BO·TT·EGA
IT minds

Program szkolenia:

iOS deep dive for advanced software developers

Informacje:

Nazwa:	**iOS deep dive for advanced software developers**
Kod:	**deep-dive**
Kategoria:	iOS
Odbiorcy:	developerzy
Czas trwania:	3 dni
Forma:	50% wykłady / 50% warsztaty

Pierwszego dnia zapoznamy się ze środowiskiem XCode. Jednak nie będziemy się skupiać na możliwościach samego IDE w kontekście pracy z kodem, tylko na istotnych aspektach związanych z projektem, takich jak: proces, konfiguracje i fazy budowania. Opanowanie samego języka Swift nie powinno stanowić problemu dla doświadczonych developerów, jednak zrozumienie zawiłości systemu zarządzania pamięcią w kontekście tego języka to już inna kwestia. Dlatego drugą część dnia poświęcimy na zrozumienie tego zagadnienia i opanowanie narzędzi które dostarcza język Swift.

Drugiego dnia uczestnicy poznają podstawowe wzorce stosowane na platformie iOS oraz podstawową budowę każdej aplikacji. Przyjrzymy się kilku wzorcom architektonicznym pod kątem łatwości budowania modularnej aplikacji i możliwości późniejszego jej testowania.

Ostatniego dnia skupimy się na zagadnieniach związanych z testowaniem. Przechodząc przez podstawy zasad testowania, praktyki i dostępne narzędzia postaramy się odpowiedzieć na pytanie, w jaki sposób najlepiej zadbać o jakość modularnej aplikacji.

Zalety szkolenia:

- Jest to trzydniowe szkolenie dla software developerów z doświadczeniem w innych technologiach.
- Szeroki zakres wiedzy: od zapoznania się ze środowiskiem, sposobami konfiguracji projektu, przez mechanizm zarządzania pamięcią aż po wzorce projektowe i testowanie aplikacji - pozwoli każdemu developerowi w miarę płynnie wejść w dowolny projekt iOS.

BOTTEGA IT minds

Szczegółowy program:

- **1. iOS Podstawy**
  - 1.1. XCode
    - 1.1.1. Ogólne omówienie funkcji IDE
    - 1.1.2. Budowanie aplikacji
      - 1.1.2.1. Ogólne omówienie procesu
      - 1.1.2.2. Targety
      - 1.1.2.3. Konfiguracje
      - 1.1.2.4. Ustawienia budowania
      - 1.1.2.5. Fazy budowania
    - 1.1.3. Podpisywanie aplikacji
    - 1.1.4. Wiersz poleceń
  - 1.2. Swift - Zarządzanie pamięcią
    - 1.2.1. ARC - wprowadzenie
      - 1.2.1.1. Historia
      - 1.2.1.2. Zasada działania
    - 1.2.2. Cykle referencji
      - 1.2.2.1. Słabe i silne dowiązania w obiektach
      - 1.2.2.2. Bloki
    - 1.2.3. Optionale
      - 1.2.3.1. Praca z opcjonalnymi zmiennymi
- **2. Budowa aplikacji iOS**
  - 2.1. Przegląd podstawowych zagadnień.
  - 2.2. Popularne wzorce architektoniczne

BOTTEGA IT minds

- 2.2.1. MVC
- 2.2.2. MVVM
- 2.2.3. MVP
- 2.3. Popularne wzorce projektowe
- 2.3.1. Delegate
- 2.3.2. Data Source
- 2.4. Modularność
- 2.4.1. Dlaczego poprawna hermetyzacja jest tak ważna?
- 2.4.2. Architektura hexagonalna
- 2.4.2.1. Praktyczne przykłady zastosowań w iOS
- 2.5. Interfejs użytkownika
- 2.5.1. Klasyczny UIKit
- 2.5.1.1. Wprowadzenie
- 2.5.1.2. Zasady budowy interfejsu
- 2.5.2. Nowoczesny SwiftUI
- 2.5.2.1. Wprowadzenie
- 2.5.2.2. Zasady budowy interfejsu
- 2.5.2.3. Mikro wprowadzenie do frameworka Combine

3. Testowanie aplikacji

- 3.1. Jak pisać kod podatny na testowanie
- 3.1.1. Dobre praktyki: SOLID
- 3.1.2. Jak zmieniać istniejący kod do dającego się testować
- 3.1.3. Pułapki i typowe błędy
- 3.2. Zakres testów
- 3.2.1. Testowanie jednostkowe

BOTTEGA IT minds

- 3.2.2. Testowanie integracyjne
- 3.2.3. Testowanie funkcjonalne
- 3.3. Wybrane wzorce i techniki testowe - omówienie i przykłady zastosowania
  - 3.3.1. Mock, Stub, Sut
  - 3.3.2. Memory implementation
  - 3.3.3. Test Driven Development
    - 3.3.3.1. Cykl czerwony-zielony-refaktoring
    - 3.3.3.2. Ewolucyjny rozwój kodu
    - 3.3.3.3. Podstawowe techniki refaktoringu
- 3.4. Testowanie interfejsów użytkownika
  - 3.4.1. Co testować?
  - 3.4.2. Page Object Pattern
  - 3.4.3. Wstrzykiwanie stanu
  - 3.4.4. Architektura pozwalająca na testy dowolnego ekranu aplikacji bez konieczności “dochodzenia” do niego.