Projekt z rozproszonych i obiektowych systemów baz danych

Projekt aplikacji wykorzystującej repliki MongoDB

Autorzy:
Maciej Kiedrowski, nr indeksu: 200105
Joanna Piątek, nr indeksu: 199966

Grupa: Poniedziałek 16:10

Data oddania: 23.01.2017

Prowadzący zajęcia: Dr inż. Robert Wójcik, W4/K-9

Ocena pracy:

Spis treści

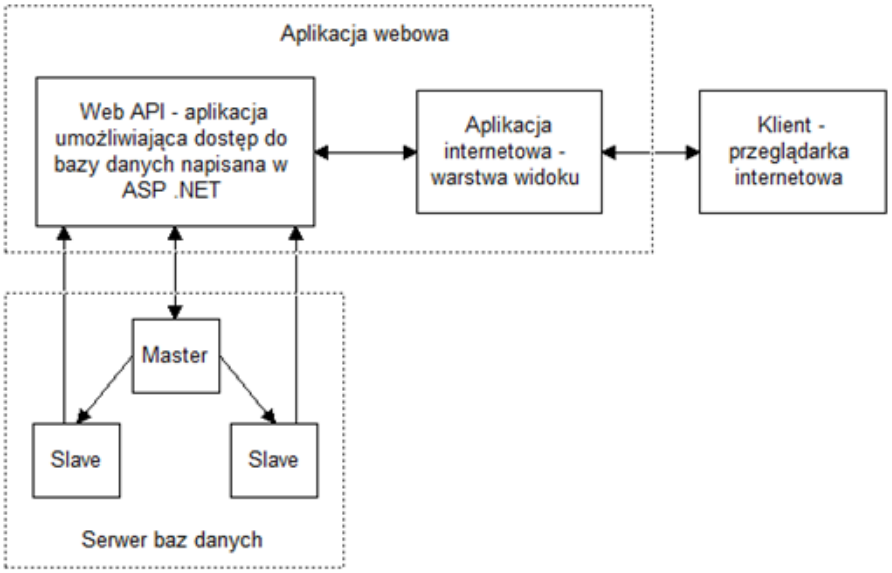
Spis rysunków

1 Wstęp

Niniejszy projekt realizowany w ramach kursu "Rozproszone i obiektowe systemy baz danych" ma na celu zaprojektowanie oraz imlementację rozproszonej aplikacji, odpornej na awarię baz danych, pozwalającej zarządzać stanem magazynowym oraz sprzedażą w sieci restauracji.

1.1 Cele projektu

Celem projektu jest stworzenie systemu pozwalającego zarządzać siecią restauracji. System pozwala na utworzenie nowej restauracji w bazie, utworzenie menu wspólnego dla całej sieci, zarządzania stanem magazynowym poszczególnych restauracji, sprzedaż prodóktów wpływający na stan magazynowy. Dane o wszystkich restauracjach przechowywane są we wspólnej, centralnej bazie danych - wymaga ona maksymalnej dostępności oraz ochrony danych przed ich utratą.
Dostęp do systemu możliwy jest poprzez dowolną przeglądarkę internetową - użytkownik za pomocą interfejsu webowego posiada możliwość wykonywania operacji na bazie danych za pośrednictwem aplikacji serwerowej. strukturaSystemu



Rysunek 1: Struktura systemu i schemat komunikacji

1.2 Założenia projektowe

Jako silnik bazy danych wybrany został system MongoDB ze względu na skalowalność, wydajność oraz architekturę zaprojektowaną z myślą o łatwej replikacji. Utworzone zostały trzy serwery bazy danych tworzące zestaw replik MongoDB. Webowa aplikacja kliencka został wykonana w oparciu o platformę ASP .NET MVC.

2 Replikacja

Proces replikacji polega na synchronizacji danych pomiędzy serwerami. Pozwala osiągnąć następujące korzyści:

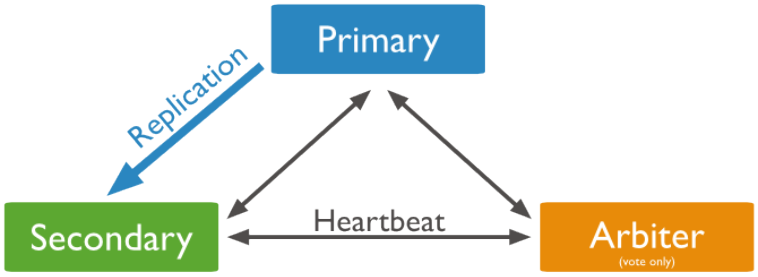
- Bezpieczeństwo danych poprzez renundancję
- Wysoką dostępność
- Skalowanie wydajności

2.1 Replikacja w MongoDB

Replikacja wbudowana w platformę *MongoDB* opiera się na *replica set* - zestawie replik.

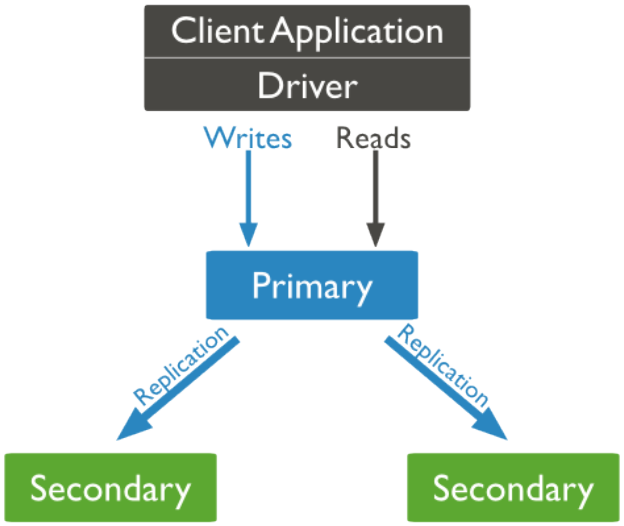
2.1.1 Zestaw replik

Zestaw replik grupa procesów *mongod* utrzymujących ten sam zestaw danych. Zestaw replik w ramach MongoDb składa się z węzłów zawierających dane oraz ewentualnego węzłą wspomagającego arbitraż.



Rysunek 2: Zestaw replik

W każdym zestawie replik jeden z węzłów pełni rolę *Primary*. Węzeł ten jest jedynym, który akceptuje operacje zapisu, jest również domyślnym węzłem dla wszystkich operacji odczytu z zestawu replik. Pozostałe węzły wchodzące w skład zestawu, a niebędące węzłem arbitrażowym działają w trybie *Secondary*. Minimalna ilość węzłów zapewniająca poprawną pracę zestawu to 3, ilośc węzłów powinna być nieparzysta.



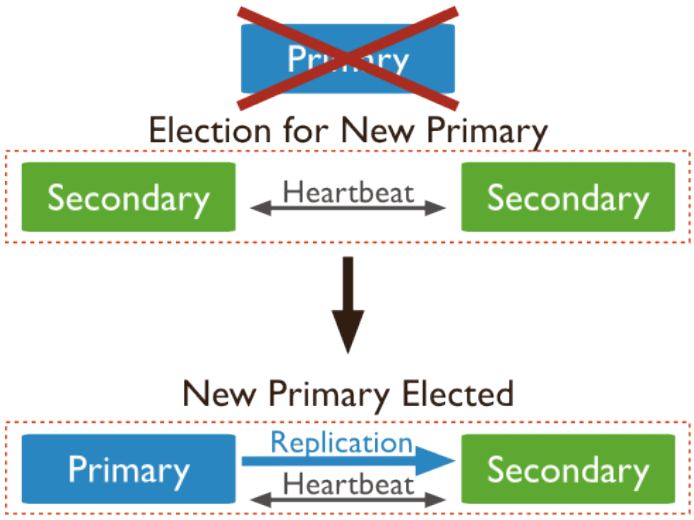
Rysunek 3: Shemat działania zestawu replik

2.1.2 Replikacja

Replikacja w *MongoDB* jest replikacją asynchroniczą. Węzły *Secondary* replikują dziennik operacji (*oplog*) węzła *Primary* i wykonują operacja w nim zawarte na przechowywanym zbiorze danych.

2.1.3 Wysoka dostępność

Zestawe replik gwarantuje wysoką dostępność danych. W przypadku awarii węzła, w szczególności węzła *Primary* uruchamiany jest proces *failover*. Ma on na celu zapewnienie płynności dostępu do danych. Pozostałe działające węzły w ramach repliki przeprowadzają głosowanie nad wyborem nowego *Primary*, po zakończeniu głosowania zestaw odzyskuje pełną sprawność. Proces *failover* trwa przeważnei poniżej minuty, z czego 10-30 sekund to wykrycie awarii węzła *Primary*, następne 10-30 sekund proces głosowania.
W trakcie głosowania zestaw pracuje w trybie *read-only* - żaden z węzłów nie posiada statusu *Primary* a zatem wszystie żądania zapisu do bazy są odrzucane.

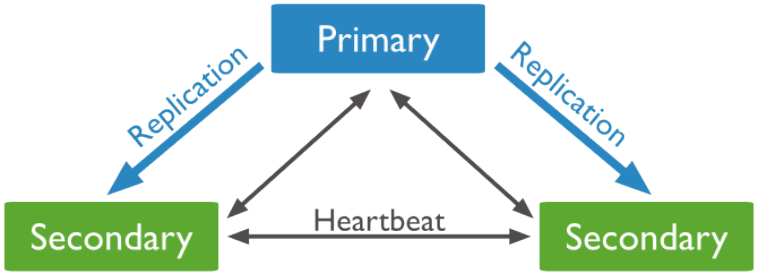


Rysunek 4: Wysoka dostępność z wykorzystaniem procesu *failover*

Warunkiem koniecznym do istnienia węzła *Primary* w zestawie, jest dostępność ponad 50% węzłów. Oznacza to, że dla zestawu składającego się z 3 instancji, jest on w pełni funkcjonalny przy awarii 1 węzła, natomiast zestaw skłądający się z 5 węzłów pozwala na awarię 2 węzłów.
W przypadku dostępnośći mniejszej liczby węzłów w zestawie, wszystkie pozostałe przechodzą w tryb *Secondary* - zestaw działa w trybie *read-only*. Mechanizm ten służy zabezpieczeniu danych przed brakiem lub niewystarczającą replikacją w systemie.

2.1.4 Dostępność

Poprawne działanie węzłów monitorowane jest za pomocą *Heartbeats*. Instancje co 2 sekundy wzajemnie informują się o poprawnym działaniu, brak informacji przez 10 sekund traktowany jest jako awaria węzłą.



Rysunek 5: Monitorowanie działania węzłów

3 Implementacja bazy danych w środowisku MongoDB

Celem implementacji było stworzenie zestawu replik składającego się z 3 procesów *mongod* działających na niezależnych instancjach.

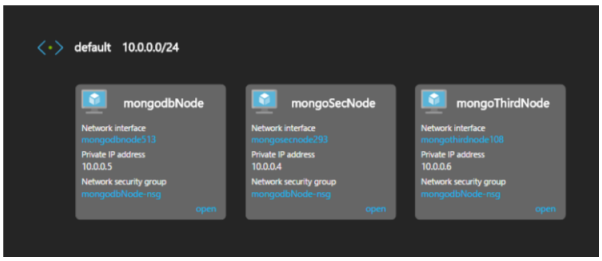
3.1 Infrastruktura

3.1.1 Instancje

Baza danych oparta została o chmurę *Azure* w modelu IaaS - *infrastruktura jako usługa*. Utworzone zostały 3 maszyny wirtualne z wykorzystanim obrazów dostarczanych przez *Bitnami*. Systemem operacyjnym działającym na maszynach wirtualnych jest Ubuntu 14.04, natomiast wersja MongoDB to 3.2.9.

3.1.2 Sieć wewnętrzna

Maszyny wirtualne zostały połączone przez prywatną sieć wirtualną, zilustrowaną poniżej.



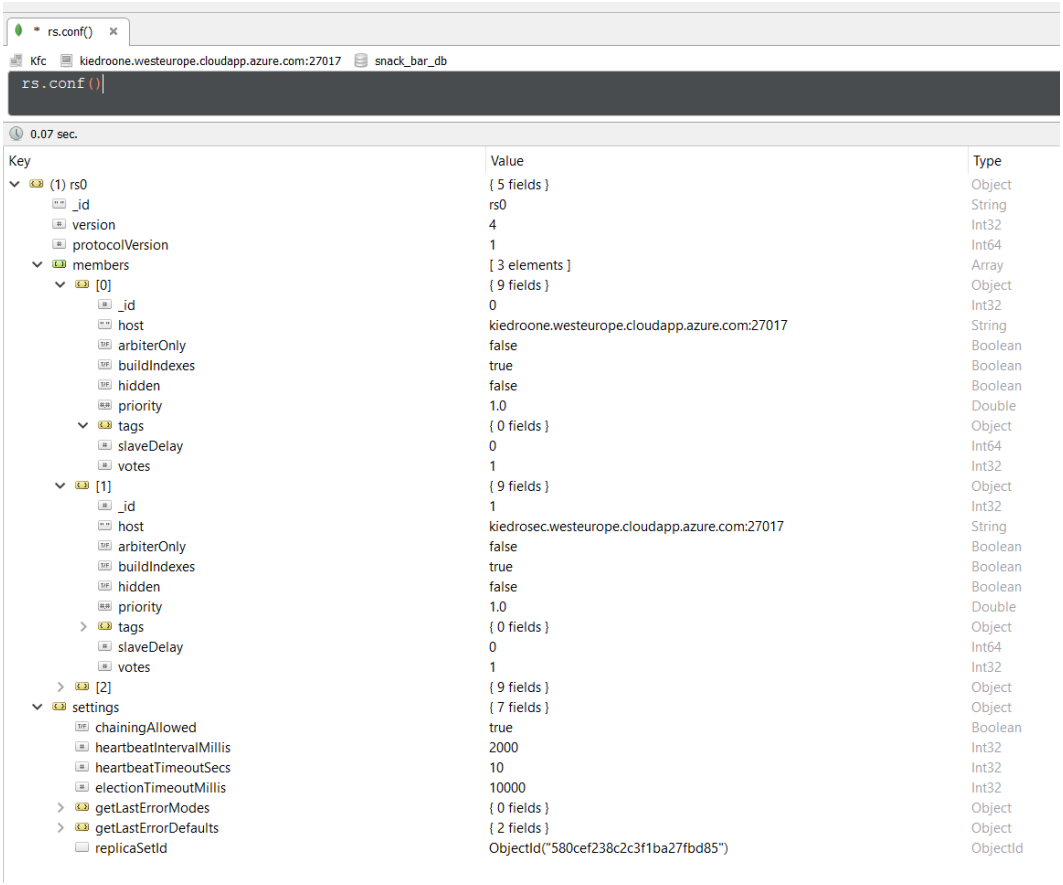
Rysunek 6: Sieć wirtualna

3.1.3 Sieć publiczna

Aby umożliwić połączenie z zestawem replik od dowolnego klienta, bez wymogu jego wdrożenia w chmurze *Azure*, każdej z instancji maszyn wirtualnych został przydzielony publiczny adres IP, oraz nazwa DNS umożliwiająca dostęp do maszyn.

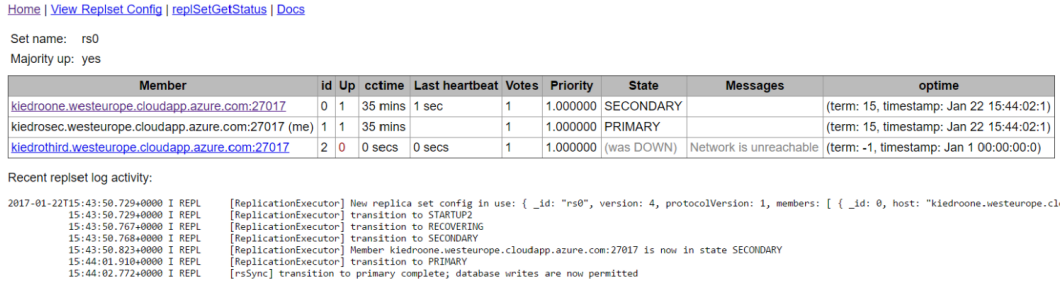
3.2 Konfiguracja MongoDB

Korzystając z *mongo shell* utworzony został zestaw replik oraz jego konfiguracja.



Rysunek 7: Konfiguracja zestawu replik

Korzystając z pliku konfiguracyjnego uaktywiono prosty interfejs REST, pozwalający na łatwą diagnozę zestawu replik.



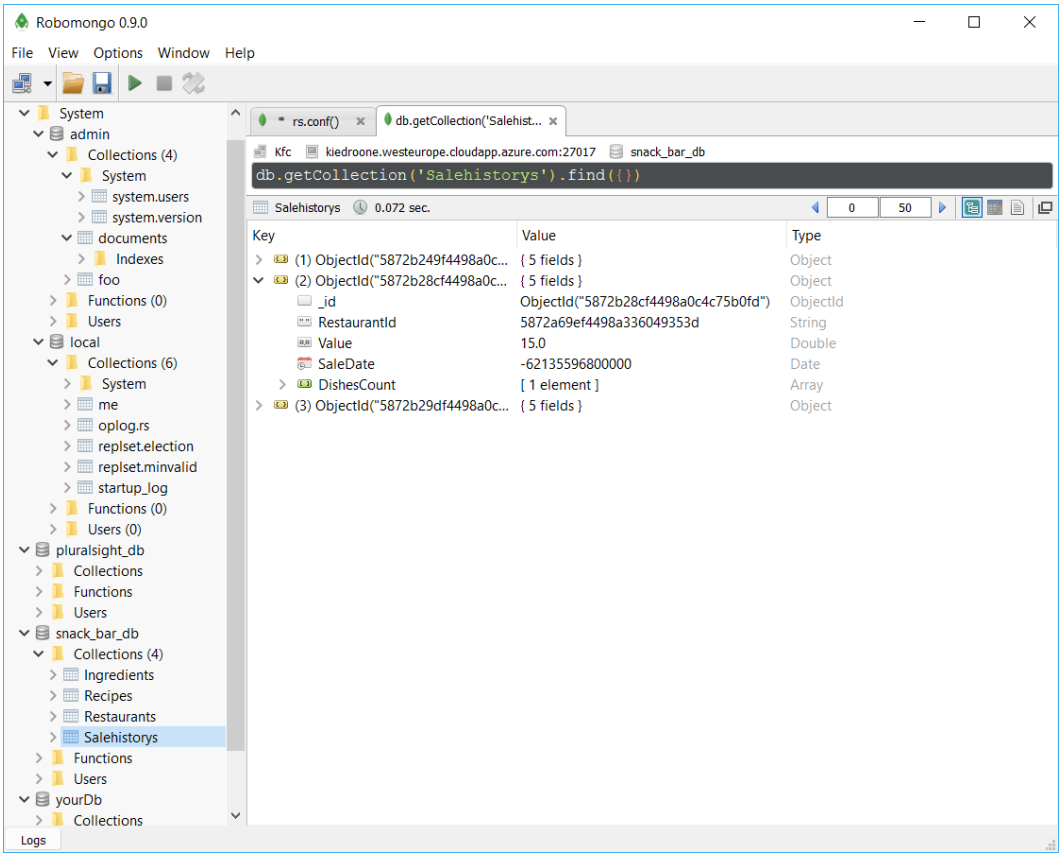
Rysunek 8: Interfejs REST

3.3 Schemat bazy danych

MongoDB należy do baz NoSQL typu dokumentowego, jedną z cech charaktrystycznych baz tego typu jest brak ustalonego schamatu bazy danych. Kolekcje tworzone dynamicznie w trakcie działania aplikacji pozwalają na przechowywanie dowolnych danych - jedynym warunkiem jest możliwość ich serializacji w formacie JSON.
Ta cecha baz NoSQL upraszcza wstępną konfigurację bazy danych.

3.4 Weryfikacja

Do weryfikacji poprawnego działania bazy danych - możliwości połączenia ze wszystkimi węzłami, odczytu, zapisu, działania mechanizmu replikacji posłużyło narzędzie *Robomongo* w wersji 0.9.



Rysunek 9: Robomongo

4 Projekt i implementacja aplikacji

4.1 Diagram przypadków użycia

Funkcje aplikacji, czyli diagram przypadków użycia przedstawia rysunek **??**.