



Mikroserwisy i SOA

Adam Jaworski, Dawid Gruszecki



Agenda

1. Wprowadzenie do SOA
2. Omówienie funkcji ESB
3. Zalety i wady Service Oriented Architecture
4. Mikroserwisy
5. Porównanie mikroserwisów z SOA
6. Przykłady wdrożenia



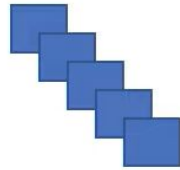
SOA - service oriented architecture

- koncepcja tworzenia systemów informatycznych, w której główny nacisk stawia się na definiowanie usług
- aplikacje składają się z luźno powiązanych, interoperacyjnie powiązanych usług
- obejmuje zestaw metod organizacyjnych i technicznych mających na celu powiązanie biznesowej strony organizacji z jej zasobami informatycznymi

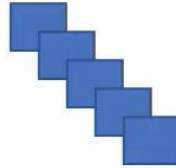
Google Map
Service

Payment
Service
(PayPal)

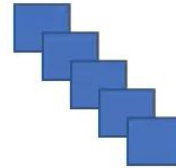
Login Service
(Facebook)



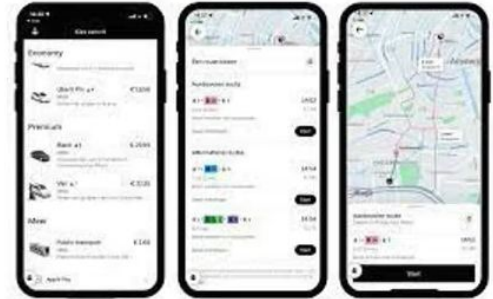
Service 1



Service 2



Service 3

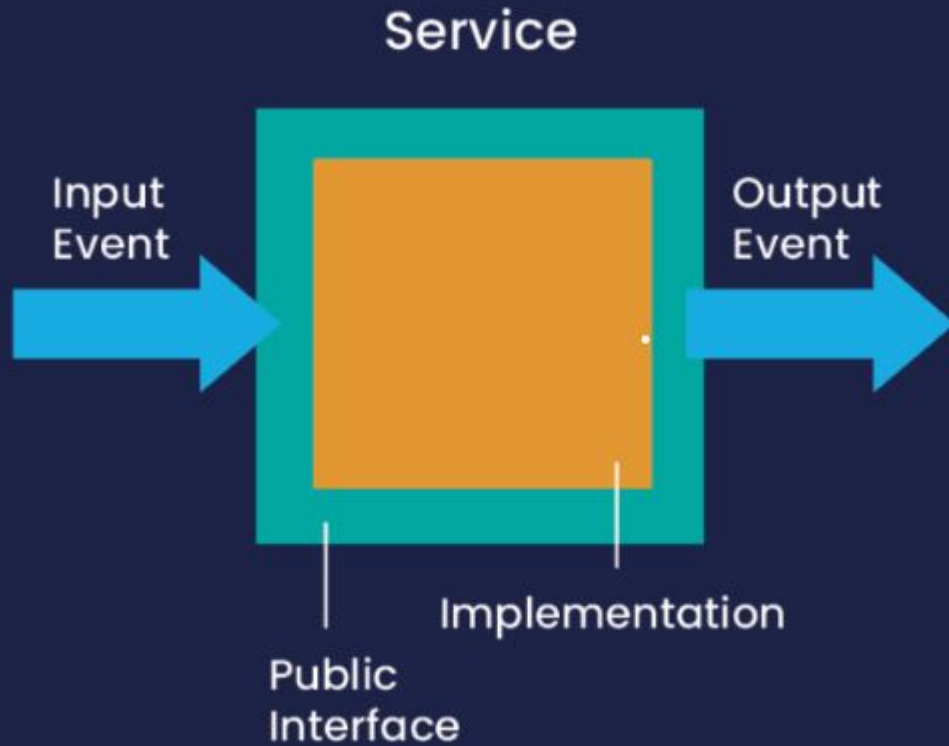


Uber App

Definicja usługi

- każdy element oprogramowania mogący działać niezależnie od innych, posiadający zdefiniowany interfejs, który udostępnia realizowane funkcje
- może składać się z innych usług
- “czarna skrzynka” dla konsumenta
- jest samowystarczalna







Podstawowe cechy

- interfejsy pracują w trybie modelu RPC (remote procedure call) - co pozwala na wywoływanie zdalnych procedur bez konieczności wiedzy o protokole komunikacyjnym czy typie danych
- wiadomości zawierają dane w formacie XML
- interfejsy mogą obsługiwać dwa typy transmisji - synchroniczny i asynchroniczny


Typy usług oferowanych przez SOA

- usługi funkcjonalne - odpowiadające za realizację
- usługi dla przedsiębiorstw - służące implementacji funkcjonalności
- usługi aplikacji - wykorzystywane do opracowywania i wdrażania aplikacji
- usługi infrastrukturalne - instrumentalne dla procesów backendowych



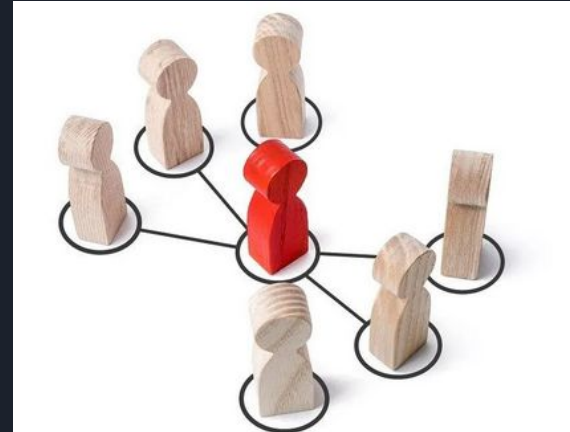
Pryncypia dla Service Oriented Architecture

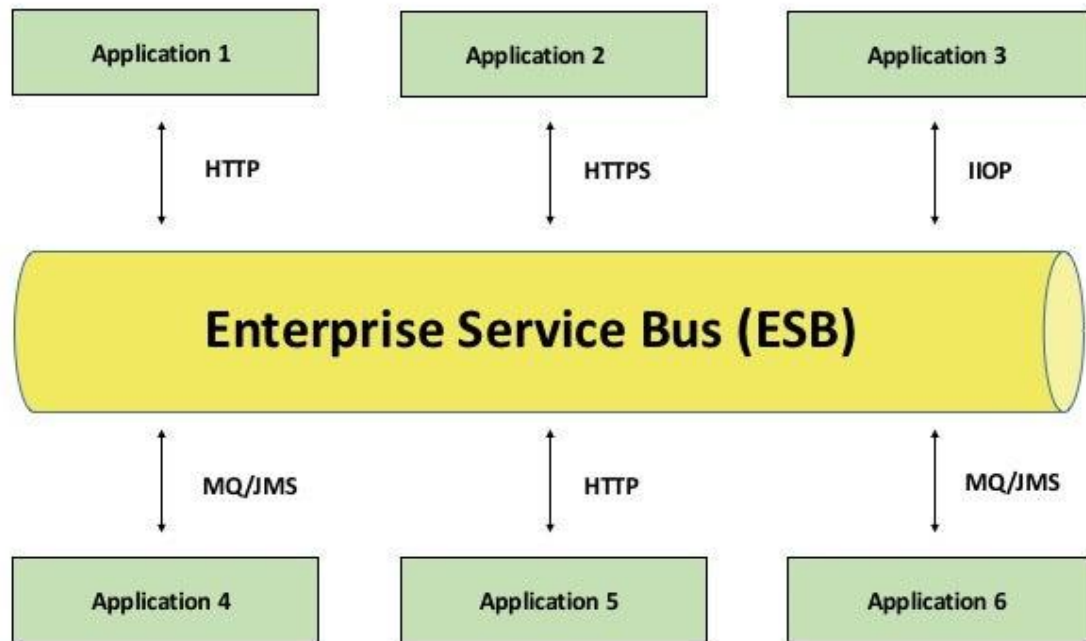


- 
- **standaryzacja opisów usług** - według jednolitego szablonu
 - **luźne powiązanie** - niezależność usług pomiędzy sobą na poziomie implementacyjnym
 - **abstrakcja usług** - usługi enkapsulują sposób realizacji swojej funkcjonalności
 - **reużywalność usług** - poprzez odpowiednia dekompozycje logiki
 - **bezstanowość usług** - poprzez nie przechowywanie żadnego stanu między kolejnymi żądaniami klienta
 - **autonomia usług** - każda usługa odpowiedzialna za swoje działanie

ESB - Enterprise Service Bus

- oprogramowanie pośredniczące, które umożliwia integrację różnorodnych aplikacji i usług w jednym scentralizowanym środowisku
- umożliwia komunikację oraz wymianę danych pomiędzy systemami
- zapewnia funkcje routingu, transformacji danych
- pełni funkcje monitorowania i zarządzania transakcjami, co umożliwia kontrolę i śledzenie przepływu danych







Zalety SOA

- możliwość ponownego (wielokrotnego) użycia zaimplementowanych usług
- łatwa integracja między różnymi aplikacjami i systemami
- elastyczność oraz skalowalność dzięki rozdzieleniu na mniejsze, łatwiejsze do zarządzania komponenty
- modułowa struktura ułatwiająca rozwój aplikacji i jej przyszłe utrzymanie
- możliwość zróżnicowania dostawców poszczególnych komponentów oprogramowania



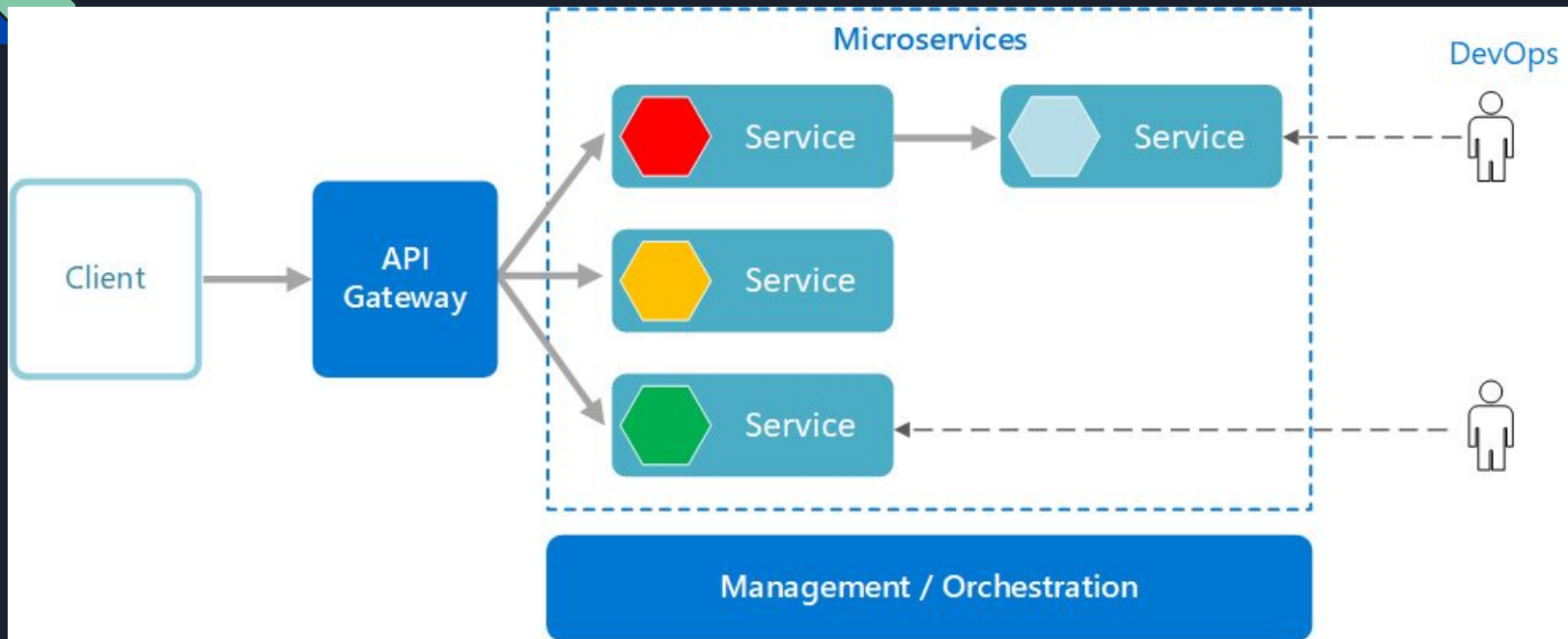
Wady SOA

- złożoność wdrażania i zarządzania wymagająca wielu zasobów
- potencjalne zwiększenie ruchu sieciowego
- ESB jako pojedynczy punkt awarii - potrzeba odpowiedniej redundancji i monitorowania
- trudność w śledzeniu i debugowaniu - spowodowane rozbiciem systemu na mniejsze usługi
- potrzeba zapewnienia bezpieczeństwa ze względu na zdecentralizowaną naturę SOA



Architektura Mikroserwisowa

- styl projektowania aplikacji, który polega na podziale całości na mniejsze, niezależne moduły, które komunikują się ze sobą za pomocą API
- Każdy mikroserwis jest odpowiedzialny za wykonanie określonej funkcji i może być rozwijany, wdrażany oraz skalowany niezależnie od innych.





Zalety architektury mikroservisowej

- Elastyczność wdrażania
- Skalowalność
- Odporność na błędy
- Technologiczna różnorodność
- Łatwiejsze zarządzanie



Wady architektury mikroservisowej

- Złożoność zarządzania
- Zwiększone nakłady na komunikację
- Redundancja danych
- Trudności z transakcjami
- Koszty infrastrukturalne



SOA

vs



Microservices

Architektura




SOA

Coarse - Grained



Microservices

Fine - Grained

- 
1. SOA - architektura gruboziarnista:
 - a. aplikacja podzielona na duże, niezależne usługi
 - b. większa złożoność i awaryjność

 2. Mikroserwisy - architektura drobnoziarnista:
 - a. podział na małe, autonomiczne serwisy
 - b. większa elastyczność w wdrażaniu



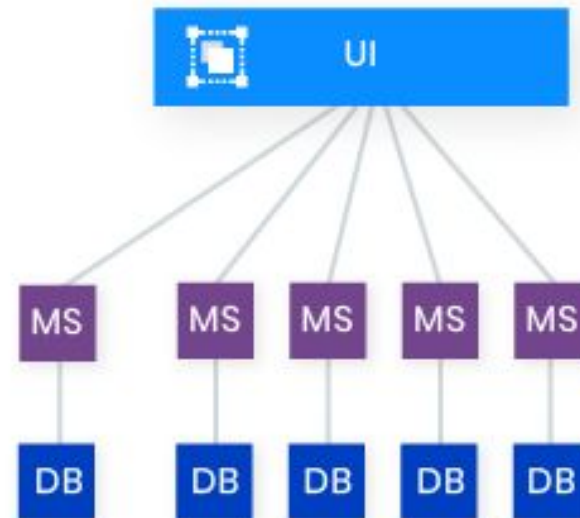
Komunikacja



1. Mikroserwisy:
 - a. każda usługa jest opracowywana niezależnie, z własnym protokołem komunikacyjnym
 - b. w celu utrzymania prostoty, wykorzystują lekkie protokoły komunikacyjne takie jak HTTP/REST i JMS
2. SOA:
 - a. zarządza i koordynuje usługi, które świadczy przy pomocy ESB
 - b. otwartość na heterogeniczne protokoły takie jak SOAP, AMQP



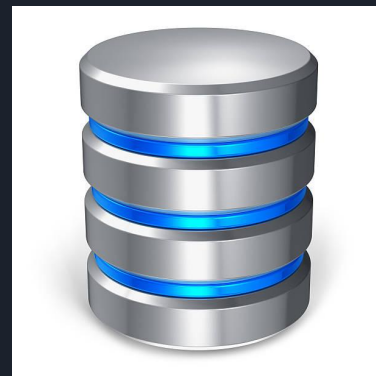
Service - Oriented



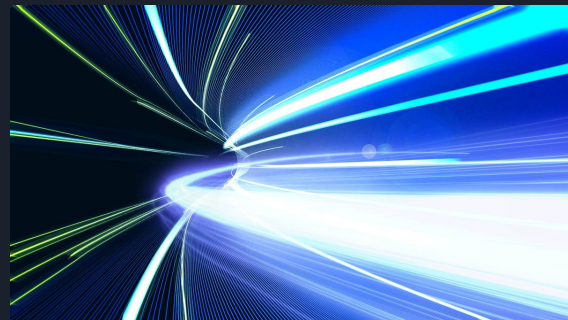
Microservices

Przechowywanie

1. każdy mikroservis może mieć swoją bazę danych co pozwala na:
 - a. niezależne zarządzanie danymi przez poszczególne serwisy
 - b. elastyczność w wyborze typu bazy danych
2. architektura SOA obejmuje pojedynczą, współdzieloną przez wszystkie usługi warstwę przechowywania danych
 - a. łatwość w integracji danych
 - b. większa zależność między usługami



Szybkość działania



1. SOA:

- a. Skupia się na większych, złożonych usługach
- b. Architektura oparta na usługach obejmuje bardziej ogólne funkcjonalności
- c. Modyfikacje i wdrożenia mogą być bardziej skomplikowane ze względu na rozległość usług

2. Mikrouługi:

- a. Koncentrują się na mniejszych, bardziej jednoznacznych funkcjonalnościach
- b. Każda mikrousluga jest autonomicznym komponentem, co umożliwia szybsze iteracje
- c. Modyfikacje, testowanie i wdrażanie mogą odbywać się niezależnie od innych usług, przyspieszając proces rozwoju



Niezawodność i zarządzanie awariami

1. SOA:
 - a. złożona natura usług sprawia, że awaria jednej usługi wpływa na cały system
 - b. zarządzanie awariami może być skomplikowane ze względu na zależności pomiędzy usługami
 - c. wymaga starannego planowania i implementacji strategii odzyskiwania po awarii
2. Mikroserwisy:
 - a. dzięki autonomiczności mamy ograniczony wpływ awarii na cały system
 - b. możliwość izolacji i szybkiej reakcji na awarie sprawia, że zarządzanie nimi jest bardziej elastyczne



Przykłady wdrożenia



MIKROSERWISY



amazon



Uber

NETFLIX

ebay



SOA

IBM



Azure

ORACLE