

## Program szkolenia:

# Techniki efektywnego testowania kodu dla programistów .NET

## Informacje:

<b>Nazwa:</b>	<b>Techniki efektywnego testowania kodu dla programistów .NET</b>
<b>Kod:</b>	<b>NET-test</b>
<b>Kategoria:</b>	.NET
<b>Odbiorcy:</b>	developerzy, testerzy
<b>Czas trwania:</b>	3 dni
<b>Forma:</b>	30% wykłady, 70% ćwiczenia

---

Podczas szkolenia uczestnicy poznają techniki pragmatycznego podejścia do wytwarzania testowalnego oprogramowania

Podczas warsztatów praktycznych uczestnicy posiadą umiejętność pisania kodu który skutecznie można testować, jak i same techniki testowania przy wykorzystaniu narzędzi powszechnie stosowanych w środowisku .NET. Szkolenie przeznaczone jest zarówno dla grup które nigdy nie miały do czynienia z testami oraz dla grup które już testują, ale czują, że z testów można wyciągnąć znacznie więcej wartości niż dotychczas.

## Zalety szkolenia:

- Tworzenie skutecznych i łatwych w utrzymaniu testów automatycznych
- Najlepsze wzorce i praktyki (w tym TDD)
- Sensowne testowanie integracyjne
- Praca z popularnymi narzędziami do testowania (XUnit lub NUnit, SpecFlow)
- Testy integracyjne przy wykorzystaniu Microsoft Testing library, NSubstitute lub Moq

## Szczegółowy program:

### 1. Jednostkowe testowanie kodu

- 1.1. Dlaczego warto automatycznie testować kod?
- 1.2. Dobre testy jednostkowe
- 1.3. Struktura testów jednostkowych
- 1.4. Nazewnictwo testów
- 1.5. Konstrukcje wspierające testowalność kodu
- 1.6. Antywzorce dla testowalnego kodu
- 1.7. Pisanie czytelnych asercji (FluentValidation)
- 1.8. Ćwiczenia praktyczne (w trakcie, przeplatane z teorią)
- 1.9. Przyjrzenie się metryką testów (code coverage)

### 2. Wzorce tworzenia testów

- 2.1. Testowania zachowania a nie metod
- 2.2. Techniki przygotowania stanu testu
- 2.3. Techniki weryfikacji rezultatu testów

### 3. Test-Driven Development

- 3.1. Podstawowe założenia
- 3.2. Cykl red-green-refactor
- 3.3. Przejrzysta struktura testu
- 3.4. Wybór kolejnych funkcji do zaimplementowania
- 3.5. Sprawne uruchamianie testów z IDE (przydatne wtyczki, skróty klawiszowe)
- 3.6. Ćwiczenia praktyczne z TDD w formie Kata/Coding Dojo
- 3.7. Korzyści ze stosowania TDD

### 4. Bezpieczny refactoring

4.1. Wyjaśnienie istoty refactoringu

4.2. Przydatne przekształcenia kodu

4.3. Ćwiczenia praktyczne

## 5. Mockowanie w akcji

5.1. Wprowadzenie do "test doubles" (różnice między test doubles, mocks, stubs, fakes, dummies)

5.2. Potrzeba, kiedy i dlaczego warto?

5.3. Mockowanie z wybranym frameworkiem (NSubstitute, Moq)

5.4. Ćwiczenia praktyczne

## 6. Testowanie z bazą danych

6.1. Rozwiązywanie problemów z zależnościami między testami

6.2. Zarządzanie transakcjami bazodanowymi w testach

6.3. Utrzymywanie zestawu danych testowych

6.4. Testowanie z bazą danych w pamięci

6.5. Kiedy warto mockować warstwę dostępu do bazy danych

6.6. Ćwiczenia praktyczne

## 7. Pisanie testowalnego kodu

7.1. Wysoka kohezja

7.2. Niski coupling

7.3. Wzorce wspomagające pisanie testowalnego kodu

## 8. Wybrane tematy z testowania akceptacyjnego

8.1. Wprowadzenie do BDD (Behaviour-Driven Development)

8.2. BDD z wykorzystaniem SpecFlow

8.3. Testy akceptacyjne

8.3.1. systemowe

8.3.2. komponentów

## 9. Weryfikacja testów za pomocą mutantów