BoardBase Case Study

Zaawansowane systemy baz danych - Projekt Etap 1

Krzysztof Dąbrowski 293101

Spis rycin

Spis tabel

> **Kod projektu**
>
> Kod projektu i sprawozdań jest dostępny na GitHub BoardBase

1 Koncepcja Systemu

Wielu miłośników gier planszowych chciałoby wiedzieć jakie gry planszowe są dostępne na rynku czy w jakich wersjach zostały wydane czy móc łatwo sprawdzić inne informacje o grach planszowych.
Aplikacja bazodanowa *BoardBase* pomoże rozwiązać te problemy.

W ramach aplikacji *BoardBase* będą przechowywane dane o grach planszowych, ich kategorii, mechanikach, ocenach graczy oraz rozgrywanych partiach. Dzięki temu gracze będą mogli wyszukać informacje o interesujących ich tytułach. Gracze będą mogli dzielić się swoimi ocenami gier oraz recenzjami. Na tej podstawie generowane będą rankingi najciekawszych gier.

Użytkownicy będą mogli monitorować swoją kolekcję gier oraz rozgrywane partie oraz interesujące ich gry, których zakup rozważają.

Administratorzy będą dodawać nowe gry do katalogu.

Dzięki tym funkcjom aplikacja *BoardBase* ułatwi znajdowanie interesujących graczy gier oraz dowiadywanie się więcej na temat ich ulubionych tytułów.

1.1 Dane przechowywane w bazie

Główne informacje, które będą przechowywane.

- Informacje o grach planszowych
- Kategorie i mechaniki gier w postaci słowników
- Dane użytkowników
- Oceny gier
- Recenzje gier
- Kolekcje gier danych użytkowników
- Lisy życzeń użytkowników
- Dziennik partii gier użytkowników

1.2 Użytkownicy systemu

Role dostępne w systemie.

- Gość: przegląda katalog i recenzje
- Użytkownik: ocenia gry, pisze recenzje, prowadzi kolekcję i dziennik partii
- Administrator: zarządza danymi gier, kategoriami

1.3 Usługi udostępniane użytkownikom

Przypadki użycia systemu.

- Wyszukiwanie gier w katalogu
- Oglądanie rankingów gier
- Wyszukanie podobnych tytułów do wskazanej gry
- Podgląd szczegółowych informacji o grze
- Ocena gry w postaci gwiazdek lub punktów
- Napisanie recenzji gry
- Katalogowanie posiadanych gier
- Rejestracja rozegranych partii
- Uwierzytelnianie: rejestracja/logowanie
- Zapisywanie gier na listę życzeń

2 Wybrany System zarządzania bazami danych (SZBD)

Słyszałem, że SZBD Postgres pozwala na instalowanie wielu modułów, które znacząco rozwijają jego działanie [1]. Możliwe, że nie będę korzystał z nich w tym projekcie, ale mimo tego uznałem to za bardzo ciekawe. Z tego powodu wybieram SZBD **Postgres**.
Dodatkowo nie używałem wcześniej Postgresa i chcę się go nauczyć.

2.1 Instalacja

Zastanawiałem się czy zainstalować Postgress bezpośrednio na komputerze, czy uruchomić go w kontenerze Docker. Skłaniałem się w strone kontenera, ponieważ instalacja wydaje sie znacznie prostsza. Planuję też uruchamiać mój projekt na komputerze z systemem Windows i Mac OS, co wydaje się również łatwiejsze z użyciem kontenera. Nie byłem pewny czy zastosowanie kontenera jest dobrą praktyką w środowiskach produkcyjnych, ale przeczytanie dyskusji Why not run production postgres in docker? [2] rozwiało moje wątpliwości.

2.2 Konfiguracja

W celu skonfigurowania SZBD postanowiłem dostosować przykładowy plik konfiguracyjny. Zgodnie z oficjalną instrukcją [3] i dokumentacją, do której odnosi [4], skopiowałem plik `/usr/share/postgresql/postgresql.conf.sample` z kontenera do mojego projektu i dostosowałem go do moich potrzeb.

Nie mam jeszcze zbyt dużej wiedzy o konfiguracji SZBD Postgres, więc zostawiłem większość ustawień na wartościach domyślnych.

Pozostawiłem odkomentowane ustawienia

```
listen_addresses = '*'
autovacuum_worker_slots = 16
```

oraz ustawiłem kilka ustawień związanych z logowaniem, żeby mieć więcej informacji o działaniu SZBD i potencjalnych problemach. Lekko zwiększyłem też dostępną pamięć, ponieważ mam zapas RAMu na komputerze, a oryginalne ustawienia wydawały mi się niskie. Zmieniłem też nazwę pliku logów na granularność na poziomie dnia, ponieważ nie będę potrzebował większej ilości plików logów.

```
shared_buffers = 256MB
work_mem = 8MB
log_filename = 'postgresql-%Y-%m-%d.log'
log_duration = on
log_statement = 'all'
track_io_timing = on
track_functions = all
```

Żeby wczytać konfigurację w obrazie Postgres skonfigurowałem w pliku *docker-compose.yml* opcje `command: -c config_file=/etc/postgresql.conf` i `volumes: - ./postgresql.conf:/etc/postgresql.conf`.

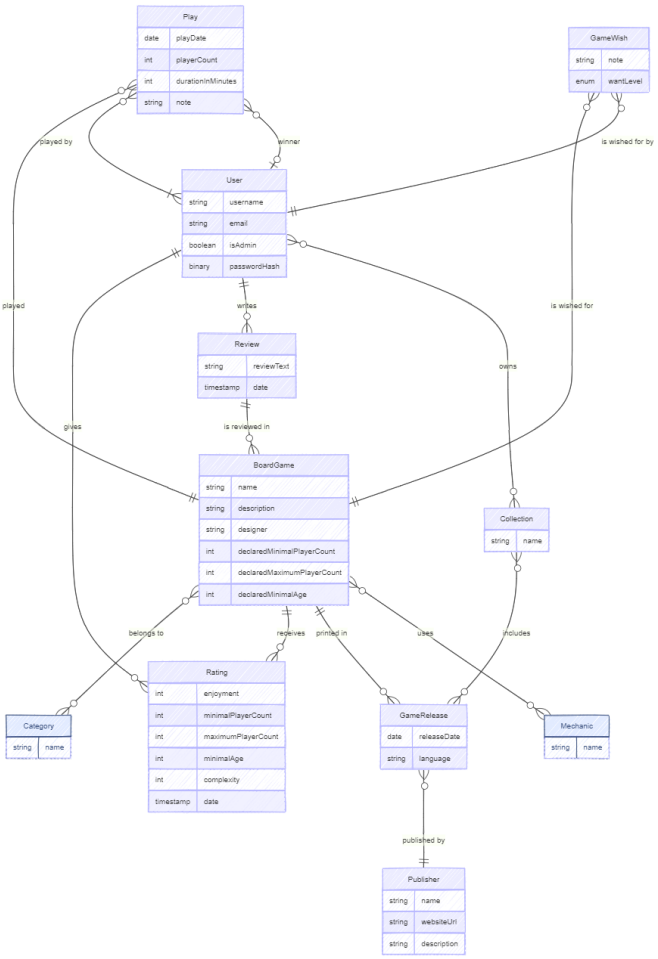
2.3 Testowanie działania

Aby sprawdzić czy SZBD działa poprawnie uruchomiłem kontener z moim plikiem konfiguracyjnym.

Połączyłem się do bazy za pomocą `psql` i sprawdziłem czy działa poprawnie.
Polecenie `SELECT version();` zwróciło poprawną wersję: `PostgreSQL 18.0 (Debian 18.0-1.pgdg13+3) on x86_64-pc-linux-gnu, compiled by gcc (Debian 14.2.0-19) 14.2.0, 64-bit`.
Oznacza to, że SZBD działa i odpowiada na zapytania.

3 Diagram Entity Relationship Diagram (ERD)

Wysokopoziomowy model relacji przedstawiony na Rysunek 1.



Rysunek 1: Diagram ERD

Źródła

1. Fireship I replaced my entire tech stack with Postgres... - YouTube. https://www.youtube.com/watch?v=3JW732GrMdg. Accessed 13 paź 2025

2. Why not run production postgres in docker? : r/PostgreSQL. https://www.reddit.com/r/PostgreSQL/comments/1c2rbow/why_not_run_production_postgres_in_docker/. Accessed 13 paź 2025

3. Charboneau T How to Use the Postgres Docker Official Image | Docker. https://www.docker.com/blog/how-to-use-the-postgres-docker-official-image/#Why-should-you-containerize-Postgres. Accessed 13 paź 2025

4. PostgreSQL Docker Official Image Documentation. https://github.com/docker-library/docs/blob/master/postgres/README.md#database-configuration. Accessed 14 paź 2025

Zastosowane skróty

ERD Entity Relationship Diagram
SZBD System zarządzania bazami danych