Dotnet architect

Bartosz Sendek

December 2, 2021

1 Introduction

2 Data consistency

Mocna spójność W aplikacjach dystrybuowanych dane są rozmieszone na osobnych storeach. Nie używamy wtedy mechanizmu locków transakcyjnych gdyż powołodowałoby to znaczny overhead. W standardowym modelu mówimy o mocnej spójności- zmiany muszą być atomowe albo wszystkie zajdą albo żadna. W cloudzie używamy tylko mocnej spójności jeżeli na pewno dzieje się to w obrębie jednego storea i plusy przwyższają problemy skalowalności. Mocną spójnośc można implementować używając NoSQL baz. Używa się wtedy quorumów i wersjonowania.

Note: Nie wszystkie dane powinny być traktowane tak samo. Krytyczne dane mogą być traktowane pod względem spójności inaczej niż dane pomocnicze w zależności od wymagań biznesowych.

Spójność ewentualna W przypadku dystrybuowanych systemów zazwyczaj chcemy zeby w miare propagacji w czasie danych aplikacje mogly wcxiąż używać swoich wersji tych danych. Twierdzenie CAP mówi że system dystrybuowany może ziamplementować tylko 2 z poniższych cech w tym samym czasie:

- Spójność
- Dostępność
- Tolerancja partycjonowania

Aplikacja musi w krytycznych obszarach wykrywać niekonsekwentnośc danych i obsługiwać swoją logiką tego typu błedy. Spójność ewentualna jest również konwekwencją cacheowania danych. Jeżeli aplikacja przetrzymuje dane w cache'u ma dane które są w starym stanie. Ważne jest ustawianie expiration policy lub Cache Aside pattern. Jest to implementacja istniejącego w wiekszości cachey pojęć:

- Read-through- jezeli w cacheu nie ma zczytaj bezposrendio z bazy

- Write-through- wpisz do cachea i cache od razu do bazy. Operacja czeka az zapisze sie do bayz dlatego nie poprawia to performanceu zapisu.
- Write-Behind poprawia performance zapisu, po pewnym czasei(ex: 10 sekund, 10 minut, 10 dni) dane sa asynchronicznie zapisywane do bazy.

Więcej patrz https://docs.microsoft.com/en-us/previous-versions/msp-n-p/dn589799(v=pandp.10).

Takie rozwiazanie nie zapewniaja spójności danych ale zmniejszaja prawdopodobienstwo jego wystapienia. W pryzkladowej apliakcji uzytkonwik kupuje na ecomie produkt ktory zostal wyprzedany. Ilosc egzemplarzy w magazynie nie zostala zpropagowana do frontu z ktorego uzytkownik korzysta. System powinien wtedy zorientowac sie i poinformowac uzytkownika o problemie w momencie zakupu. Innych rozwiazaniem (w przypdku wiekszej dystrybucji systemu gdzie w skladaniu zamowienia nie ma SOT) zamowienie powinnno zostac odlozone do kolejki zamowien i tam logika powinna sprawdzac czy istneije dostateczna liczba egzemplarzy i w razie jej braku poinformowac uzytkownika o anulowaniu zamówienia.

Idempotencja W przypadku bledu komunikacji, upadku maszyny pojawia sie innny problem. Bledy takie sa tymczasowe i serwis powinien ponawiac probe wykonania zadania. Istnieje możliwość wykonania wtedy operacji podwójnie. Operacje takie powinny byc idempotentne. To znaczy wykonanie tej samej operacji dwa razy powinno wynikować tym samym stanem systemu. W przypadku sytuacji z natury nie idemponentnych jak na pryzkład wykorzystanie zewnetrznęgo serwisu płatności wprowadza się sztuczną idemponentność- przypisuje się każdej operacij ID i upewnia że id nei istneije juz w systemie. Technika ta nazywa się de-duping. W przypadku gdy aplikacje modyfikują te same dane system upewnić że modyfikacje te nie wchodzą ze sobą w konflikt. Nie myślimy wtedy o CRUDzie a o atomicznych operacjach patrz CQRS i ES.

Jako cache używane sa szybkie bazy danych jak na pryzkład key-value Redis.

3 Data Replication

Replikacja między data storeami.

Master-Master serwisy lub kopie tego samego serwisu maja oba SOT. Konflikty sa rozwiazywane logika. MAster-subordinate tutaj jeden serwis moze modyfikowac dane, a reszta ma je w wersji jedynie read-only.

4 JWT

JSON Web Token- JWT to otwarty standard definiujący bezpieczny sposób komunikacji informacji między senderem a receiverem. Jest stworzony w formacie JSON i podpisany kryptograficznie by potwierdzić ooryingalność. JWT składa się z 3 części.

- Header

- Payload
- Signature

Header zaweira metadane o typie danych i algorytmie używanym do enkrypcji. Składa się z trzech części: metadanych dla tokenu, typu sygnatury i algorytmu kryptograficznego. Składa się z dwóch pól:

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

przykładowy JWT header. Payload reprezentuje faktyczną informację przesyłaną w formacie JSON.

```
{
  "sub": "1234567890",
  "name": "Joydip Kanjilal",
  "admin": true,
  "jti": "cdafc246-109d-4ac9-9aa1-eb689fad9357",
  "iat": 1611497332,
  "exp": 1611500932
}
```

Payload zaweira claimy- customowe i zarezerwowane. Lista zarezerwowanych claimsów.

Active Directory authority Celem będzie stworzenie API autentykowanego JWT.

Klient dostaje token z AD. następnie z każdym połączeniem token jest używany aby auentykować klienta w API. AD jest darmowym featurem azure. Rejestrujemy aplikacje w AD, Audience w JWT to resource ID. Dodajemy konfigurację w configure services. Tenant ID to unikatowy identyfikator acitve directory, idzie do authority

Microsooft identity platform to ewolucja AD i jest nieskoorelowana z ASP.NET Core Identity.

CSRF CSRF- cross site request forgery to atak w którym przeglądarak użytkownika jest używana do postowania forma z jednej strony na inną. Atak przebiega następująco: użytkownik loguje się do banku i dostaje token, nastepnie uzytkownik używa formujlarza na stronie zewnetrznej który używa zapisanego tokenu by zrobić http request do banku. Atak uderza tylko w sposoby weryfikacji automatycznie podlaczajace token, nie dziala w przypadku JWT.

XSS XSS- cross site scripting attack iniekcja w kórej skrypt wchodiz w strone zaufana. Doczytac tu trzeba. Atak pozwala przejac cookeis, session tokens.