 'description',
          'start_date', 'end_date', 'client']
      }],
      order: [['views', 'DESC']],
      limit: 3
    });

    res.json(topCampaigns);
  } catch (err) {
    console.error('Invalid campaign:', err);
  }
});
```

localhost:3001/campaigns/top_campaigns



```
Układ strukturalny [x] 100% 100% 100%
[
  {
    "effect_id": 13,
    "campaign_id": 6,
    "influencer_id": 7,
    "views": 200000,
    "likes": 15000,
    "comments": 1300,
    "shares": 500,
    "report_date": "2025-07-02",
    "campaign": {
      "campaign_id": 6,
      "name": "Letnia Promocja",
      "description": "Kampania letnia dla młodych",
      "start_date": "2025-06-01",
      "end_date": "2025-07-01",
      "client": "Firma"
    }
  },
  {
    "effect_id": 14,
    "campaign_id": 6,
    "influencer_id": 6,
    "views": 150000,
    "likes": 12000,
    "comments": 1200,
    "shares": 400,
    "report_date": "2025-07-02",
    "campaign": {
      "campaign_id": 6,
      "name": "Letnia Promocja",
      "description": "Kampania letnia dla młodych",
      "start_date": "2025-06-01",
      "end_date": "2025-07-01",
      "client": "Firma"
    }
  },
  {
    "effect_id": 10,
    "campaign_id": 2,
    "influencer_id": 4,
    "views": 23500,
    "likes": 1227,
    "comments": 79,
    "shares": 45,
    "report_date": "2024-10-22",
    "campaign": {
      "campaign_id": 2,
      "name": "Jesienna Promocja",
      "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.",
      "start_date": "2024-09-01",
      "end_date": "2024-10-30",
      "client": "Nike"
    }
  }
]
```

Pobranie danych z serwera (na front)

```
export async function getTopCampaigns() {
  const res = await axios.get('http://localhost:3001/campaigns/top_campaigns');
  return res.data;
}
```

Przykład zastosowania ORM

Połączenie modułów (relacje)

```
1  const Sequelize = require('sequelize');
2  const sequelize = require('../config/database');
3  const InfluencerPlatform = require('./InfluencerPlatform');
4
5  const User = require('./User')(sequelize, Sequelize);
6  const Influencer = require('./Influencer')(sequelize, Sequelize);
7  const Campaign = require('./Campaign')(sequelize, Sequelize);
8  const CampaignInfluencer = require('./CampaignInfluencer')(sequelize, Sequelize);
9  const CampaignEffect = require('./CampaignEffect')(sequelize, Sequelize);
10 const Payment = require('./Payment')(sequelize, Sequelize);
11 const Platform = require('./Platform')(sequelize, Sequelize);
12 const InfluencerPlatform = require('./InfluencerPlatform')(sequelize, Sequelize);
13
14 User.hasOne(Influencer, { foreignKey: 'user_id' });
15 Influencer.belongsTo(User, { foreignKey: 'user_id' });
16
17 Campaign.hasMany(CampaignInfluencer, { foreignKey: 'campaign_id' });
18 CampaignInfluencer.belongsTo(Campaign, { foreignKey: 'campaign_id' });
19 Influencer.hasMany(CampaignInfluencer, { foreignKey: 'influencer_id' });
20 CampaignInfluencer.belongsTo(Influencer, { foreignKey: 'influencer_id' });
21
22 Campaign.hasMany(CampaignEffect, { foreignKey: 'campaign_id' });
23 CampaignEffect.belongsTo(Campaign, { foreignKey: 'campaign_id' });
24
25 Platform.hasMany(InfluencerPlatform, { foreignKey: 'platform_id' });
26 InfluencerPlatform.belongsTo(Platform, { foreignKey: 'platform_id' });
27
28 Influencer.hasMany(InfluencerPlatform, { foreignKey: 'influencer_id' });
29 InfluencerPlatform.belongsTo(Influencer, { foreignKey: 'influencer_id' });
30
31 Campaign.hasMany(Payment, { foreignKey: 'campaign_id' });
32 Payment.belongsTo(Campaign, { foreignKey: 'campaign_id' });
33 Influencer.hasMany(Payment, { foreignKey: 'influencer_id' });
34 Payment.belongsTo(Influencer, { foreignKey: 'influencer_id' });
35
36 module.exports = {
37   sequelize,
38   User,
39   Influencer,
40   Campaign,
41   CampaignInfluencer,
42   CampaignEffect,
43   Payment,
44   Platform,
45   InfluencerPlatform
46 };

```

Przykładowy moduł ORM

```
module.exports = (sequelize, DataTypes) => {  
  return sequelize.define('Campaign', {  
    campaign_id: { type: DataTypes.INTEGER, autoIncrement: true, primaryKey: true },  
    name: { type: DataTypes.STRING(150), allowNull: false },  
    client: { type: DataTypes.STRING(150), allowNull: false },  
    start_date: { type: DataTypes.DATEONLY, allowNull: false },  
    end_date: { type: DataTypes.DATEONLY, allowNull: false },  
    budget: { type: DataTypes.DECIMAL(12,2), allowNull: false },  
    goal_description: { type: DataTypes.TEXT, allowNull: true },  
    status: { type: DataTypes.ENUM('Planned', 'Ongoing', 'Completed', 'Cancelled'), allowNull: false },  
    description: { type: DataTypes.TEXT, allowNull: false },  
  }, {  
    tableName: 'campaigns',  
    timestamps: false  
  });  
};
```