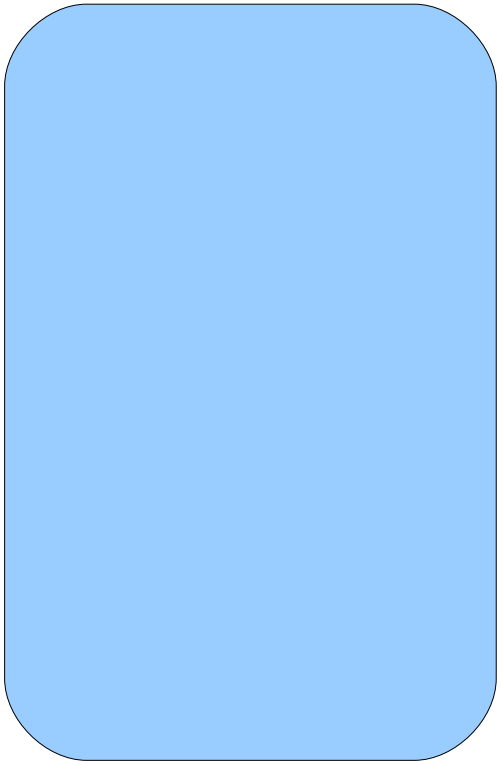
Metodyka tworzenia portali biznesowych

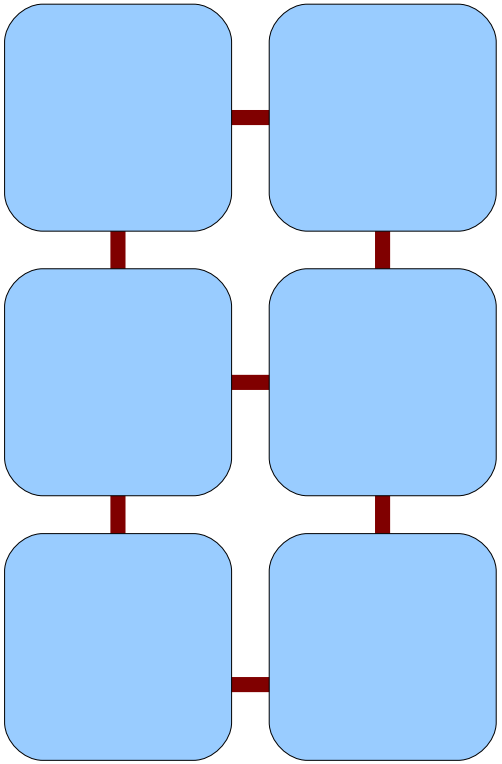
WYKŁAD 10

Aplikacje webowe

Aplikacja monolityczna



Aplikacja jako zestaw usług



Aplikacje webowe

- Aplikacja monolityczna – pojedyncza aplikacja działająca w ramach jednego procesu o strukturze wewnętrznej niekoniecznie monolitycznej (tj. mogąca składać się z kilku warstw, usług, bibliotek statycznych lub dynamicznych, itp.).

Aplikacje webowe

- Podstawowe wady aplikacji monolitycznych:
  - nieefektywne skalowanie poziome, tj. najczęściej tylko wybrane składniki aplikacji wymagają skalowania,
  - modyfikacje pojedynczego składnika aplikacji wymagają ponownego wdrożenia i przetestowania całej aplikacji.

Aplikacje webowe jako zestaw usług

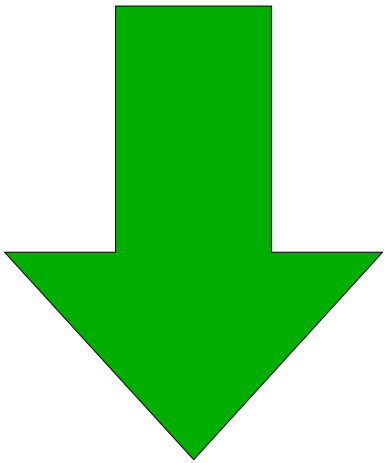
- ZALETY:
  - łatwiejsza modyfikacja aplikacji
    - wystarcza refaktoryzacja i ponowne wdrożenie wybranych usług zamiast wdrażania nowej wersji aplikacji monolitycznej
  - łatwiejsze utrzymanie aplikacji
    - awarie poszczególnych składowych nie skutkują awarią całości systemu
  - łatwiejsze skalowanie aplikacji (skalowane są tylko usługi, które tego wymagają)

Aplikacje webowe jako zestaw usług

- ZALETY (cd.):
  - eliminacja powtarzających się prac nad aplikacjami w przypadku usług współdzielonych
    - zamiast tworzenia monolitycznych aplikacji przez różne zespoły można skomponować aplikacje z istniejących i wdrożonych już usług
  - usprawnienie danej usługi powoduje, że wszystkie aplikacje z niej korzystające zyskują na tym
  - w przypadku aplikacji mobilnych i webowych można skoncentrować się na warstwie prezentacji

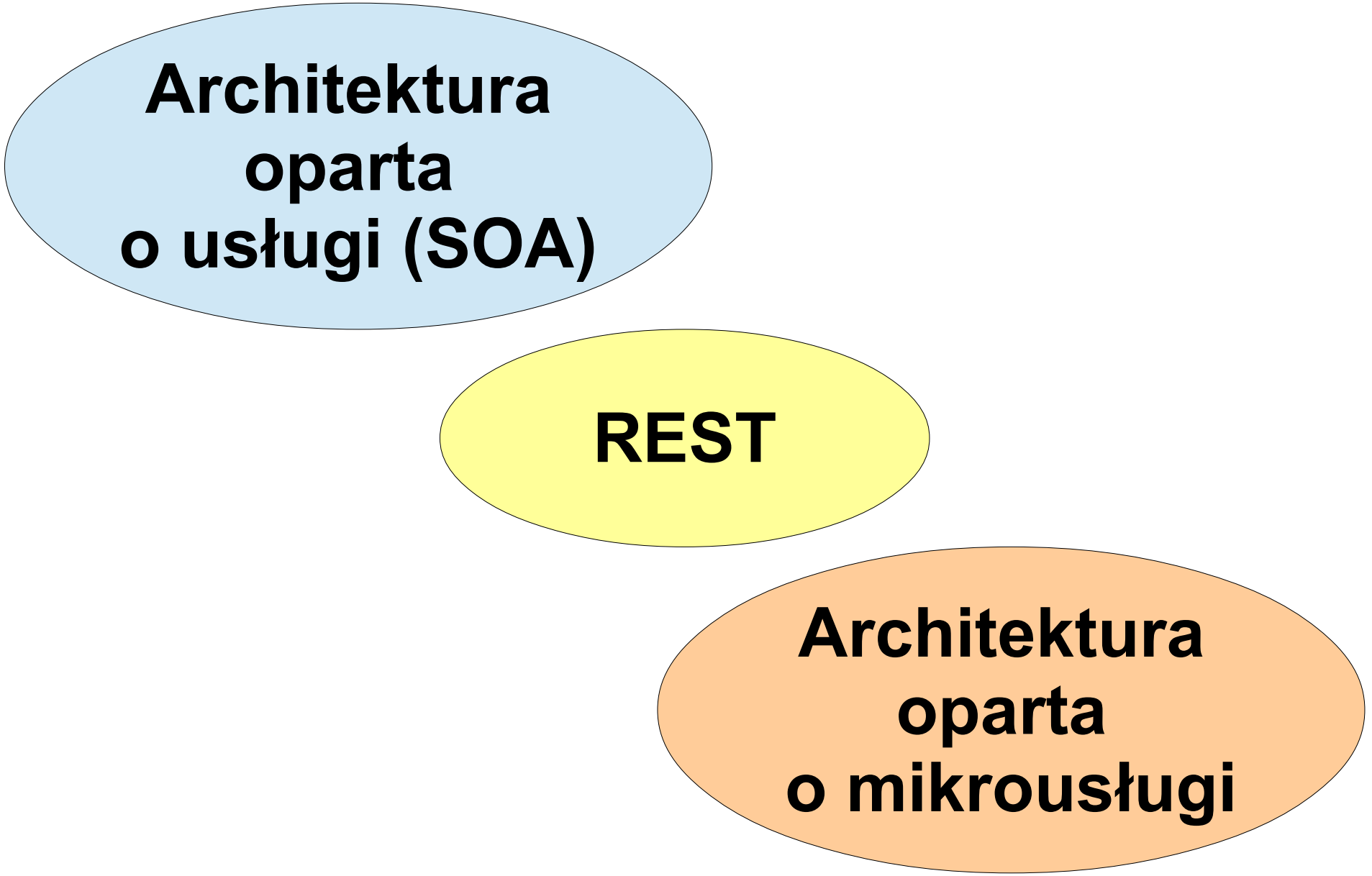
Aplikacje webowe jako zestaw usług

~~Ponowne wykorzystanie kodu~~



Wykorzystanie istniejących, działających usług

Podejścia do tworzenia aplikacji opartych o usługi



Mikrousługi

- **Architektura oparta o mikrousługi** (**mikroserwisy**) – kolejny etap w sposobie wytwarzania oprogramowania nawiązujący do architektury opartej o usługi (SOA).
- Termin „Micro-Web-Services” pojawił się w 2005 roku.
- Koncepcja mikrousług została zaprezentowana na konferencji GeeCon 2013.

Mikrousługi

- **Architektura oparta o mikrousługi** (**mikroserwisy**) – sposób projektowania aplikacji jako autonomicznych usług, które mogą być:
  - implementowane,
  - testowane,
  - wersjonowane,
  - wdrażane

  niezależnie.

Mikrousługi

- Idea mikrousług oparta jest na filozofii systemu Unix:

„Do one thing and do it well”

- Najważniejsze prawo przy budowaniu aplikacji składającej się z mikrousług:

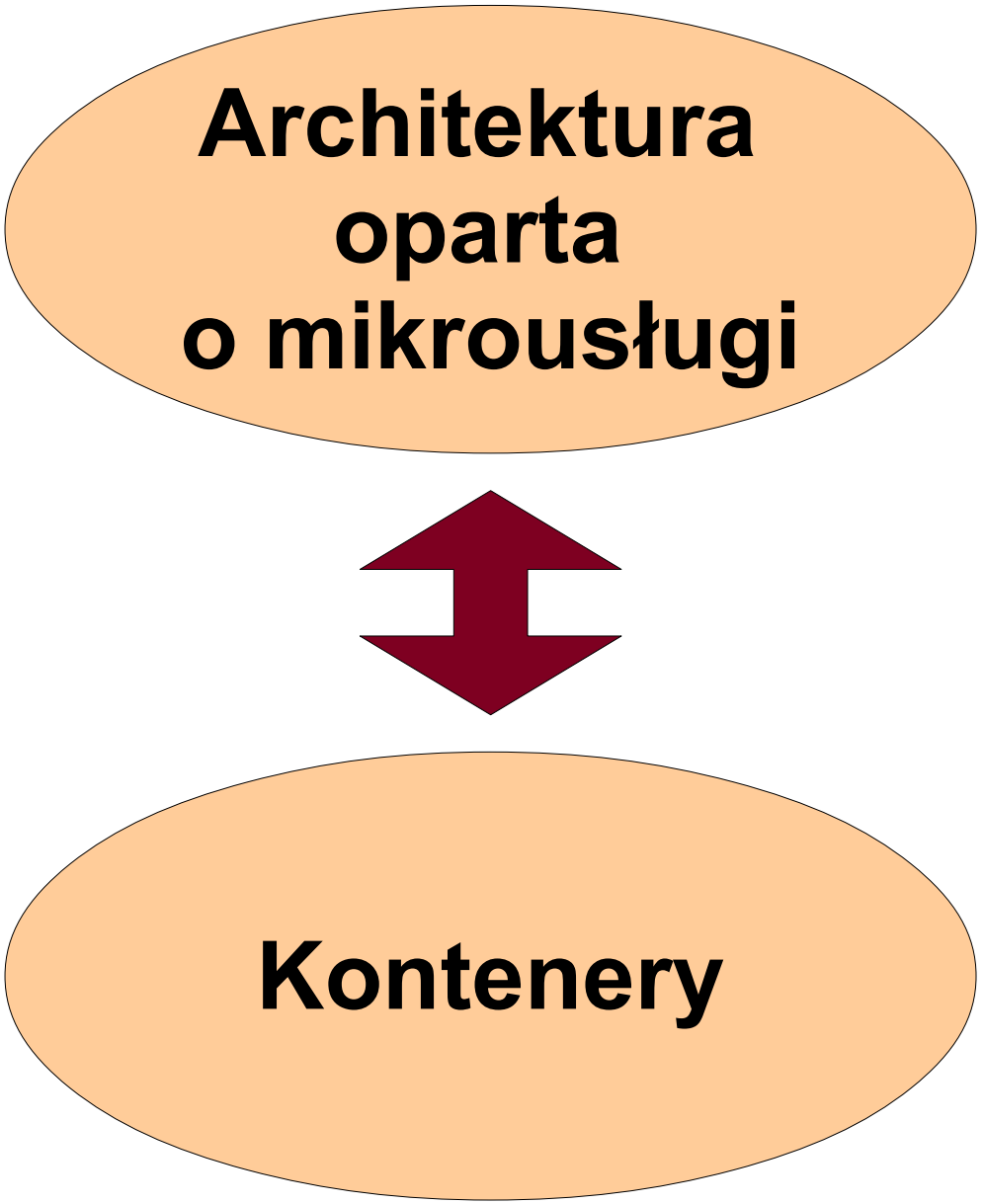
„*High in cohesion, low in coupling*”

(mikrousługi wysoce spójne, luźne powiązanie pomiędzy mikrousługami)

Mikrousługi

- Mikrousługi stanowią elementy składowe aplikacji rozproszonych.
- Aby efektywnie uruchomić aplikację rozproszoną można wykorzystać mikrousługi dostępne jako rozwiązania w chmurze.

Mikrousługi



Architektura mikroserwisów w połączeniu z konteneryzacją segreguje funkcjonalność aplikacji w mniejsze usługi (miroserwisy).

Mikrousługi

- ZALETY MIKROUSŁUG W STOSUNKU DO SOA:
  - prostsza architektura
  - zasadniczo każda mikrousługa realizuje pojedynczą funkcję (usługi sieciowe w architekturze SOA mogły być dowolnej wielkości)
  - możliwość wykorzystania automatycznie skalującej się infrastruktury chmurowej (ma to istotne znaczenie przy współdzieleniu usług)
  - nowe technologie kontenerów (np. Docker)
  - wykorzystanie prostych, standardowych narzędzi webowych

Mikrousługi

- PROBLEMY ROZWIĄZYWANE W OSTATNICH LATACH:
  - uruchamianie wielu instancji mikrousług
  - odkrywanie usług wzajemnie (*Service Discovery*)
  - rozproszone logowanie
  - korelacja logów

Mikrousługi

- Wysoka spójność mikrousług
  - koncepcja nawiązująca do wydzielania w programowaniu fragmentów kodu do funkcji
  - zamykanie jednej i wyodrębnionej funkcji w konkretnej mikrousłudze
  - łatwe zrozumienie co jest treścią mikrousługi i jaka jest jej kluczowa odpowiedzialność

Mikrousługi

- Luźne powiązanie pomiędzy mikrousługami
  - zmiana implementacji mikrousługi nie pociąga konieczności dokonania wielu zmian w implementacjach innych mikrousług

Mikrousługi

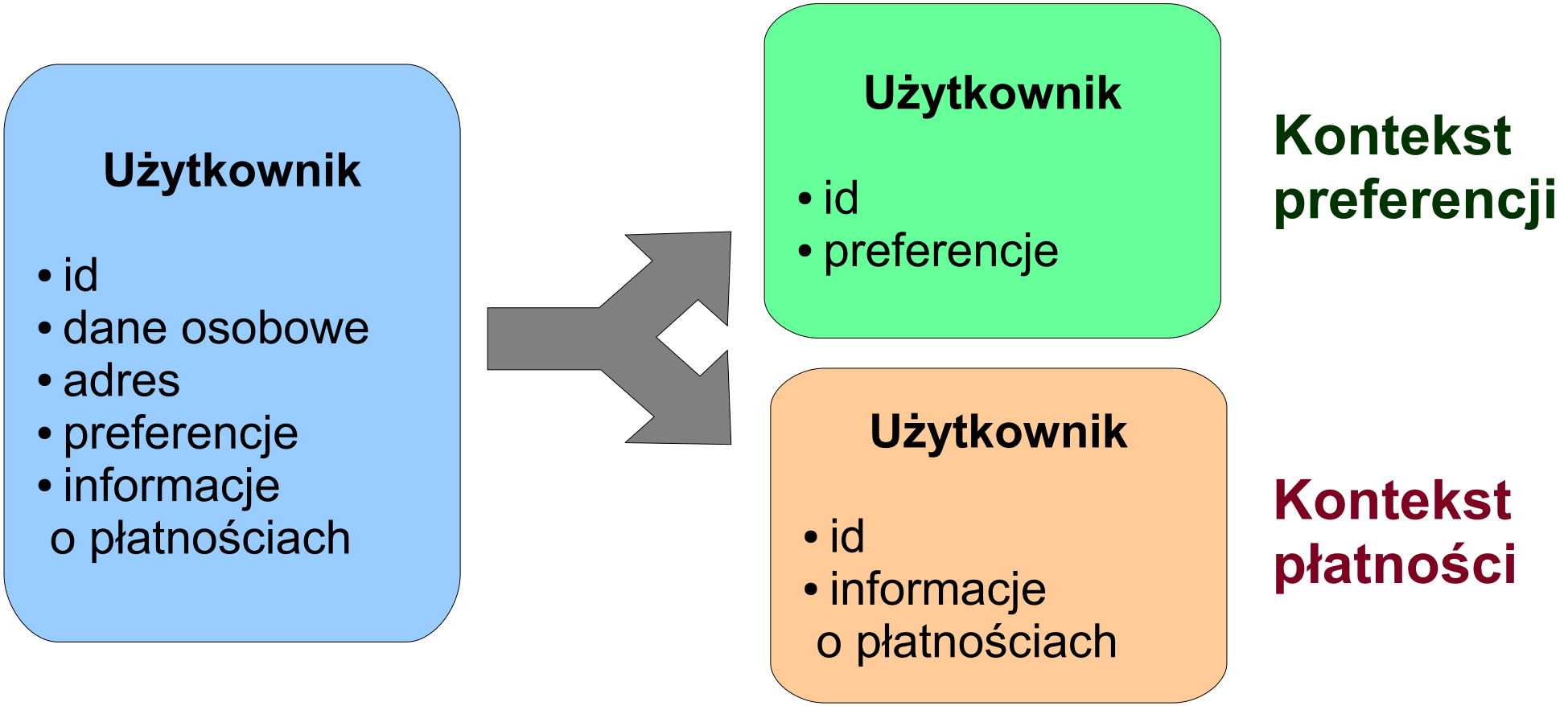
- WAŻNE PROBLEMY:
  - jawne wyznaczenie granic pomiędzy mikrousługami,
  - zapewnienie komunikacji pomiędzy mikrosługami za pomocą odpowiedniego API.

Mikrousługi

- Granice pomiędzy mikrousługami
  - identyfikacja oddzielnych kontekstów, np. użytkownik portalu e-commerce może być elementem kontekstu preferencji oraz kontekstu płatności,
  - każdy z kontekstów powinien wykorzystywać tylko specyficzne atrybuty użytkownika.

Mikrousługi

- Granice pomiędzy mikrousługami
  - przykład rozdzielenia modelu użytkownika ze względu na różne konteksty:



Uruchamianie mikrousług

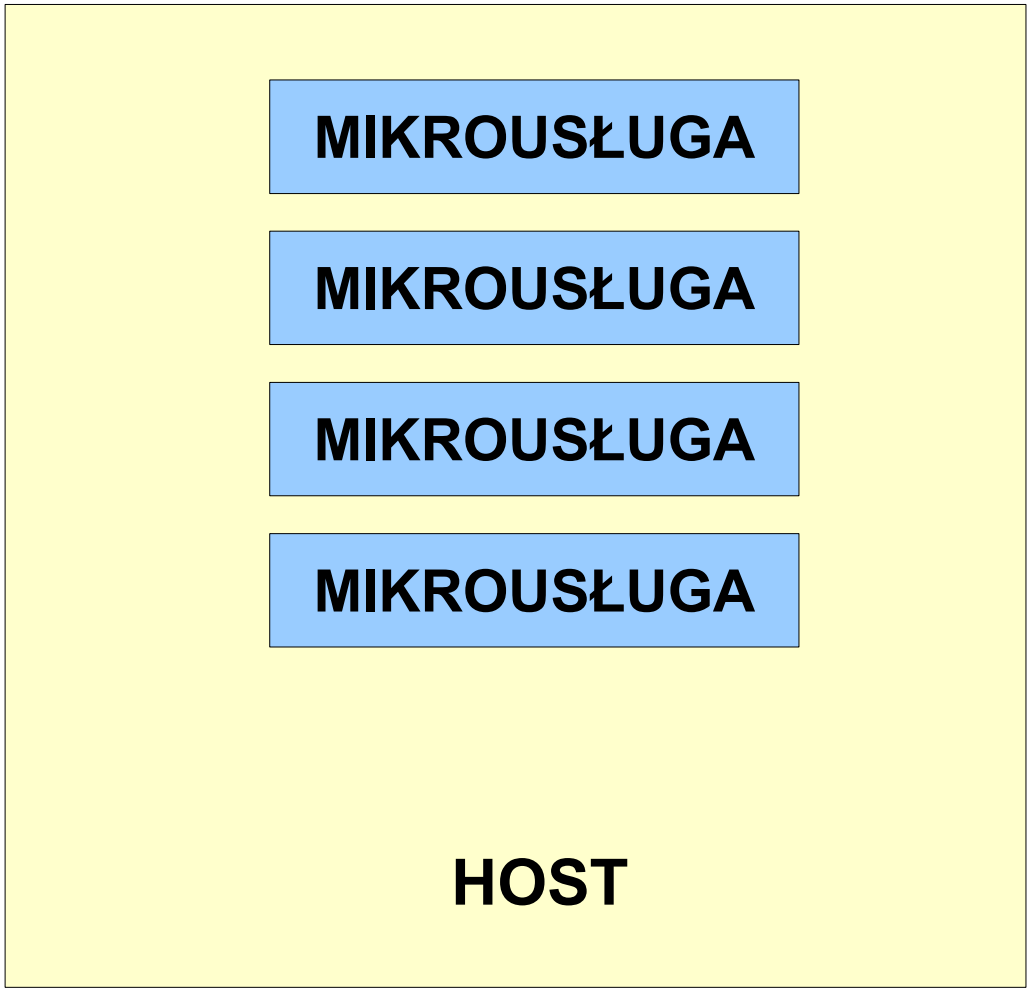
- Pytanie związane z uruchamianiem mikrousług:

  Ile mikrousług powinno działać na jednej maszynie?

- Terminologia:
  - host – system operacyjny, na którym można zainstalować i uruchomić usługi
    - przy braku wirtualizacji: maszyna fizyczna ↔ host

Uruchamianie mikrousług

- Wiele mikrousług na jednym hoście

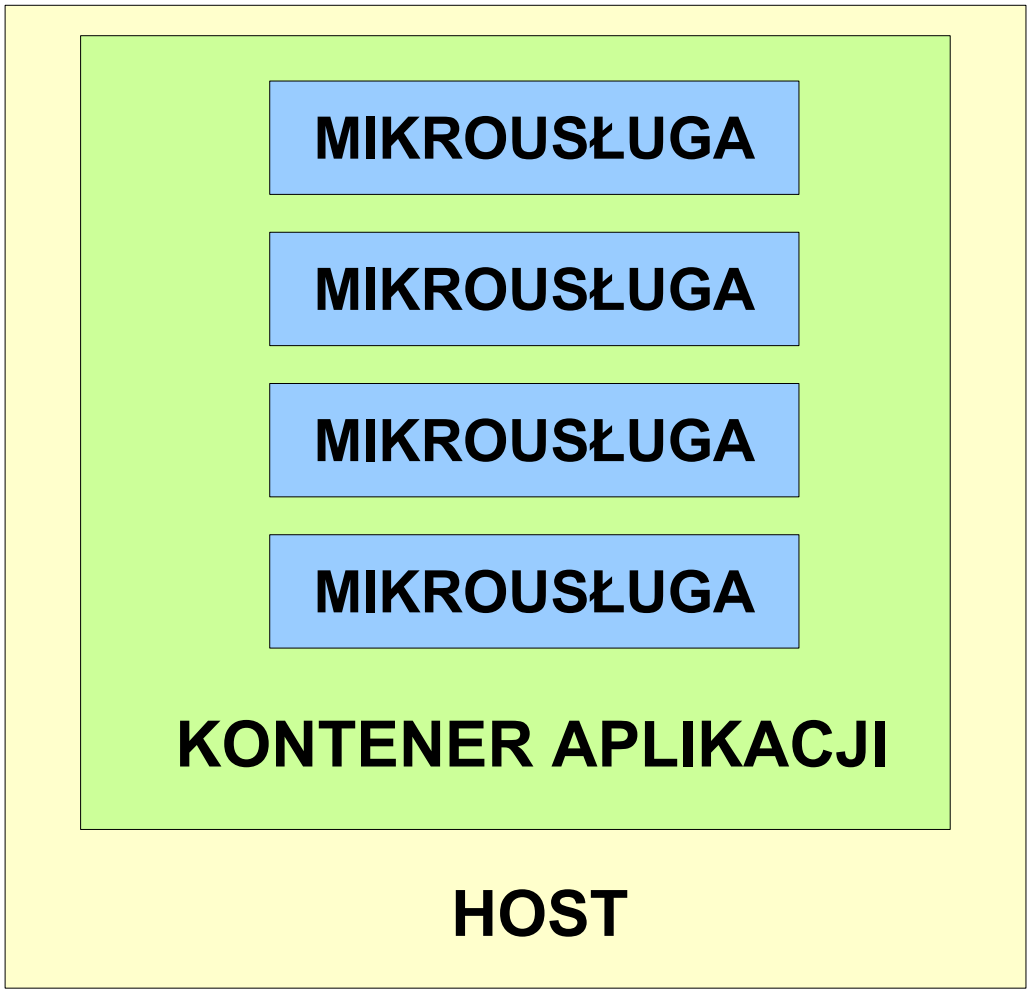


Uruchamianie mikrousług

- Wiele mikrousług na jednym hoście
  - ZALETY:
    - prosta konfiguracja
    - małe koszty
  - WADY:
    - utrudnione monitorowanie
    - uruchomienie usługi wykorzystującej znacząco zasoby wpływa na działanie pozostałych usług
    - instalacja danej usługi może mieć wpływ na działanie pozostałych usług

Uruchamianie mikrousług

- Wiele mikrousług w kontenerze aplikacji na jednym hoście

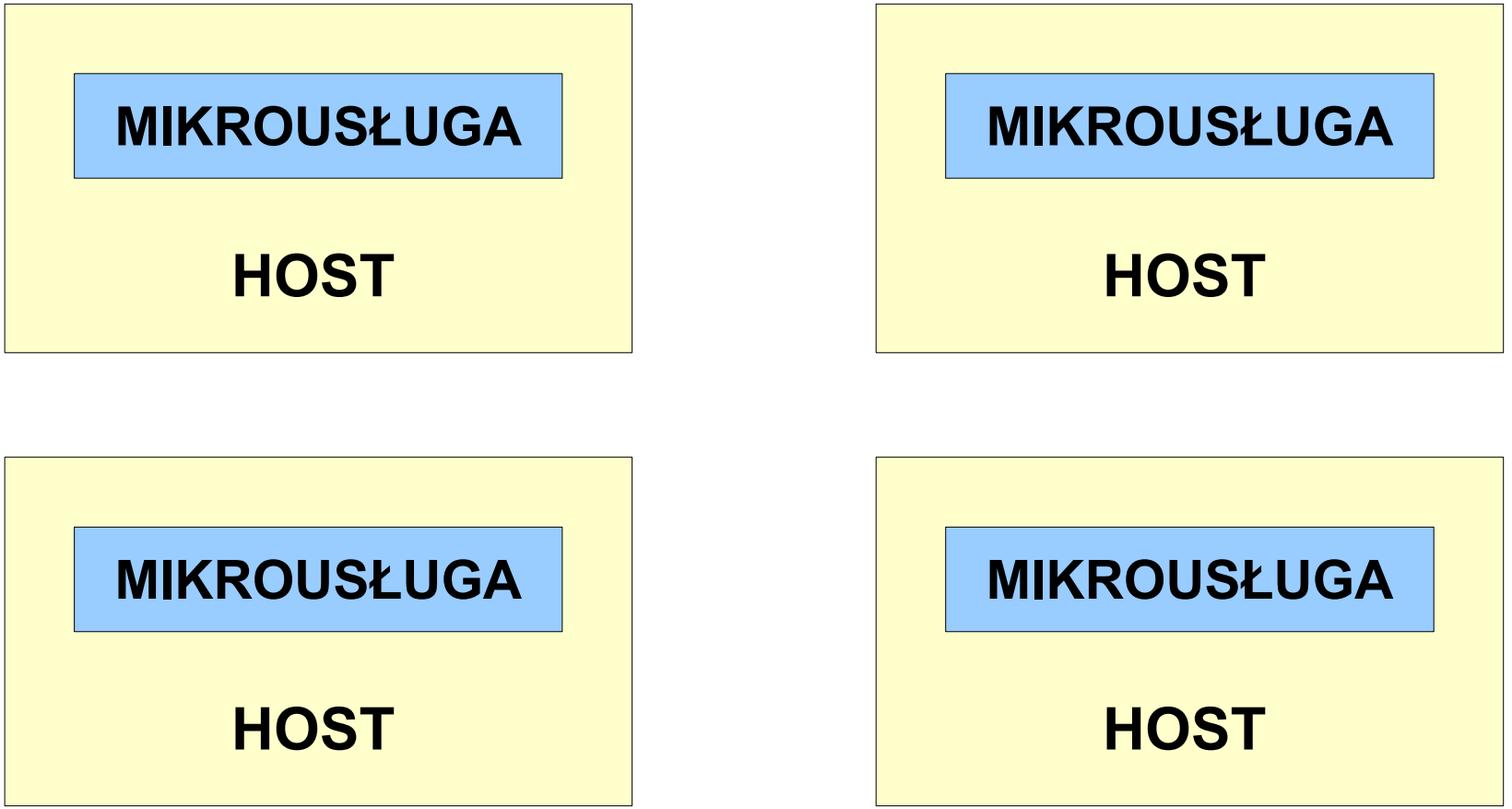


Uruchamianie mikrousług

- Wiele mikrousług w kontenerze aplikacji na jednym hoście
  - ZALETY:
    - zmniejszenie narzutów związanych z działaniem środowisk wykonawczych dla języków (np.. JVM, CLR)
    - większe możliwości wspólnego zarządzania
  - WADY:
    - ograniczony wybór technologii
    - konieczność dostosowania się do wymogów technologii

Uruchamianie mikrousług

- Jedna mikrousługa na jednym hoście



Uruchamianie mikrousług

- Jedna mikrousługa na jednym hoście
  - ZALETY:
    - proste zarządzanie mikrousługami i usuwanie awarii
    - łatwiejsze skalowanie danej usługi, niezależne od innych usług
    - mniejsza złożoność
  - WADY:
    - konieczność zarządzania wieloma hostami
    - większe koszty

Komunikacja pomiędzy mikrousługami

- Mikrousługi mogą komunikować się w różny sposób, np.:
  - protokół HTTP
  - podejście REST
  - protokół AMQP (Advanced Message Queuing Protocol)

Komunikacja pomiędzy mikrousługami

- **Komunikacja synchroniczna**
  - obiekt wywołujący jest zablokowany do czasu otrzymania wyniku
- **Komunikacja asynchroniczna**
  - obiekt wywołujący nie czeka aż operacja zostanie ukończona

Komunikacja pomiędzy mikrousługami

- **Komunikacja „żądanie-odpowiedź”**
  - klient inicjuje żądanie i czeka na odpowiedź
- **Komunikacja sterowana zdarzeniami**
  - klient nie inicjuje żądań a tylko informuje o określonym zdarzeniu (inne strony wiedzą co z tym faktem zrobić)
  - klient generujący zdarzenie nie ma wiedzy, które strony zareagują na to zdarzenie

Sposoby pobierania danych z wielu mikrousług

- Bramka API
  - tworzona jest dodatkowa mikrousługa (bramka API) agregująca dane z kilku mikrousług,
  - aplikacje klienckie nie muszą samodzielnie pobierać danych z kilku mikrousług,
  - bramka API pozwala zmniejszyć liczbę powiązań z aplikacją główną.

Sposoby pobierania danych z wielu mikrousług

- Widoki zmaterializowane i tabele tylko do odczytu
  - tabele tylko do odczytu agregują dane z kilku mikrousług oraz posiadają strukturę opdowiadającą potrzebom aplikacji klienckich,
  - widoki zmaterializowane realizują operację łączenia zbiorów danych z różnych baz danych (zarządzanych przez różne mikrousługi).

Sposoby pobierania danych z wielu mikrousług

- Zimne dane w centralnych bazach danych
  - wykorzystywane są do tworzenia raportów nie wymagających danych pozyskiwanych w czasie rzeczywistym,
  - dane z mikrousług eksportowane są do większej bazy danych i wykorzystywane wyłącznie do raportowania.

Kontenery

- W praktyce, każda mikrousługa jest pakowana wraz z zależnościami do kontenera.
- Skalowanie mikrousługi w kontenerze jest inteligentnie zarządzane przez orkiestrator.

Kontenery

- Przykładowe orkiestratory:
  - Docker Swarm
  - Google Kubernetes
  - Azure Service Fabric
  - Mesosphere DC/OS