BOTTEGA IT minds

Program szkolenia:

Architektura aplikacji i systemów - Wzorce architektoniczne dla projektantów

Informacje:

Nazwa:	**Architektura aplikacji i systemów - Wzorce architektoniczne dla projektantów**
Kod:	**Arch-patterns**
Kategoria:	Wzorce architektoniczne
Odbiorcy:	architekci
Czas trwania:	3 dni
Forma:	50% wykłady / 50% warsztaty

Szkolenie prezentuje wybrane Wzorce Projektowe i Architektoncizne w praktycznym i niepodręcznikowym ujęciu osadzonym w kontekście projektowania aplikacji Webowych, platform, systemów i frameworków.
Podczas szkolenia prezentowane są przykłady praktycznego zastosowania zaczerpnięte z rzeczywistych systemów klas: ERP, narzędzia wizualne, systemy rozproszone, serwery.

Podczas szkolenia uczestnicy nabędą zintegrowaną wiedzę na temat zdobyczy nowoczesnej inżynierii oprogramowania pozwalającą im na tworzenie zaawansowanych systemów.
Omawiane zagadnienia leżą u podstaw nowoczesnych frameworków i technologii – co zwiększa poziom ich zrozumienia i pozwala na świadome korzystanie.
Przedstawiamy techniki łączenie wzorców w struktury wyższego rzędu.

Szkolenie przeznaczone dla projektantów i architektów pragnących poszerzyć swe kompetencje w zakresie standardowych styli architektonicznych.

Program jest ogólną ramą merytoryczną. Jego realizacja wykracza poza 3 dni. W trakcie analizy przedszkoleniowej wybieramy wzorce, które będą przydatne dla zespołu i skupiamy się tylko na nich w trakcie szkolenie. W ciągu 3 dni zwykle jesteśmy w stanie zaadresować ok 70% wymienionych wzorców.

Zalety szkolenia:

- Nowoczesne architektury (CqRS - wspierająca DDD)
- Możliwość oparcia przykładów o Kafka
- Dobór klasy rozwiązania do klasy problemu

BOTTEGA IT minds

Szczegółowy program:

1. Dokumentowanie architektury w podejściu 4C - Context, Containers, Components, Classes

- 1.1. Context
  - 1.1.1. Kontekst w jakim będzie działał system
  - 1.1.2. Główni aktorzy
  - 1.1.3. Procesy na poziomie organizacji w które zaangażowany jest system
  - 1.1.4. Rola innych systemów
- 1.2. Containers
  - 1.2.1. Kontenery techniczne w jakich będzie rozmieszczony system
  - 1.2.2. Wzorce integracji
  - 1.2.3. Protokoły komunikacji
- 1.3. Components
  - 1.3.1. Komponenty i Serwisy w ujęciu SOA
  - 1.3.2. Dane kanoniczne wymieniane pomiędzy Serwisami
  - 1.3.3. Projektowanie API
  - 1.3.4. Zasada jednego źródła prawdy
- 1.4. Classes
  - 1.4.1. Wstęp do technik i wzorców projektowania na poziomie klas

2. Architektura systemu - przegląd podejść

- 2.1. Modularyzacja
  - 2.1.1. Poziomy separacji
  - 2.1.2. Techniki integracji modułów
- 2.2. Architektura Micro Kernel
- 2.3. Command + Command Handler

BO·TT·EGA IT minds

BO·TT·EGA IT minds

- 3.1.2.11. Specyfikacje
- 3.1.2.12. Fabryki
- 3.1.2.13. Repozytoria
- 3.1.2.14. Modelowanie niezmienników
- 3.1.2.15. Poziomy modelu: Capacity, Operations, Policy – dostrajanie modelu, Decission Support
- 3.1.3. Strategiczne testowanie warstw – ogólny przegląd podejścia
- 3.1.3.1. Problem eksplozji kombinatorycznej przypadków testowych
- 3.1.3.2. Testowanie górnych warstw w podejściu end2end
- 3.1.3.3. Testowanie dolnych warstw w podejściu jednostkowym

3.2. Praktyczne wykorzystanie technik Inversion of Control (DI, Events, AOP)

- 3.2.1. Dependency Injection – podstawowa technika IoC
- 3.2.1.1. Wykorzystanie zamiast wzorców fabrykujących
- 3.2.1.2. Budowanie konkretnych Strategii, Dekoratorów itd. w zależności od stanu aplikacji (kontekst, konfiguracja)
- 3.2.1.3. Otwartość na rozbudowę dzięki wzorcom Strategii
- 3.2.2. Systemy sterowane zdarzeniami – silniejsza technika IoC
- 3.2.2.1. Użycie do tworzenie rozszerzalnych architektur opartych o pluginy
- 3.2.2.2. Użycie do tworzenia skalowalnych systemów wysokiej wydajności (wykorzystanie kolejek, np. JMS)
- 3.2.2.3. Sagi - Modelowanie złożonych procesów zdarzeniowych
- 3.2.3. Aspect Oriented Programming
- 3.2.3.1. Wstęp do AOP
- 3.2.3.2. Techniki Interceptorów
- 3.2.3.3. Przykłady zastosowania AOP

3.3. Projektowanie systemów otwartych na rozbudowę

- 3.3.1. pluginy

BO·TT·EGA IT minds

BOTTEGA IT minds

4.2.2. Wywołania API

4.2.3. Silniki procesów biznesowych

5. Command-query Responsibility Segregation – rozszerzona architektura warstwowa

5.1. Wsparcie dla Domain Driven Design

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

5.3. Zorientowanie na skalowanie i rozszerzalność

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

5.4.1. Rozwiązanie problemów z wydajnością

5.4.2. Architektura zewnętrznego indeksu z użyciem noSQL

6. noSQL

6.1. Zastosowania

6.2. Rodzaje baz - dobór do problemu

7. Testability – projektowanie architektur aplikacji zorientowanych na testy

7.1. Strategiczne testowanie

7.1.1. Problem eksplozji kombinatorycznej przypadków testowych

7.1.2. Mapowanie warstw aplikacji na piramidę testów

7.1.3. Strategia

7.1.3.1. Warstwa domenowa - testowanie jednostkowe

7.1.3.2. Warstwa serwisów - testowanie end2end

7.2. Dążenie do uruchamiania logiki poza serwerem – zwiększanie produktywności, redukcja czasu używanego na redeploy

7.3. Zagadnienia podatności architektury na testy: problemy i pułapki

7.4. Techniki testowania jednostkowego: dummy, fake, stub, mock

7.5. Narzędzia testowania jednostkowego i integracyjnego (JUnit, Mockito)

7.6. 3 poziomy testów

7.6.1. Specification by Example - cele biznesowe