Rozproszone Systemy Baz Danych - Opis zadania projektowego

Jakub Jaśków 268416, Justyna Ziemichód 268431 (G2)

1 Temat i cel projektu

Temat: „Rozproszony system bazodanowy przeznaczony do obsługi liczników wody."
Cel projektu: Zaprojektowanie oraz implementacja aplikacji umożliwiającej dostęp do rozproszonej bazy spisów liczników wody.

2 Opis działania i funkcje systemu

System opiera się na dwóch lub trzech serwerach, na których funkcjonują bazy danych przechowujące informacje dotyczące odczytów liczników wody, takie jak numer urządzenia pomiarowego, lokalizacja, data pomiaru czy ilość zużytej wody. Struktura tych baz jest z góry określona, co umożliwia udostępnianie danych użytkownikom poprzez aplikację działającą na serwerze WWW, dostępną z poziomu przeglądarki internetowej.

Rozwiązanie pozwala na zdalny dostęp do dowolnego serwera bazodanowego obsługującego dany obszar działania przedsiębiorstwa wodociągowego. Użytkownicy mogą przeglądać dane, dodawać nowe odczyty lub aktualizować istniejące wpisy w bazie.

Aby zapewnić spójność informacji w całym systemie, wprowadzono mechanizm replikacji danych pomiędzy serwerami [2, 6].

3 Założenia architektoniczne przyjęte podczas realizacji systemu

W projekcie zostanie zastosowany trójwarstwowy model architektury klient/serwer, oparty na koncepcji „cienkiego klienta" [3]. Logika biznesowa będzie przetwarzana po stronie serwera aplikacyjnego, dane będą przechowywane w warstwie bazodanowej, a po stronie klienta znajdzie się jedynie warstwa prezentacji obsługiwana przez przeglądarkę internetową.

Aplikacja biznesowa będzie udostępniana poprzez serwer WWW. Komunikacja z systemem bazodanowym będzie odbywać się za pomocą mechanizmów aplikacyjnych kierujących zapytania do odpowiedniego serwera danych. W projekcie zostaną wykorzystane bazy danych jednego typu, co zapewni spójność i prostotę zarządzania [2].

Aby zwiększyć wydajność i równomiernie rozłożyć ruch między serwerami aplikacji, zastosowany zostanie mechanizm równoważenia obciążenia (load balancing) w trybie cyklicznym (round robin) [5]. Dzięki temu każde żądanie użytkownika trafi do kolejnego dostępnego serwera, co pozwoli uniknąć przeciążenia pojedynczych węzłów i zapewni lepszą skalowalność systemu.

System zostanie również przygotowany do uruchamiania w lekkich środowiskach kontenerowych, co umożliwi szybkie wdrażanie, aktualizacje oraz łatwe zwiększanie liczby instancji aplikacji [1].

W warstwie bazodanowej zastosowana zostanie replikacja danych pomiędzy serwerami [6]. Główny serwer będzie obsługiwał operacje zapisu, natomiast serwery podrzędne będą przechowywać ich bieżące kopie. Takie rozwiązanie pozwoli rozłożyć obciążenie przy odczytach i zwiększy dostępność systemu w przypadku awarii serwera głównego.

4 Wykorzystywane narzędzia, technologie projektowania oraz implementacji systemu

Dostęp do baz danych pomiarów wodociągów będzie realizowany za pośrednictwem serwerów zarządzających systemem **PostgreSQL** [6]. Logika aplikacji zostanie zaimplementowana po stronie serwera z wykorzystaniem frameworka **Flask** w języku **Python** [4], który obsłuży komunikację z bazą danych oraz przetwarzanie żądań użytkowników.

Dane pomiarowe, takie jak odczyty liczników, lokalizacje punktów czy daty odczytów, będą przechowywane w bazie **PostgreSQL**. W celu zapewnienia spójności i wysokiej dostępności danych zastosowana zostanie replikacja strumieniowa (**streaming replication**) [6]. Dodatkowo do zarządzania połączeniami z bazą oraz równoważenia obciążenia między węzłami zostanie wykorzystane narzędzie **pgpool** [5], które pozwoli na efektywne kierowanie zapytań odczytu i zapisu do właściwych serwerów.

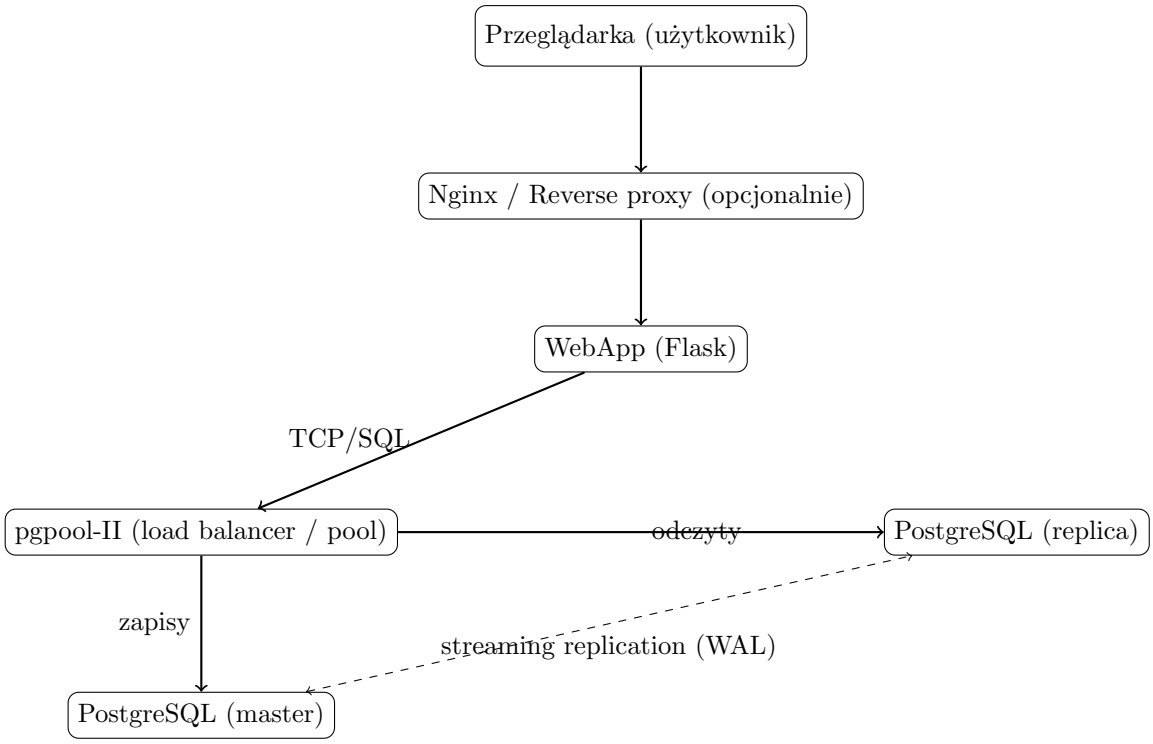
Środowisko uruchomieniowe systemu zostanie przygotowane w oparciu o lekką konteneryzację, co ułatwi wdrażanie, testowanie i utrzymanie aplikacji. Poszczególne komponenty — serwer aplikacyjny, bazy danych oraz narzędzia pomocnicze — będą definiowane i uruchamiane przy użyciu pliku konfiguracyjnego **Docker Compose** [1], który umożliwi automatyczne tworzenie i łączenie kontenerów w spójne środowisko systemowe.

Warstwa prezentacji zostanie zrealizowana w formie aplikacji webowej dostępnej z poziomu przeglądarki internetowej, umożliwiającej przeglądanie i aktualizację informacji o pomiarach.

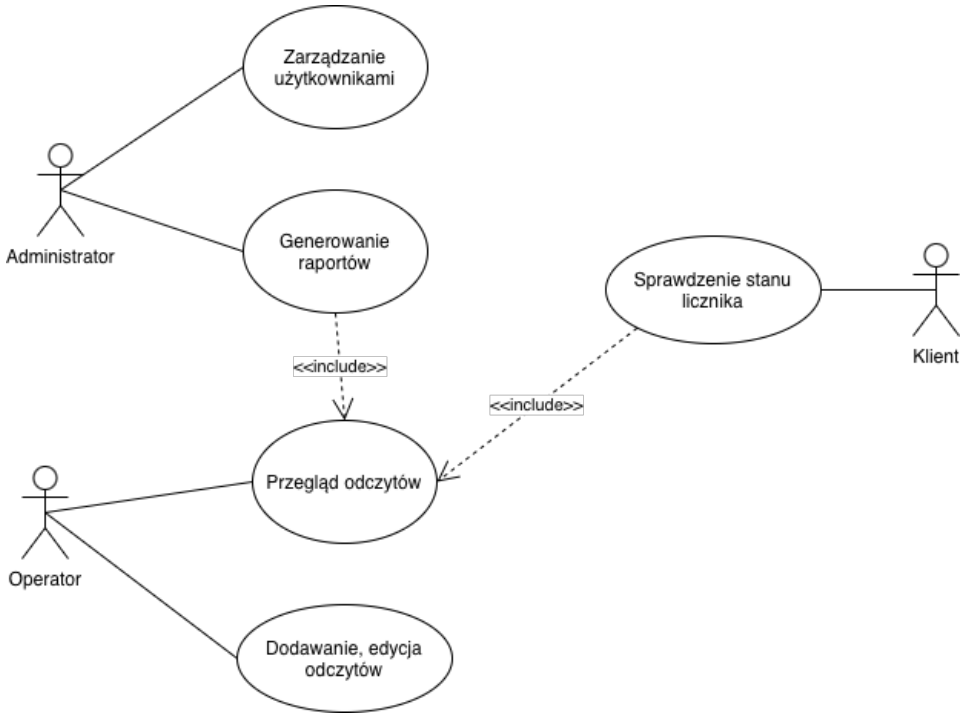
Do opisu i projektowania funkcjonalności systemu wykorzystany zostanie zunifikowany język modelowania **UML**, służący do tworzenia diagramów przedstawiających strukturę i sposób działania aplikacji [2, 3].

5 Struktura systemu i schemat komunikacji

Poniżej znajduje się prosty schemat strukturalny pokazujący główne komponenty i kierunki komunikacji (wersja uproszczona):



6 Diagram przypadków użycia (Use Case)



7 Scenariusze użycia (przykładowe)

UC1: Zarządzanie użytkownikami (Administrator)

Aktor: Administrator.
Cel: Dodanie nowego operatora / zmiana uprawnień.
Pre-kondycje: Administrator zalogowany.
Przebieg podstawowy:

1. Administrator otwiera panel administracyjny.
2. Wybiera opcję „Użytkownicy" i klik „Dodaj użytkownika".
3. Wprowadza dane użytkownika (login, rola, e-mail) i zatwierdza.
4. System tworzy konto i wysyła powiadomienie (opcjonalnie e-mail).
5. Administrator może przypisać uprawnienia dostępu do funkcji (np. tylko odczyt, odczyt+zapis).

Przebieg alternatywny: Jeśli login jest już zajęty — system wyświetla komunikat o błędzie i prosi o zmianę.

UC2: Przegląd odczytów (Operator / Klient)

Aktor: Operator lub Klient.
Cel: Wyświetlenie listy odczytów dla danego adresu / okresu.
Pre-kondycje: Użytkownik zalogowany (dla klienta: dostęp ograniczony do własnych liczników).
Przebieg podstawowy:

1. Użytkownik wybiera filtr (adres, okres dat).
2. System przekazuje zapytanie do warstwy aplikacji (Flask).
3. WebApp wysyła zapytanie SQL przez pgpool.
4. pgpool rozdziela zapytanie do odpowiedniej repliki (odczyt) i zwraca wyniki.
5. Wyniki są wyświetlane użytkownikowi w postaci tabeli/grafu.

UC3: Dodawanie / edycja odczytu (Operator)

Aktor: Operator.
Cel: Wprowadzenie nowego odczytu licznika.
Pre-kondycje: Operator zalogowany, posiada uprawnienia zapisu.
Przebieg podstawowy:

1. Operator wchodzi do formularza „Dodaj odczyt".
2. Wprowadza numer licznika, datę pomiaru, wartość odczytu i ewentualne notatki.
3. WebApp przesyła transakcję zapisu do pgpool, który kieruje ją do mastera.
4. Master zapisuje transakcję, WAL jest aktualizowany i strumieniowo przesyłany do replik.
5. System potwierdza sukces operacji i wyświetla komunikat.

Przebieg alternatywny: Jeśli zapis nie powiedzie się (np. błąd walidacji), system zwraca odpowiedni komunikat wraz z kodem błędu.

UC4: Generowanie raportów (Administrator)

Aktor: Administrator.
Cel: Wygenerowanie miesięcznego raportu zużycia wody dla obszaru.
Pre-kondycje: Administrator zalogowany.
Przebieg podstawowy:

1. Administrator wybiera zakres dat i obszar geograficzny.
2. System uruchamia zapytania agregujące (najczęściej wykonywane na replikach).
3. Wynik agregacji jest eksportowany do formatu PDF/CSV.
4. Plik jest dostępny do pobrania lub wysyłany e-mailem.

UC5: Sprawdzanie stanu licznika (Klient)

Aktor: Klient (mieszkaniec).
Cel: Szybkie sprawdzenie ostatniego odczytu i zużycia.
Przebieg podstawowy:

1. Klient loguje się i wybiera swój licznik z listy.
2. System pobiera ostatni odczyt (odczyt z repliki) i pokazuje wykres trendu.
3. Klient otrzymuje informację o ostatnim odczycie oraz o ewentualnych alertach (np. nietypowe skoki zużycia).

Literatura

[1] Docker Inc. *Docker and Docker Compose Documentation*, 2024. URL `https://docs.docker.com/`.

[2] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems*. Pearson, 7th edition, 2016.

[3] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002.

[4] Miguel Grinberg. *Flask Web Development: Developing Web Applications with Python and Flask*. O'Reilly Media, 2018.

[5] pgpool Development Team. *pgpool-II Documentation*, 2024. URL `https://www.pgpool.net/docs/latest/en/html/`. Opis load balancing, connection pooling, failover.

[6] PostgreSQL Global Development Group. *PostgreSQL Documentation*, 2024. URL `https://www.postgresql.org/docs/`. Sekcje: Streaming Replication, WAL.