Vattenfall IT Services Poland

Domain-Driven Design Pragmatycznie

Michał Michaluk

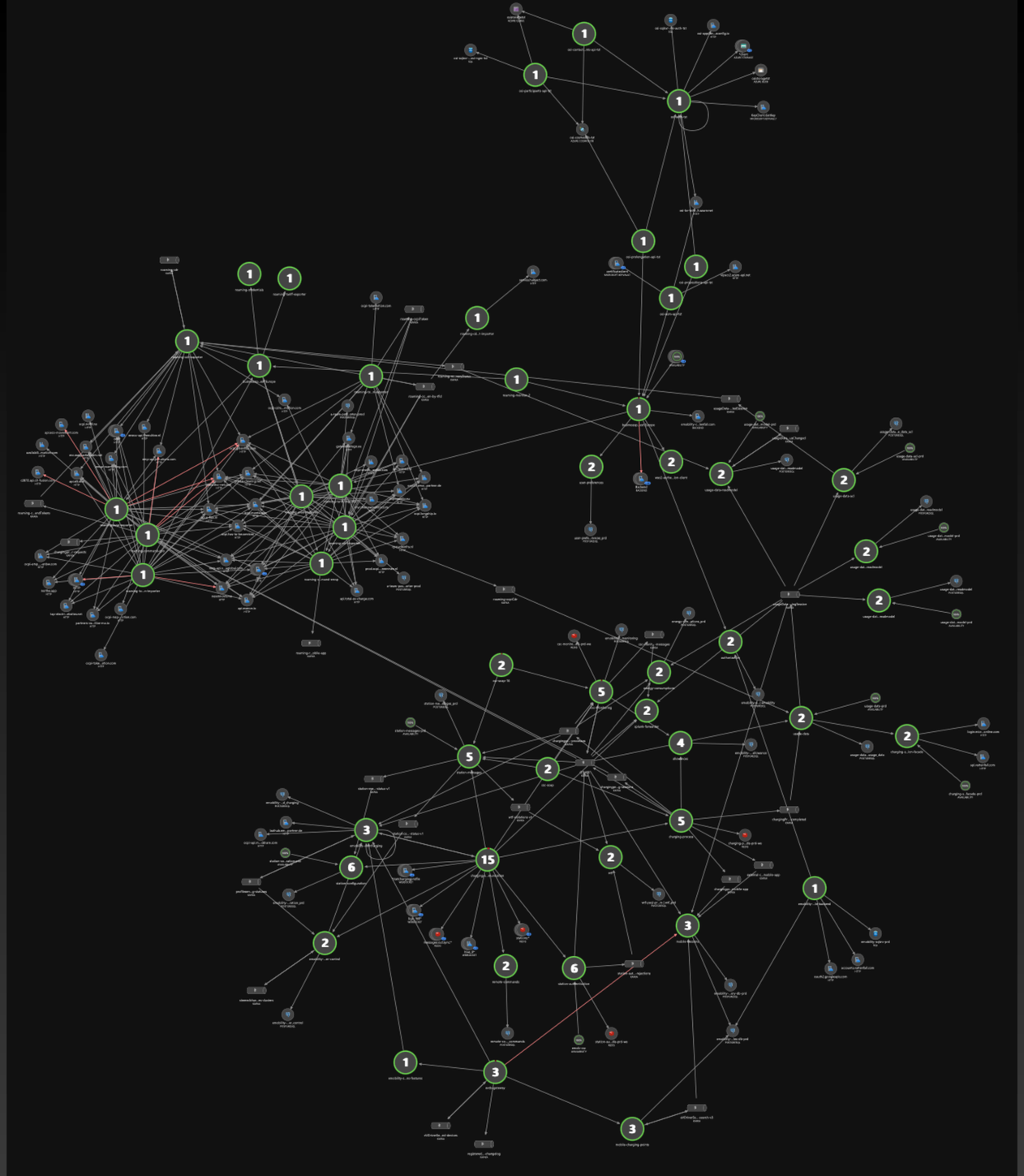
Wyzwania realnych aplikacji biznesowych

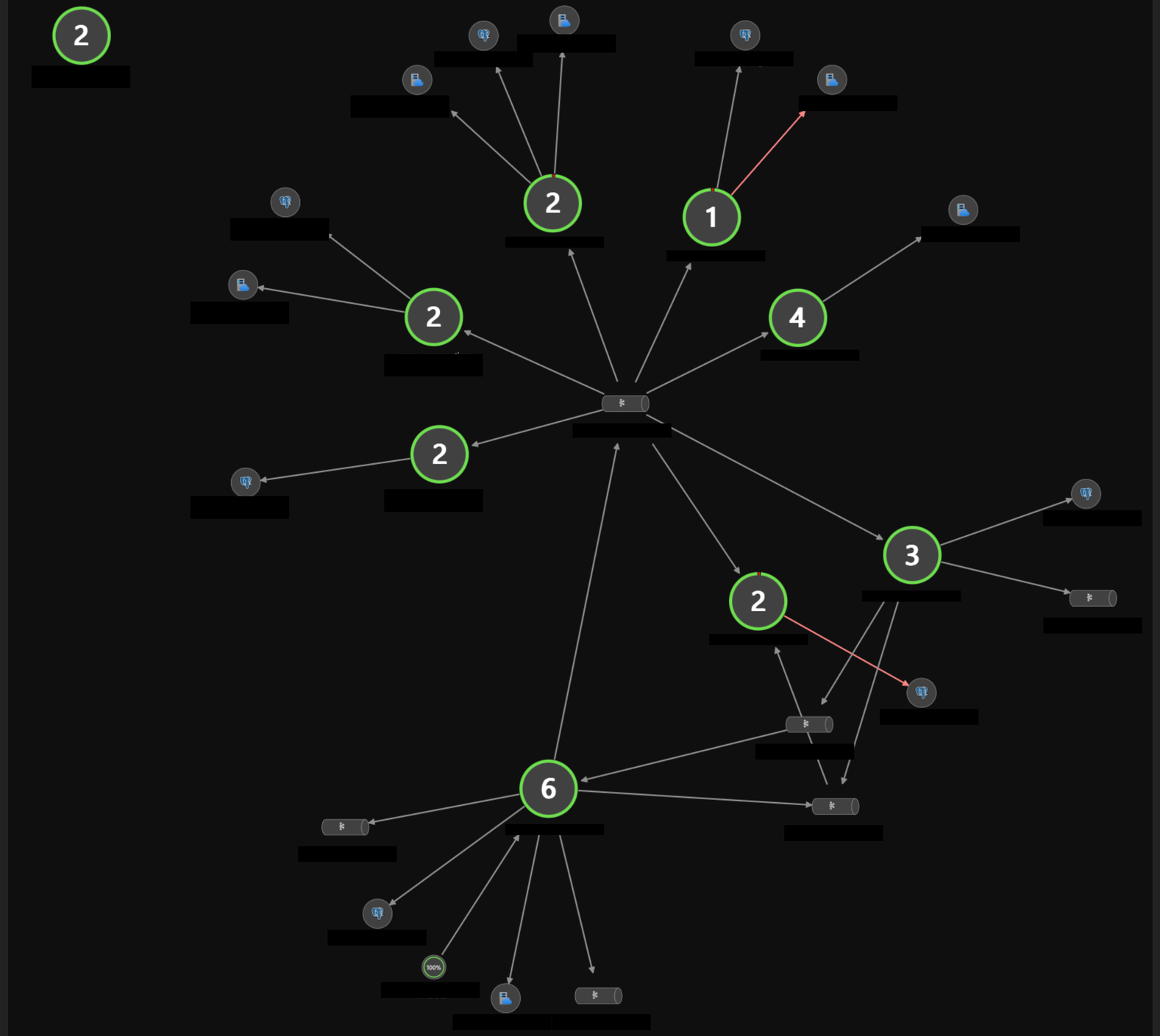
- Zrozumieć potrzeby biznesu / użytkowników
- Zrozumieć złożoność dziedziny
  - kalkulacje
  - automatyczne decyzje
  - przepływ danych - transformacje / łączenie informacji
  - …
- Integracje między systemami
- Zarządzanie całą tą złożonością na przestrzeni lat

Techniki radzenia sobie z tymi wyzwaniami

- Domain-Driven Design
- Mikro serwisy
- Modularyzacja wewnątrz serwisu
- Architektura Portów i Adapterów
- Programowanie Obiektowe
  / Funkcyjne
  / Deklaratywne

Domain-Driven Design




2
2
1
2
4
2
2
3
6
100%

Mikro serwisy

W czym pomaga mikro serwisy?

- Niezależny deployment
  - dostosowany do potrzeb biznesu
  - zespoły pracują niezależnie
- Niezależne skalowanie
  - tam gdzie to potrzebne
  - obciążenia nie propagują się na cały system
- Niezależna technologia
  - tanie ciągłe upgrady technologii
  - zespoły wybierają same technologię pod problem i preferencje

Prezentowany przykład w liczbach

Baza danych (Postgresql):

10 GB dokumentów, 11 314 464 szt
29 GB eventów, 41 360 818 szt

Messaging (Kafka):

397 277 386 wiadomości przetworzonych
84 625 790 wiadomości wysłanych (total)
858 005 wiadomości po akcjach z frontu

Object Oriented Programming

W czym pomaga Object Oriented Programming?

- Czysta implementacja złożoność dziedziny
- Łatwe testowanie
- Zarządzanie złożonością systemu na przestrzeni lat

czas na kodzik

Ports & Adapters

W czym pomaga architektura Ports and Adapters?

- Umożliwia implementacja złożoność dziedziny w oderwaniu od technicznych detali
- Zarządzanie złożonością integracji między:
  - Systemami
  - Naszymi microserwisami
  - Modułami monolitu

Ports & Adapters

Model domeny

- biblioteka obiektów i/lub funkcji,
  które rozwiązują „trudne” zagadnienia biznesowe
  tą biblioteką posłużymy się by zaimplementować kompletną aplikację
- możliwy model obiektowy lub funkcyjny łatwość stosowania immutable
- przetestowana jednostkowo, zanim powstanie infra

Ports & Adapters

Serwis aplikacyjny (port wejściowy)

- interakcje z obiektami domenowymi i portami
- często proceduralny prosty kod
  (bo szczegóły są w domenie lub adapterach)
- transakcje bazodanowe
- security

Interfejsy portów (port wyjściowy)

- wstrzyknięte do Serwisów aplikacyjnych lub modelu
- udostępnia „wyjście na zewnątrz” aplikacji do
  - baz danych
  - serwisów zewnętrznych
  - innych efektów ubocznych: drukarka

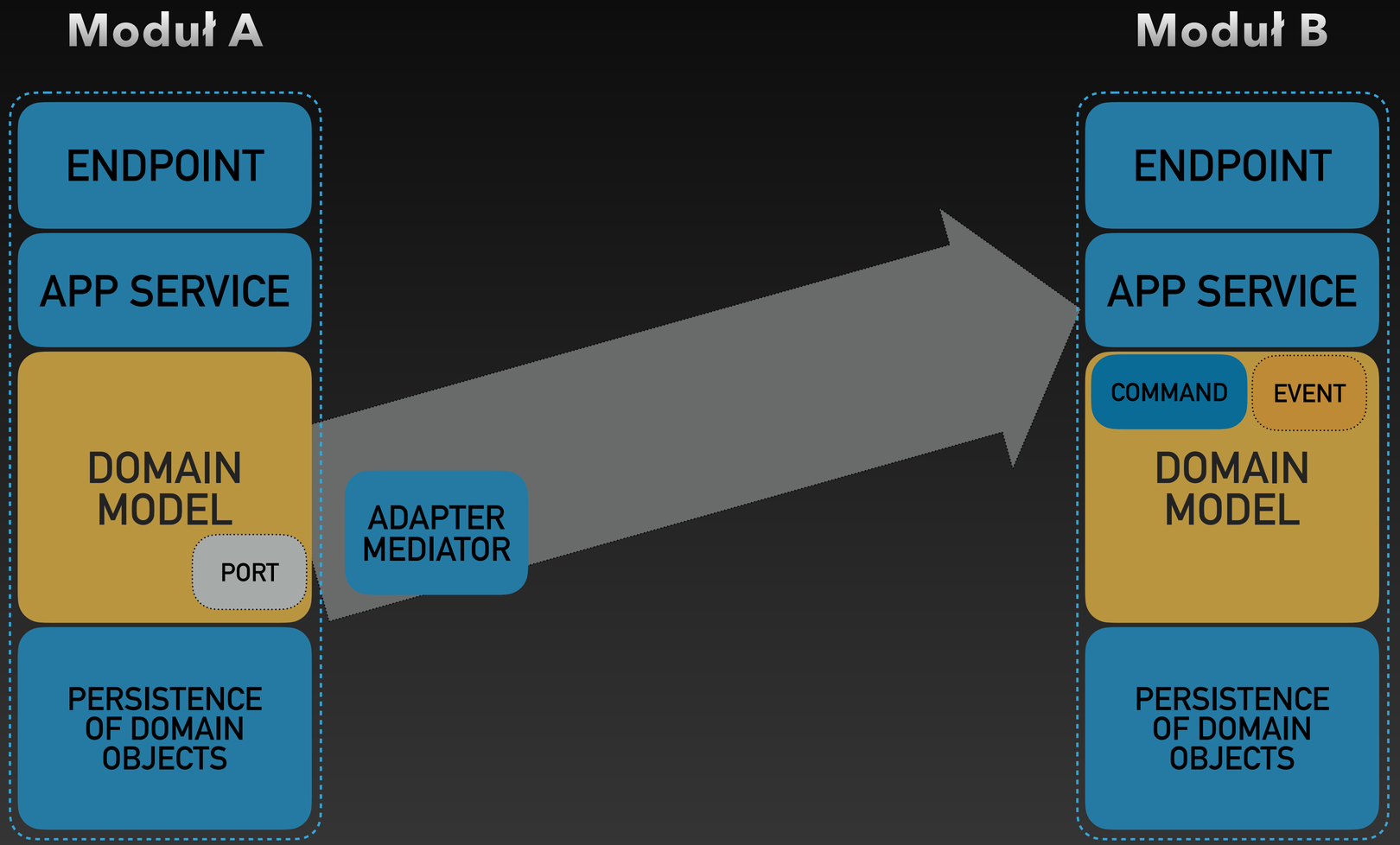
Ports & Adapters

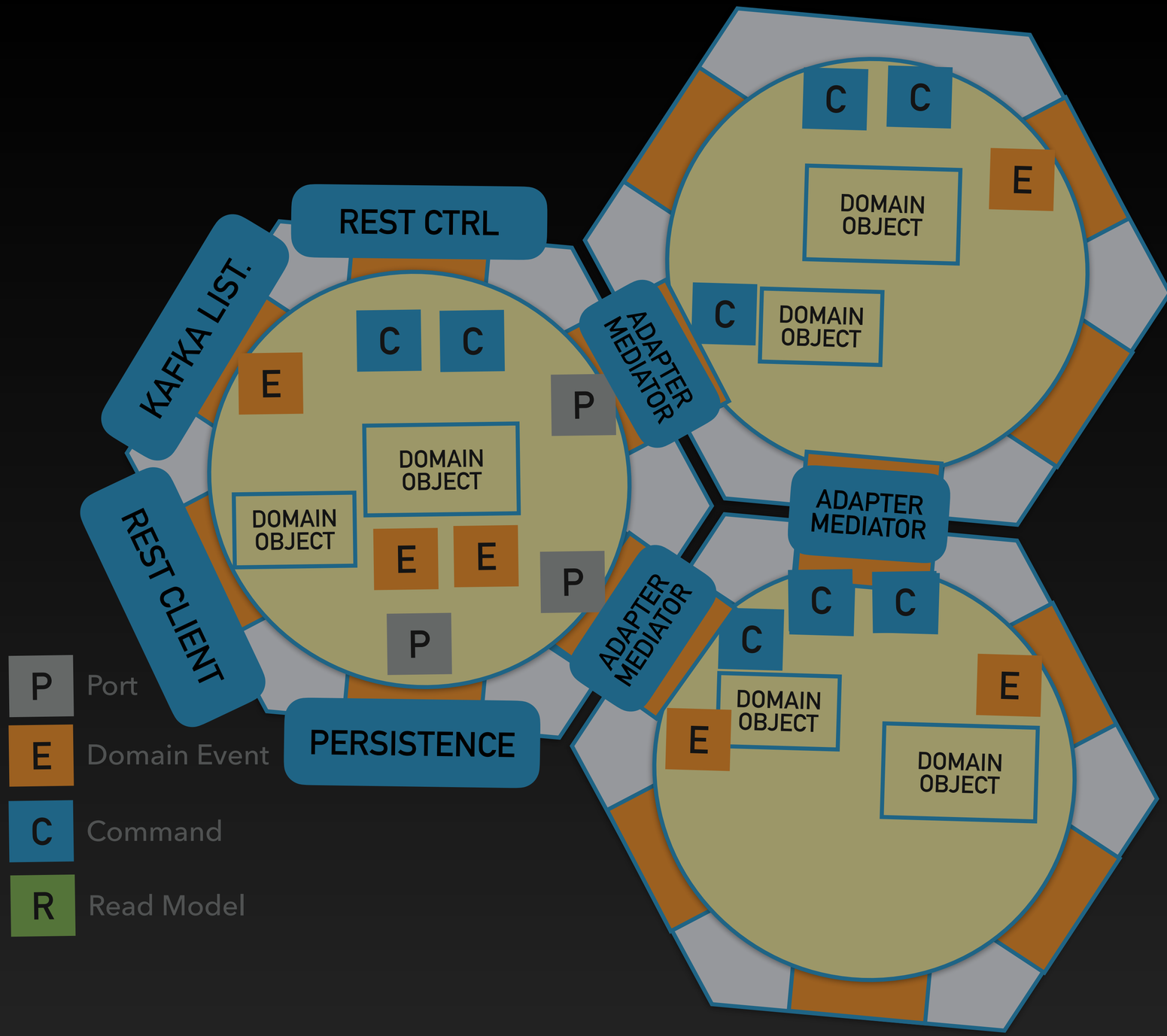
Adaptery

- skupione na jednym technicznym zadaniu
- korzystają z modelu domeny i framework-ów / bibliotek / driverów
- mogą posłużyć do integracji z innymi modułami dziedzinowymi mediujące = wymieniające dane i wywołania między modułami

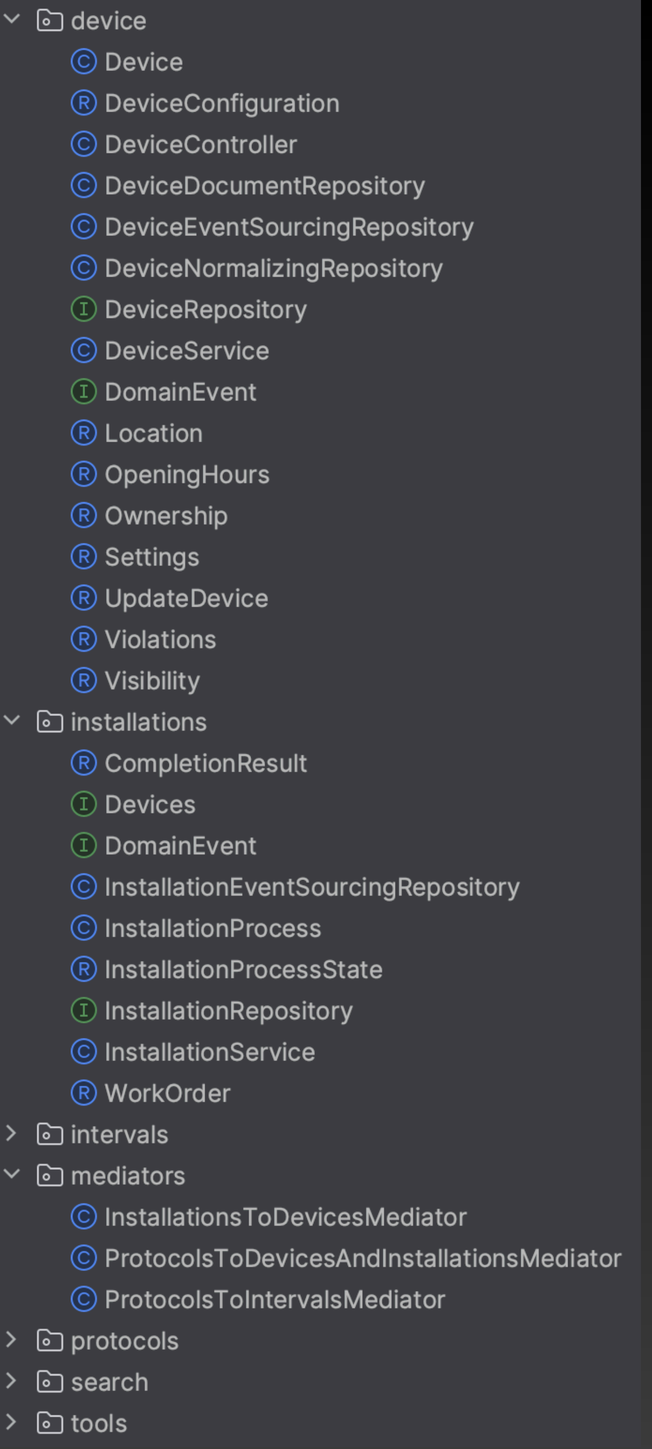
czas na kodzik

Integracja modułów




REST CTRL
KAFKA LIST.
REST CLIENT
PERSISTENCE
ADAPTER MEDIATOR
ADAPTER MEDIATOR
ADAPTER MEDIATOR
C
C
E
P
DOMAIN OBJECT
DOMAIN OBJECT
E
E
P
P
C
C
E
DOMAIN OBJECT
C
DOMAIN OBJECT
C
C
C
E
DOMAIN OBJECT
E
DOMAIN OBJECT
P Port
E Domain Event
C Command
R Read Model

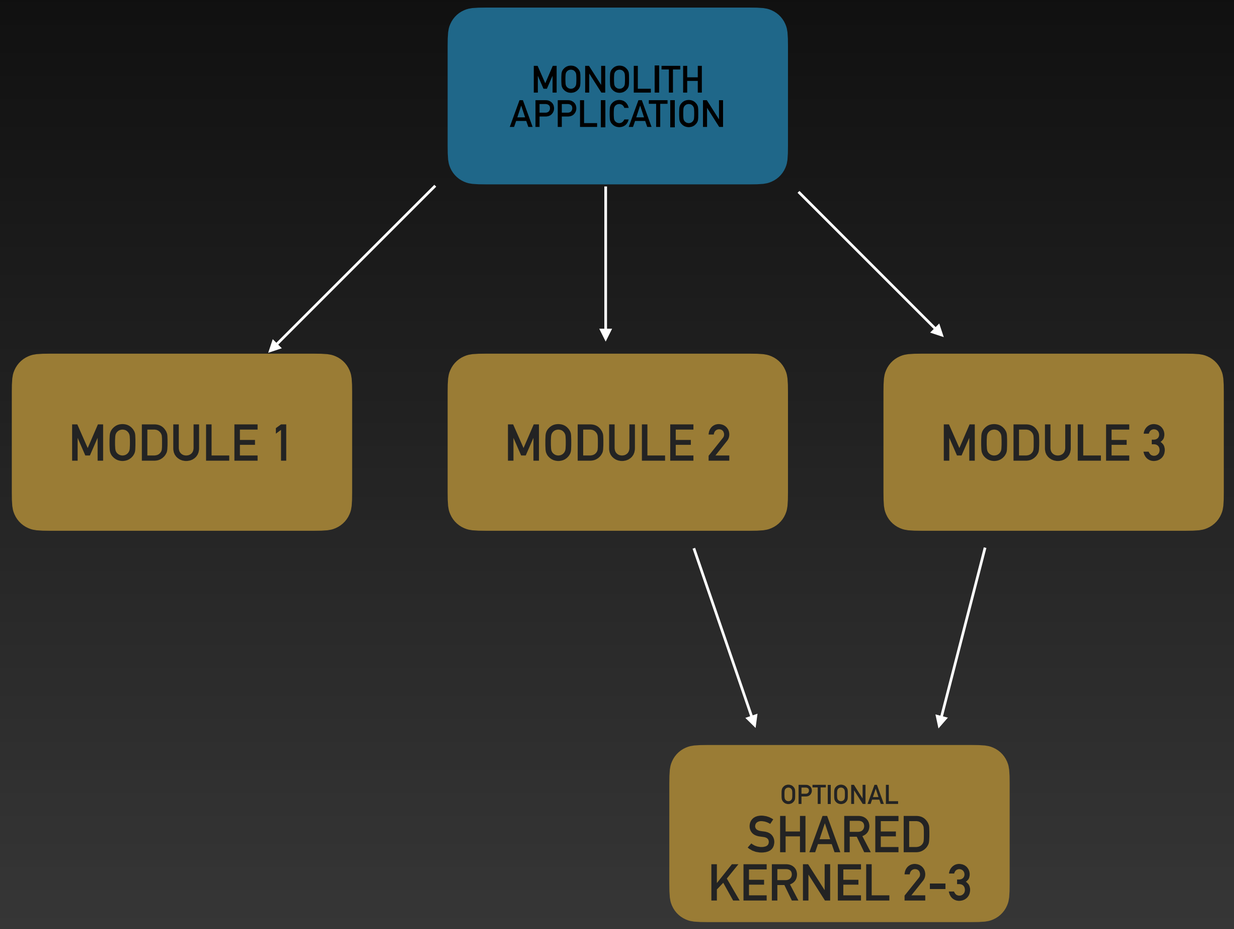
czas na kodzik



Modularyzacja na pakietach

- Vertical slice - rest, business, persystencja w jednym pakiecie
- Łatwo jest użyć widoczności klas package private
- Publiczny serwis i value obiekty z parametrów metod potrzebne dla innych modułów serwis z API
- Publiczne interfejsy portów wyjściowych do innych modułów
- Pakiet z mediatorami - integracjami między modułami
  - implementuje interfejs [portu wyjściowego moduł A]
  - wywołuje serwis [port wejściowy modułu B]
  - mapuje value obiekty wejścia i wyjścia
- Inner klasy by optycznie zmniejszyć objętość pakietu:
  - Entity + JpaRepository wewnątrz implementacji Repository
  - DomainEventy wewnątrz marker interfejsu

Modularyzacja z Maven / Gradle



- Moduły maven gradle per moduł dziedzinowy
- BRAK zależności między tymi modułami
- Przepływa danych między modułami realizujemy jako Porty i Mediatory
- Mediatory, konfiguracja techniczna i „funkcja main” są w Monolith Application
- Shared kernel - biblioteka wspólnych obiektów biznesowych (common)

GitHub

Michał Michaluk

Software developer
Consultant & Trainer

@ Vattenfall IT Services Poland

DDD / Craftsmanship / Architecture
Legacy Code Refactoring

https://github.com/michal-michaluk/

michal.michaluk@bottega.com.pl

Michał Michaluk

Vattenfall IT Services Poland