PROGRAM SZKOLENIA TESTOWANIE

Szkolenie Test-driven development

Nauczysz się projektować kod małymi krokami, zaczynając od testu, który mówi, czego potrzebujesz.

CZAS TRWANIA	POZIOM	FORMA
2 dni	**średniozaawansowany**	**warsztat dla zespołu, zdalnie lub stacjonarnie**

Z TDD piszesz mniej kodu, który potem trzeba poprawiać, bo test najpierw pyta, jak moduł ma być używany, a dopiero potem, jak ma działać. W efekcie dostajesz mniejsze moduły, prostsze API i zestaw testów, który powstaje przy okazji, a nie na końcu sprintu. To podejście działa także wtedy, gdy kod pisze za Ciebie asystent AI: czerwony test staje się dla niego specyfikacją.

To szkolenie skupia się na samym procesie: rytmie red, green, refactor, wielkości kroków i decyzjach projektowych. Większość czasu spędzisz przy klawiaturze, pracując w parach na katach i na zadaniu zbliżonym do codziennej pracy. Wyjdziesz z listą testów i gotowym, przetestowanym modułem z tego zadania.

Jeśli dopiero zaczynasz z testami, zacznij od szkolenia Testy jednostkowe, a jeśli problemem jest duży, nieprzetestowany kod, wybierz Legacy Code.

PROGRAM

2 dni

- Czym jest TDD, a czym nie jest?
  - TDD jako technika projektowania, nie testowania
  - Test First a Test After
  - Trzy prawa TDD według Roberta C. Martina
- Cykl red, green, refactor
  - Vitest w trybie watch jako szybka pętla informacji zwrotnej
  - Najmniejszy test, który nie przechodzi
  - Najprostsza implementacja: fake it, obvious implementation
  - Triangulacja kolejnymi przykładami
  - Refaktoryzacja pod ochroną zielonych testów
- Baby steps
  - Lista testów jako plan pracy
  - Kolejność przypadków: od prostych do brzegowych
  - Za duży krok: cofnięcie zmian i mniejszy test

- Praktyka: katy w parach
  - String Calculator, Bowling Game, Roman Numerals
  - Ping-pong pairing
  - Rotacja ról i rytm pracy w parze
  - Ta sama kata drugi raz: porównanie rozwiązań
- Dublery testowe w TDD
  - Kiedy dubler pomaga projektować, a kiedy przeszkadza
  - Wstrzykiwanie zależności zamiast mockowania modułów
  - Testowanie zachowania, a nie implementacji
- Outside-in i inside-out
  - Szkoła klasyczna (Detroit/Chicago) i londyńska (mockist)
  - Test akceptacyjny jako punkt startowy
  - Dobór podejścia do problemu
- Warsztat: zadanie z codziennej pracy
  - Lista testów rozpisana z wymagań zadania
  - Moduł budowany w parach od testu akceptacyjnego
  - Przegląd gotowego modułu i jego testów
- TDD w codziennej pracy
  - Co sprawdza TypeScript, a co musi sprawdzić test
  - Komponent UI zaczynany od testu z Testing Library
  - Test, który odtwarza zgłoszony błąd
- TDD z asystentem AI
  - Czerwony test jako specyfikacja dla asystenta AI
  - Testy dopisane przez AI pod gotowy kod: dlaczego to nie TDD

DLA KOGO?

- Dla programistów JavaScript i TypeScript, którzy piszą już testy.
- Dla osób, którym testy pisane po fakcie wydają się stratą czasu.
- Dla zespołów, które chcą wprowadzić TDD i pair programming.
- Dla zespołów, które pracują z asystentem AI i chcą panować nad kodem.

CO MUSISZ UMIEĆ?

- Pisać kod w JavaScript lub TypeScript.
- Napisać prosty test jednostkowy w Vitest lub Jest.
- Korzystać z asystenta AI w edytorze (moduł o TDD z asystentem AI).

CZEGO SIĘ NAUCZYSZ?

- Rozpiszesz zadanie na listę testów i przejdziesz ją małymi krokami.
- Utrzymasz rytm red, green, refactor bez skakania do implementacji.

- Przećwiczysz TDD w parze na katach i prawdziwym zadaniu.
- Dobierzesz dubler testowy, który pomaga projektować, zamiast betonować implementację.
- Wybierzesz outside-in albo inside-out zależnie od problemu.
- Rozdzielisz to, co sprawdza TypeScript, od tego, co musi sprawdzić test.
- Napiszesz komponent UI i poprawkę błędu, zaczynając od testu.
- Poprowadzisz asystenta AI czerwonym testem zamiast opisem w prompcie.

TIP: Więcej o podejściu znajdziesz we wpisach Test First oraz TDD i pair programming.

OPINIE UCZESTNIKÓW

> *„Gruntownie, krok po kroku stworzyliśmy aplikację sklepu internetowego, po drodze pisząc testy jednostkowe, e2e, web komponenty, autorski backend oparty na Node.js + Express.js, jak również wykorzystaliśmy narzędzia CI."*
>
> Łukasz, JavaScript Developer (WarsawJS MasterClass)

> *„The whole team was able to learn from the mini-workshops and presentations he regularly shared with us, covering advanced Git tricks, React, JavaScript and more."*
>
> Paul Ngei, współpracownik z zespołu w Box (rekomendacja na LinkedIn)

TRENER

Piotr Kowalski

W branży IT od 2008 roku. Od 2014 jest mentorem dla wielu developerów. Jako trener spędził na sali szkoleniowej ponad 1000 godzin, a na LinkedIn ma 52 rekomendacje. Każde jego szkolenie to dawka solidnej wiedzy oraz humoru.

Więcej: piecioshka.pl/piotr-kowalski

ZORGANIZUJMY TO SZKOLENIE W TWOIM ZESPOLE

Program dopasuję do Waszego projektu i poziomu grupy. Napisz, z czym pracujecie, a odpowiem z propozycją terminu i zakresu.

KONTAKT **piecioshka.pl/kontakt**
E-MAIL **piecioshka+blog [małpka] gmail.com**
SZKOLENIE **piecioshka.pl/szkolenia/tdd/**