Unit testen

AVD Informatica

**Jaar 1 – 2020/2021

**Onderwerp** Unit testen applicatie Vita E-Bikes

**Door** Dennis van de Meerendonk;

Christian van der Klooster;

Xanh Nguyen;

Peter Moorthamer.

**Klas** 1B – Groep 1

Inhoudsopgave

[1 Inleiding 3](#_Toc74685165)

[2 Unit test User class 4](#_Toc74685166)

[2.1 Unit test User -> constructor 6](#_Toc74685167)

[2.2 Unit test User -> login 6](#_Toc74685168)

[2.3 Unit test User -> register 7](#_Toc74685169)

[2.4 Unit test User -> setAll 8](#_Toc74685170)

[2.5 Unit test User -> findUserByEmail 8](#_Toc74685171)

[3 Unit test Contact class 9](#_Toc74685172)

[3.1 Unit test Contact -> constructor 9](#_Toc74685173)

[4 Unit test Option class 10](#_Toc74685174)

[4.1 Unit test Option -> constructor 10](#_Toc74685175)

[5 Unit test Order class 11](#_Toc74685176)

[5.1 Unit test Order -> constructor 11](#_Toc74685177)

[6 Unit test Product class 12](#_Toc74685178)

[6.1 Unit test Product -> constructor 12](#_Toc74685179)

[7 Unit test Search class 13](#_Toc74685180)

[7.1 Unit test Search -> constructor 13](#_Toc74685181)

# Inleiding

In dit document een beschrijving en resultaten van het unittesten van de Vita E-Bikes applicatie.

We gebruiken voor het unittesten PHPUnit versie 9.5.5 van Sebastian Bergmann.

# 

# Unit test User class

De User class bestaat uit de volgende functies:

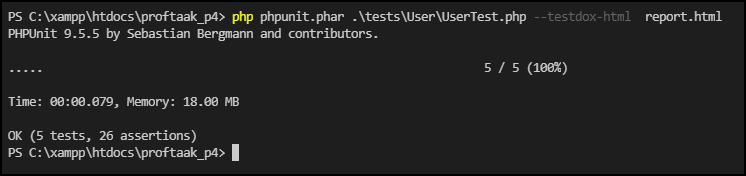
1. De constructor;
2. setAll;
3. register;
4. login;
5. findUserByEmail.

Voor deze klasse hebben vijf unit testen gemaakt, één voor ieder functie. Er is een unittest bestand userTest.php aangemaakt, waarin de 5 testen zijn verwerkt.

Wanneer we de userTest.php unit test uitvoeren met het commando

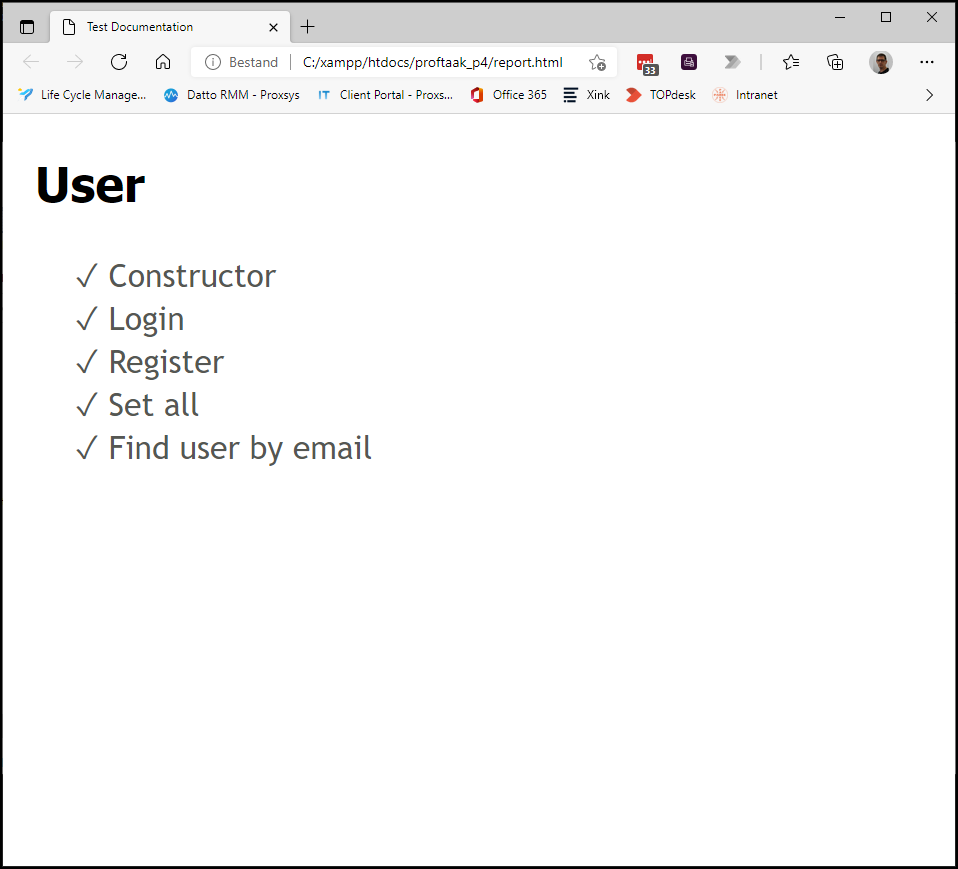
**php phpunit.phar UserTest.php –testdox-html report.html**

dan zien we in onderstaande figuur dat er 5 test zijn uitgevoerd met 26 assertions en dat alle tests zijn geslaagd (5/5 100%).



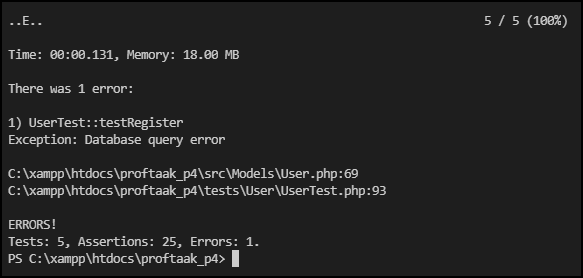
Figuur 1. Uitvoeren unittest UserTest.php

Door de optie **–testdox-html report.html** mee te geven, wordt er een report.html document gegenereerd waar je, heel summier, de resultaten in een webbrowser kunt bekijken.



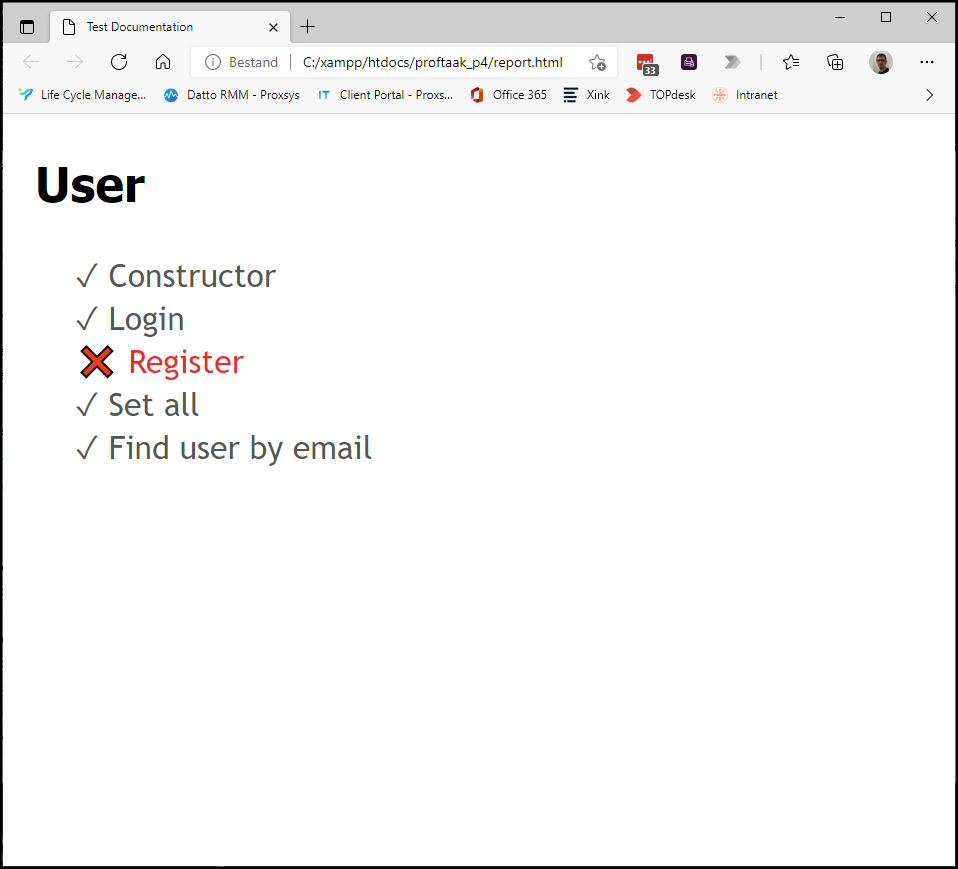
Figuur 2. Testresultaten in webbrowser

Wanneer een test niet slaagt, dan wordt aangegeven wat er fout is gegaan, en op welke regel in het test bestand dit is opgetreden.



Figuur 3. Melding wanneer er een fout uit de unittest komt

Wanneer je een er ook een web rapport van laat maken, dan zie je alleen welke test er is mislukt, zonder een verdere reden.



Figuur 4. Testresultaten in webbrowser met mislukte test

De basis van het unittest bestand ziet er als volgt uit

<?php

require\_once "vendor/autoload.php";

use PHPUnit\Framework\TestCase;

class UserTest extends TestCase

{

    public function testConstructor()

    {

    }

public function testLogin()

    {

    }

public function testRegister()

    {

    }

public function testSetAll()

    {

    }

public function testFindUserByEmail()

    {

    }

}

Figuur 5. Basis unittest bestand

## Unit test User -> constructor

De unittest testConstructor maakt een nieuw user object aan en worden email en wachtwoord in het object gezet.

Er worden *assertEquals* en *assertNotEquals* testen uitgevoerd op het object.

    public function testConstructor()

    {

        $user = new User();

        $user->set('email', 'unittest@test.com');

        $user->set('password', 'NepWachtwoord1');

        $this->assertEquals($user->get('email'), 'unittest@test.com');

        $this->assertEquals($user->get('password'), 'NepWachtwoord1');

        $this->assertNotEquals($user->get('email'), 'fake@fake.nl');

        $this->assertNotEquals($user->get('password'), 'W@chtW00rd');

    }

Figuur 6. unittest User constructor

## Unit test User -> login

De unittest testLogin maakt een nieuw user object aan en worden email en wachtwoord in het object gezet. De login functie wordt getest. De gebruiker moet wel in de database staan!

Er worden *assertEquals* testen uitgevoerd op het object, door data van het object op te vragen.

    public function testLogin()

    {

        $user = new User();

        $user->set('email', 'unittest@test.com');

        $user->set('password', 'NepWachtwoord1');

        $row = $user->login();

        $this->assertEquals($row->id, 11);

        $this->assertEquals($row->email, 'unittest@test.com');

        $this->assertEquals($row->first\_name, 'Unit');

        $this->assertEquals($row->last\_name, 'Test');

        $this->assertEquals($row->street, 'Teststraat');

        $this->assertEquals($row->house\_number, '99');

        $this->assertEquals($row->postal\_code, '1234AA');

        $this->assertEquals($row->city, 'Testdorp');

        $this->assertEquals($row->phone, '+31 111 111111');

        $this->assertEquals($row->active, '1');

        $this->assertEquals($row->function, 'customer');

    }

Figuur 7. unittest User login

## Unit test User -> register

De unittest testRegister maakt een nieuw user object aan. In het user object worden de diverse attributen toegevoegd.

De register functie krijgt na het toevoegen van de user aan de database, het id terug. Dat wordt dan ook getest.

Er worden *assertEquals* test uitgevoerd op het object, door het user object toe te voegen aan de database.

Om deze test te laten slagen moet je eerst weten wat het id is wat zal worden ingevoegd. Voer daarvoor onderstaande query uit om het nieuwe autoincrement id te vinden van de users tabel:

SELECT auto\_increment AS NEXT\_ID  
FROM `information\_schema`.`tables`  
WHERE table\_name = "users" AND table\_schema = "vitaebikes"

Uit de query komt een getal. dit is het volgende id wat wordt gebruikt bij het aanmaken van een nieuwe gebruiker. Vul dit getal in als resultaat voor de assertEquals methode. Voer de test uit. Deze moet OK als resultaat geven.

Verwijder daarna de aangemaakte test gebruiker weer uit de database, zodat je onderstaande gebruiker bij een volgende test weer kunt gebruiken.

    public function testRegister()

    {

        $user = new User();

        $user->set('email','unittest2@test.com');

        $user->set('password','NepWachtwoord2');

        $user->set('firstname','Unit 2');

        $user->set('lastname','Test');

        $user->set('street','Testerlaan');

        $user->set('houseNumber','125');

        $user->set('postalCode','9999ZZ');

        $user->set('city','Testerdam');

        $user->set('telephone','+31 222 222222');

        $user->set('active','1');  
        $user->set('function','admin');

        $newUser = $user->register();

        $this->assertEquals($newUser, 21);

    }

Figuur 8. Unittest User registreer

## Unit test User -> setAll

De unittest testSetAll maakt een nieuw user object aan. In het user object worden de alle attributen (op attribuut active en function) toegeovegd aan het object.

Er worden *assertEquals* en *assertNotEquals* testen uitgevoerd op het object, door get op user object uit te voeren.

    public function testSetAll(){

        $user = new User();

        $user->setAll('unittest2@test.com','Unit 2','Test','Testerlaan','125',

                      '9999ZZ', 'Testerdam','+31 222 222222','NepWachtwoord2');

        $this->assertEquals($user->get('email'),'unittest2@test.com');

        $this->assertNotEquals($user->get('email'),'fakeemail@dummy.com');

        $this->assertEquals($user->get('street'),'Testerlaan');

        $this->assertNotEquals($user->get('street'),'Foutstraat');

        $this->assertEquals($user->get('postalCode'),'9999ZZ');

        $this->assertNotEquals($user->get('postalCode'),'1111AA');

        $this->assertEquals($user->get('telephone'),'+31 222 222222');

        $this->assertNotEquals($user->get('telephone'),'+31 111 111111');

    }

Figuur 9. unittest User setAll

## Unit test User -> findUserByEmail

De unittest testFindUserByEmail maakt een nieuw user object aan. Daarna wordt er met *assertTrue* en *assertFalse* de findUserByEmail functie getest of een email adres voorkomt in de database.

public function testFindUserByEmail(){

        $user = new User;

        $email = 'unittest@test.com';

        $this->assertTrue($user->findUserByEmail($email));

        $this->assertFalse($user->findUserByEmail('dummy@test.com'));

    }

Figuur 10. unittest User findUserByEmail

# Unit test Contact class

Van de Contact class testen we de constructor.

## Unit test Contact -> constructor

De unittest testConstructor maakt een nieuw contact object aan en worden firstName, lastName, email en tekst in het object gezet.

Er worden *assertEquals* testen uitgevoerd op het object.

public function testConstructor()

    {

        $contact = new Contact();

        $contact->set('firstName', 'Kees');

        $contact->set('lastName', 'de Tester');

        $contact->set('email', 'kdetester@dummy.com');

        $contact->set('text', 'Test contact formulier');

        $this->assertEquals($contact->get('firstName'), 'Kees');

        $this->assertEquals($contact->get('lastName'), 'de Tester');

        $this->assertEquals($contact->get('email'), 'kdetester@dummy.com');

        $this->assertEquals($contact->get('text'), 'Test contact formulier');

        $this->assertEquals($contact->get('firstName'), 'Kees');

        $this->assertEquals($contact->get('firstName'), 'Kees');

    }

Figuur . unittest Contact constructor

# Unit test Option class

Van de Contact class testen we de constructor.

## Unit test Option -> constructor

De unittest testConstructor maakt een nieuw option object aan en worden id, name, price, category en image in het object gezet.

Er worden *assertEquals* en *assertNotEquals* testen uitgevoerd op het object.

/\*

     \* Test constructor, stel id, name, price, category en image in op het object,

     \* en controlleer met assertEquals en assertNotEquals.

     \*/

    public function testConstructor()

    {

        $option = new Option();

        $option->set('id', '1');

        $option->set('name', 'fietstas');

        $option->set('price', '35.00');

        $option->set('category', 'Tassen');

        $option->set('image', 'fietstas.jpg');

        $this->assertEquals($option->get('id'), '1');

        $this->assertEquals($option->get('price'), '35.00');

        $this->assertEquals($option->get('image'), 'fietstas.jpg');

        $this->assertNotEquals($option->get('name'), 'xxxxx');

        $this->assertNotEquals($option->get('price'), '0.00');

        $this->assertNotEquals($option->get('category'), 'qqqqqq');

    }

Figuur . unittest Option constructor

# Unit test Order class

Van de Contact class testen we de constructor.

## Unit test Order -> constructor

De unittest testConstructor maakt een nieuw user object aan en worden id, date, delivery\_date, payment\_option en status in het object gezet.

Er worden *assertEquals* en *assertNotEquals* testen uitgevoerd op het object.

/\*

     \* Test constructor, stel id, date, delivery\_date, payment\_option en status   
 in op het object, en controleer met assertEquals en assertNotEquals.

     \*/

    public function testConstructor()

    {

        $order = new Order();

        $order->set('id', '1');

        $order->set('date', '2021-06-17');

        $order->set('delivery\_date', '2021-09-21');

        $order->set('payment\_option', 'iDeal');

        $order->set('status', 'closed');

        $this->assertEquals($order->get('id'), '1');

        $this->assertEquals($order->get('date'), '2021-06-17');

        $this->assertNotEquals($order->get('payment\_option'), 'Visa');

        $this->assertNotEquals($order->get('status'), 'open');

    }

Figuur . unittest Order class

# Unit test Product class

Van de Contact class testen we de constructor.

## Unit test Product -> constructor

De unittest testConstructor maakt een nieuw user object aan en worden id, name, description, price en image in het object gezet.

Er worden *assertEquals* en *assertNotEquals* testen uitgevoerd op het object.

/\*

     \* Test constructor, stel id, name, description, price en image in op het object,

     \* en controleer met assertEquals en assertNotEquals.

     \*/

    public function testConstructor()

    {

        $product = new Product();

        $product->set('id', '99');

        $product->set('name', 'Vita-XL');

        $product->set('description', 'Luxe uitvoering');

        $product->set('price', '2100.00');

        $product->set('image', 'vitaxl.jpg');

        $this->assertEquals($product->get('id'), '99');

        $this->assertEquals($product->get('name'), 'Vita-XL');

        $this->assertNotEquals($product->get('price'), '0.00');

        $this->assertNotEquals($product->get('image'), 'bike.jpg');

    }

Figuur . unittest Product constructor

# Unit test Search class

Van de Contact class testen we de constructor.

## Unit test Search -> constructor

De unittest testConstructor maakt een nieuw user object aan en worden email en wachtwoord in het object gezet.

Er worden *assertEquals* en *assertNotEquals* testen uitgevoerd op het object.

/\*

     \* Test constructor, stel id, term en date in op het object,

     \* en controleer met assertEquals en assertNotEquals.

     \*/

    public function testConstructor()

    {

        $search = new Search();

        $search->set('id', '25');

        $search->set('term', 'motor');

        $search->set('date', '2021-06-15');

        $this->assertEquals($search->get('id'), '25');

        $this->assertEquals($search->get('term'), 'motor');

        $this->assertNotEquals($search->get('term'), 'fiets');

        $this->assertNotEquals($search->get('date'), '1999-01-01');

    }

Figuur . unittest Search class