Intro do TDD

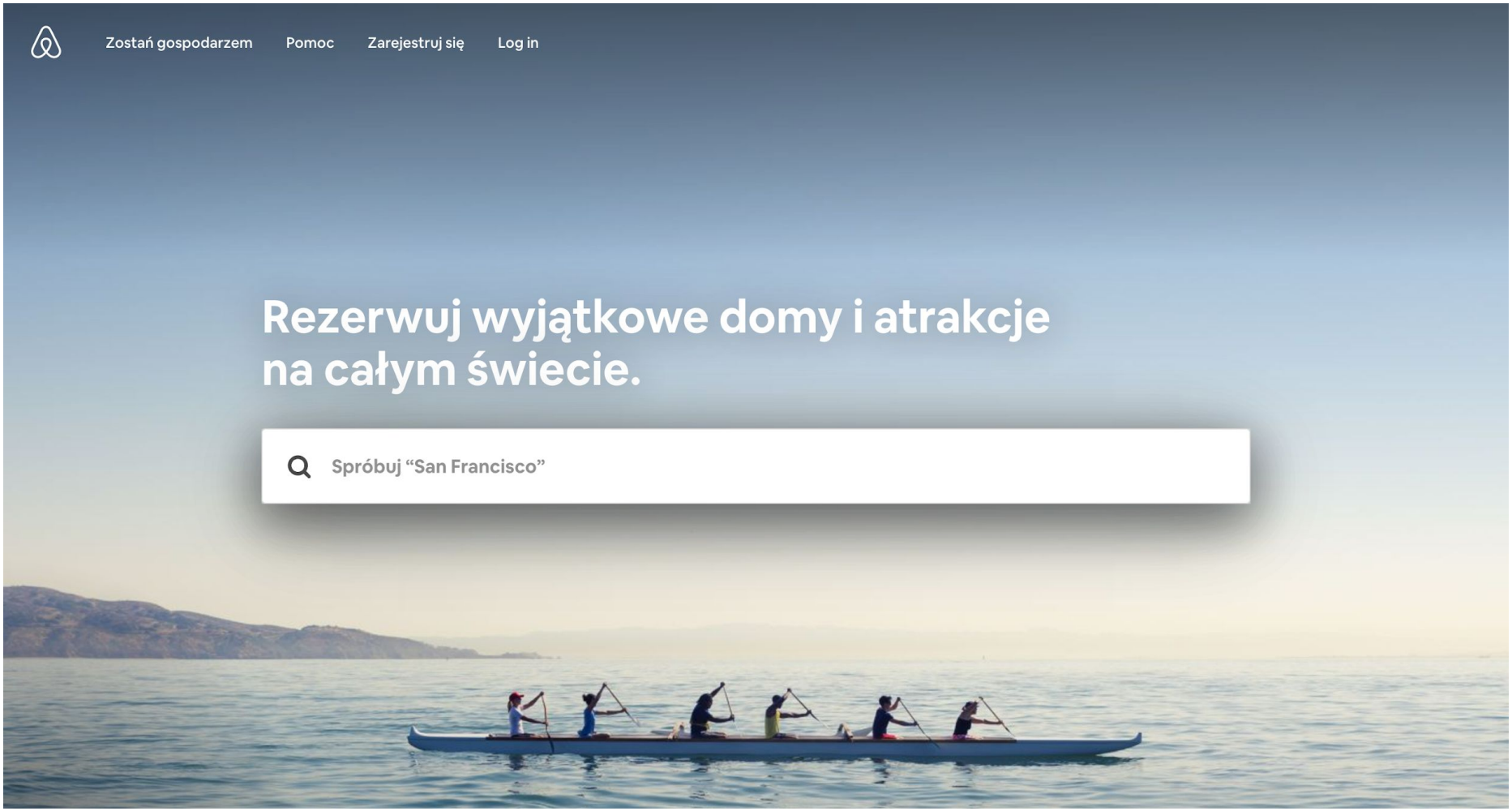
Damian Kampik

Agenda

- Przyczyny zakażeń bugami, czyli po co testować
- Zapobieganie i leczenie, czyli o TDD
- Sposoby dawkowania, czyli narzędzia wspomagające proces

Przyczyny zakażeń bugami

u2i


Zostań gospodarzem
Pomoc
Zarejestruj się
Log in
Rezerwuj wyjątkowe domy i atrakcje na całym świecie.
Spróbuj “San Francisco”

u2i

u2i

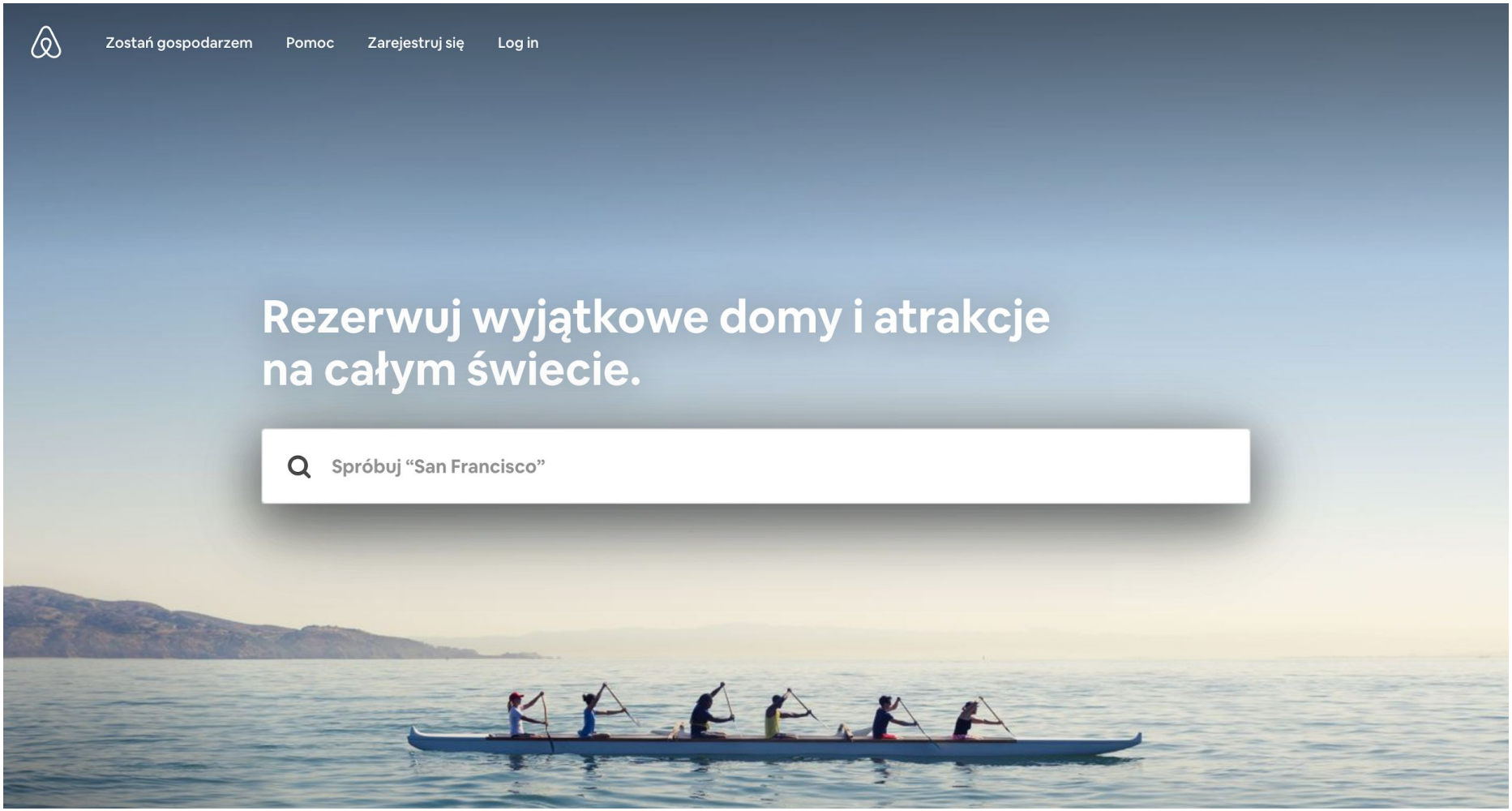
```ruby
module ThatIsUsedEverywhere
  class AndYouDontKnowAboutIt
    def very_important_method(params)
      params.first
    end
  end
end
```

u2i

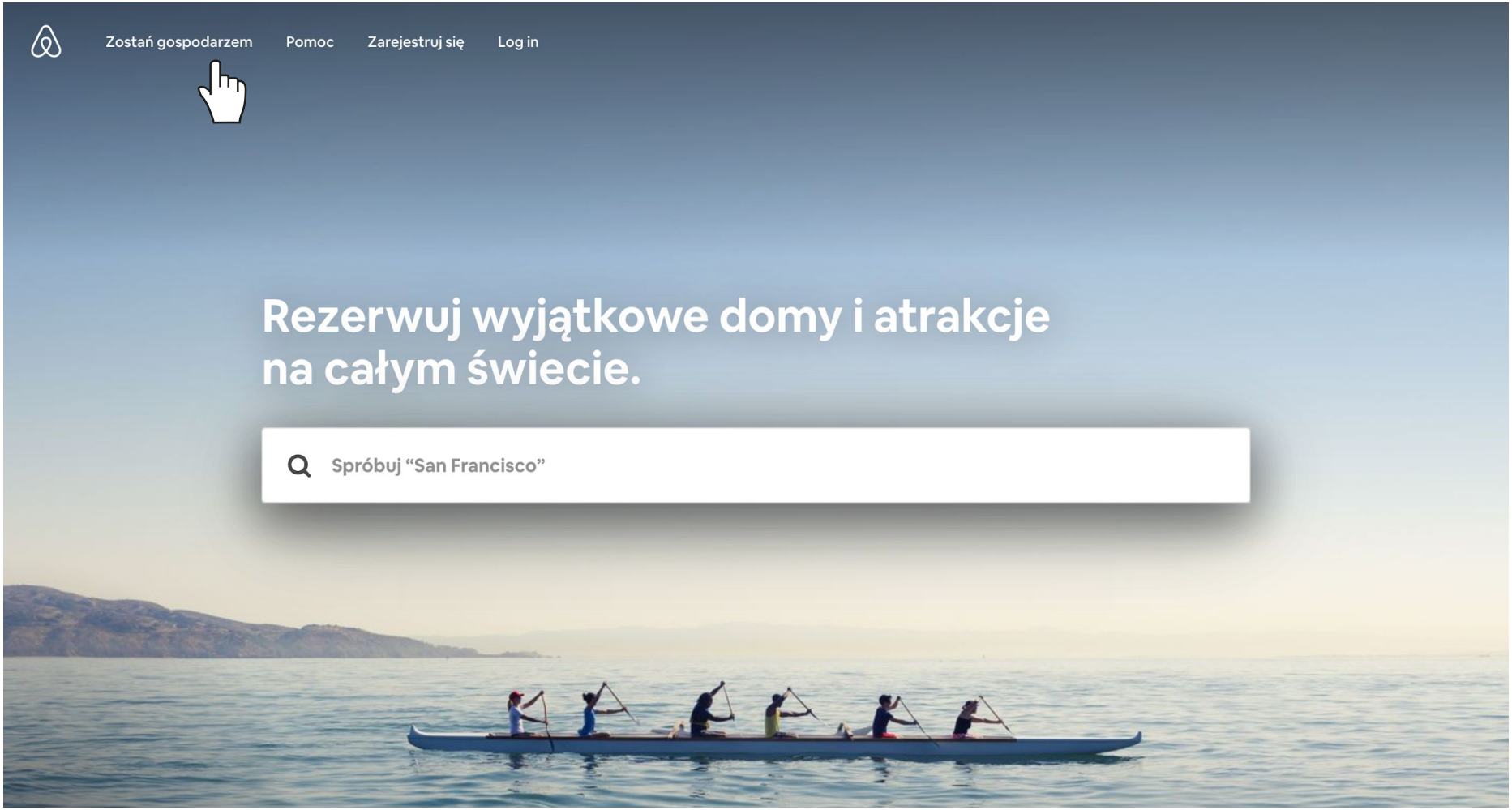
```ruby
module ThatIsUsedEverywhere
  class AndYouDontKnowAboutIt
    def very_important_method(params)
      params.first
    end
  end
end
```

```ruby
module ThatIsUsedEverywhere
  class AndYouDontKnowAboutIt
    def very_important_method(params)
      params.last
    end
  end
end
```

u2i


Zostań gospodarzem
Pomoc
Zarejestruj się
Log in
Rezerwuj wyjątkowe domy i atrakcje na całym świecie.
Spróbuj “San Francisco”

u2i

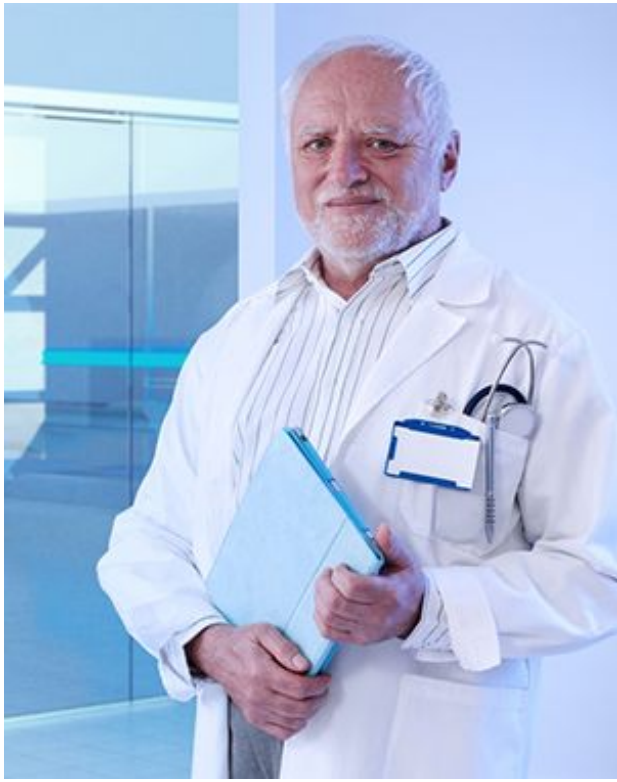

Zostań gospodarzem
Pomoc
Zarejestruj się
Log in
Rezerwuj wyjątkowe domy i atrakcje na całym świecie.
Spróbuj “San Francisco”

u2i


500
There was an error.

u2i

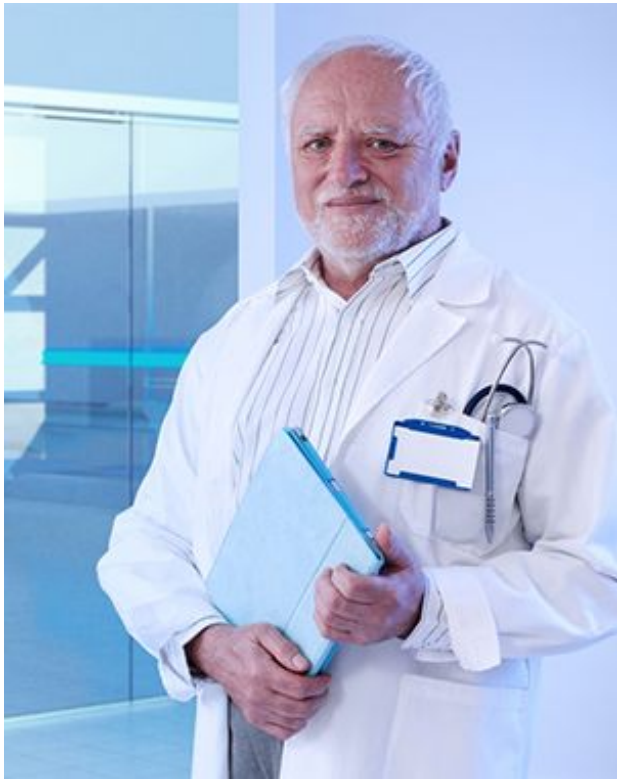
Diagnoza:



u2i

Diagnoza:

BRAK TESTÓW



u2i

Zapobieganie i leczenie

u2i

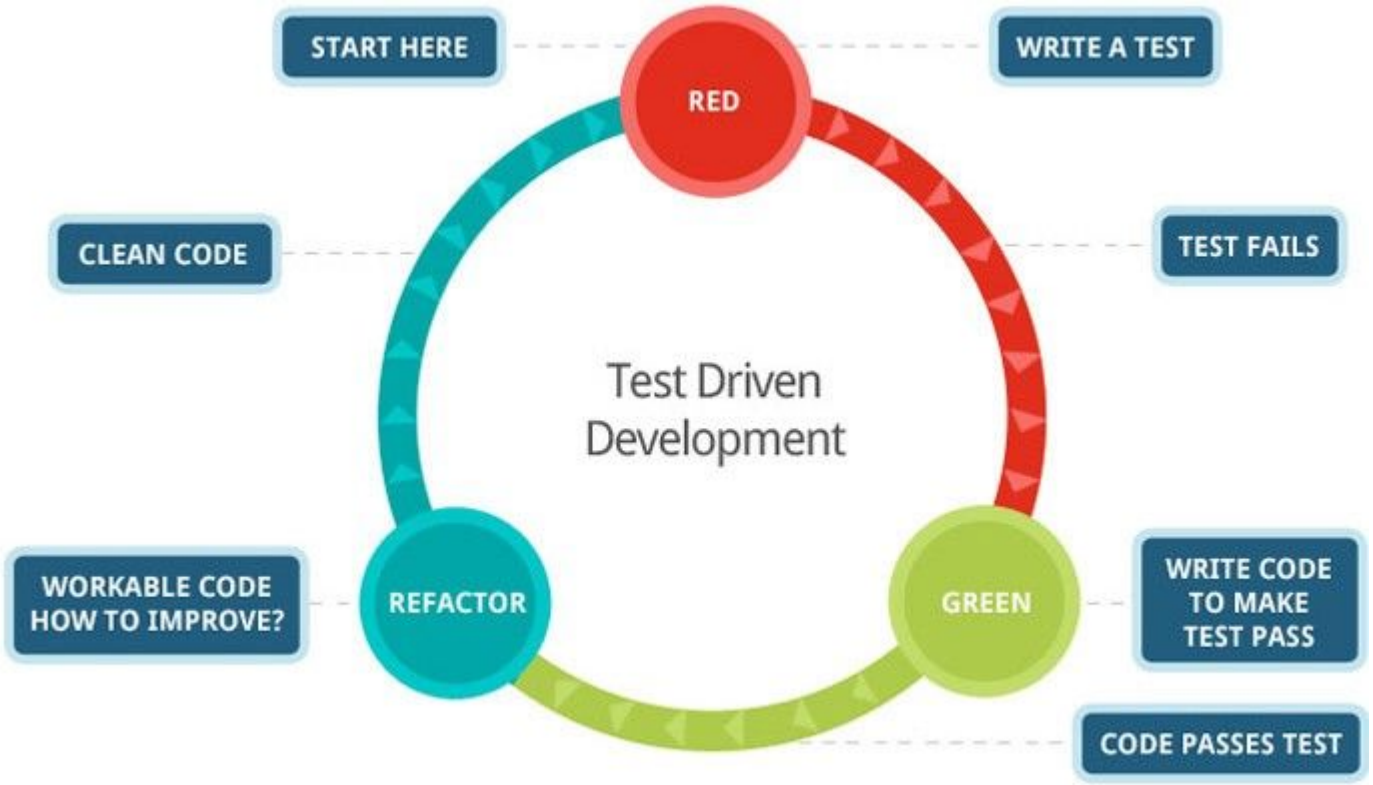
APAP?

Ibuprofen?

C6H12O6?

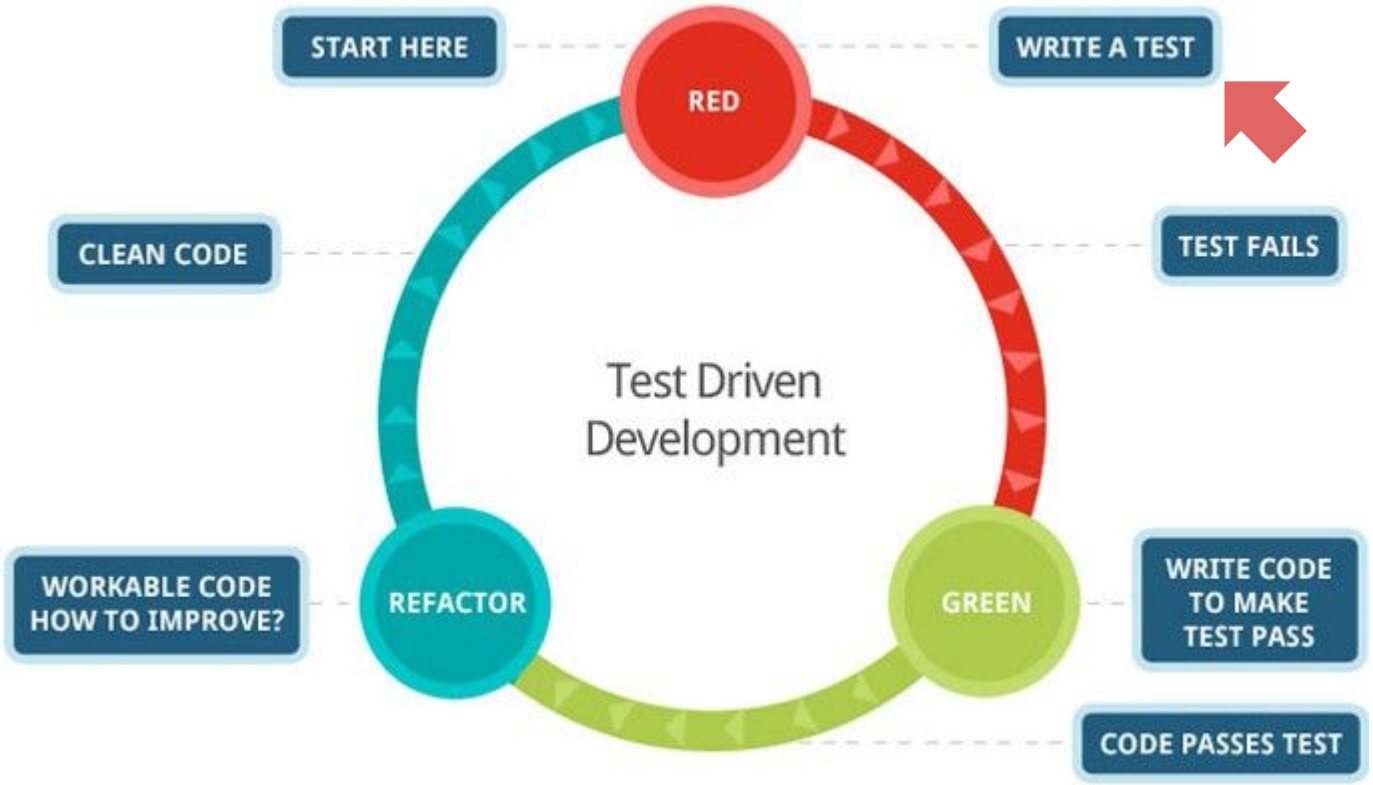
TDD!

TDD


START HERE
RED
WRITE A TEST
CLEAN CODE
TEST FAILS
Test Driven
Development
WORKABLE CODE
HOW TO IMPROVE?
REFACTOR
GREEN
WRITE CODE
TO MAKE
TEST PASS
CODE PASSES TEST

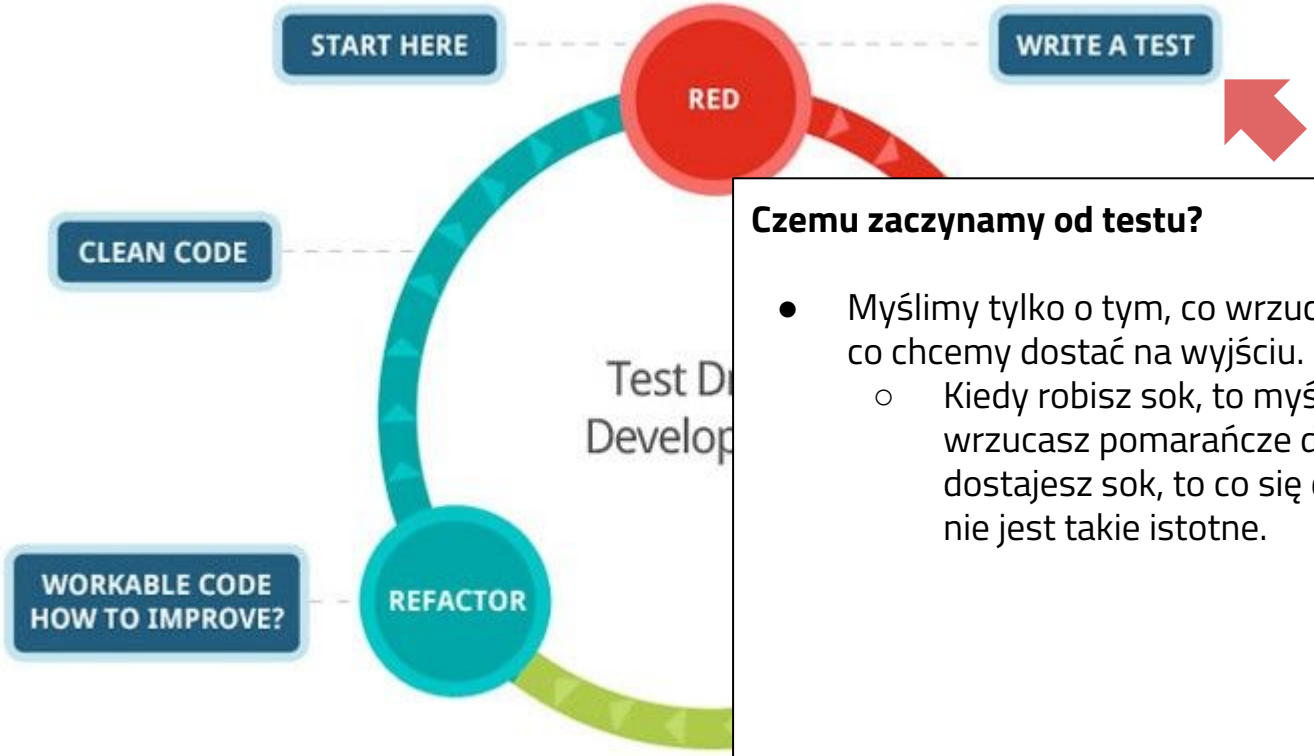
u2i

TDD


START HERE
RED
WRITE A TEST
CLEAN CODE
TEST FAILS
Test Driven
Development
WORKABLE CODE
HOW TO IMPROVE?
REFACTOR
GREEN
WRITE CODE
TO MAKE
TEST PASS
CODE PASSES TEST

u2i

TDD

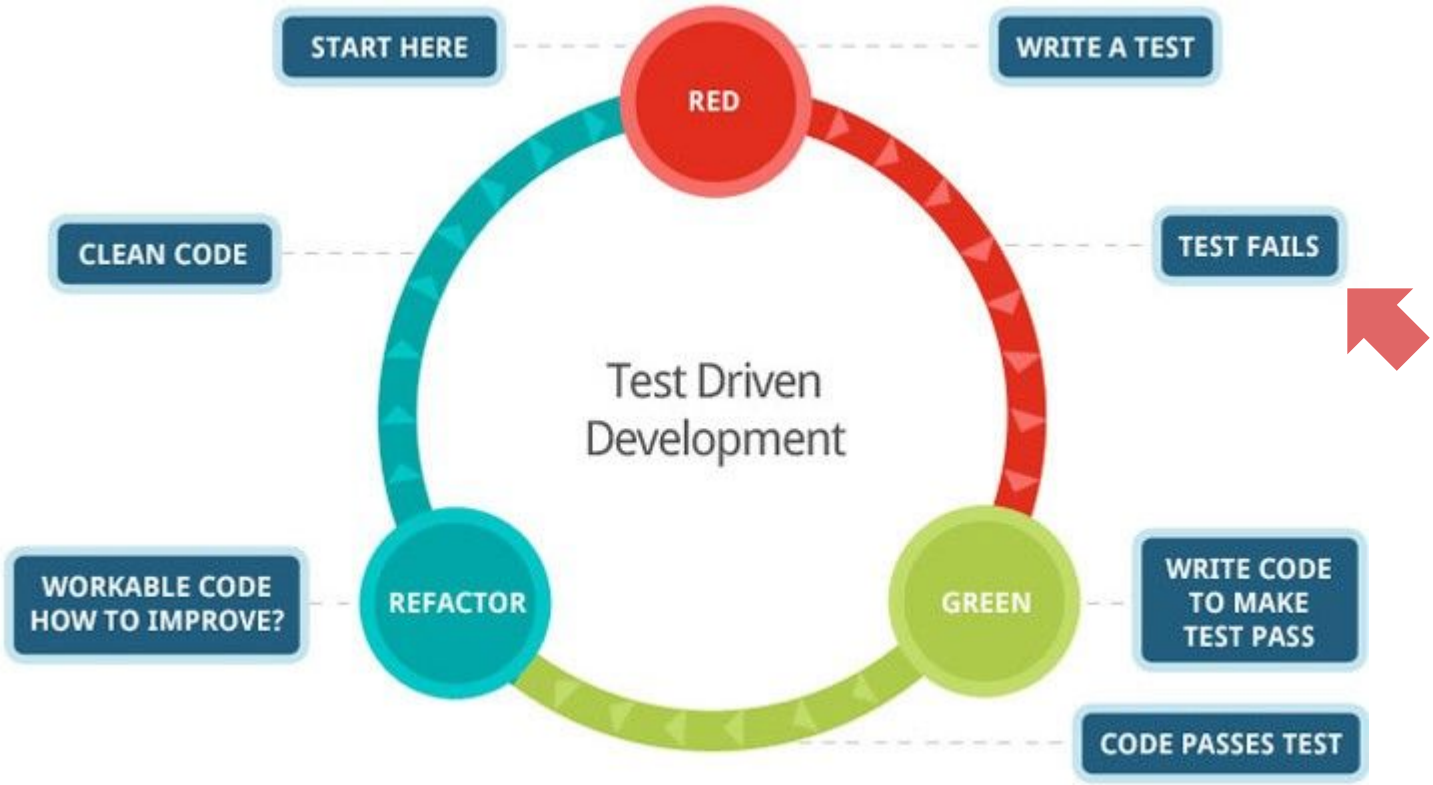


Czemu zaczynamy od testu?

- Myślimy tylko o tym, co wrzucamy na wejście i co chcemy dostać na wyjściu.
  - Kiedy robisz sok, to myślisz o tym, że wrzucasz pomarańcze do sokowirówki i dostajesz sok, to co się dzieje pomiędzy nie jest takie istotne.

u2i

TDD


START HERE
RED
WRITE A TEST
CLEAN CODE
TEST FAILS
Test Driven
Development
WORKABLE CODE
HOW TO IMPROVE?
REFACTOR
GREEN
WRITE CODE
TO MAKE
TEST PASS
CODE PASSES TEST

u2i

TDD

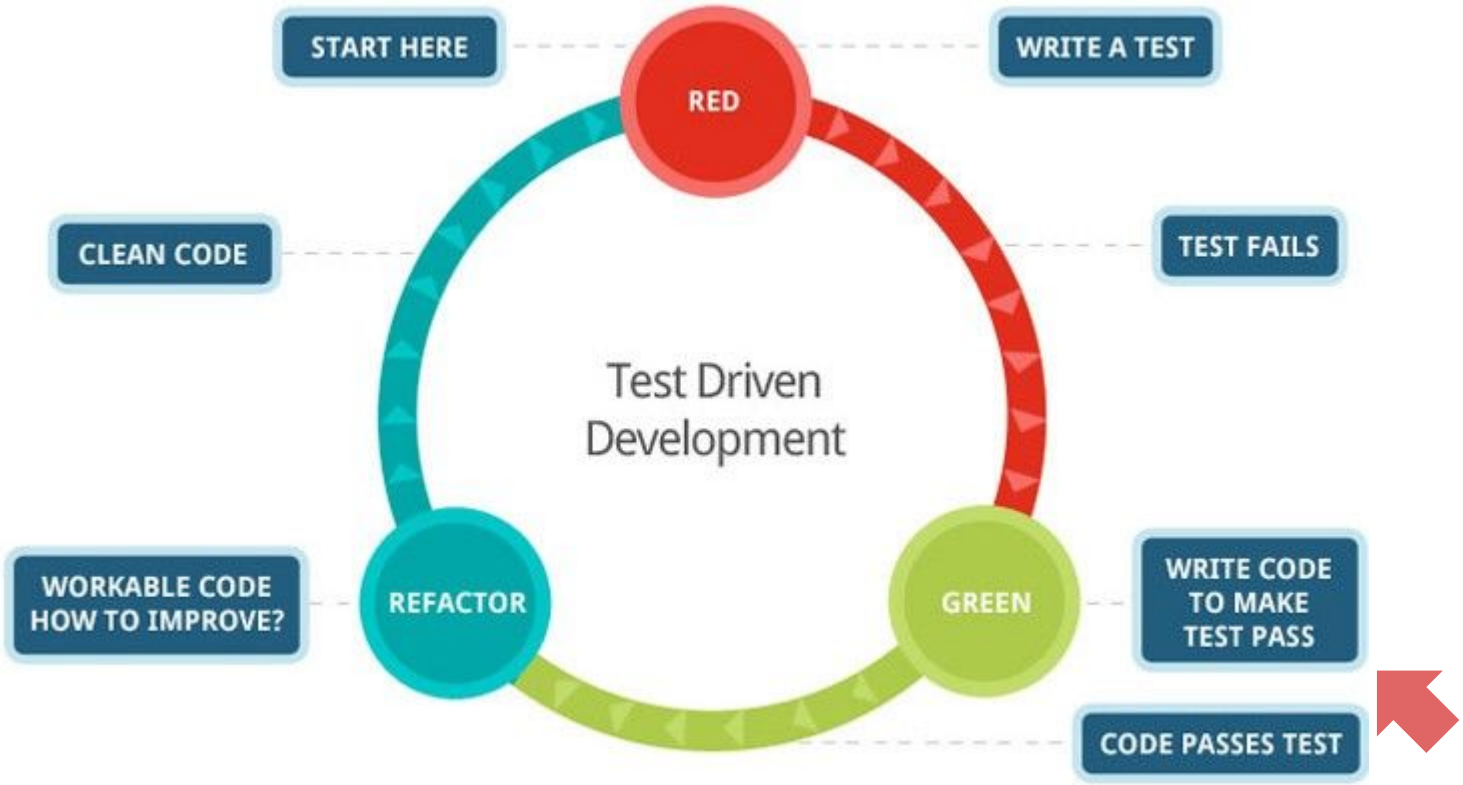
Dlaczego test się wywala?

Bo nie ma czego testować!

u2i

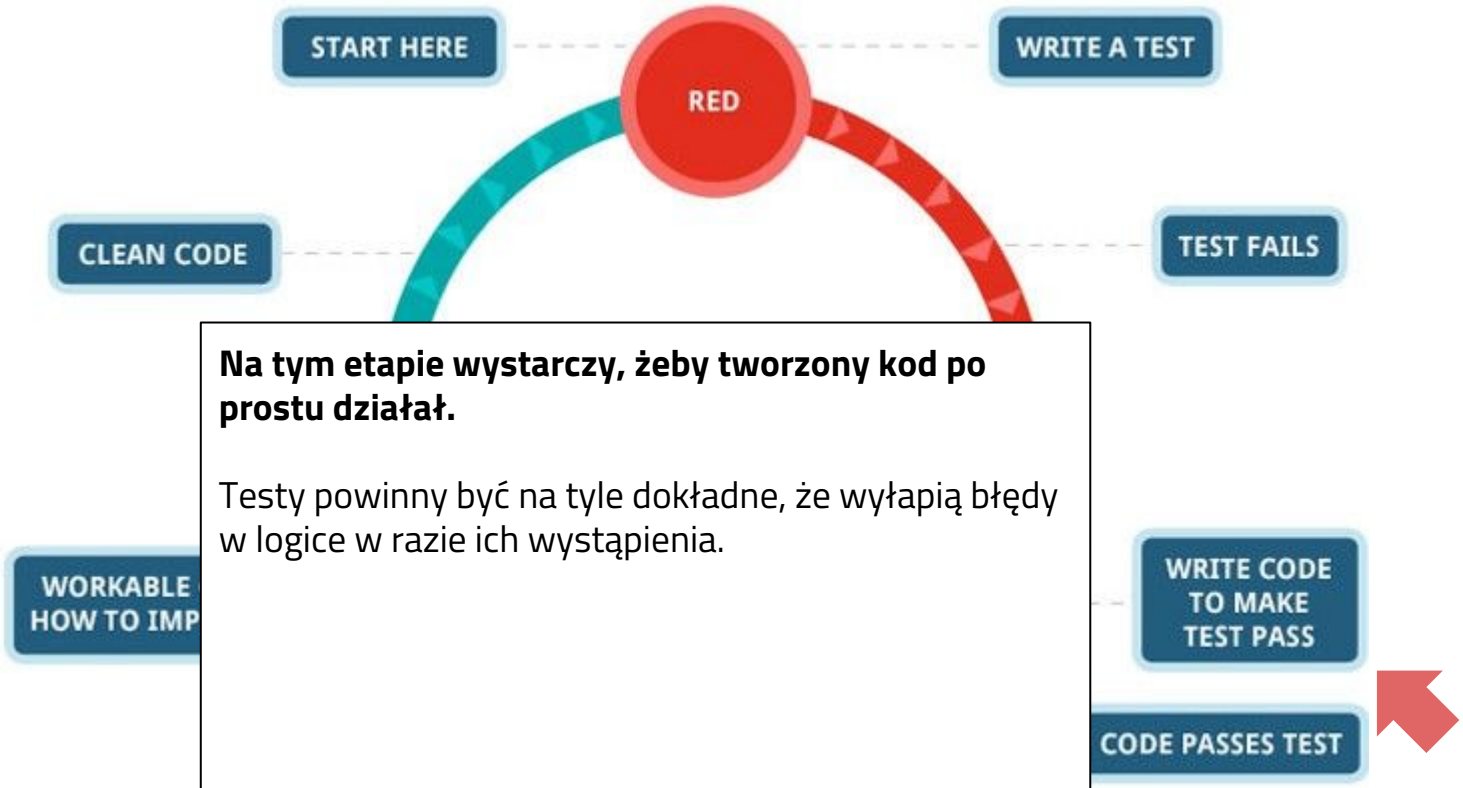
Test to +/- wymaganie wobec kodu

TDD


START HERE
RED
WRITE A TEST
CLEAN CODE
TEST FAILS
Test Driven
Development
WORKABLE CODE
HOW TO IMPROVE?
REFACTOR
GREEN
WRITE CODE
TO MAKE
TEST PASS
CODE PASSES TEST

u2i

TDD

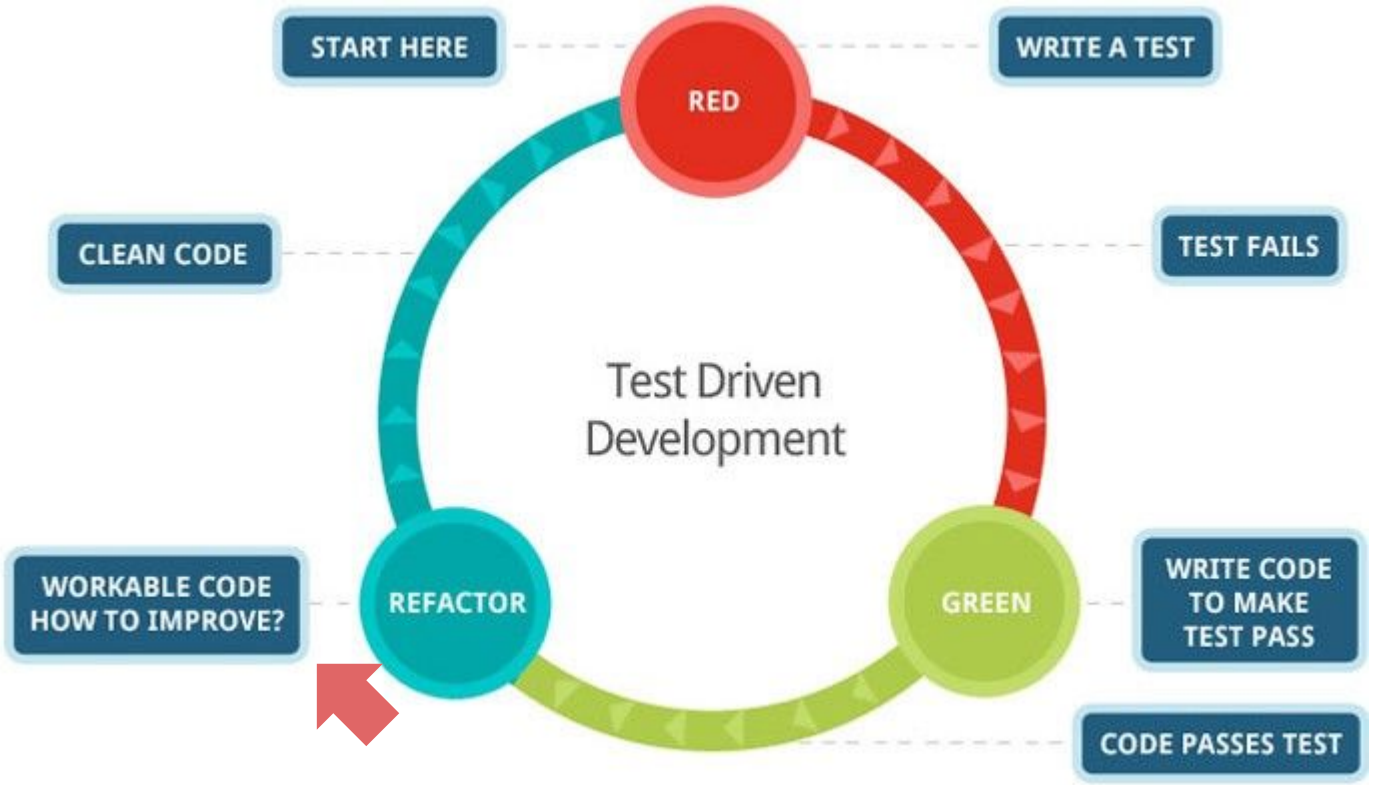


Na tym etapie wystarczy, żeby tworzony kod po prostu działał.

Testy powinny być na tyle dokładne, że wyłapią błędy w logice w razie ich wystąpienia.

u2i

TDD


START HERE
RED
WRITE A TEST
CLEAN CODE
TEST FAILS
Test Driven
Development
WORKABLE CODE
HOW TO IMPROVE?
REFACTOR
GREEN
WRITE CODE
TO MAKE
TEST PASS
CODE PASSES TEST

u2i

TDD

Jeżeli mamy działający kod, to możemy się skupić na jego usprawnianiu.

W razie niepowodzenia możemy wrócić do wcześniejszej działającej iteracji zamiast błądzić jak dziecko we mgle :D

u2i

TDD


START HERE
RED
WRITE A TEST
CLEAN CODE
TEST FAILS
Test Driven
Development
WORKABLE CODE
HOW TO IMPROVE?
REFACTOR
GREEN
WRITE CODE
TO MAKE
TEST PASS
CODE PASSES TEST

u2i

Plusy i minusy

Plusy i minusy

+ Szybkie wychwytywanie błędów

Plusy i minusy

+ Szybkie wychwytywanie błędów
+ Niskie koszty poprawiania błędów

Plusy i minusy

+ Szybkie wychwytywanie błędów
+ Niskie koszty poprawiania błędów
+ Bardziej przemyślany kod

Plusy i minusy

+ Szybkie wychwytywanie błędów
+ Niskie koszty poprawiania błędów
+ Bardziej przemyślany kod
+ Możliwość przetestowania funkcjonalności bez uruchamiania całej aplikacji

Plusy i minusy

+ Szybkie wychwytywanie błędów
+ Niskie koszty poprawiania błędów
+ Bardziej przemyślany kod
+ Możliwość przetestowania funkcjonalności bez uruchamiania całej aplikacji
+ Testy jako dokumentacja

Plusy i minusy

+ Szybkie wychwytywanie błędów
+ Niskie koszty poprawiania błędów
+ Bardziej przemyślany kod
+ Możliwość przetestowania funkcjonalności bez uruchamiania całej aplikacji
+ Testy jako dokumentacja
– Więcej czasu potrzebnego do stworzenia funkcjonalności

Plusy i minusy

+ Szybkie wychwytywanie błędów
+ Niskie koszty poprawiania błędów
+ Bardziej przemyślany kod
+ Możliwość przetestowania funkcjonalności bez uruchamiania całej aplikacji
+ Testy jako dokumentacja
– Więcej czasu potrzebnego do stworzenia funkcjonalności
– Konieczność utrzymywania testów

Frameworki xUnit

Smalltalk	SUnit

Frameworki xUnit

Smalltalk	SUnit
Java	
Python	
C++	
Ruby	

u2i

Frameworki xUnit

Smalltalk	SUnit
Java	JUnit
Python	
C++	
Ruby	

Frameworki xUnit

Smalltalk	SUnit
Java	JUnit
Python	PyUnit
C++	
Ruby	

Frameworki xUnit

Smalltalk	SUnit
Java	JUnit
Python	PyUnit
C++	CppUnit
Ruby	

u2i

Frameworki xUnit

Smalltalk	SUnit
Java	JUnit
Python	PyUnit
C++	CppUnit
Ruby	Minitest :D

Słowo mocy - assert

assert **true**

Test przechodzi

Słowo mocy - assert

assert **false**

Test nie przechodzi

Asercje

- assert @traits.any?, "empty subjects"
- assert_empty @labels
- assert_equal 2, @traits.size
- assert_in_delta @traits.size, 1, 1
- assert_in_epsilon @traits.size, 1, 1
- assert_includes @traits, "skinny jeans"
- assert_instance_of Hipster, @hipster
- assert_kind_of Enumerable, @labels
- assert_match @traits.first, /silly/
- assert_nil @labels.first
- assert_operator @labels.size, :== , 0
- assert_output("Size: 2") { print "Size: #{@traits.size}"}
- assert_raises(NoMethodError) { @traits.foo }
- assert_respond_to @traits, :count
- assert_same @traits, @traits, "It's the same object silly"
- assert_send [@traits, :values_at, 0]
- assert_silent { "no stdout or stderr" }
- assert_throws(Exception, 'is empty') {throw Exception if @traits.any?}

http://mattsears.com/articles/2011/12/10/minitest-quick-reference/

u2i

Przykładowy test

```
def test_sample
  assert my_name() == 'Damian'
end
```

Do dzieła!

u2i

Ankieta

https://www.surveymonkey.com/r/X8M6FVB

u2i

Dziękuję za uwagę!

damian.kampik@u2i.com

u2i