Dokumentacja Projektu

Zespół Z01, łączony z PAP

1 Cel Projektu

Celem projektu jest stworzenie systemu bazy danych oraz aplikacji. Projekt obejmuje zaprojektowanie bazy danych, implementację aplikacji, oraz testowanie systemu.

1.1 Skład zespołu

- Dominik Śledziewski 331447
- Jan Szwagierczak 331444
- Piotr Szkoda 331442
- Tomasz Okoń 331414

1.2 Zakres Projektu

Projekt obejmuje:

- Projekt bazy danych: model ER, model relacyjny
- Skrypty DDL do stworzenia schematu bazy danych
- Skrypty do załadowania danych
- Definicje sekwencji, wyzwalaczy, procedur, funkcji
- Skrypty testujące bazę danych
- Aplikację w języku Java (używając JDBC/JPA)

2 Opis Aplikacji

Aplikacja **YapYap** to platforma społecznościowa, umożliwiająca użytkownikom tworzenie, wstawianie, lajkowanie postów, wysyłanie wiadomości tekstowych, tworzenie list znajomych czy wieloosobowych czatów. Wymiana wiadomości jest zrealizowana za pomocą WebSocket, zapewniając ich natychmiastowe przesyłanie i odbieranie.

2.1 Funkcjonalności

- Rejestracja, logowanie
- Edytowanie nazwy użytkownika
- Tworzenie i zarządzanie konwersacjami (indywidualnymi i grupowymi)
- Wysyłanie i odbieranie wiadomości
- Reakcje na wiadomości i posty
- Postowanie i komentowanie
- Zarządzanie listą znajomych

3 Stack Technologiczny

3.1 Frontend

- Framework: React.ts
- Stan aplikacji: Context API, React Hooks
- WebSocket: STOMP over SockJS client
- HTTP Client: Axios

3.2 Backend

- Framework: Spring Boot (Java)
- Bezpieczeństwo: Spring Security z JWT
- WebSocket: STOMP

3.3 Baza Danych

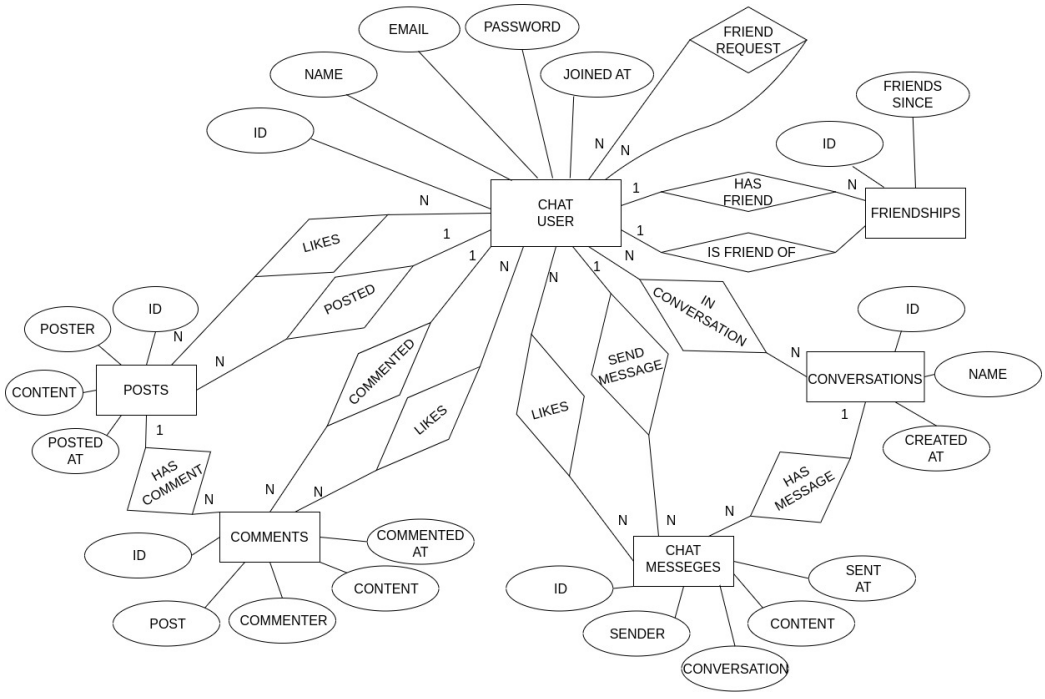
- Oracle Database
- ORM: JPA/Hibernate
- Połączenie: JDBC

3.4 Narzędzia Deweloperskie

- Docker, Git
- CI/CD: GitLab, DigitalOcean

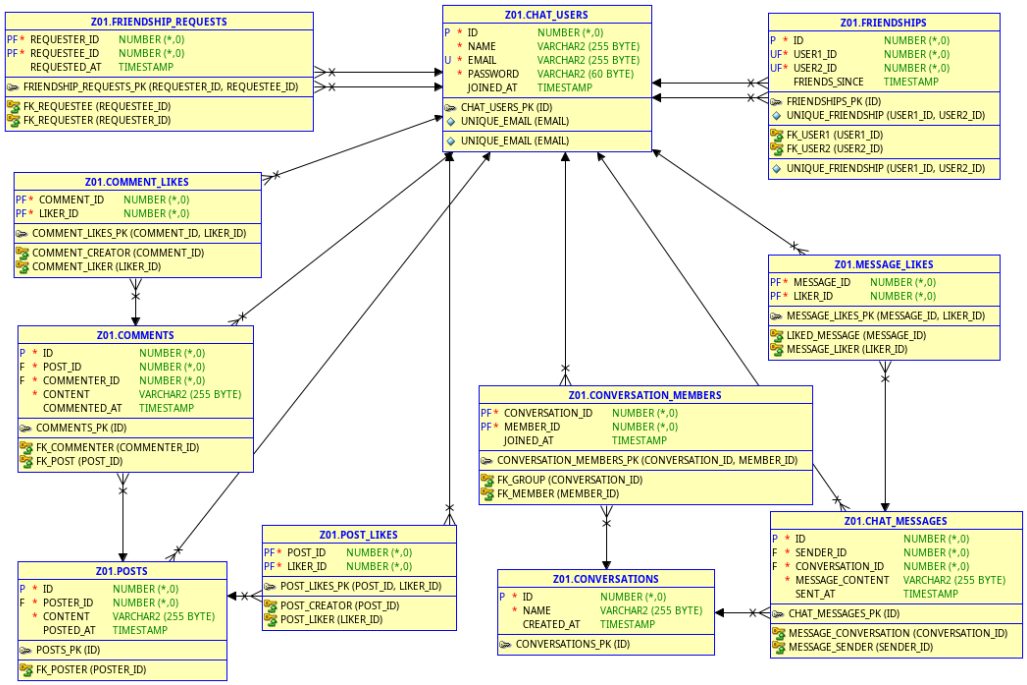
4 Baza danych

4.1 Model ER



Rysunek 1: Model ER bazy danych

4.2 Model relacyjny



Rysunek 2: Relacyjny model bazy danych

5 Zadania Bazodanowe

Wszystkie i pełne skrypty bazodanowe znajdują się w projekcie w folderze **DB**. Są to:

- **DB/00_drop_tables.sql**
- **DB/01_database.sql**
- **DB/Functions.sql**
- **DB/Procedures.sql**
- **DB/Tests.sql**
- **DB/Triggers.sql**
- **DB/delete_data.sql**

Poniżej zostaną przedstawione wybrane z nich fragmenty.

5.1 Tworzenie Tabel

Wybrany przykładowy skrypt do tworzenia tabel w bazie danych:

```
CREATE TABLE CHAT_USERS (
    ID INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
                                                    PRIMARY KEY NOT NULL,
    NAME VARCHAR2(255) NOT NULL,
    EMAIL VARCHAR2(255) NOT NULL,
    PASSWORD VARCHAR2(60) NOT NULL,
    JOINED_AT TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT email_format CHECK (EMAIL LIKE '%@%.%'),
    CONSTRAINT unique_email UNIQUE (EMAIL)
);
```

5.2 Procedury i Funkcje

Przykład procedury obliczającej aktywność użytkownika:

```
CREATE OR REPLACE FUNCTION calculate_user_activity(p_user_id IN NUMBER)
    RETURN NUMBER IS
    v_posts_count NUMBER;
```

```
    v_comments_count NUMBER;
    v_messages_count NUMBER;
    v_activity_score NUMBER;
BEGIN
    SELECT COUNT(*) ..... /* logika */

    v_activity_score := ......

    RETURN ROUND(v_activity_score, 2);
END;
```

5.3 Skrypty Testowe

Przykład skryptów testowych sprawdzających działanie zaprojektowanej bazy danych.

Testowanie Aktywności Użytkownika

```
WITH user_activity AS (
    SELECT
        cu.ID,
        cu.NAME,
        calculate_user_activity(cu.ID) as activity_score,
        COUNT(DISTINCT  .......
    FROM CHAT_USERS cu
             LEFT JOIN FRIENDSHIPS f1 ON cu.ID = f1.USER1_ID
             LEFT JOIN ........
    GROUP BY cu.ID, cu.NAME
)
SELECT ........
    RANK() OVER (ORDER BY activity_score DESC) as activity_rank
FROM user_activity
WHERE activity_score > 0
ORDER BY activity_score DESC;
```

6 Analiza rozwiązania

6.1 Mocne strony projektu

- **Normalizacja bazy danych:** Struktura bazy danych jest dobrze znormalizowana, co pozwala uniknąć redundancji i zapewnia spójność danych. Tabele są zaprojektowane zgodnie z zasadą pierwszej, drugiej i trzeciej postaci normalnej (1NF, 2NF, 3NF), co gwarantuje eliminację zbędnych powtórzeń.

- **Integralność referencyjna:** Wszystkie tabele zawierają klucze obce, które zapewniają spójność danych. Zastosowanie klauzul `ON DELETE CASCADE` pozwala na automatyczne usuwanie powiązanych danych, co zapewnia integralność bazy.

- **Bezpieczeństwo danych:** Zastosowanie ograniczeń, takich jak `CHECK` (np. na format e-maila w tabeli `CHAT_USERS`) oraz `UNIQUE`, zapewnia poprawność danych wprowadzanych do bazy.

- **Elastyczność w obsłudze postów i komentarzy:** Struktura tabel `POSTS`, `COMMENTS`, `POST_LIKES` i `COMMENT_LIKES` zapewnia pełną elastyczność w zakresie interakcji użytkowników z postami i komentarzami, umożliwiając łatwe rozbudowywanie aplikacji o nowe funkcjonalności, takie jak dodatkowe reakcje na posty.

- **Optymalizacja zapytań:** Zastosowanie tabel pośrednich (np. `MESSAGE_LIKES`, `CONVERSATION_MEMBERS`) pozwala na wydajniejsze przetwarzanie zapytań dotyczących wielu do wielu relacji, co jest korzystne w kontekście skalowalności aplikacji.

6.2 Obszary do poprawy

- **Brak tabeli audytu:** W projektowanej bazie danych brakuje tabeli audytu, która śledziłaby operacje na danych (np. kto i kiedy edytował posty, wiadomości czy inne istotne dane). Tego typu tabela mogłaby poprawić bezpieczeństwo i transparentność aplikacji.

- **Brak obsługi wersji danych:** W przypadku aplikacji społecznościowych warto rozważyć mechanizm wersjonowania danych (np. w tabeli `POSTS` czy `COMMENTS`), szczególnie w kontekście edytowania postów lub komentarzy. Dzięki temu użytkownicy mogliby śledzić historię zmian i odzyskiwać poprzednie wersje.

- **Skalowalność bazy danych:** W przypadku dużych danych (np. w przypadku rosnącej liczby postów i komentarzy) może pojawić się problem z wydajnością. Dobrą praktyką byłoby wprowadzenie archiwizacji starszych danych dotyczących np. `POSTS`, `COMMENTS` czy `CHAT_MESSAGES`.
- **Brak tabeli do zarządzania rolami i uprawnieniami:** Aplikacja umożliwia użytkownikom tworzenie postów, wiadomości i zarządzanie znajomymi. Warto byłoby rozważyć dodanie tabeli, która będzie przechowywać role użytkowników (np. administrator, moderator, użytkownik), co umożliwi elastyczne zarządzanie uprawnieniami w aplikacji.
- **Brak obsługi multimediów:** Projekt bazy danych nie uwzględnia możliwości przechowywania plików multimedialnych (np. zdjęć, filmów) powiązanych z postami czy wiadomościami.

6.3 Podsumowanie

Proponowany projekt bazy danych jest dobrym rozwiązaniem, które spełnia większość wymagań aplikacji społecznościowej. Niemniej jednak, istnieje pole do jej poprawy. Wprowadzenie proponowanych poprawek mogłoby poprawić wydajność bazy oraz jej elastyczność w miarę rozwoju aplikacji.