BOTTEGA IT minds

Program szkolenia:

Modularyzacja systemu - analiza granic i projektowanie API

Informacje:

Nazwa:	**Modularyzacja systemu - analiza granic i projektowanie API**
Kod:	**Arch-mod**
Kategoria:	Architektura systemów i aplikacji
Odbiorcy:	architekci, developerzy
Czas trwania:	3 dni
Forma:	40% wykłady / 60% warsztaty

Na tym szkoleniu nie zadowalamy się prostokątami połączonymi kreskami, nad którymi widnieje napis "communication".

Niezależnie od tego czy projektujesz modularny monolit czy rozproszony system (przykładowo microservices) musisz zadbać o odpowiednie określenie granic modułów. Bez tego skończysz a zapachami architektury "feature envy" albo "data envy", co w runtime wygląda jak orgia modułów. Dobrze określone granice zapewnią ci autonomię, czyli możliwość niezależnego eksperymentowania z modelem, z kodem, z pomysłami biznesowymi bez wchodzenia zespołów sobie wzajemnie w drogę.

Program szkolenia przeprowadzi Cię przez kompletny proces:

- Pozyskiwanie wiedzy o procesach biznesowych z wykorzystaniem Event Stormingu na poziomie procesowym
- Analiza i projektowanie granic z wykorzystaniem strategicznych technik Domain-driven design, które zmuszają nas do zastanowienia się nad granicami pod-domen biznesowych i kontekstów modeli
- Projektowanie API modułów i wybór technik integracyjnych z wykorzystaniem notacji C4 dokumentowania architektury

Zalety szkolenia:

- Integracja technik Event Stormingu, strategicznego Domain Driven Design, C4
- Sprawdzone heurystyki określania granic
- Wybrane wzorce projektowe na poziomie API i integracji systemów

BOTTEGA IT minds

Szczegółowy program:

1. Kontekst strategiczny (punkt opcjonalny)

1.1. Pytania

1.1.1. Ryzyka i szanse

1.1.2. Otoczenie systemu

1.1.3. Poszukiwanie Core Domain

1.1.4. Integracja z 3rd party/legacy

1.2. Techniki pozyskiwania informacji

1.2.1. Event Storming - Big Picture

1.2.2. Domain Story telling

1.2.3. User story mapping

1.2.4. Value stream mapping

1.3. Techniki dokumentowania

1.3.1. C4 - context level

2. Projektowanie granic logicznych - komponenty

2.1. Pytania

2.1.1. Spójność danych

2.1.2. Poszukanie właściciela danych i reguł

2.1.3. Klasa złożoności logiki biznesowej i technicznej

2.1.4. Autonomia pracy zespołu

2.1.5. Samowystarczalność w razie awarii innych modułów

2.2. Techniki zbierania informacji

2.2.1. Event Storming - process level

2.2.2. Analiza zmienności pod kątem głównych pytań biznesowych

BOTTEGA IT minds

2.2.3. Analiza alternatywnych wejść i wyjść z procesu

2.2.4. Analiza granic słownictwa

2.2.5. Analiza zmiany istoty obiektów biznesowych

2.3. Techniki analityczne

2.3.1. Odkrywanie splątanych i ukrytych pod-domen biznesowych

2.3.2. Destylacja Bounded Context

2.3.3. Mapowanie kontekstów w celu określenia granic modeli

2.3.3.1. Mapa

2.3.3.2. Strategie mapowania

2.3.3.2.1. Open host

2.3.3.2.2. Shared kernel

2.3.3.2.3. Customer/supplier

2.3.3.2.4. Published language

2.3.3.2.5. Anti-corruption layer

2.4. Techniki dokumentowania

2.4.1. C4 - components level

3. Projektowanie API komponentów

3.1. Zapobieganie przeciekaniu kontekstów

3.1.1. Dobór architektury na podstawie mapy kontekstów

3.2. Wybór stylu API (niekoniecznie wzorca implementacji)

3.2.1. Command

3.2.1.1. Command jako dokument Restful

3.2.1.2. Poprawne mapowanie zasobów i endpointów

3.2.2. Event

3.2.2.1. Najlepsze praktyki

BOTTEGA IT minds

3.2.2.1.1. Język publiczny

3.2.2.1.2. Wewnętrzna saga/process manager emitujący zdarzenia publiczne

3.2.2.2. Antywzorce

3.2.2.2.1. Opresyjne zdarzenia

3.2.2.2.2. Przeciekanie domen

3.2.3. Query

3.3. Podejścia techniczne

3.3.1. Restful

3.3.1.1. Dokument jako command a nie encja z bazy

3.3.2. Command-query Responsibility segregation

3.3.2.1. Dobór read modelu do klasy problemu

3.4. Wzorce

3.4.1. Service

3.4.2. Command + handler

3.4.3. Event bus

3.4.4. Event broker

3.5. Adresowanie dodatkowych wymagań

3.5.1. Wersjonowanie API

3.5.2. Multi-tenancy

3.6. Dobór strategii testowania

3.6.1. Opracowanie macierzy: zakres testu VS cel testy

3.6.1.1. Cele

3.6.1.1.1. Sprawdzenie perfekcji działania logiki

3.6.1.1.2. Akceptacja ogólnego postępu prac

3.6.1.1.3. Regresja

BOTTEGA IT minds

3.6.1.1.4. Środowisko integracyjne

3.6.1.2. Zakres

3.6.1.2.1. Unit

3.6.1.2.2. End2end komponentowy

3.6.1.2.3. End2end systemowy

3.6.2. Narzędzia

4. Integracja komponentów

4.1. Orkiestracja

4.2. Choreografia

4.3. Wzorce

4.3.1. Saga/Process manager

4.3.2. Generyczne API

4.3.3. Inbox pattern

4.3.4. Outbox pattern

4.3.4.1. Zapewnienie at least once delivery

5. Przygotowanie kodu modularnego monolitu do transformacji w microservices

5.1. Pułapki

5.1.1. Granice spójność

5.1.2. Lokalny event broker musi mieć kontrakt taki sam jak zdalny