SZKOLENIE ŚREDNIO ZAAWANSOWANE

# Testy automatyczne i Test Driven Design w JavaScript

JS/TDD

Czas trwania: 2 dni (16h)

Testowanie automatyczne pozwala szybciej dostarczać oprogramowanie, przy tym unikając typowych błędów oraz regresji - dzięki temu programista może poświecić więcej czasu na dostarczanie funkcjonalności, a nie szukanie błędów

## Cele szkolenia

- Zaznajomienie z narzędziami oraz metodyką tworzenia testów na każdym poziomie oraz według sprawdzonych podejść

## Zalety

- Szkolenie skupia się na budowaniu praktycznych umiejętności na każdym poziomie testów - od testowania jednostkowego, przez integrację po testowanie funkcyjne e2e
- Podczas szkolenia uczestnicy poznają praktyczne techniki pracy i różne podejścia do testowania, w tym podejście TDD, czyli najpierw piszemy testy, a potem kod
- Praktyka przed teorią - wszystkie szkolenia technologiczne prowadzone są w formie warsztatowej. Konieczna teoria jest wyjaśniana na przykładzie praktycznych zadań
- Konkretne umiejętności - w ramach każdego szkolenia rozwijamy praktyczne umiejętności związane z daną technologią i tematyką
- Nauka z praktykami - wszyscy trenerzy na co dzień pracują w projektach, gwarantuje to dostęp do eksperckiej wiedzy i praktycznego know-how

## Dla kogo?

- Programiści chcący pisać lepszy i testowalny kod oraz poznać techniki test first oraz design by specification

## Wymagania

- Podstawowa znajomość JavaScript

## Program

1. Wprowadzenie
    a. Dlaczego testować kod?

   b. Rodzaje aplikacji i testowanie
   c. Piramida testów - wady i zalety każdej warstwy
2. Testy statyczne
   a. Statyczna analiza kodu
   b. Narzędzia i Reguły
   c. Dobre praktyki
   d. Automatyzacja, wtyczki i githooks
3. Testy Jednostkowe
   a. Konfiguracja narzędzi
   b. Struktura i nazewnictwo testów
   c. Asercje Jasmine i Chai
   d. Raportowanie
   e. Zamienniki - Stubs i Mocks
   f. Testowanie Czasu - Asynchroniczność
   g. Wzorzec Arrange/Act/Assert
   h. Testy parametryczne i fuzztesty
4. Testy Integracyjne
   a. Testy współpracujących obiektów
   b. Test Driven Design - testy najpierw
   c. Cykl Red-Green-Refactor
   d. Wykrywanie regresji
   e. Pokrycie kodu testami
   f. Testowanie interfejsów
   g. Testowanie zapytań HTTP
   h. Metoda czarnej skrzynki - blackbox testing
   i. Programowanie kontraktowe
   j. Metoda migawek - Snapshot testing
   k. Tworzenie utrzymywalnych i stabilnych testów
5. Refaktoryzacja kodu
   a. Testowalne aplikacje
   b. Code smells, antywzorce a dobre praktyki
   c. Modularna aplikacja i enkapsulacja
   d. Refaktoryzacja do jednej odpowiedzialności
   e. Refaktoryzacja do otwarte zamknięte
   f. Refaktoryzacja do segregowanych kontraktów
   g. Refaktoryzacja do odwróconych zależności
   h. Testowanie z dependency injection
6. Testy End-to-End
   a. Omówienie koncepcji testów e2e
   b. Wady i zalety testowania w przeglądarce
   c. Narzędzia i środowisko e2e
   d. Porównanie Selenium, WebDriver, Protractor, Cypress i inne
   e. Behavior Driven Design/Specification by Example
   f. Testy akceptacyjne

   g. Scenariusze testowe
   h. Akcje asynchroniczne
   i. Mockowanie zależności
7. Testy a Ciągła Integracja i Dostarczanie (CD/CI)
   a. Automatyzacja Testów
   b. Synchronizacja z Repozytorium kodu
   c. Testy jako część Code Review
   d. Raportowanie wyników testów
   e. Omówienie Continuous deployment and delivery