Mikroserwisy

- Czym są mikroserwisy?
- Porównanie: monolit vs mikroserwisy
- Praktyczne rozważania

Mikroserwisy

- **Czym są mikroserwisy?**
- Porównanie: monolit vs mikroserwisy
- Praktyczne rozważania

Spring Cloud

Czym są mikroserwisy?

- modny temat od kilku lat ;)
- możemy je opisać jako:
  - **styl** architektoniczny
  - **alternatywa** dla monolitycznych rozwiązań
  - **dekompozycję systemu**
    - na zestaw małych serwisów, funkcjonujących w niezależnych procesach i komunikujących się ze sobą

Spring Cloud

Definicje ekspertów

- Fine-grained SOA
  - **Adrian Cockroft (Netflix)**
- Developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API
  - **Martin Fowler**

Spring Cloud

MIkroserwisy - praktyczna definicja

- **Zestaw małych serwisów** składający się na całość
  - zamiast pojedyńczej, monolitycznej aplikacji
- …każdy z nich **w osobnym procesie**
  - pojedyńczy plik wykonywalny
- …**komunikujących się** ze sobą
  - HTTP/REST, messaging etc.
- …**osobno pisane** i utrzymywane
- …**enkapsulują biznesowe funkcjonalności**
  - zamiast konstruktów języka (klasy, pakiety)
- …**niezależnie wymieniane** / upgrade (może być trudno)

Spring Cloud

Czym nie są mikroserwisy?

- **To nie** to samo co **SOA**
  - SOA skupia się na integracji różnych aplikacji
  - mikroserwisy dekomponują pojedyńczą aplikację
  - w obu przypadkach serwisy są głównym komponentem, ale inna jest ich charakterystyka
    - SOA: business, enterprise, application, infrastructure
    - mikro: functional, infrastructure
- idealnym rozwiązaniem dla każdego systemu
  - każda architektura ma minusy i związane z nią ryzyka

Spring Cloud

Mikroserwisy

- Czym są mikroserwisy?
- Porównanie: **monolit** vs mikroserwisy
- Praktyczne rozważania

Aplikacja - monolit vs mikroserwisy

- Wyobraźmy sobie aplikację zakupową
- Funkcjonalności:
  - wyszukiwanie produktu
  - zarządzanie artykułami
  - koszyk zakupów
  - etc.
- klient przekierowywany jest do danej funkcjonalności

Spring Cloud

Aplikcja monolityczna - wyzwania

- utrudniona ewolucja, trudności z dodaniem
  - nowego typu klienta
  - nowego serwisu
  - nowego sposobu zapisu danych

Spring Cloud

Aplikcja monolityczna - wyzwania

- Zarządzanie cyklem życia produkcji:
  - pojedynczy codebase
  - jeden deployment dla całej aplikacji
- Problemy z jedną funkcjonalnością, wykryte np. na UAT’ach, wstrzymują pozostałe!

Spring Cloud

Aplikcja monolityczna - charakterystyka

- pojedyńczy, wykonywalny plik
  - **łatwa do ogólnego zrozumienia**, trudno się w nią wgryźć
- modułowość aplikacja oparta o jezyk, w którym jest napisana (pakiety, klasy, metody, frameworki etc.

Spring Cloud

Aplikcja monolityczna - zalety

- łatwo **testować**
  - do pewnego rozmiaru
- łatwy **deploy**
- łatwo **zarządzać**
  - do pewnego rozmiaru
- łatwo **zarządzać zmianami**
  - do pewnego momentu
- łatwo **skalować**
  - typowa web’owa aplikacja - load balancer i dodatkowe instancje

Spring Cloud

Aplikcja monolityczna - wady

- **przywiązanie do** framework'u / **języka**
  - trudno eksperymentować z nowymi technologiami
  - trudno dobrać najlepszą technologię dla zadania
- **zrozumienie kodu**
  - 1 dev: trudności ze zrozumieniem codebase dużej aplikacji
  - 1 team: trudność w zarządzaniu dużą aplikacją.
    - Amazon: **2 Pizza rule**
- deployment pojedyńczej aplikcji
  - **brak niezależnego** deployment'u zmian w jednym komponencie
  - **zmiany** są '**zakładnikami**' wykrytych problemów w innych zmianach

Spring Cloud

Mikroserwisy

- Czym są mikroserwisy?
- Porównanie: monolit vs **mikroserwisy**
- Praktyczne rozważania

Spring Cloud

Rozwiązanie mikroserwisowe

- **małe**, **niezależnie** deploy'owane **aplikacje**
- wymusza **przejrzystą architekturę** interfejsów
- zakres **zmian dotyczy jednego** serwisu

Spring Cloud

Rozwiązanie mikroserwisowe

- Niezależnie deploy’owane aplikacje, pozwalają na:
  - **osobne codebase** dla każdej aplikacji
  - możliwość używania **wielu języków** / framework’ów

Spring Cloud

Rozwiązanie mikroserwisowe

- Komunikacja oparta o **lekkie protokoły** (HTTP, TCP etc.):
  - wymusza przejrzystą architekturę interfejsów
  - enkapsuluje biznesowe funkcjonalności
  - **komunikacja poprzez API**,
  - **brak wspólnej bazy** danych

Spring Cloud

Rozwiązanie mikroserwisowe

- pojedyńczy serwis jest łatwy do
  - zrozumienia
  - zmiany
  - testowania
  - deploy’u
  - zarządzania
  - wymiany
- przez mały team lub nawet jednego dev’a / dev’kę

Spring Cloud

Decentralizacja zarządzania

- używanie **najlepiej dobranego** narzędzia dla zadania
- serwisy **rozwijane w** swoim tempie, w **zależności od potrzeb**
- ”**tolerant readers**”
  - *Be conservative in what you do, be liberal in what you accept from others. -- Jon Postel*
- kontrakty typu **consumer-driven**
  - serwisy ewoluują, nadal wypełniając oczekiwania klientów

Spring Cloud

Polyglot Persistence

- różne rozwiązania bazodanowe dla różnych serwisów
- RDBMS nie zawsze jest najlepsze
  - (kontrowersyjne dla niektórych - brak transakcji, modelu wspólnego dla całej aplikacji)

Spring Cloud

Aplikacja mikroserwisowa - zalety

- **niski próg wejścia** dla nowych osób
- **pojedynczy serwis** bardzo **łatwy**
  - testowanie, deploy, wersjonowanie, skalowanie, zarządzanie
- **cykl produkcji niezależny** (teoretycznie) od innych koponentów
- **brak przywiązania do języka** / frameworku
- **dynamiczne skalowanie**
- **łatwa** adaptacja **nowych technologii**

Spring Cloud

Aplikacja mikroserwisowa - wyzwania

- **eventual consistency**
- **Fallacies of Distributed Computing**
- Serwisy mogą nie być dostępne ;)
  - ”**Everything fails all the time**”
    - -- Werner Vogels, CTO Amazon
  - potrzebny ”design for failure”
  - zwiekszona potrzeba monitorowania
- **Refactoring Module Boundaries**

Spring Cloud

Fallacies of Distributed Computing

- można **polegać** na sieci
- **latency** = 0
- **przepustowość** jest nieskończona
- sieć jest **bezpieczna**
- **topologia** sieci nie zmienia się
- jest **jeden administrator**
- **przesył** danych **nic nie kosztuje**
- sieć jest **homogeniczna**

Spring Cloud

Mikroserwisy

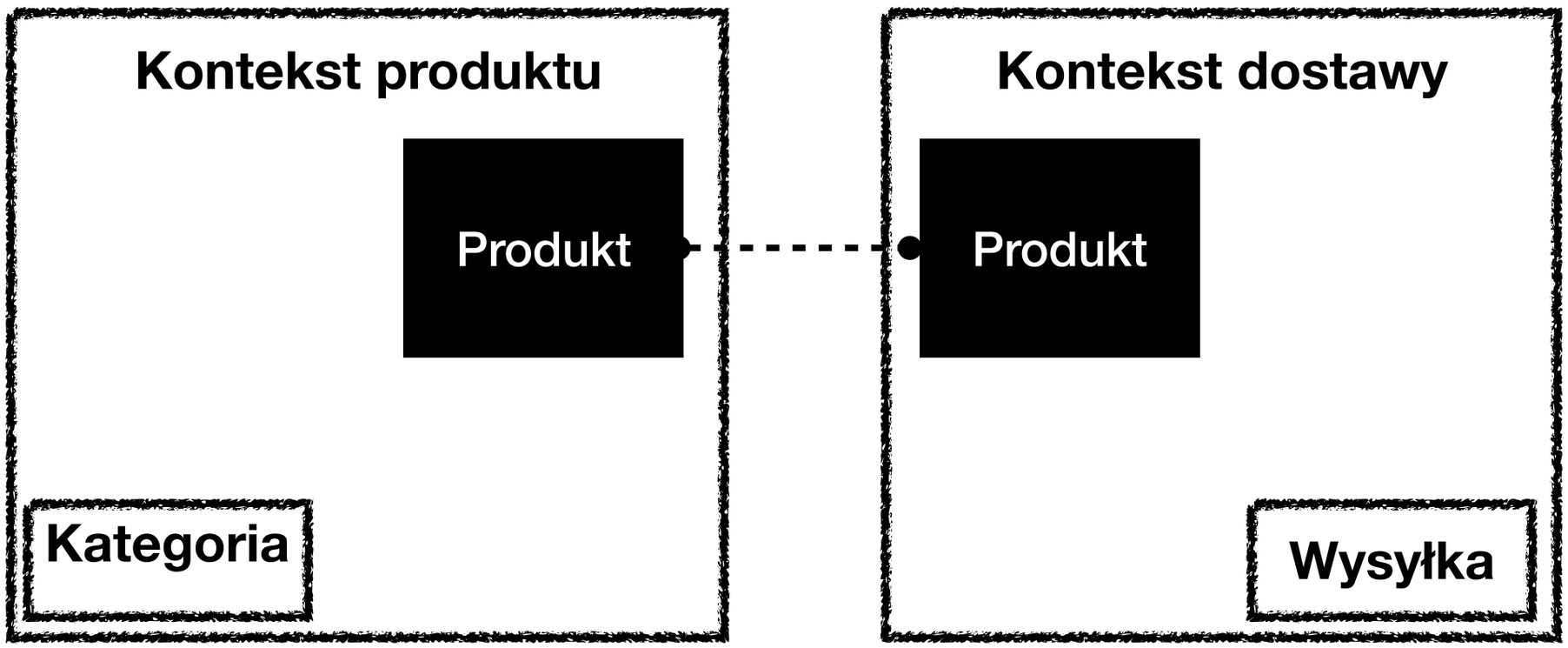
- Czym są mikroserwisy?
- Porównanie: monolit vs mikroserwisy
- **Praktyczne rozważania**

Mikroserwisy - praktyczne rozważania

- **Jak rozbić monolit** na mikroserwisy**?**
- Głównym czynnikiem, **biznesowe funkcjonalności**
  - **noun**-based (katalog, koszyk, klient)
  - **verb**-based (wyszukaj, zamów)
- **Single Responsibility Principle**
- **Bounded Context**

Spring Cloud

Mikroserwisy - praktyczne rozważania

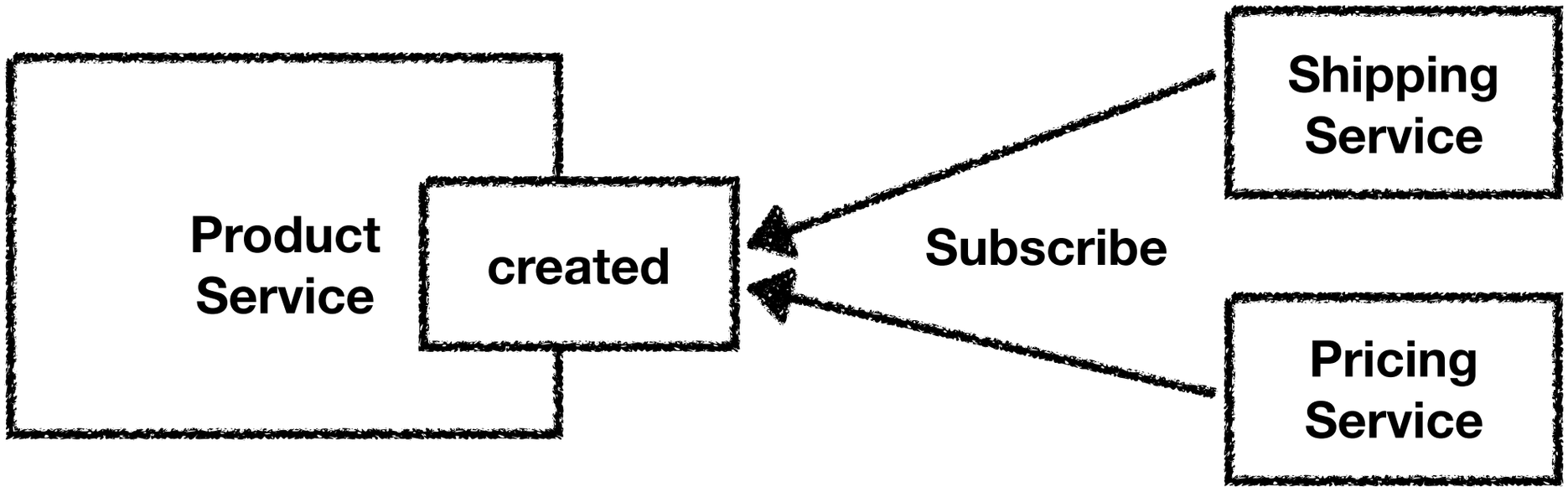


Bounded Context = high cohesion

Spring Cloud

Mikroserwisy - praktyczne rozważania

- **Choreography over orchestration**
  - dodanie produktu do katalogu vs wysyłki: najprawdopodobniej przez inną osobę i w innym czasie
  - dla zachowania spójności serwisy powinny subskrybować zdarzenie dodania / usunięcia produktu.



Spring Cloud

Mikroserwisy - praktyczne rozważania

- **Jak bardzo mikro ?**
  - Rozmiar nie ma znaczenia…
  - mały na tyle, żeby poradził sobie z nim **jeden dev** (ka)
  - **2 Pizza Rule**
  - **dokumentacja do** przeczytania i **zrozumienia**
  - **dziesiątki** smaczków, **nie setki**
  - **przewidywalny** = łatwy do eksperymentowania z nim

Spring Cloud

Mikroserwisy - praktyczne rozważania

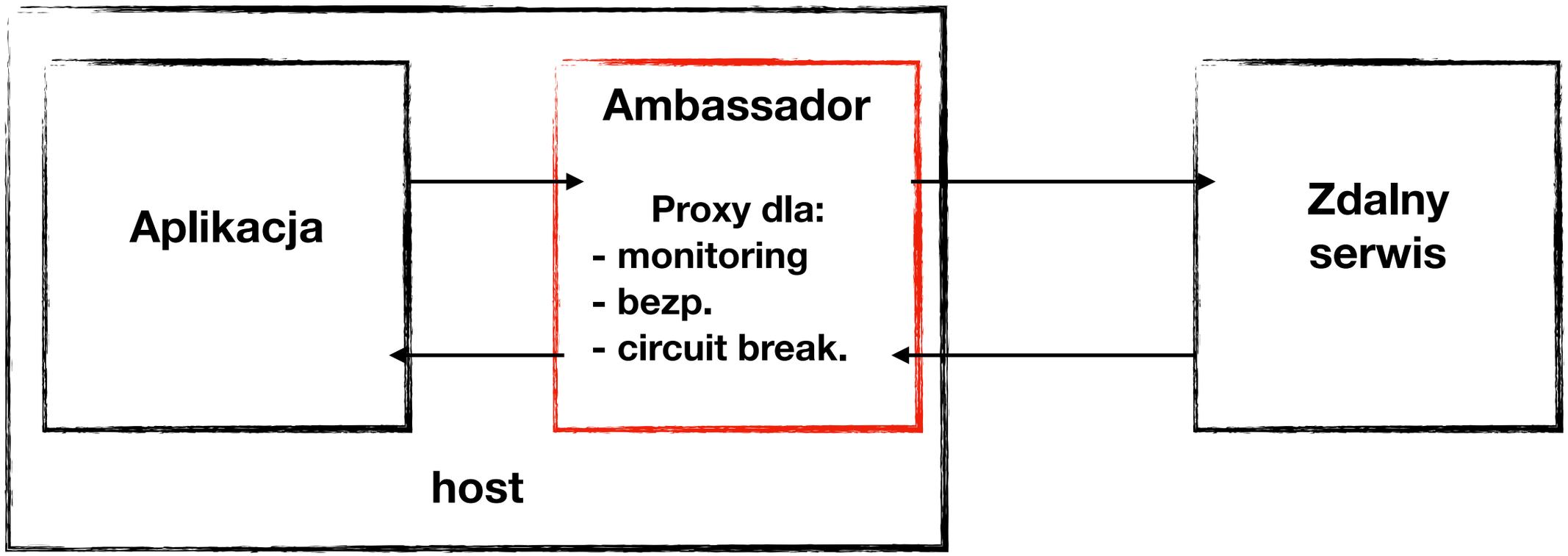
- **Różnice z SOA**
  - SOA
    - integracja systemów
    - ’inteligentna’ technologia integracji, ‘głupie’ serwisy
  - mikro
    - indywidualna aplikacja
    - **’głupia’ technologia integracji**, **‘inteligentne’ serwisy**
      - analogia
      - ps aux | grep something | awk ’{print $2}’

Spring Cloud

Przykładowe wzorce

Ambassador

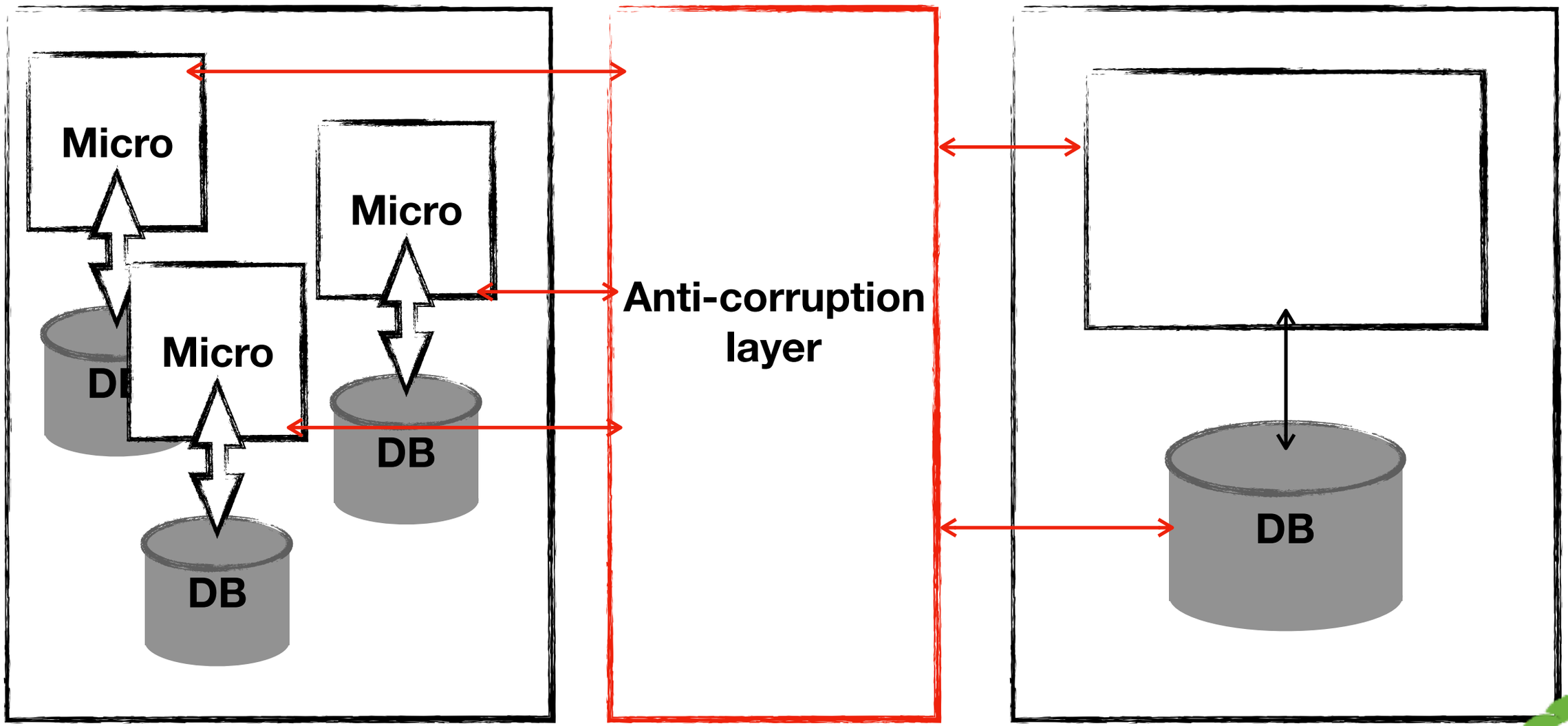
odciążenie klienta



Spring Cloud

Anti-corruption layer

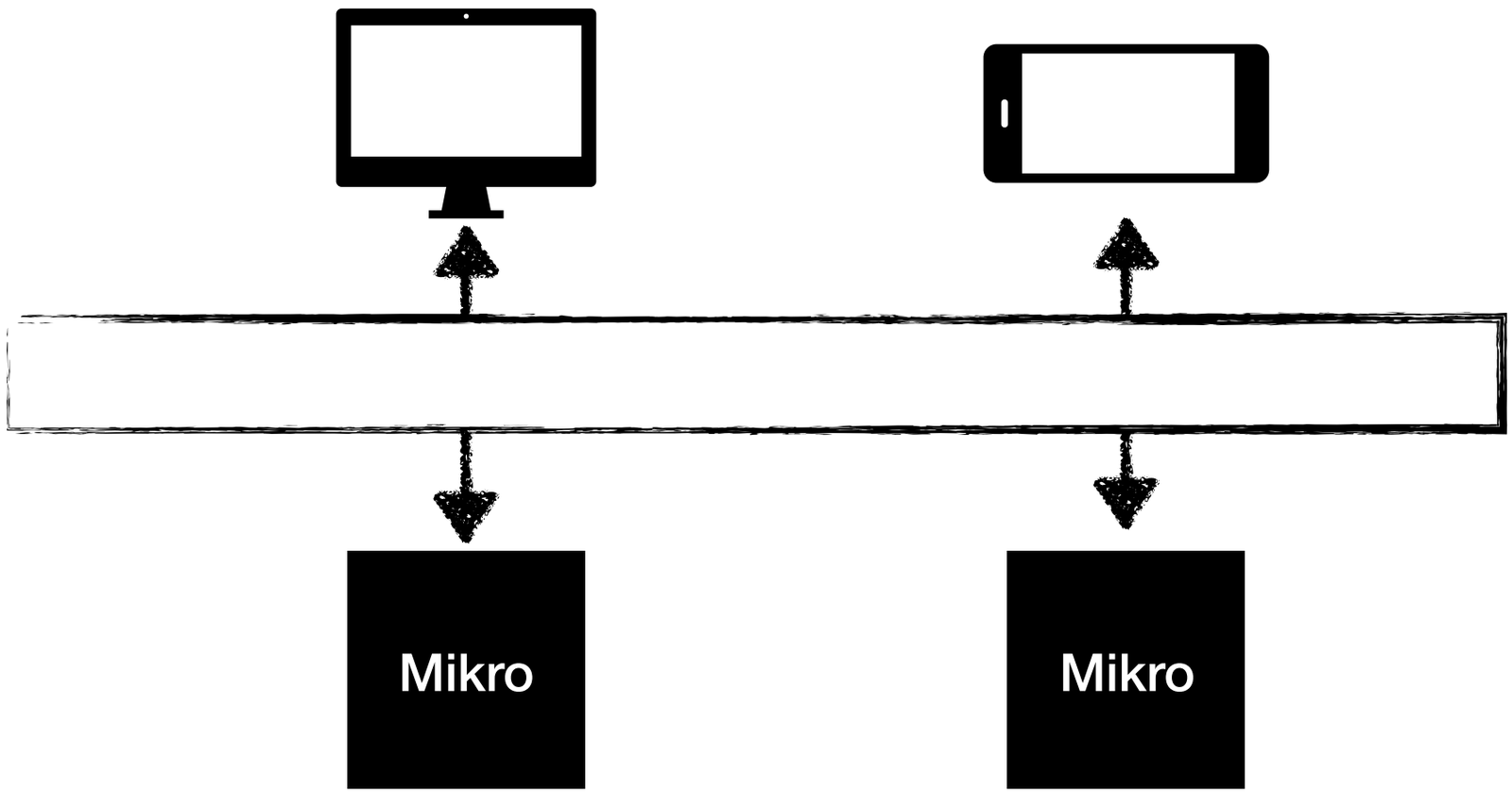
fasada pomiędzy nowym i starym



Spring Cloud

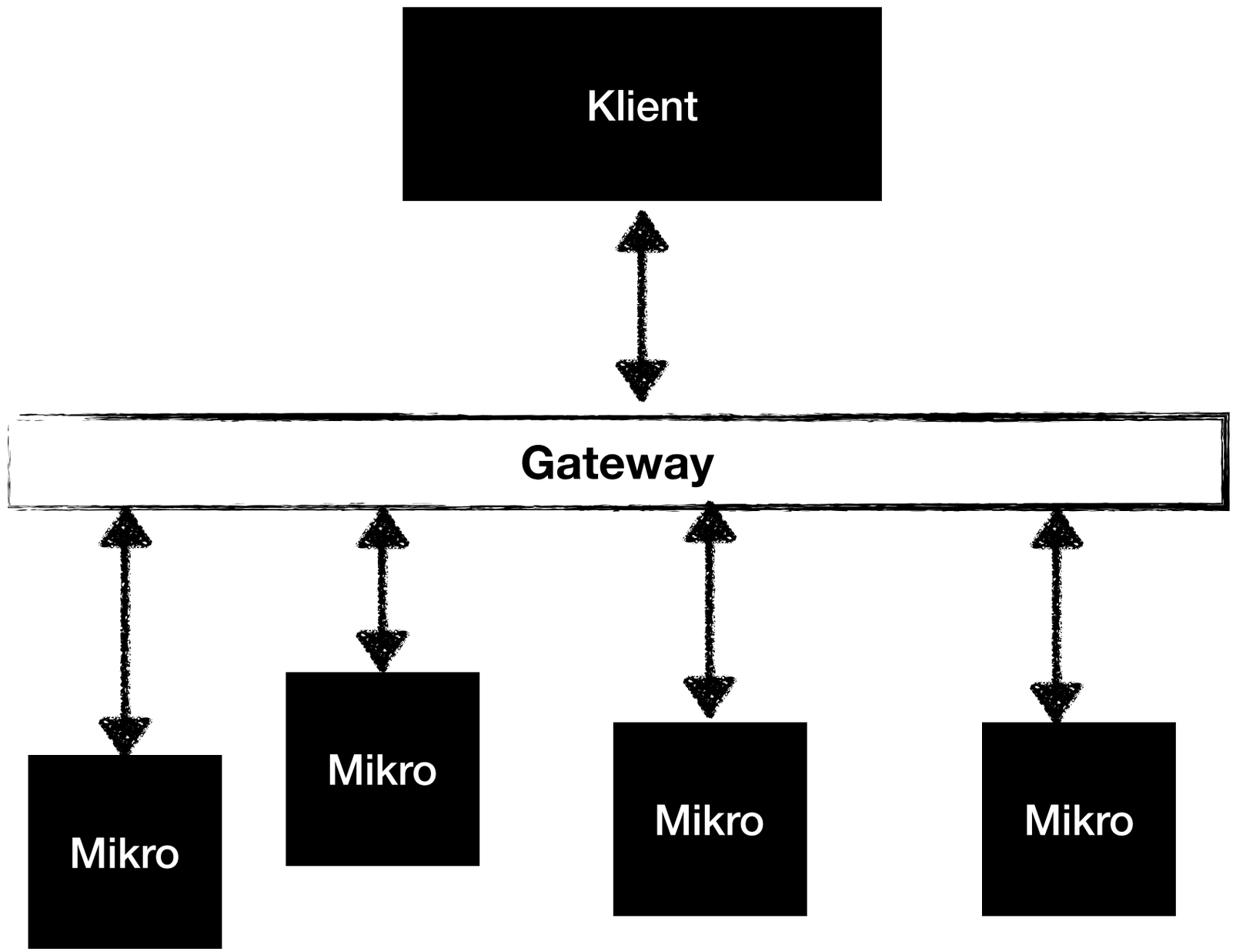
Backends & frontends

osobne serwisy dla różnych klientów



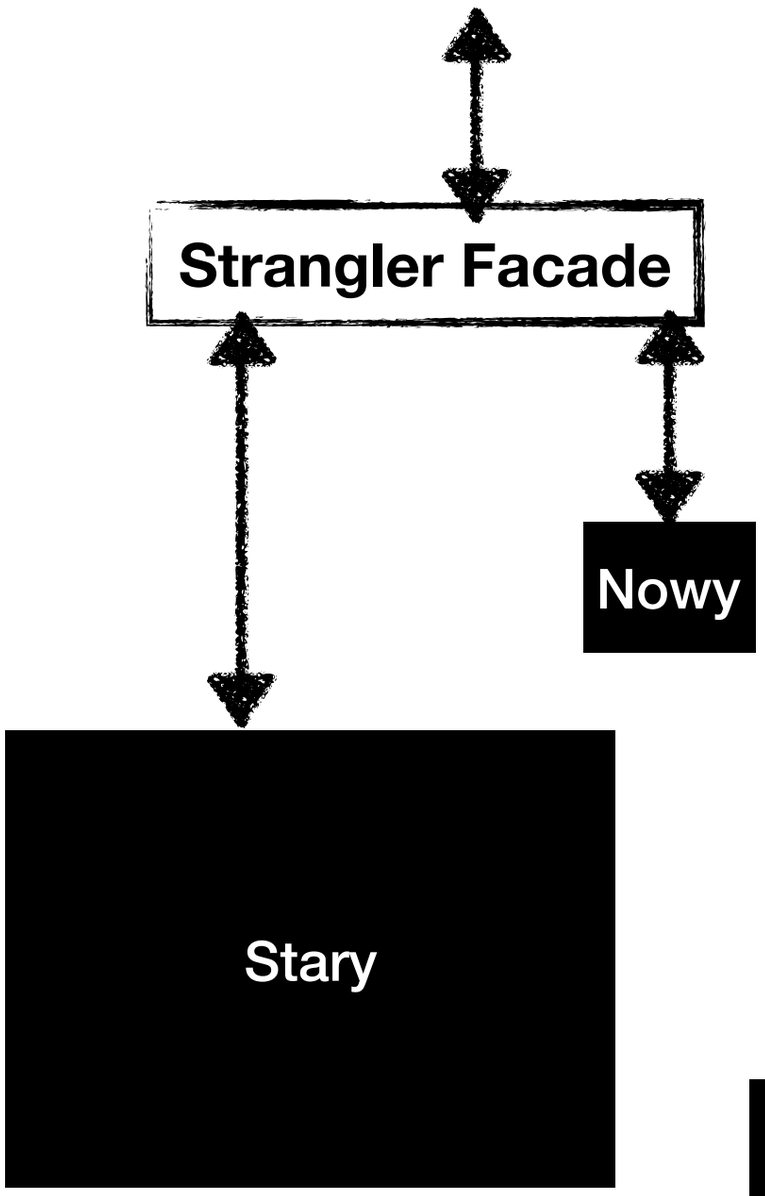
Spring Cloud

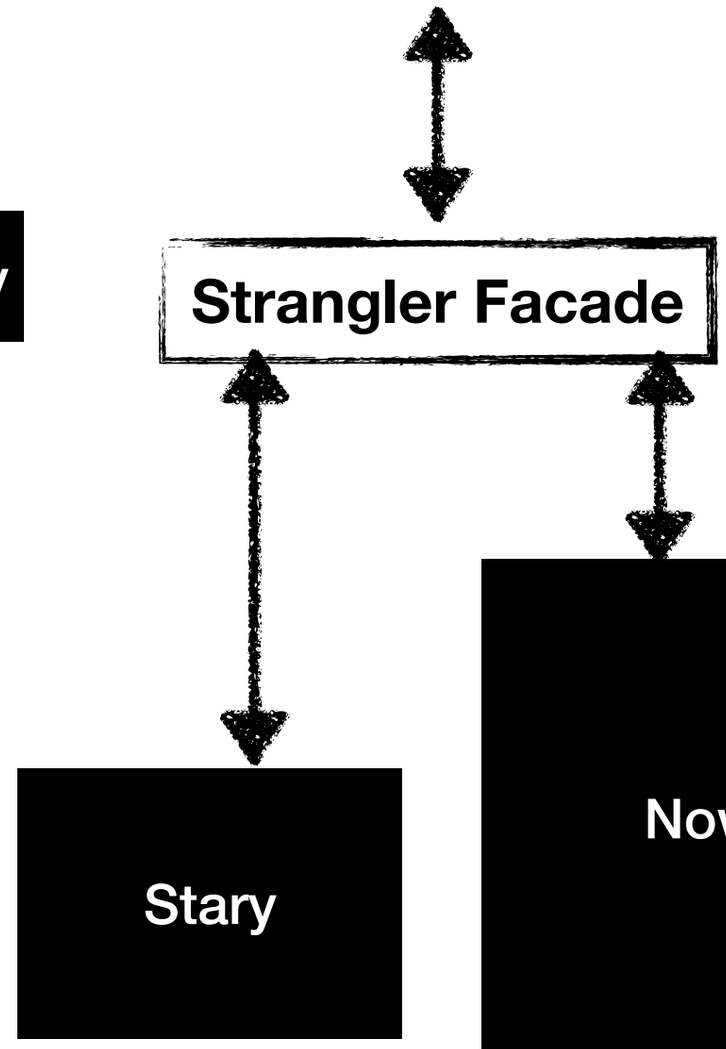
Gateway Routing & Aggregation

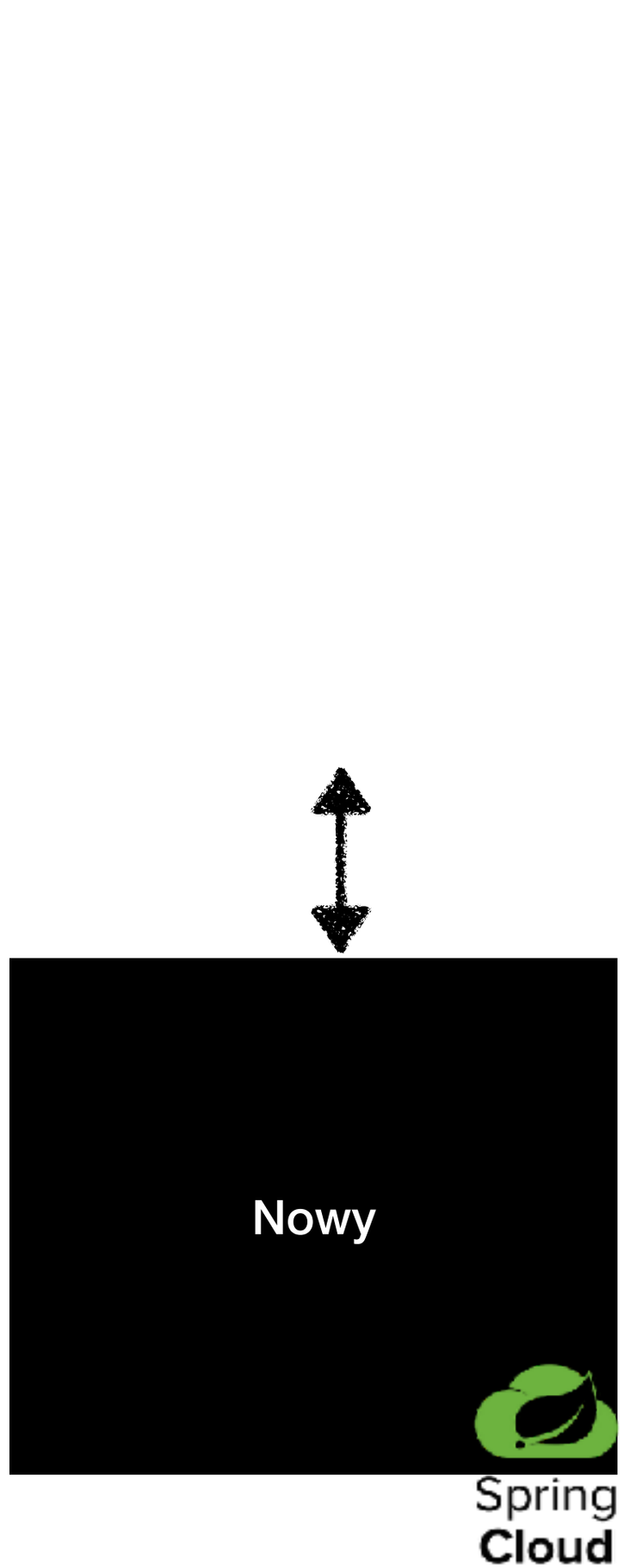


Spring Cloud

Strangler


Strangler Facade
Nowy
Stary


Strangler Facade
Stary
Nowy


Nowy
Spring
Cloud

Podsumowanie modułu

- **Mikroserwisy to** jeszcze jedno **architektoniczne podejście** do tworzenia systemów:
  - **dekompozycja** monolitu **na niezależnie** istniejące, **komunikujące** się ze sobą **procesy**
- Mikroserwisy tak jak i rozwiązania monolityczne, **mają swoje zalety i wady**
- **mem frontend backend :)**

Spring Cloud