

Program szkolenia:

Projektowanie microservisów z użyciem DDD

Informacje:

Nazwa:	Projektowanie microservisów z użyciem DDD
Kod:	arch-ms-ddd
Kategoria:	Microservices
Odbiorcy:	developerzy, architekci, analitycy
Czas trwania:	3 dni
Forma:	40% wykłady / 60% warsztaty

Warsztat jest przeznaczony dla uczestników, którzy zastanawiają się jak pogodzić zrozumienie dziedziny problemu z decyzjami technicznymi.

Nieodpowiednie ustalenie granic, odpowiedzialności i hermetyzacji usług prowadzi w krótkim czasie do katastrofy całego systemu. Mikrousługi zamiast pomagać i spełniać składane obietnice powodują dodatkowy narzut złożoności technicznej i chaos organizacyjny. Dzięki zastosowaniu strategicznych wzorców DDD oraz analizy Event Storming zorientowanej na zachowania i reguły jesteśmy w stanie racjonalnie podejść do wymienionych problemów i przygotować się na większość typowych sytuacji, które naruszają architekturę rozwiązania.

Warsztat prezentuje spójną metodykę, która integruje rzetelną analizę, sprawdzone style architektoniczne i najlepsze wzorce implementacyjne.

Zalety szkolenia:

- Dobre określenie granic i odpowiedzialności microservices
- Strategiczne podejście do integracji usług - ich zależności i api
- Wyłanianie serwisów poprzez zachowania i spójność procesu dzięki Event Stormingowi

Szczegółowy program:

1. Wstęp do Event Stormingu

1.1. Problem jaki chcemy rozwiązać

1.2. Stosowalność

1.3. Metodyka

1.3.1. Mechanika

1.3.2. Role i odpowiedzialność

2. Sesja strategiczna - demonstracja z udziałem uczestników

2.1. Łagodne wejście w nomenklaturę poprzez stopniowe wprowadzanie notacji

2.2. Odkrycie procesów biznesowych

2.3. Wstępna destylacja Bounded Context

2.4. Określanie klasy problemu z jakim mamy do czynienia w każdym BC

2.4.1. Szacowanie ryzyk

2.4.2. Drivery architektoniczne

2.5. Odkrywanie ukrytych Bounded Context

2.5.1. Conway's Law vs SOA:Single Source of Truth

2.5.2. Destylacja dziedziny kontekstów tak aby były reużywalne

2.6. Opracowanie strategii integracji BC

2.6.1. PublishedLanguage

2.6.2. OpenHost

2.6.3. SharedKernel

2.6.4. AnticorruptionLayer

2.6.5. Customer-Supplier

2.6.6. Conformist

2.7. Propozycja modułów technicznych na podstawie granic BC (jeden BC to potencjalnie kilka modułów)

2.7.1. Przygotowanie modułów do życia w izolacji jako microservices

2.7.2. Destylacja API

3. Sesja taktyczna - demonstracja z udziałem uczestników lub praca własna

3.1. Kryteria wyboru kontekstów, w których będziemy stosować DDD

3.2. Sesja ES z pogłębionym poszukiwaniem reguł domenowych

3.2.1. Określanie granic agregatów - reguły i heurystyki

3.2.1.1. Typowe problemy

3.2.1.1.1. Źle obrany korzeń

3.2.1.1.2. Zbyt duży agregat - brak kohezji

3.2.1.1.3. Mylenie obiektów biznesowych z widokami (projekcjami)

3.2.2. Najlepsze praktyki

3.3. Przykład implementacji kilku agregatów

3.3.1. Testowanie

3.3.2. Mapowanie relacyjno-obiektowe

3.3.3. Optimistic Locking

4. Architektura systemu - przegląd podejść

4.1. Modularyzacja

4.1.1. Poziomy separacji

4.1.2. Techniki integracji modułów

4.2. Command + Command Handler

4.3. Ports and Adapters

4.4. Restful

4.4.1. Poziomy dojrzałości

4.4.2. Rest jako protokół aplikacyjny a nie transportowy

4.5. Event Driven Architecture

4.5.1. Architektury oparte o zdarzenia

4.5.1.1. Events broker

4.5.1.2. Events Bus

4.5.1.3. Eventual Consistency

4.5.2. Zależności czasowe pomiędzy zdarzeniami

4.5.3. Sagi – orkiestracja zdarzeń

5. Command-query Responsibility Segregation – rozszerzona architektura warstwowa

5.1. Rozwiązanie problemów z niedopasowaniem ORM do przeglądu danych w Gridach

5.2. Zorientowanie na skalowanie i rozszerzalność

5.3. Tworzenie dedykowanych modeli: do odczytu, do operacji biznesowych

5.3.1. Rozwiązanie problemów z wydajnością

5.3.2. Architektura zewnętrznego indeksu z użyciem noSQL