Politechnika Łódzka
nstytut Informatyki

Spis treści

1 Wstęp

1.1 Wprowadzenie

W ostatnich latach w świecie IT można było zauważyć trend polegający na promowaniu budowania dużych systemów jako aplikacji rozproszonych składających się z wielu serwisów. Prelegenci na różnych konferencjach programistycznych przepowiadali schyłek wszelkich problemów podczas budowania wielkich systemów wykorzystywanych w biznesie. W dniu dzisiejszym rozwiązanie to określane jest architekturą **mikroserwisową**. Sama koncepcja nie jest czymś zupełnie nowym i możemy określić ją jako rozwinięcie architektury SOA (Service Oriented Architecture). Idea tworzenia oprogramowania w architekturze mikroserwisowej polega na budowaniu niewielkich autonomicznych komponentów z których każdy odpowiada za konkretne zadanie. Elementy te współpracując ściśle ze sobą pozwalają na dostarczenie wymaganej logiki biznesowej.

Wielu architektów oraz zespołów programistów wizja rozbicia swojej monolitycznej aplikacji na architekturę mikroserwisową zachęciła do prób budowania takich rozwiązań. Pomimo początkowego entuzjazmu popartego niezaprzeczalnymi zaletami mikroserwisów takimi jak:

- możliwości stosowania różnych technologi
- skalowalności
- odporności na awarie

dały o sobie znać wady, które spowodowały, że projektowanie, implementacja oraz utrzymanie takiej architektury stało się olbrzymim wyzwaniem dla firm produkujących oprogramowanie. Jednym z największych problemów okazał się sposób komunikacji pomiędzy kolejnymi mikroserwisami. Tym samym projektowanie interfejsów **API** wymagało od architektów oraz programistów rozwiązania problemów w następujących kwestiach:

- wyboru formatu wymiany danych (JSON, XML itp.)

- zaprojektowania ścieżek wywołania serwisów
- obsługi błędów
- wydajności (ilość danych przy jednym wywołaniu serwisu, czas oczekiwania na odpowiedź)
- implementacji uwierzytelniania

Mimo tych przeszkód korporacje pokroju Amazon, Netflix czy Google budują swoje olbrzymie systemy w oparciu o mikroserwisy wykorzystując zróżnicowany stos technologi. Architektura mikroserwisowa przyczyniła się do spopularyzowania takich technologii jak chmura obliczeniowa (Amazon Web Services, Azure itp.) oraz rozwiązań opartych na kontenerach (Docker, Kubernetes).

Technologiczny potentat jakim jest niewątpliwe firma Google opierając swój biznes na usługach sieciowych stworzyła w tym celu technologię opartą na protokole RPC (Remote Procedure Call), która miałaby być panaceum na wyżej wymienione problemy w stosunku do "klasycznego" podejścia opartego na technologi REST.

1.2 Cel i założenia pracy

Celem niniejszej pracy jest zaprojektowanie oraz implementacja scenariuszy, które symulowałyby typowe problemy z jakimi spotykamy się podczas tworzenia architektury mikroserwisowej. Założenia każdego scenariusza obejmują:

- identyfikację oraz opis rozpatrywanego problemu,
- implementację mikroserwisów w technologiach RPC oraz REST,
- ocenę efektywności zastosowanych rozwiązań popartą wnioskami lub pomiarami

Scenariusze zostały opracowane tak, aby zaprezentować podstawowe problemy w projektowaniu oraz implementacji mikroserwisów i nie wyczerpują każdego możliwego aspektu.