Advanced Database Systems - I - K.Jankiewicz - 2023/2024 summer

Aplikacja do monitorowania i rejestrowania stanu zdrowia w czasie rzeczywistym

Aplikacja ma za zadnie zapisywać pomiary parametrów użytkowników takie jak tętno, ciśnienie krwi i temperaturę przesyłane z różnych urządzeń jak np. smartwatch, czujnik tętna. Dane zostają zapisywane w bazie danych oraz są analizowane pod kątem odchyleń. Podczas przekroczenia krytycznych wartości użytkownik lub osoba śledząca jego pomiary zostaje o ich powiadomiona.

Użyta baza danych to Redis z wykorzystaniem modelu danych klucz wartość.

Użyłem bazy danych Redis ponieważ jest ona przystosowania do otrzymywania bardzo dużej ilości danych gdzie pomiary odbywają się co klika sekund. Można łatwo i szybko otrzymać zbiór ostatnich danych. Aplikacja nie zawiera skomplikowanych modeli a wręcz opiera się na bardzo prostych gdzie idealnie wpasowuje się w typy słownikowe.

Wykorzystane specyficzne wartości:

- Konfiguracja – w projekcie jest dodany plik redis.conf który umożliwia dostosowanie ustawień bazy danych. Jak np. ustawienie hasła, ograniczenie pamięci maszyny lub opcje snapshotu. Użyłem aby zabezpieczyć lepiej bazę danych i nadać jej hasło. Dodatkowo zabezpieczyłem się przed stratą danych tworząc snapshoty,
- PubSub – Jest to mechanizm pozwalający na komunikacje między różnymi procesami za pośrednictwem wzoraca publikacji-subskrybcji. W aplikacji za jego pomocą zostały wysyłane sygnały aby powiadomić o przekroczonych wartościach krytycznych.

Schemat bazy danych:

1. Użytkownicy
- Typ: Hash
- Klucz : user:{username}
- Wartość: id, username, password (zaszyfrowane), email
2. Parametry
- Typ: Sorted set
- Klucz: user:{username}:heart_rate/pulse/temperaturę
- Wartość: Member: timestamp, Score: int (jest to wartość danego pomiaru)
3. Informacje o powiadomieniach i wartościach krytycznych
- Typ: hash
- Klucz: user:{username}:follow_users:{username(obserwujący)}
- Wartość: Field: critical_pulse, critical_heart_rate, critical_temperature Value: int(wartości krytyczne)
4. Historia powiadomień
- Typ: Lista
- Klucz: alerts:{username}:history:{ username(obserwujący)}
- Wartość: wartość teksowa typu json z znacznikiem czasu oraz treścią powiadomień

Opis Metod

Autentykacja i Rejestracja

- **Rejestracja** (register()):
  - Sprawdzenie, czy użytkownik już istnieje.
  - Hashowanie hasła przed zapisaniem.
  - Przechowywanie danych użytkownika w Redis.
- **Logowanie** (login()):
  - Walidacja użytkownika poprzez sprawdzenie hasła z wykorzystaniem funkcji check_password_hash.

Zarządzanie parametrami

- **Rejestracja parametrów ciała** (register_vitals()):
  - Zapisywanie pulsu, tętna oraz temperatury jako zbiorów posortowanych w Redis, co pozwala na łatwe sortowanie i odczytywanie danych chronologicznie.
  - Publikacja alertów, gdy wartości przekroczą określone progi dla obserwujących użytkowników.

Zarządzanie Obserwacjami i Alertami

- **Dodawanie obserwującego** (add_follow_user()):
    - Możliwość ustawienia wartości krytycznych dla parametrów zdrowotnych, które jeżeli zostaną przekroczone, generują alert.
- **Otrzymywanie alertów** (get_alerts()):
    - Odczytanie i formatowanie alertów z historii dla danego użytkownika i obserwującego.

Redis jest bardzo szybką i wydajną bazą danych jednak są dość zauważalne jego braki. O ile do zapisywania dużej ilość prostych danych nadaję się świetnie to komplikacja modelu przechowywanych przyćmiewa jego zalety. Opieranie całej aplikacji na tej bazie było błędne. Gdybym zaczynał pisać projekt na nowo użył bym Redisa tylko do przechowywania parametrów ciała. A bardziej złożone modele jak informacje o użytkownikach kto kogo śledzi czy przy jakich wartościach mają zostać wysyłane powiadomienia przeniósł bym do bazy np. MongoDb.