BO·TT·EGA
IT minds

Program szkolenia:

Quality assurance w architekturze mikrousługowej na platformie Java

Informacje:

Nazwa:	**Quality assurance w architekturze mikrousługowej na platformie Java**
Kod:	**qa-ms**
Kategoria:	Architektura Java
Czas trwania:	2 dni
Forma:	30% wykłady, 70% ćwiczenia

Praktyki dostarczania oprogramowania typowe dla systemów monolitycznych nie sprawdzają się w świecie mikrousług. Mnogość ruchomych elementów architektury i procesu wytwórczego sprawiają, że samo testowanie nie jest wystarczające do dostarczenia wysokiej jakości produktu. Czego zatem potrzebujemy? Holistycznego podejścia do quality assurance.

Podczas tego szkolenia dowiemy się jak na jakość produktu wpływają decyzje architektoniczne, struktury organizacyjne i komunikacja. Poznamy praktyczne techniki efektywnego testowania klas, modułów, interfejsów webowych, listenerów i producentów wiadomości. Dowiemy się jak testować aplikacje w obliczu istnienia kolaboratorów takich jak bazy danych, brokery wiadomości, czy usługi HTTP. Omówimy także sposoby ograniczania skutków potencjalnych awarii na środowiskach produkcyjnych i szybkiego reagowania na problemy, zanim zgłosi je nam zdenerwowany klient. Przeanalizujemy pod kątem zapewniania jakości każdy element procesu dostawczego.

Zalety szkolenia:

- Racjonalna strategia testowania
- Testowalna architektura
- Osadzenie w procesie wdrażania

BOTTEGA IT minds

Szczegółowy program:

1. Architektura i model C4

- 1.1. Czym jest i na jakie pytania odpowiadać ma architektura?
- 1.2. Model C4
  - 1.2.1. Na jakim poziomie abstrakcji operują mikrousługi?
  - 1.2.2. Na jakim poziomie abstrakcji operują testy?

2. Wprowadzenie do architektury mikrousługowej

- 2.1. Na czym polega autonomia mikrousług?
- 2.2. Korzyści, problemy i wyzwania
- 2.3. Mikrousługi a struktura organizacyjna zespołów
  - 2.3.1. Testowanie vs Quality Assurance
  - 2.3.2. Zespół QA vs QA per zespół

3. Techniki integracyjne

- 3.1. Komunikacja synchroniczna vs komunikacja asynchroniczna
- 3.2. REST API
  - 3.2.1. Protokół HTTP
  - 3.2.2. REST jako protokół aplikacyjny a nie transportowy
- 3.3. Messaging
  - 3.3.1. Rodzaje wiadomości i kanałów komunikacyjnych
- 3.4. Jak wybór technik integracyjnych wpływa na jakość?
  - 3.4.1. SLA/SLI/SLO
- 3.5. Integracja z systemami zewnętrznych dostawców

4. Testowanie - systematyzacja pojęć

- 4.1. Znajomość domeny jako klucz do jakości

BO·TT·EGA
IT minds

- 4.2. Budowa testu - najlepsze praktyki i narzędzia
- 4.3. Test quadrant
- 4.4. Przechodniość implikacji jako sposób na testy integracyjne

5. Testowanie systemu w architekturze mikrousługowej

- 5.1. Wyzwania
- 5.2. Testy end-to-end w rzeczywistości
  - 5.2.1. Środowiska wdrożeniowe i ich wiarygodność
- 5.3. Testy kontraktowe w heterogenicznym środowisku
  - 5.3.1. Provider contracts
  - 5.3.2. Consumer contracts
  - 5.3.3. Consumer-driven contracts
    - 5.3.3.1. PACT
    - 5.3.3.2. Spring Cloud Contract

6. Testowanie mikrousługi

- 6.1. Piramida testów - fakty i mity
- 6.2. Architektura aplikacyjna a testowalność
- 6.3. Identyfikacja szwów i techniki testowania jednostkowego
- 6.4. Test doubles jako narzędzie do testowania w izolacji
- 6.5. Techniki ukrywania złożoności w testach
- 6.6. Testowanie wywołań asynchronicznych
- 6.7. Testowanie integracyjne udostępnianego REST API
  - 6.7.1. RestAssured
  - 6.7.2. Spring MockMVC
- 6.8. Testowanie integracyjne handlerów wiadomości
- 6.9. Testowanie integracyjne w obecności kolaboratorów

BOTTEGA IT minds

6.9.1. Wiremock

6.9.2. Bazy/brokery pamięciowe

6.9.3. Testcontainers

6.10. Jak zagwarantować niezależność testów integracyjnych?

6.11. Testowanie wymagań niefunkcjonalnych

6.12. TDD a dobry design

7. Dobre praktyki dostarczania oprogramowania

7.1. Miejsca i sposoby kontroli jakości w łańcuchu dostawczym

7.2. Co po wdrożeniu na produkcję?

7.3. Techniki zwinne a automatyzacji testów

8. Jak mieć pewność, że system działa po wdrożeniu?

8.1. Monitorowanie semantyczne

8.2. Metryki techniczne i biznesowe

8.3. Design for failure