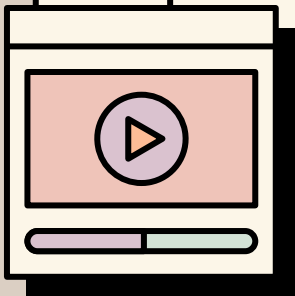
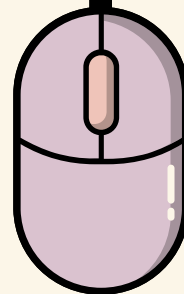
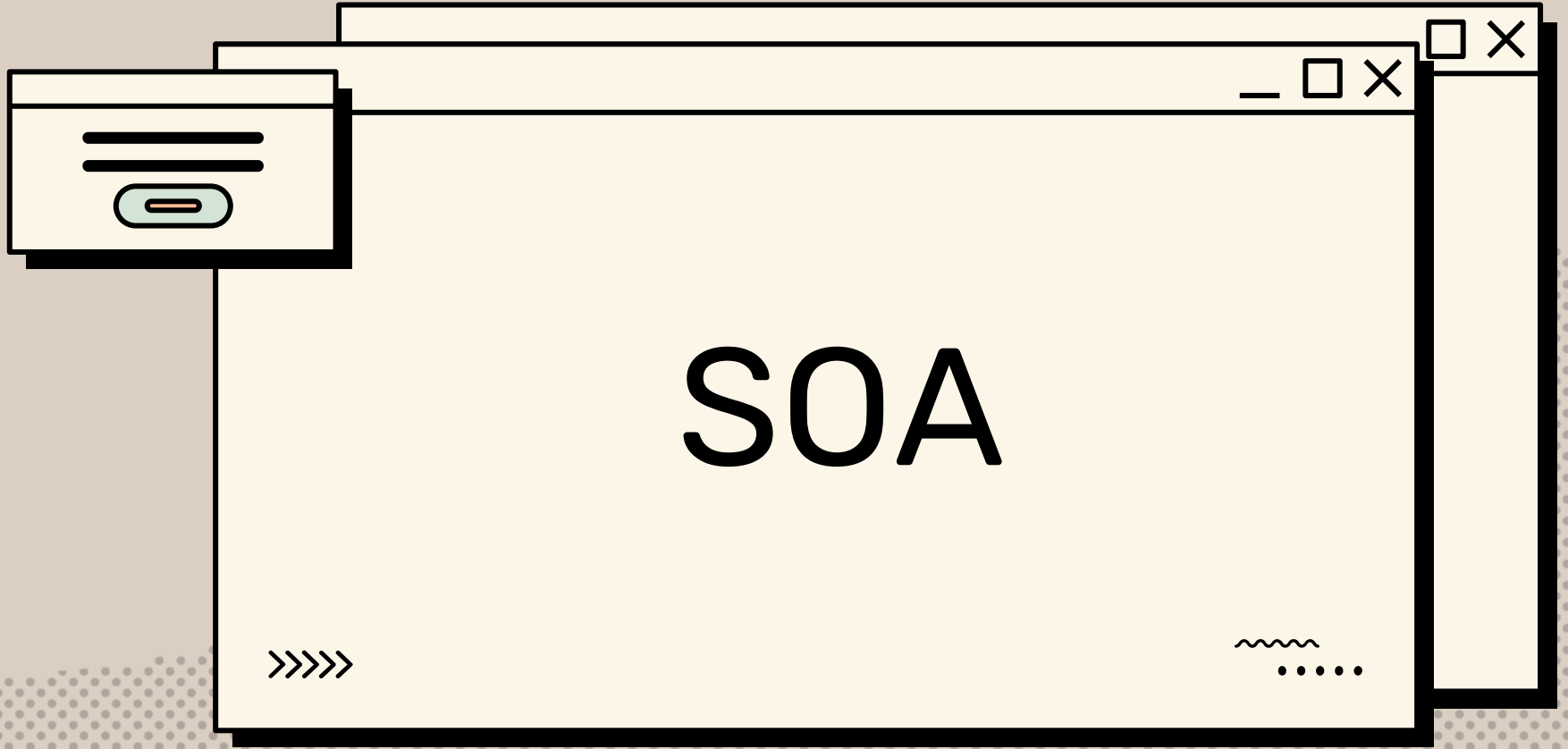


# SOA i Mikroserwisy

Mateusz Sznurawa, Michał Chmiel





.....

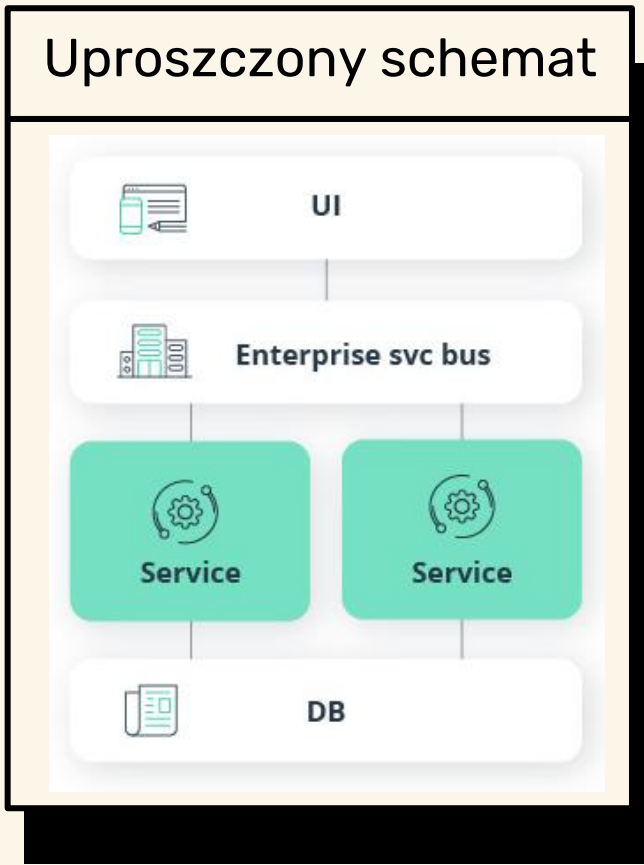
# SOA - Service-oriented architecture

Strukturalna koncepcja tworzenia oprogramowania komputerowego, w której aplikacje są rozbijane na mniejsze, modularne usługi, wykorzystywane później przez różne części systemu. Usługi komunikują się ze sobą przez standardowe interfejsy, dzięki protokołom komunikacyjnym. Architektura ta może być implementowana na różnych platformach i z użyciem różnych technologii.



>>>>

## Uproszczony schemat



# Podstawowe założenia



<b>Modularność</b>	Usługi w SOA powinny być projektowane jako niezależne moduły, które następnie łączą się i modyfikują, nie ingerując w inne części systemu
<b>Interoperacyjność</b>	Wykorzystując standardowe protokoły i interfejsy usługi w SOA mają możliwość współpracy z różnymi systemami i technologiami, co zapewnia łatwiejszą integrację i rozbudowę systemów informatycznych
<b>Reusability</b>	Usługi SOA powinny być projektowane tak, by można było je wielokrotnie wykorzystać w różnych konfiguracjach i kontekstach, co pozwala zaoszczędzić czas i zasoby
<b>Skalowalność</b>	Architektura SOA umożliwia łatwe skalowanie systemów informatycznych w zakresie liczby usług i wydajności, co w obliczu rosnących potrzeb, np. biznesowych, jest kluczowe



## a poza tym..



### Loose Coupling

Minimalizacja zależności między serwisami

### Service Abstraction

Ukrywanie szczegółów wewnętrznych implementacji serwisu

### Service Statelessness

Serwisy nie powinny przechowywać żadnego stanu (danych) pomiędzy żądaniem klienta

### Service Autonomy

Serwis ma pełną kontrolę nad tym co enkapsuluje

### Service Composability

Rozbijanie większych problemów na mniejsze



# Protokoły i standardy



SOAP – Simple Object Access  
Protocol

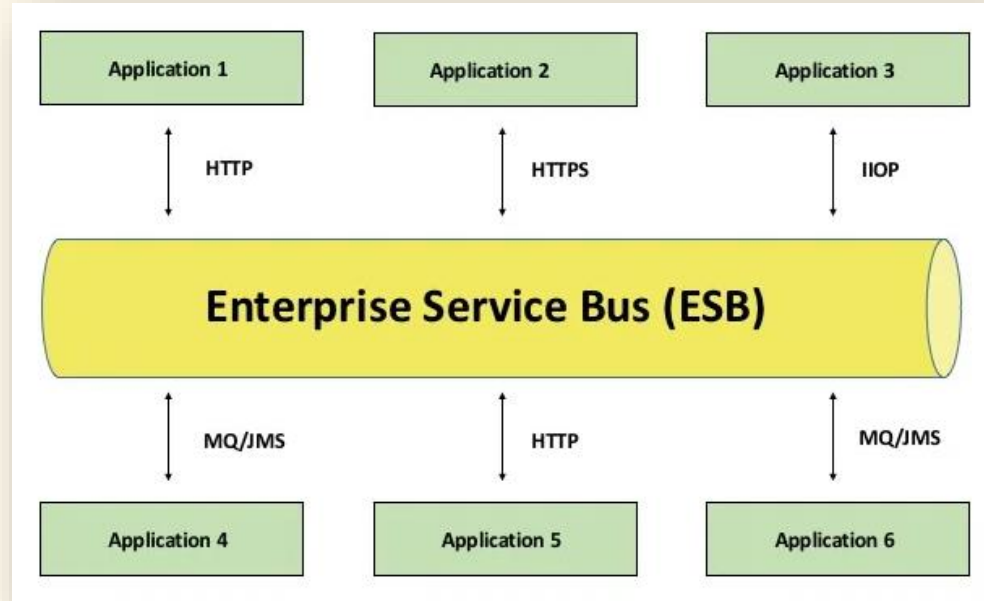
REST – Representational State  
Transfer

WSDL – Web Services  
Description Language)

UDDI – Universal Description,  
Discovery, and Integration

JSON – JavaScript Object  
Notation

# ESB





## ESB



ESB stanowi rodzaj „komunikacyjnego mostu” pomiędzy różnymi usługami architektury SOA. Jego główne zadanie to umożliwienie komunikacji i wymiany danych, niezależnie od technologii czy języka programowania.

# Zalety

>>>>

- SOA nie jest związane z konkretną technologią czy platformą, co otwiera możliwości implementacji (to bardziej szkoła myślenia, paradygmat)
- Zaimplementowane usługi można wielokrotnie wykorzystać
- Spójność między aplikacjami i systemami
- Łatwość wprowadzania zmian w module bez wpływu na wiele aplikacji
- Modułowa struktura ułatwia rozwój aplikacji i jej proste skalowanie
- Dostawców poszczególnych komponentów oprogramowania można zróżnicować



# Wady

- Złożoność – architektura SOA może być trudna do zrozumienia i wdrożenia
- Implementacja tego rozwiązania jest obciążona na początku dużymi kosztami
- ESB jako centralny punkt komunikacyjny może ulec awarii, wymaga więc redundancji
- Rozbicie systemu na wiele modułów może sprawiać trudności w debugowaniu
- Komunikacja między serwisami może generować duży ruch sieciowy





The illustration depicts a web browser interface. On the left, a sidebar contains a hamburger menu icon (three horizontal lines) and a rounded rectangular button with a teal-to-orange gradient. The main content area is a large rectangle with a thick black border. At the top right of this area are window control icons: a minus sign, a square, and an 'X'. The word 'Mikroserwis' is centered in the main area in a large, black, sans-serif font. In the bottom left corner of the main area are five right-pointing chevrons '>>>>>'. In the bottom right corner are a wavy line and five dots '.....'. The background is a light beige color with a grey dotted pattern at the bottom.

# Mikroserwis

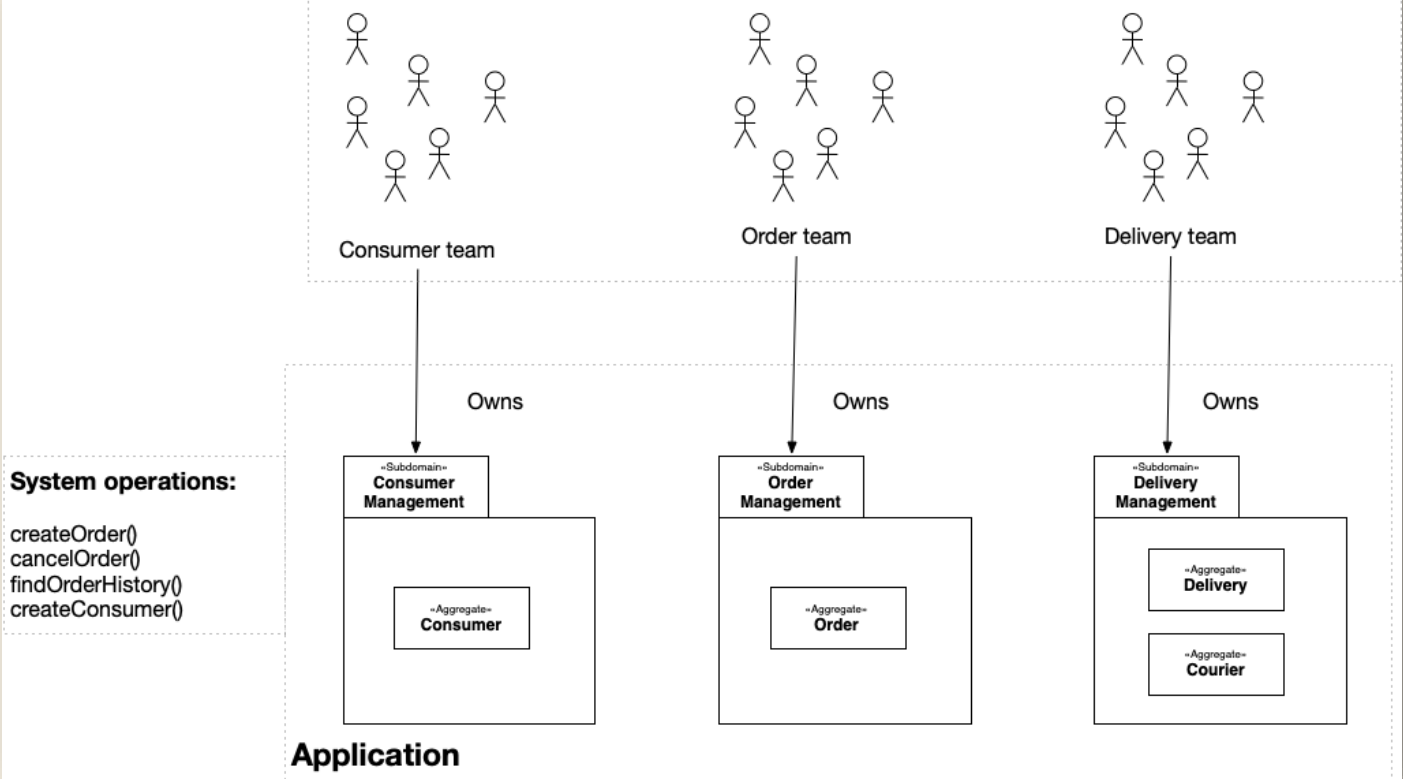
>>>>>

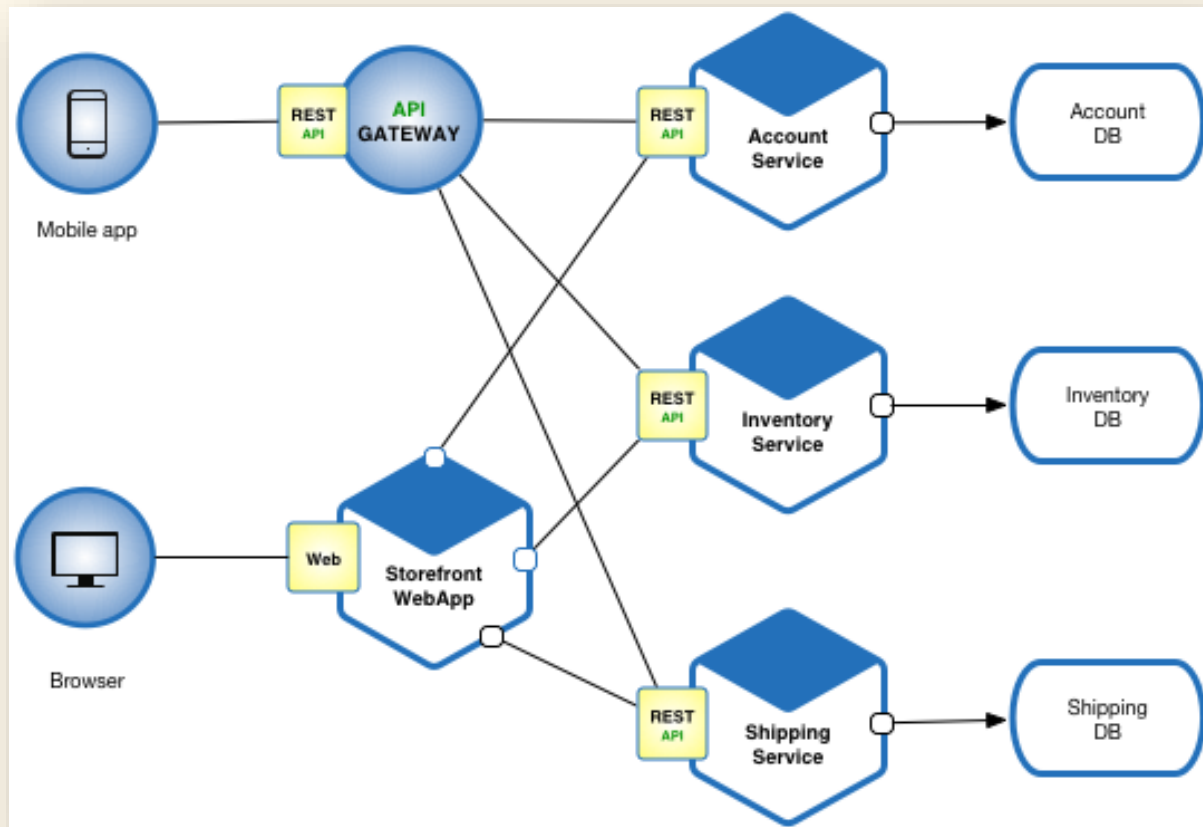
.....

# Architektura mikroservisowa

Architektura mikroservisowa to podejście architektoniczne, które zakłada budowanie aplikacji w oparciu o małe, niezależne i samodzielne serwisy, zwane mikroservisami. Każdy mikroservis jest odpowiedzialny za realizację jednej, dobrze zdefiniowanej funkcji biznesowej.

### DevOps (Stream-aligned) teams







# Cechy architektury mikroservisowej

- Rozbicie na serwisy
- Samodzielność
- Komunikacja przez API
- Elastyczność technologiczna





# ZALETY

- Skalowalność
- Łatwość zarządzania
- Technologiczna różnorodność
- Wysoka dostępność



# WADY

- Złożoność komunikacji
- Nadmierna fragmentacja
- Nadmierny narzut

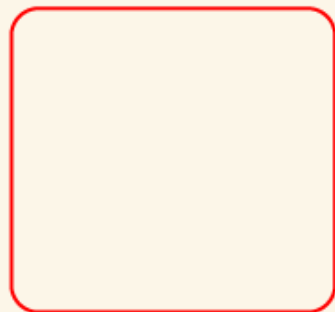


# Porównanie

>>>>

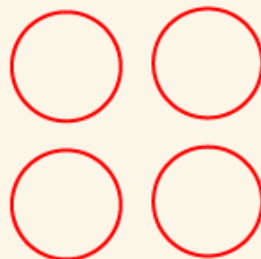
~~~~~  
.....

## Monolithic Vs SOA Vs Microservices



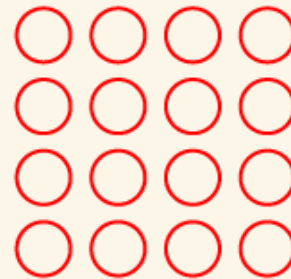
**Monolithic**

Single Unit



**SOA**

Coarse-grained



**Microservices**

Fine-grained

# Data Storage

.....

~~~~~  
>>>>>

## SOA

- SOA przewiduje pojedynczą, współdzieloną warstwę do przechowania danych przez różne usługi.
- Dane często przechowywane w jednym centralnym repozytorium
- Duży nacisk kładziony jest na maksymalizację ponownego wykorzystania danych, co może prowadzić do ich większej spójności w całym systemie

## Mikroserwisy

- Każdy mikroserwis ma własną przestrzeń do przechowywania danych, związany z jego funkcjonalnością.
- Nacisk położony jest na niezależność danych, która daje większą elastyczność i lepszą skalowalność



.....

# Komunikacja

~~~~~  
>>>>

SOA

Mikroserwisy



- Wykorzystujemy scentralizowany kanał ESB, by różnymi protokołami łączyć ze sobą serwisy
- Protokoły obejmują np. SOAP, AMQP, MSMQ
- W przypadku awarii ESB skutki tego odczuje cały system

- Wykorzystuje się proste protokoły komunikacyjne jak HTTP/REST, JMS,
- Interfejsy API wykorzystywane w tej architekturze pozwalają na bezpośrednią wymianę danych między serwisami, bez centralnego kanału komunikacji.
- Taka kompleksowość komunikacji może być jednak wyzwaniem jeśli chodzi o zarządzanie

.....

# Szybkość działania

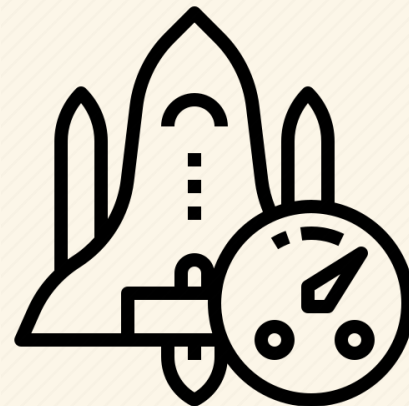
~~~~~  
>>>>>

## SOA

- Ze względu na scentralizowany charakter komunikacyjny (ESB) czas odpowiedzi może być dłuższy.
- SOA może oferować dobrą szybkość działania przy prostej implementacji, która spada jednak wraz z dodawaniem kolejnych serwisów

## Mikroserwisy

- Komunikacja między serwisami odbywa się bezpośrednio, z użyciem prostych mechanizmów komunikacyjnych, może być więc szybka
- Każdy serwis jest niezależny i ma własne zasoby, można więc optymalizować system skalując go



.....

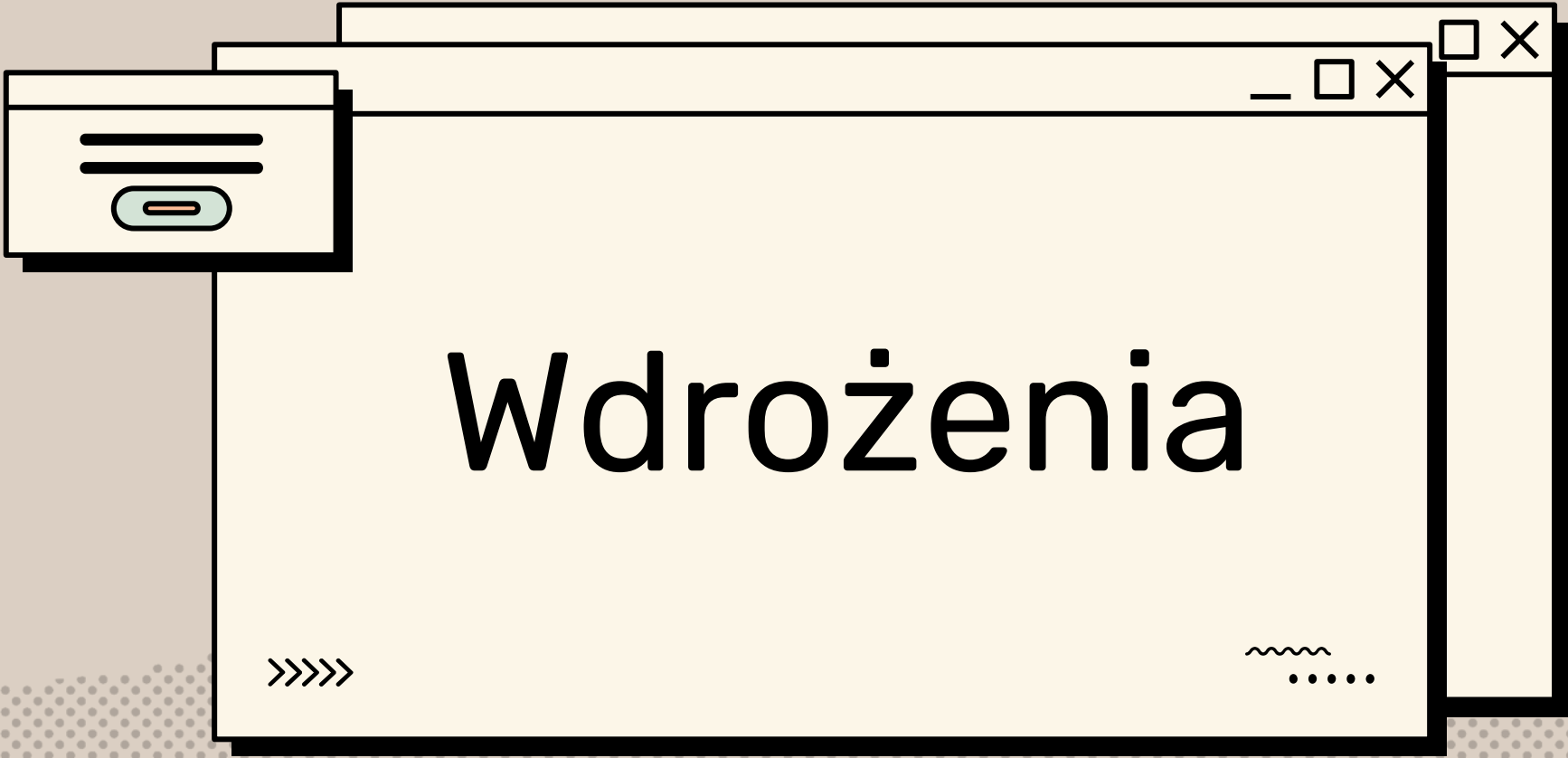
## SOA

## Mikroserwisy ~~~~

- Aplikację wdraża się zwykle jako pojedynczą jednostkę
- Występuje loose coupling, ale z powodów współdzielonych zasobów jest mniejszy niż w mikroserwisach
- Lepsze rozwiązanie dla złożonych przedsiębiorstw korzystających z reusability, oraz dla scentralizowanych zespołów z przemyślaną strategią rozwoju i zarządzania zasobami.

- Każdy serwis można wdrażać i rozwijać osobno
- Loose coupling z minimalnymi zależnościami między serwisami
- To rozwiązanie dobrze sprawdzi się w organizacjach stawiających na dynamiczny rozwój i elastyczność, chcących ograniczyć wpływ awarii na cały system.





# Wdrożenia



# Mikroserwisy



Spotify




Netflix



Uber



# SOA



eCommerce



Oracle

