Projekt bazy do gry komputerowej podobnej do wordle lub literalnie.fun

Autor: Jakub Surdej

Nr albumu: 14009

Grupa laboratoryjna: 21

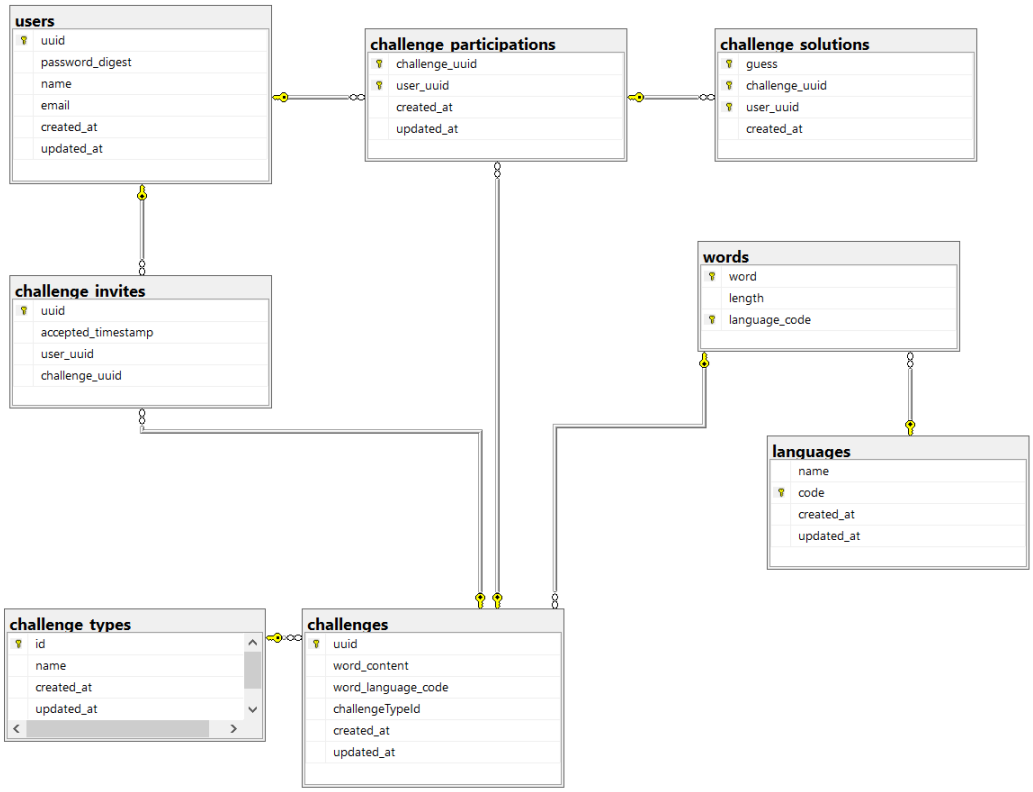
Opis bazy danych

Zaprojektowana baza danych służy do przetrzymywania danych aplikacji podobnej do wymienionej w tytule gier. Baza zawiera funkcjonalności takie jak dodawanie słów, gier, użytkowników, czy zaproszeń. Gra polega na odgadnięciu pseudolosowo wybranego słowa.

Wykaz tabel

Baza składa się z 8 tabel:

1. users
2. languages
3. words
4. challenges
5. challenge_types
6. challenge_participations
7. challenge_solutions
8. challenge_invites



Wykaz widoków

random

Widok stworzony do pobierania pseudolosowej wartości hexadecymalnej w funkcjach (użycie funkcji w funkcji jest często problematyczne bez tego).

```
CREATE VIEW [dbo].[random]
AS
    SELECT CRYPT_GEN_RANDOM(4) AS random_value;
```

Przykład użycia:

```
SELECT * FROM random;
```

scoreboard

Widok wyświetlający punktację dla użytkowników (za punkt uznawane jest prawidłowe zgadnięcie słowa)

```
CREATE VIEW [dbo].[scoreboard] AS
    SELECT COUNT(*) as correct_answers, [dbo].[challenge_participations].[user_uuid]
```

```
FROM [dbo].[challenge_participations]
    LEFT JOIN [dbo].[challenges] ON [dbo].[challenge_participations].
[challenge_uuid] = [dbo].[challenges].[uuid]
    LEFT JOIN [dbo].[challenge_solutions] ON [dbo].[challenges].[word_content] =
[dbo].[challenge_solutions].[guess] GROUP BY [dbo].[challenge_participations].
[user_uuid];
```

Przykład użycia:

```
SELECT * FROM scoreboard ORDER BY correct_answers DESC;
```

popular_words

Widok wyświetlający słowa wraz z ilością ich użycia w wyzwaniach

```
CREATE VIEW [dbo].[popular_words] AS
    SELECT [dbo].[challenges].[word_content], [dbo].[languages].[name], COUNT(*) as
[count] FROM challenges
    LEFT JOIN [dbo].[languages] ON [dbo].[challenges].[word_language_code] = [dbo].
[languages].[code]
    GROUP BY [dbo].[languages].[name], [dbo].[challenges].[word_content];
```

Przykład użycia:

```
SELECT * FROM popular_words ORDER BY count DESC;
```

Wykaz funkcji

get_random_word

Zwraca pseudolosowe słowo dla danego języka. Jako domyślny język używa angielski.

```
CREATE FUNCTION get_random_word
    (@lang NVARCHAR(3) = 'en')
RETURNS NVARCHAR(100)
AS
BEGIN
    DECLARE @word NVARCHAR(100)
    SET @word = (SELECT TOP 1 [word] FROM [dbo].[words] WHERE [language_code] =
@lang ORDER BY (SELECT TOP 1 * FROM [dbo].[random]))
    RETURN @word
END;
```

Przykład użycia

```
SELECT [dbo].[get_random_word] (DEFAULT)
```

users_by_challenge_count

Zwraca uuid użytkowników wraz z ilością wyzwań, w których brali udział (dla danego języka- domyślny angielski)

```
CREATE FUNCTION users_by_challenge_count
    (@lang NVARCHAR(3) = 'en')
RETURNS TABLE
AS
RETURN
    SELECT [user_uuid], COUNT([challenge_participations].[challenge_uuid]) AS
[challenge_count]
    FROM [dbo].[users] RIGHT JOIN [dbo].[challenge_participations]
    ON [challenge_participations].[user_uuid] = [users].[uuid]
    LEFT JOIN [dbo].[challenges] ON [challenges].[uuid] = [challenge_uuid]
    WHERE [word_language_code] = @lang
    GROUP BY [user_uuid];
```

Przykład użycia

```
SELECT * FROM users_by_challenge_count('en') ORDER BY challenge_count DESC;
```

solutions_for_lang

Zwraca liczbę prób odgadnięcia prawidłowego hasła dla wszystkich (łącznie) wyzwań dla danego języka

```
CREATE FUNCTION solutions_for_langs ()
RETURNS TABLE
AS
RETURN
    SELECT [word_language_code], COUNT(*) as [count]
    FROM [dbo].[challenge_solutions]
    LEFT JOIN [dbo].[challenges]
    ON [challenges].[uuid] = [challenge_solutions].[challenge_uuid]
    GROUP BY [word_language_code];
```

Przykład użycia:

```
SELECT * FROM solutions_for_langs();
```

Wykaz procedur

insert_word

Insertuje słowo do bazy danych. Automatycznie dodaje jego długość.

```
CREATE PROCEDURE [dbo].[insert_word]
    @language_code NVARCHAR(3),
    @word NVARCHAR(100)
AS
```

```
BEGIN
    INSERT INTO [dbo].[words] ([word], [language_code], [length])
    VALUES (@word, @language_code, LEN(@word))
END
```

Przykład użycia:

```
EXEC insert_word 'pl', 'zwięźle'
```

start_game

Tworzy nowe wyzwanie dla danego języka i typu, wybiera dla niego pseudolosowo słowo, zaprasza do udziału w nim 10 użytkowników.

```
CREATE PROCEDURE [dbo].[start_game]
    @language_code NVARCHAR(3),
    @type_id INT
AS
BEGIN
    BEGIN TRANSACTION;
    SAVE TRANSACTION [start_game_transaction];

    DECLARE @word NVARCHAR(100)
    DECLARE @challenge_uuid NVARCHAR(36)
    DECLARE @random_players_uuids TABLE (uuid NVARCHAR(36))

    BEGIN TRY
        SET @word = [dbo].[get_random_word] (@language_code);
        SET @challenge_uuid = NEWID()

        INSERT INTO [dbo].[challenges] ([uuid], [word_content],
[word_language_code], [challengeTypeId], [updated_at])
        VALUES (@challenge_uuid, @word, @language_code, @type_id, GETDATE());

        --- selecting random 10 players to invite
        INSERT INTO @random_players_uuids SELECT TOP 10 [uuid] FROM [dbo].[users]
ORDER BY CRYPT_GEN_RANDOM(4)

        INSERT INTO [dbo].[challenge_invites] ([uuid], [user_uuid],
[challenge_uuid])
        SELECT NEWID(), [uuid], @challenge_uuid FROM @random_players_uuids
    COMMIT TRANSACTION
    END TRY

    BEGIN CATCH
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION [start_game_transaction]; -- rollback to
start_game_transaction save point
       END
```

```
    END CATCH
END
```

Przykład użycia:

```
EXEC start_game 'pl', 1
```

drop_challenge_duplicates

Wyszukuje wyzwań, dla których powtarza się słowo i zostawia tylko najstarsze z nich. Nie ma praktycznego zastosowania.

```
CREATE PROCEDURE [drop_challenge_duplicates]
AS
BEGIN
WITH [duplicates] AS (
    SELECT ROW_NUMBER() OVER(PARTITION BY [word_content], [word_language_code] ORDER
BY [created_at] DESC)
    AS [rn]
    FROM [dbo].[challenges]
)
DELETE [duplicates] WHERE [rn] > 1
END;
```

Przykład użycia:

```
EXEC drop_challenge_duplicates
```

Przykładowe instrukcje

```
SELECT TOP 10 * FROM challenges WHERE [word_language_code] = 'en' ORDER BY
[created_at];
```

Zwraca 10 najnowszych wyzwań dla języka angielskiego.

```
SELECT [word_language_code], COUNT(*) as [count]
FROM [dbo].[challenge_solutions]
LEFT JOIN [dbo].[challenges]
ON [challenges].[uuid] = [challenge_solutions].[challenge_uuid]
WHERE [challenge_solutions].[guess] = [challenges].[word_content]
GROUP BY [word_language_code];
```

Zwraca liczbę poprawnych odpowiedzi dla każdego języka.

```
SELECT ROUTINE_NAME, ROUTINE_TYPE FROM INFORMATION_SCHEMA.ROUTINES
```

Zwraca listę dodanych funkcji i procedur.

```
SELECT * FROM users WHERE email LIKE '%k%';
```

Zwraca listę użytkowników, którzy mają literę `k` w adresie email.

```
SELECT TOP 11 user_uuid, COUNT(*) as participations_count
FROM users RIGHT JOIN challenge_participations
ON users.uuid = challenge_participations.user_uuid
GROUP BY user_uuid
ORDER BY participations_count ASC;
```

Zwraca 11 użytkowników z najmniejszą liczbą partycypacji w wyzwaniach.

```
SELECT TOP 5 *
FROM users_by_challenge_count('en') AS challenge_users
LEFT JOIN users ON challenge_users.user_uuid = users.uuid
ORDER BY challenge_count DESC;
```

Zwraca 5 użytkowników z nawiększą liczbą partycypacji w wyzwaniach w języku angielskim;

```
SELECT * FROM challenge_invites
WHERE accepted_timestamp < DATEADD(day, -1, CAST(GETDATE() AS date));
```

Zwraca wszystkie zaproszenia do gry zaakceptowane wcześniej niż wczoraj.

```
SELECT * FROM challenges WHERE word_content LIKE 'a%a';
```

Zwraca wszystkie wyzwania, w których słowo zaczyna i kończy się na `a`

```
SELECT session_id FROM sys.dm_exec_requests WHERE connection_id IS NOT NULL;
```

Zwraca identyfikator sesji dla transakcji wywołanych przez użytkownika (lub aplikację połączoną z bazą).

```
DECLARE @query_start bigint
DECLARE @query_end bigint
SELECT @query_start=(SELECT DATEDIFF_BIG(ms, '1970-01-01 00:00:00', GETUTCDATE()))

EXEC start_game 'pl', 1

SELECT @query_end=(SELECT DATEDIFF_BIG(ms, '1970-01-01 00:00:00', GETUTCDATE()))

SELECT @query_end - @query_start AS time_to_start_game_in_milliseconds
```

Zwraca czas tworzenia nowej gry (w milisekundach) (oraz tworzy nową grę)

Uruchamianie projektu

Projekt został napisany z pomocą narzędzia Prisma, oraz języka TypeScript, który pozwolił stowrzyć narzędzie zapełniające bazę przykładowymi danymi. Projekt zawiera definicję bazy danych w pliku docker-compose.yml, która pozwala uruchomić projekt bez konieczności instalowania bazy danych Microsoft SQL Server na systemie hosta komputera.

Projekt można uruchomić na 3 sposoby:

- za pomocą bazy Microsoft SQL Server zainstalowanej na maszynie hosta, oraz pliku zawierającego strukturę bazy danych, oraz zapełnionego przykładowymi wartościami
- za pomocą bazy Microsoft SQL Server zainstalowanej na maszynie hosta, oraz narzędzi Yarn oraz Node.js (rekomendowana wersja Node: 17.4.0)
- za pomocą bazy danych w kontenerze Docker, oraz ww. narzędzi

Uruchamianie bazy z pliku zrzutu (najszybszy, rekomendowany)

Sposób wymaga jedynie uruchomienia pliku 14009_jakub_surdej_gr_7.sql.

Uruchamianie bazy za pomocą Node

Sposób ten wymaga uzupełnienia connection stringa (w przypadku Dockera można posłużyć się komendą `cp .env.example .env`), oraz uruchomienia kolejno poleceń:

```
yarn
yarn deploy
yarn seed
```

Uruchamianie bazy za pomocą Node i Dockera

Wymaga uruchomienia bazy z pliku docker-compose za pomocą polecenia

```
docker-compose up -d
```

oraz wykonania poleceń z kroku wymienionego poprzednio.