**Recenzje gier komputerowych**

**Skład zespołu**

* Karol Hamielec
* Michał Janeczko

**Źródło**

Steam

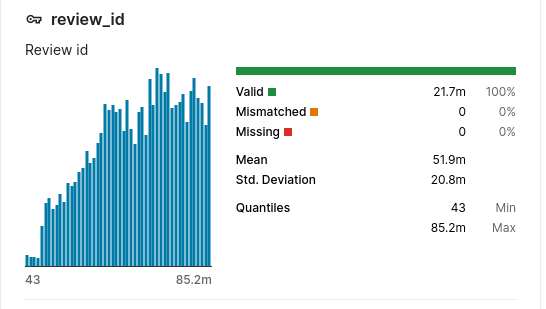
**Dane**

Dane użyte w projekcie są dostępne na stronie <https://www.kaggle.com/datasets/najzeko/steam-reviews-2021> w formacie csv. Dane zajmują łącznie 8.17GB i obejmują recenzje gier komputerowych z 2021 dla ponad 300 gier dostępnych na tej platformie w wielu językach. Dane składają się z 23 kolumn:

1. # - Index
2. app\_id – numer id gry w platformie
3. app\_name – nazwa gry na platformie
4. review\_id – numer id recenzji
5. language – język, w którym została napisana recenzja
6. review – tekst recenzji
7. timestamp\_created – czas utworzenia recenzji
8. timestamp\_updated – ostatnia aktualizacja recenzji
9. recommended – czy recenzja poleca grę
10. votes\_helpful – ilość głosów, że recenzja była pomocna
11. votes\_funny – ilość głosów, że recenzja była zabawna
12. weighted\_vote\_score – wynik obliczany na podstawie ilości pomocnych głosów
13. comment\_count – ilość komentarzy do recenzji
14. steam\_purchase – stwierdzenie, czy autor kupił grę na steamie
15. received\_for\_free – stwierdzenie, czy autor recenzji otrzymał grę za darmo
16. written\_during\_early\_access – stwierdzenie, czy recenzja była napisana na początku dostępu do gry
17. author.steamid – id autora recenzji
18. author.num\_games\_owned – ilość gier posiadanych przez autora recenzji
19. author.num\_reviews – łączna ilość recenzji autora
20. author.playtime\_forever- całkowity czas poświęcony w recenzowanej grze przez autora recenzji
21. author.playtime\_last\_two\_weeks – czas odtwarzania recenzowanej gry przez autora recenzji w ostatnich 2 tygodniach
22. author.playtime\_at\_review – czas poświęcony na grę przez autora do czasu wystawienia recenzji
23. author.last\_played – czas kiedy autor recenzji ostatni raz grał w recenzowaną grę

Kluczami w tej tablicy danych są wartości app\_id, review\_id i author\_id. Dzięki temu można jednoznacznie określić recenzję do której chcemy się odwołać.

Oprócz tego na powyższej stronie możemy porównać statystyki dla każdej kolumny, tzn. czy nie ma uszkodzeń danych, wartości średnie, odchylenie standardowe, kwantyle, unikalne wartości, wartości najczęściej występujące itp.



**Wybrane bazy danych**

* PostgreSQL – system zarządzania relacyjnymi bazami danych
* MongoDB – otwarty, nierelacyjny system zarządzania bazą danych napisany w języku C++. Charakteryzuje się brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Zamiast tego dane składowane są jako dokumenty w stylu JSON.
* Redis – otwartoźródłowe oprogramowanie działające jako nierelacyjna baza danych przechowująca dane w strukturze klucz-wartość w pamięci operacyjnej serwera, przeznaczona do działania jako klasyczna baza danych, miejsce przechowywania pamięci podręcznej oraz broker wiadomości.

Definicje z Wikipedii

**Potrzebne oprogramowanie**

* Git - rozproszony system kontroli wersji
* Docker - otwarte oprogramowanie służące do realizacji wirtualizacji na poziomie systemu operacyjnego, działające jako „platforma dla programistów i administratorów do tworzenia, wdrażania i uruchamiania aplikacji rozproszonych”.
* Python – język programowania na którym opierał się projekt
* FastApi – platforma internetowa służąca do tworzenia API
* Node.js – wieloplatformowe środowisko uruchomieniowe o kodzie do tworzenia aplikacji
* Angular – platforma do stworzenia aplikacji
* SQLAlchemy - otwartoźródłowa biblioteka programistyczna napisana w języku programowania Python, która służąca do pracy z bazami danych oraz mapowania obiektowo-relacyjnego. W tym przypadku została użyta do bazy PostgreSQL
* PyMongo - dystrybucja Pythona zawierająca narzędzia do pracy z MongoDB
* Redis-Py – oprogramowanie klienckie do pythona do obsługi Redisa w tym jezyku programowania
* Ngx-charts – biblioteka dla Angulara służącą do wizualizacji danych za pomocą wykresów

**Konfiguracja**

1. Instalacja brakujących programów lub aktualizacja istniejących: git, python i docker
2. Sklonowanie kodu z repozytorium

git clone https://github.com/lewelyn7/ztbd\_project.git

1. Przejście do katologu ztpd\_project
2. Przy pierwszym uruchomieniu zbudowanie obrazu dockera

docker compose build

1. Po zbudowaniu obrazu uruchomienie obrazu

docker compose up

1. W nowym terminalu przechodzimy do katalogu backend
2. Instalujemy potrzebne paczki

pip install -r requirements.txt

1. Uruchomiamy aplikacje

uvicorn app.main:app --reload

1. W przeglądarce sprawdzamy, czy w API zostały załadowane dostępne metody i testy

<http://127.0.0.1:8000/docs>

Rezultat powinien wyglądać podobnie do zamieszczonego niżej:

Obraz zawierający tekst

Opis wygenerowany automatycznie

1. Przechodzimy do katalogu frontend/db-perf-tester/
2. Instaluje pakiety do widoku aplikacji

npm install

1. Włączamy aplikacje

npx ng serve

1. Sprawdzamy, czy pod wskazanym linkiem znajduje się aplikacja z wszystkimi testami wydajnościowymi
2. Uruchomiamy interesujące nas testy

**Kod dostępny w projekcie**

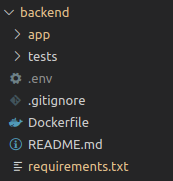
W repozytorium znajduje się podział na kilka części.

W folderze domyślnym znajdują się pliki do zbudowania obrazu Dockeru, zbudowania angulara oraz stworzenia środowiska

Obraz zawierający tekst

Opis wygenerowany automatycznie

W folderze backend znajduje się cała aplikacja od tej strony w folderze app. Oprócz tego znajduje się plik z wymaganymi bibliotekami z pythona (requirements.txt) oraz Dockerfile, który zawiera szereg instrukcji do stworzenia w pełni funkcjonalnego obrazu



W folderze app mamy dostępne kilka podfolderów.

W folderze core mamy dostępne pliki config oraz utils, które pozwalają się poprawnie połączyć z bazami danych oraz implementują wyjątki.

W folderze daos mamy zaimplementowane DAO dla autora, recenzji i gry. DAO to komponent dostarczający jednolity interfejs do komunikacji między aplikacją, a bazą danych

W folderze databases znajdują się pliki, które tworzą sesje dla baz danych

W routers znajduje się funkcje możliwe do użycia na każdej bazie danych oraz znajdują tam testy wydajnościowe, które pozwalają nam sprawdzić szybkość działania bazy danych dla odpowiednich funkcji oraz wybranej ilości iteracji

Jeśli chodzi o folder frontend to mamy tam wygenerowany kod w angularze, który służy do stworzenia aplikacji. Znajdują się tam pliki html, css, które odpowiadają za pojawienie się elementów w aplikacji. Z bardziej interesujących części mamy plik app.component.html gdzie są dostępne parametry wizualne aplikacji

W pliku app.components.ts wyliczamy odpowiednie wartości z przeprowadzonych testów. Takimi wartościami są wartości minimalne, maksymalne, średnia oraz odchylenie standardowe dla każdej bazy osobno. Oprócz tego tam dodajemy listę testów, które możemy włączyć w aplikacji z odpowiednią liczbą iteracji.

**Testy**

Wszystkie testy zostały napisane w języku python. Mają one tę samą strukturę dla każdej bazy danych i wywołują one kolejne bazy danych z tymi samymi parametrami oraz wykonują tą samą czynność na bazie danych. Obliczają one czas na wykonanie danej czynności przez policzeniem różnicy czasu. Następnie otrzymany czas zamieniamy na ms, aby móc wynik przedstawić na wykresie.

Scenariusze testowe:

* Case0 – wyszukiwanie elementu w kolumnie author\_id przy odwołaniu do innego DAO
* Case1 – wyszukiwanie istniejącego elementu w kolumnie language
* Case1 – wyszukiwanie nieistniejącego elementu w kolumnie language
* Case2 – wyszukiwanie recenzji na podstawie fragmentu recenzji w kolumnie review
* Case3 – wyszukiwanie tekstu bez wskazania kolumny
* Case4 – dodanie nowego elementu do bazy danych
* Case5 – próba usunięcia istniejącego elementu z bazy danych
* Case6 – próba usunięcia nieistniejącego elementu z bazy danych
* Case7 – próba dodania nowego elementu, która się nie udaje z powodu istnienia już takiego elementu w bazie
* Case8 – dodanie nowego elementu do bazy danych, a następnie jego usunięcie

Każdy test jest powtarzany określoną ilość razy, którą ustalamy w aplikacji w polu Iterations. Uzyskujemy w ten sposób bardziej rzeczywiste pomiary, które pozwalają odrzucić rezultaty związane z błędem pomiarowym urządzenia. Wyniki następnie są prezentowane na wykresie gdzie jest podział dla każdej bazy danych, a następnie są obliczane kwantyle. Całkowite wyniki, które zawierają średnią, wartości minimalne i maksymalne są pokazane w tabeli Results table. Kwantyle są natomiast na wykresie po najechaniu na odpowiadający bazie danych słupek.

**Wyniki**

**Wnioski**