

## Program szkolenia:

# Wieloplatformowe aplikacje mobilne przy użyciu frameworka Flutter i języka Dart

## Informacje:

<b>Nazwa:</b>	<b>Wieloplatformowe aplikacje mobilne przy użyciu frameworka Flutter i języka Dart</b>
<b>Kod:</b>	<b>Flutter-apps</b>
<b>Kategoria:</b>	Flutter
<b>Odbiorcy:</b>	developerzy, architekci
<b>Czas trwania:</b>	3 dni
<b>Forma:</b>	30% wykłady / 70% warsztaty

W trakcie szkolenia uczestnik nauczy się posługiwać językiem Dart i tworzyć kompletne, wydajne i przetestowane automatycznie aplikacje mobilne na platformę iOS i Android z wykorzystaniem frameworka Flutter. Dodatkowo, pozna dobre praktyki strukturyzowania aplikacji pod względem architektonicznym, separując kluczową logikę biznesową od widoku aplikacji.

Flutter jest pierwszym stabilnym frameworkiem pozwalającym tworzyć wieloplatformowe aplikacje mobilne, które pod względem wydajności i User Experience nie ustępują aplikacjom natywnym.

## Zalety szkolenia:

- Architektura aplikacji
- Wydajność
- Realne klasy problemów

## Szczegółowy program:

### 1. Wprowadzenie do Fluttera

- 1.1. Architektura frameworka
- 1.2. Aplikacje w Darcie i Flutterze a aplikacje natywne
- 1.3. Porównanie Fluttera z alternatywnymi rozwiązaniami

### 2. Język Dart

- 2.1. Podstawy składni
- 2.2. Deklaracja zmiennych stałych i funkcji
- 2.3. Optymalizacja alokacji obiektów poprzez używanie const
- 2.4. Kontrola przepływu if/switch/for/while
- 2.5. Funkcje
- 2.6. Obsługa wyjątków
- 2.7. Konstrukcje języka do programowania obiektowego
- 2.8. Konstrukcje umożliwiające programowanie funkcyjne
- 2.9. Programowanie asynchroniczne

### 3. Tworzenie aplikacji mobilnych przy użyciu Fluttera

- 3.1. Struktura projektu i narzędzia
- 3.2. Tworzenie widoku aplikacji
  - 3.2.1. Tworzenie hierarchii widgetów
  - 3.2.2. Budowanie layoutów aplikacji przy użyciu widgetów z biblioteki standardowej
  - 3.2.3. Obsługa interakcji użytkownika
  - 3.2.4. Widgety stanowe i bezstanowe
  - 3.2.5. Implementacja nawigacji i routingu
  - 3.2.6. Zarządzanie stanem widoków

3.2.7. Widok jako funkcja stanu aplikacji

3.2.8. Animacje

3.2.9. Optymalizacja wydajności

3.2.9.1. Selektywne odświeżanie hierarchii widgetów

3.2.9.2. Redukcja alokacji poprzez użycie const widgetów

3.2.10. Renderowanie własnych widgetów

3.3. Persistencja danych i komunikacja z backendem

3.3.1. Konsumowanie RESTowego API i serializacja JSONów

3.3.2. Persistencja prostych danych w key-value store (preferences)

3.3.3. Persistencja plików

3.3.4. Persistencja danych w bazie SQLite

3.4. Integracja aplikacji z natywną platformą

3.5. Testowanie automatyczne

3.5.1. Testowanie jednostkowe poszczególnych elementów aplikacji

3.5.2. Testy end-to-end aplikacji mobilnej

## 4. Tworzenie aplikacji otwartych na rozbudowę

4.1. Użyteczne wzorce projektowe i architektoniczne pozwalające na tworzenie rozbudowanych projektów

4.1.1. Oddzielenie logiki aplikacyjnej od widoku

4.1.2. Zasada odwracania zależności

4.1.3. Modularyzacja aplikacji

4.2. Porównanie wzorców i najlepszych praktyk do zarządzania stanem aplikacji

4.2.1. Provider and Scoped Model

4.2.2. Redux

4.2.3. BLoC

4.2.4. MobX

