BO·TT·EGA
IT minds

Program szkolenia:

Craftsmanship - przybornik profesjonalisty: najlepsze techniki developerskie i architektoniczne

Informacje:

Nazwa:	**Craftsmanship - przybornik profesjonalisty: najlepsze techniki developerskie i architektoniczne**
Kod:	**Craft-practices-przybornik**
Kategoria:	Craftsmanship
Odbiorcy:	developerzy
Czas trwania:	3 dni
Forma:	50% wykłady / 50% warsztaty

Szkolenie stanowi syntezę kluczowych elementów klasycznej i nowoczesnej inżynierii oprogramowania. Daje ogólny pogląd na praktyczne aspekty wykorzystania omawianych technik w projektach Omawiane zagadnienia leżą u podstaw nowoczesnych frameworków i technologii – co zwiększa poziom ich zrozumienia i pozwala na świadome korzystanie.

Profesjonalista w naszym ujęciu:

- doskonale włada technikami developerskimi i stylami architektonicznymi,
- potrafi porozumieć się z biznesem przy pomocy technik DDD,
- dobiera właściwe narzędzie do klasy problemu
- dostarcza kod wysokiej jakości.

Szkolenie przeznaczone dla programistów i projektantów pragnących poszerzyć swe kompetencje w zakresie profesjonalnych technik zwiększających jakość kodu i projektu. Zdobyta wiedza przekłada się w praktyczny sposób na produktywność mierzoną w szerszej perspektywie czasu.

Forma szkolenia

Szkolenie prowadzone jest w formie warsztatu połączonego z częścią teoretyczną. Począwszy od pierwszego dnia uczestnicy pracują nad refactoringiem i rozwojem aplikacji biznesowej.

System przechodzi od podejścia opartego o anemiczny model danych do systemu podzielonego na trzy oddzielne stosy:

- Kluczowa dla systemu logika biznesowa zrealizowana w architekturze Hexagonalnej
- Wprowadzanie i modyfikacji danych słownikowych oparte o architekturę warstwową CRUD
- Projekcje udostępnione tylko do odczytu w architekturze warstwowej

Zalety szkolenia:

- Sprawdzone techniki SOLID, GRASP i DDD
- Osadzenie technik w architekturze aplikacji i systemu
- Całość w kontekście testowania automatycznego
- Realne przykłady

BO·TT·EGA
IT minds

Szczegółowy program:

1. Techniki Object Oriented

- 1.1. Ukierunkowanie myślenia w stylu OO
- 1.2. Code smell
- 1.3. Podstawowe założenia paradygmatu obiektowego
  - 1.3.1. Abstrakcja
    - 1.3.1.1. Poziomy abstrakcji
    - 1.3.1.2. Wycieki abstrakcji
    - 1.3.1.3. Single Level Of Abstraction Principle
  - 1.3.2. Enkapsulacja
    - 1.3.2.1. Tell Don’t Ask Principle
    - 1.3.2.2. Efektywne wykorzystanie wzorca Fasady
    - 1.3.2.3. Refactoring anemicznego modelu dziedziny
  - 1.3.3. Kompozycja, dziedziczenie, delegacja
    - 1.3.3.1. Zamknięcie kodu na rozbudowę
    - 1.3.3.2. Zastępowania dziedziczenia kompozycją – praktyczne zalety zmiany podejścia
      - 1.3.3.2.1. Dziedziczenie nie nadaje się do modelowania ról
      - 1.3.3.2.2. Liskov Substitution Principle
      - 1.3.3.2.3. Wzorzec Role Object i Extension Object
- 1.4. GRASP - General Responsibility Assignment Software Patterns
  - 1.4.1. Wysoka spójność i luźne powiązania
    - 1.4.1.1. Jak ją osiągnąć
    - 1.4.1.2. Jak wykrywać zapachy kodu
- 1.5. Praktyczne wykorzystanie SOLID

BOTTEGA IT minds

- 4.1.3. Layers vs Tiers
- 4.2. Architektura Hexagonalna
- 4.2.1. Podstawowe założenia
- 4.2.1.1. Otwartość na rozbudowę
- 4.2.1.2. Odporność na zmiany wymagań technologicznych
- 4.2.2. Model domenowy
- 4.2.3. Porty wejściowe
- 4.2.3.1. Otwarcie aplikacji na różne źródła danych
- 4.2.3.2. Przykładowa implementacja adapterów w specyfikacji REST
- 4.2.4. Porty wyjściowe
- 4.2.4.1. Jak implementować model domenowy niezależny od bazy danych
- 4.2.4.2. Przykładowa implementacja adapterów pamięciowych oraz SQL
- 4.2.5. Odwzorowanie architektury Hexagonalnej za pomocą pakietów oraz modułów
- 4.3. Ciągłe walidowanie założeń wybranej architektury za pomocą testów automatycznych

5. Wzorce integracyjne

- 5.1. Transactional Outbox
- 5.2. Event Notification
- 5.3. Event Carried State Transfer
- 5.4. Saga

6. Testability – projektowanie pod kątem wsparcia dla TDD

- 6.1. Podejście Specify First
- 6.2. Behaviour Driven Development
- 6.3. Projektowanie pod kątem testów z wykorzystaniem technik OO
- 6.4. Wykorzystanie Dependency Inejection

BO·TT·EGA
IT minds