BO·TT·EGA
IT minds

Program szkolenia:

Clean Architecture i testowanie automatyczne w środowisku Androida

Informacje:

Nazwa:	**Clean Architecture i testowanie automatyczne w środowisku Androida**
Kod:	**android-clean**
Kategoria:	Android
Grupa docelowa:	developerzy
Czas trwania:	2 dni
Forma:	40% wykłady / 60% warsztaty

Szkolenie zostało przygotowane z myślą o programistach Android, którzy szukają lepszych sposobów na strukturyzację architektury aplikacji i organizację porządku w kodzie.

Podczas szkolenia tworzona jest od podstaw aplikacja oparta o Clean Architecture oraz rozwiązania, które z niej wynikają: testability logiki biznesowej i modularyzacja.

Materiały wstępne

Przed szkoleniem możesz zapoznać się z serią naszych artykułów: Zaawansowane programowanie na platformie Android.

Zalety szkolenia:

- Aspekty bezpieczeństwa
- Architektura i wzorce projektowe
- Bazujemy na Clean Architecture

BOTTEGA IT minds

Szczegółowy program:

1. Aplikowanie Clean Architecture

- 1.1. Uniezależnienie logiki aplikacji od frameworka za pomocą Dependency Inversion Principle
- 1.2. Modelowanie logiki biznesowej
- 1.3. Persystencja danych w kontekście Clean Architecture
- 1.4. Komunikacja z usługami zewnętrznymi w kontekście Clean Architecture
- 1.5. Model View Presenter
  - 1.5.1. Separacja logiki widoku aplikacji od frameworka
  - 1.5.2. Poprawna implementacja MVP w kontekście cyklu życia Activity i zmian konfiguracji
  - 1.5.3. Obsługa operacji asynchronicznych
- 1.6. Techniki Inversion of Control
  - 1.6.1. Dependency Injection (Dynamiczne wstrzykiwanie zależności)
    - 1.6.1.1. Dostarczanie zależności przy użyciu narzędzia Dagger 2
    - 1.6.1.2. Definiowanie własnych scopów kontenera wstrzykiwania zależności
    - 1.6.1.3. Wstrzykiwanie zaślepek na czas testów
  - 1.6.2. Events
    - 1.6.2.1. Separacja i decoupling testowanych modułów
    - 1.6.2.2. Luźne wiązanie komponentów z użyciem Event Bus
- 1.7. Dobre praktyki i użyteczne wzorce projektowe na platformie Android

2. Testowanie i programowanie Object Oriented

- 2.1. Podstawy testowania
  - 2.1.1. Rodzaje testów i przykłady ich wykorzystania
  - 2.1.2. Zakres testów (Jednostkowe, Integracyjne, End 2 End)
  - 2.1.3. Rola testów (Akceptacyjne, Regresywne)

BO·TT·EGA
IT minds

- 2.1.4. Strategia budowania piramidy testów
- 2.2. Projektowanie przypadków testowych
  - 2.2.1. Testowanie przypadków granicznych
  - 2.2.2. Nacisk na testy jednostkowe w celu osiągnięcia wysokiego pokrycia testami
  - 2.2.3. Rozwarstwienie logiki na Aplikacyjną i Domenową
  - 2.2.4. Modelowanie logiki przy pomocy Building Blocks z Domain Driven Design
  - 2.2.5. Powtarzalność testów, wyeliminowanie losowości z testów
  - 2.2.6. Najlepsze praktyki tworzenia przypadków testowych
- 2.3. Testowanie jednostkowe
  - 2.3.1. Określanie zakresu jednostki i nazwy testów
  - 2.3.2. Implementacja testów (JUnit, Spock)
  - 2.3.3. Tworzenie własnych asercji
  - 2.3.4. Techniki: Mock, Stub, Fake (Mockito and Spock)
  - 2.3.5. Dobór technik do potrzeb - czym się kierować
  - 2.3.6. Zalety i wady testowania w izolacji
  - 2.3.7. Nagrywanie zachowania
  - 2.3.8. Weryfikacja wywołań
- 2.4. Testability - podatność kodu na testy
  - 2.4.1. Jak pisać kod, który daje się testować
  - 2.4.2. Najlepsze praktyki: SOLID
  - 2.4.3. Wybrane wzorce projektowe, które zwiększają testability: Factory, Strategy, Value Object
  - 2.4.4. Wprowadzenie do testowania z użyciem narzędzia Spock
  - 2.4.5. Code smell - "zapachy" nietestowalnego kodu
- 2.5. Android

BO·TT·EGA
IT minds

- 2.5.1. Testowanie w środowisku Andorida
- 2.5.2. Mockowanie w środowisku Androida
- 2.5.3. Narzędzia, pułapki, najlepsze praktyki
- 2.5.4. Tworzenie szybkich testów z wykorzystaniem biblioteki Robolectric
- 2.5.5. Werbalizacja asercji z użyciem AssertJ oraz rozszerzenia AssertJ - Android
- 2.5.6. Przegląd narzędzi wspomagających testowanie