BOTTEGA IT minds

Program szkolenia:

Wprowadzenie do iOS (Swift)

Informacje:

Nazwa:	**Wprowadzenie do iOS (Swift)**
Kod:	**ios-swift**
Kategoria:	iOS
Odbiorcy:	developerzy
Czas trwania:	3 dni
Forma:	50% wykłady / 50% warsztaty

Jest to szkolenie wprowadzające w meandry budowania aplikacji dla systemu iOS (opcja tylko z językiem Swift). Trzydniowe szkolenie zaprojektowane jest w taki sposób, aby po jego zakończeniu uczestnik był w stanie samodzielnie zbudować i przetestować prostą aplikację.

Pierwszy dzień - to intensywne wprowadzenie w dwie podstawowe rzeczy wymagane aby móc pracować z systemem iOS: zarządzanie pamięcią i język Swift. Są to dwa powiązane ze sobą elementy, których zrozumienie i przyswojenie pozwala w kolejnych dniach skupić się na eksplorowaniu dobrych praktyk budowy i testowania aplikacji.

Drugiego dnia uczestnicy poznają podstawowe wzorce stosowane na platformie iOS oraz podstawową budowę każdej aplikacji. Przyjrzymy się kilku wzorcom architektonicznym pod kątem łatwości budowania aplikacji i możliwości późniejszego jej testowania. Obszerną częścią tego dnia jest wprowadzenie do budowania interfejsu aplikacji. W tej części uczestnicy poznają dwa podstawowe frameworki: klasyczny UIKit oraz nowoczesny SwiftUI.

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 aplikacji. Które części aplikacji powinny być testowane, a które niekoniecznie. Odkryjemy, w jaki sposób powinna być zbudowana aplikacja i testy aby dało się przetestować dowolny jej ekran bez konieczności “dochodzenia” do niego w wielu krokach.

BOTTEGA IT minds

Szczegółowy program:

1. Swift - podstawy

- 1.1. Składnia języka - ogólne wprowadzenie
- 1.2. Zarządzanie pamięcią
  - 1.2.1. Wprowadzenie teoretyczne
  - 1.2.2. ARC
    - 1.2.2.1. Podstawy
    - 1.2.2.2. Praktyczne przykłady
    - 1.2.2.3. Jak unikać wycieków pamięci (i dlaczego to jest groźne)
- 1.3. Zmienne i stałe
  - 1.3.1. Optionals
- 1.4. Podstawowe typy danych
  - 1.4.1. String, interpolacja stringów
  - 1.4.2. Liczby
  - 1.4.3. Kolekcje
- 1.5. Sterowanie przepływem
  - 1.5.1. Instrukcje warunkowe
  - 1.5.2. Pętle
- 1.6. Operatory
- 1.7. Klasy i struktury
  - 1.7.1. Właściwości klas
  - 1.7.2. Poziomy dostępu
  - 1.7.3. Protokoły
  - 1.7.4. Polimorfizm i typecasting

BOTTEGA IT minds

1.7.5. Rozszerzenia klas

1.8. Bloki

1.8.1. Wprowadzenie

1.8.2. Deklarowanie bloków

1.8.3. Wywoływanie i przekazywanie

1.8.4. Zarządzanie pamięcią

1.9. Szybki przegląd paradygmatów programowania w kontekście języka Swift

1.9.1. Programowanie imperatywne

1.9.2. Programowanie deklaratywne

1.10. Moduły

1.10.1. Swift Package Manager

2. Budowa aplikacji iOS

2.1. Przegląd podstawowych zagadnień.

2.2. Popularne wzorce architektoniczne

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

BOTTEGA IT minds

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

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?

BOTTEGA IT minds

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