BAZY DANYCH 2

Marek Kocik, Michał Pawlik

November 2024

Spis treści

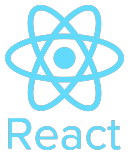
1 Wstęp

Projekt obejmuje stworzenie aplikacji webowej z frontendem w React i backendem opartym na Java Spring, z bazą danych PostgreSQL do przechowywania danych. Platforma korzysta z API Riot Games do pobierania statystyk i szczegółów rozgrywek, co pozwala na automatyczną aktualizację profili oraz bieżącą ocenę poziomu graczy.

2 Wybrane technologie

2.1 Frontend

React to popularna biblioteka JavaScript służąca do budowania nowoczesnych interface. Wybraliśmy React ze względu na jego dużą popularność dostępność gotowych komponentów oraz materaiałów edukacyjnych.



2.2 Backend

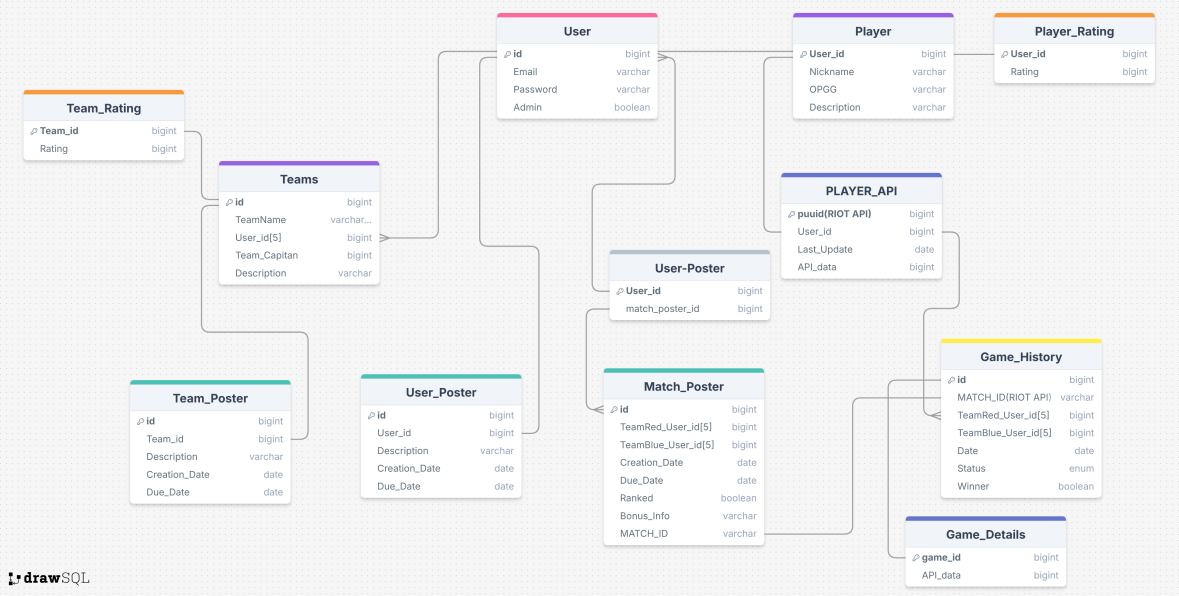
2.2.1 Do Backendu wybraliśmy język Java i framework Spring, które są stosowane w środowisku komercyjnym. Java i Spring oferują stabilność i są warte nauczenia przez ich typowo komercyjny charakter.

2.2.2 Hibernate to framework ORM (Object-relational Mapping), który ułatwia zarządzanie danymi i ich walidację danych w bazie. Wybraliśmy Hibernate, ponieważ jest popularnym wyborem jako orm w aplikacjach z springiem.

2.3 Baza danych

PostgreSQL to popularna baza danych o dużych możliwościach skalowania. Wybraliśmy PostgreSQL ze względu na jego elastyczność i wsparcie dla zaawansowanych zapytań SQL.

3 Projekt Bazy danych



3.1 **User** - Przechowuje informacje o użytkownikach, takie jak e-mail, hasło oraz poziom dostępu.

3.2 **Player** - Zawiera dane o graczach, w tym pseudonim, link do strony OPGG oraz opis.

3.3 **Teams** - Reprezentuje drużyny, z nazwą zespołu, kapitanem i opisem odnosi się również do użytkowników będących członkami zespołu.

3.4 **Team Rating** - Przechowuje oceny poszczególnych drużyn, gdzie każda drużyna ma przypisaną jedną ocenę.

3.5 **Player Rating** - Przechowuje indywidualne oceny graczy (użytkowników).

3.6 **PLAYER API** - Zawiera szczegółowe dane związane z kontem gracza pobrane z API RIOTu.

3.7 **Team Poster** - Umożliwia tworzenie ogłoszeń dla drużyn, z opisem, datą utworzenia i datą ważności.

3.8 **User Poster** - Pozwala użytkownikom na tworzenie ogłoszeń, z opisem, datą utworzenia i datą ważności.

3.9 **Match Poster** - Reprezentuje ogłoszenia związane z meczami, z informacją o drużynach, typem meczu, oraz dodatkowym opisem.

3.10 **Game History** - Przechowuje historię rozegranych meczów, uczestniczących graczy, datą oraz statusem i wynikiem meczu. Przechowuje też klucz do dzieki ktoremu można uzyskać bardziej szczegółowe informacje o meczy z API RIOTu.

3.11 **Game Details** - Zawiera szczegółowe informacje o grze pobrane z API RIOTu.

4 Możliwe transakcje w bazie danych

4.1 **Tworzenie gracza:** Transakcja tworzy nowego gracza platformy, tworząc nowy rekord w tabeli `Player`. Jednocześnie inicjalizuje go w innych tabelach związanych z graczem.

- **Operacje:**

4.1.1 Insert into Player - dodanie podstawowych danych na temat gracza.

4.1.2 Insert into Player_API - utworzenie pustego rekordu w tabeli.

4.1.3 Insert into Player_Rating - inicjalizacja pustego wpisu w tabeli ocen graczy.

4.2 **Usuwanie drużyny:** Proces usunięcia drużyny wymaga skasowania danych o drużynie w tabeli Teams, jej oceny w tabeli Team_Rating, i wszystkich związanych z nią ogłoszeń w tabeli Team_Poster.

- **Operacje:**

4.2.1 Delete from Teams - usunięcie rekordu z podstawowymi informacjami o drużynie.

4.2.2 Delete from Team_Rating - usunięcie rekordu z oceną drużyny.

4.2.3 Delete from Team_Poster - usunięcie ogłoszeń powiązanych z drużyną.

4.3 **Zakonczenie meczu i zapis wyniku:** Po zakończeniu meczu wynik jest zapisywany w tabeli Game_history a dodatkowe informacje są zapisywane w tabeli Game_Details. Jednocześnie oceny drużyn i graczy są aktualizowane .

- **Operacje:**

4.3.1 Update into Game_History - Zmiana statusu meczu na skonczony.

4.3.2 Insert into Game_Details - zapisanie szczegółowych informacji o meczu.

4.3.3 Update Player_Rating - aktualizacja oceny gracza na podstawie wyniku rozegranego meczu.

4.3.4 Update Team_Rating - aktualizacja oceny drużyny na podstawie wyniku rozegranego meczu.

4.4 **Dołaczenie konta Riot Games** Użytkownik może połączyć swoje konto na platformie z kontem serwisu RIOT games.

- **Operacje:**

4.4.1 Update Player - aktualizacja rekordu gracza.

4.4.2 Update Player_API - pobranie danych o graczu z serwera RIOT'u i zapisanie ich w naszej bazie danych.

4.5 **Rozpoczęcie meczu** Po określeniu gotowosci obu zespołów automatycznie tworzony jest mecz i usuwane jest ogłoszenie o meczu

- **Operacje:**

4.5.1 Delete Match_Poster - usuniecie ogloszenia o meczu.

4.5.2 Insert Game_history - dodanie rekordu meczu z statusem ongoing.

5 Prognoza charakteru encji

5.1 **User** - Użytkownicy są odczytywani przy logowaniu i autoryzacji w systemie. Aktualizacje są rzadsze, ponieważ zmiany danych konta odbywa się tylko na polecenie użytkownika.

5.2 **Player** - Profil gracza jest często pobierany, gdy inni użytkownicy sprawdzają jego statystyki, np. wyniki czy opis. Dodawanie i aktualizacja danych są rzadsze, bo maja miejsce tylko na polecenie użytkownika.

5.3 **Player_API** - Dane zewnętrznego API RIOT są często odczytywane i aktualizowane, aby na bieżąco synchronizować statystyki i zapewniać ich aktualność dla graczy.

5.4 **Teams** - Dane drużyn są często pobierane, szczególnie gdy gracze przeglądają składy zespołów. Modyfikacje są rzadkie, ponieważ drużyny zmieniają się stosunkowo rzadko, a dane o nich są stabilne.

5.5 **Posters** - Ogłoszenia użytkowników są często przeglądane przez innych graczy szukających partnerów do gry. Tworzenie i usuwanie ogłoszeń występuje regularnie, szczegolnie w przypadku meczy ktorych uzytkownicy moga rozgrywać kilka dziennie.

5.6 **Game_History / Game_Detail** - Historia rozegranych meczów jest często pobierana do celów statystycznych i analitycznych, dlatego odczyt jest intensywny, a aktualizacje praktycznie nie występują.

5.7 **Ratings** - Oceny graczy i zespołów są intensywnie wykorzystywane do systemów rankingowych i porównań, co generuje częsty odczyt. Aktualizacje sa równie częste związane z wynikami końcowymi meczy rankingowych.

Tabela 1: Prognoza charakteru encji

Encja	Liczba wierszy	Częstotliwość operacji			Obciążenie odczytem	Obciążenie zapisem	Łączne obciążenie
		Select	Insert/Delete	Update			
User	5	4	3	2	20	12,5	32,5
Player	5	6	3	2	30	12,5	42,5
Player_API	5	5	3	3	25	20	45
Teams	4	6	3	3	24	12	36
User_Poster	3	5	4	2	15	9	24
Team_Poster	3	5	3	2	15	7,5	22,5
Match_Poster	4	6	7	2	24	18	42
Game_History	7	4	7	1	28	28	56
Game_Details	7	4	7	1	28	28	56
User_Rating	5	7	3	7	35	25	60
Team_Rating	4	7	3	7	28	20	48

Tabela 2: Legenda

	częstość	liczba
1	prawie nigdy	kilka
2	bardzo rzadko	kilkanaście
3	rzadko	kilkadziesiąt
4	średnio	kilkaset
5	często	kilka tysięcy
6	bardzo często	kilkadziesiąt tysięcy
7	prawie ciągle	kilkaset tysięcy