Feature test

**Naam:** Ravda Tüküç

**Project:** Snap & cook

**Intro:**

Feature tests zijn essentieel om ervoor te zorgen dat onderdelen van een applicatie correct werken. In Laravel worden feature tests geschreven met PHPUnit en Laravel’s testhulpmiddelen. Dit document geeft een overzicht van een feature test-klasse voor een Laravel-applicatie.

**Code:**

<?php  
  
namespace Tests\Feature;  
  
// use Illuminate\Foundation\Testing\RefreshDatabase;  
use Illuminate\Support\Facades\Storage;  
use Tests\TestCase;  
use App\AIChef\Chef;  
use App\Models\Recipe;  
use App\Jobs\ProcessImage;  
use Illuminate\Http\UploadedFile;  
use Illuminate\Support\Facades\Queue;  
use Illuminate\Support\Facades\Session;  
  
class ExampleTest extends TestCase  
{  
 */\*\*  
 \* A basic test example.  
 \*/* // test the routes  
  
 public function test\_home\_route()  
 {  
 $response = $this->get('/');  
 $response->assertStatus(200);  
 }  
  
 public function test\_upload\_route()  
 {  
 $response = $this->get('/upload');  
 $response->assertStatus(405);  
 }  
  
 public function test\_wait\_route()  
 {  
 $response = $this->get('/wait');  
 $response->assertStatus(200);  
 }  
  
 public function test\_status\_route()  
 {  
 $response = $this->get('/status');  
 $response->assertStatus(500);  
 }  
  
 public function test\_job\_status\_route()  
 {  
 $response = $this->get('/job-status');  
 $response->assertStatus(500);  
 }  
  
 public function test\_like\_route()  
 {  
 $response = $this->get('/like');  
 $response->assertStatus(404);  
 }  
  
 public function test\_dislike\_route()  
 {  
 $response = $this->get('/dislike');  
 $response->assertStatus(404);  
 }  
  
 public function test\_download\_route()  
 {  
 $response = $this->get('/download');  
 $response->assertStatus(404);  
 }  
  
 public function test\_about\_route()  
 {  
 $response = $this->get('/about');  
 $response->assertStatus(200);  
 }  
  
 public function test\_contact\_route()  
 {  
 $response = $this->get('/contact');  
 $response->assertStatus(200);  
 }  
  
 public function test\_recipes\_route()  
 {  
 $response = $this->get('/recipes');  
 $response->assertStatus(500);  
 }  
  
 public function test\_scanner\_route()  
 {  
 $response = $this->get('/scanner');  
 $response->assertStatus(200);  
 }  
  
 // test the controllers  
  
 public function test\_like\_controller()  
 {  
 $response = $this->get('/like/1');  
 $response->assertStatus(405);  
 }  
  
 public function test\_dislike\_controller()  
 {  
 $response = $this->get('/dislike/1');  
 $response->assertStatus(405);  
 }  
  
 public function test\_download\_controller()  
 {  
 $response = $this->get('/download/1');  
 $response->assertStatus(302);  
 }  
  
 public function test\_job\_status\_controller()  
 {  
 $response = $this->get('/job-status');  
 $response->assertStatus(500);  
 }  
  
 public function testGetRecipe(): void  
 {  
 // Mock the Storage facade  
 Storage::*fake*('local');  
  
 // Create a fake image file  
 Storage::put('image.jpg', 'Fake image content');  
  
 // Create a new Chef object  
 $chef = new Chef();  
  
 // Call the GetRecipe method  
 $result = $chef->GetRecipe('image.jpg');  
  
 // Assert the result is as expected  
 $this->assertIsString($result);  
 }  
  
 public function testUpload(): void  
 {  
 // Mock the Storage facade  
 Storage::*fake*('local');  
  
 // Mock the Queue facade  
 Queue::*fake*();  
  
 // Create a fake image file  
 $file = UploadedFile::*fake*()->image('image.jpg');  
  
 // Call the upload method  
 $response = $this->post('/upload', [  
 'image' => $file,  
 ]);  
  
 // Assert the file was stored...  
 Storage::disk('local')->assertExists($file->hashName());  
  
 // Assert a job was pushed to the queue...  
 Queue::assertPushed(ProcessImage::class);  
  
 // Assert the response  
 $response->assertRedirect('wait');  
 }  
  
 public function testDownload(): void  
 {  
 // Create a new recipe  
 $recipe = new Recipe();  
 $recipe->path = 'image.jpg';  
 $recipe->save();  
  
 // Call the download method  
 $response = $this->get("/download/$recipe->id");  
  
 // Assert the response  
 $response->assertStatus(200);  
 }  
  
}

**Voorbeeld van de Testklasse**

De `ExampleTest`-klasse bevat tests voor verschillende routes en controllers in een Laravel-toepassing. Elke testmethode controleert een specifieke route of functionaliteit om de juiste werking van de applicatie te waarborgen.

**Verklaring:**

**Route tests**

1. \*\*test\_home\_route\*\*: Controleert of de homepagina correct wordt geladen (HTTP-status 200).

2. \*\*test\_upload\_route\*\*: Verwacht een HTTP-status van 405 voor de `/upload` route.

3. \*\*test\_wait\_route\*\*: Controleert of de `/wait` route correct wordt geladen (HTTP-status 200).

4. \*\*test\_status\_route\*\*: Verwacht een HTTP-status van 500 voor de `/status` route.

5. \*\*test\_job\_status\_route\*\*: Verwacht een HTTP-status van 500 voor de `/job-status` route.

6. \*\*test\_like\_route\*\*: Verwacht een HTTP-status van 404 voor de `/like` route.

7. \*\*test\_dislike\_route\*\*: Verwacht een HTTP-status van 404 voor de `/dislike` route.

8. \*\*test\_download\_route\*\*: Verwacht een HTTP-status van 404 voor de `/download` route.

9. \*\*test\_about\_route\*\*: Controleert of de `/about` route correct wordt geladen (HTTP-status 200).

10. \*\*test\_contact\_route\*\*: Controleert of de `/contact` route correct wordt geladen (HTTP-status 200).

11. \*\*test\_recipes\_route\*\*: Verwacht een HTTP-status van 500 voor de `/recipes` route.

12. \*\*test\_scanner\_route\*\*: Controleert of de `/scanner` route correct wordt geladen (HTTP-status 200).

**Controller tests**

1. \*\*test\_like\_controller\*\*: Verwacht een HTTP-status van 405 voor de `like` controller.

2. \*\*test\_dislike\_controller\*\*: Verwacht een HTTP-status van 405 voor de `dislike` controller.

3. \*\*test\_download\_controller\*\*: Verwacht een HTTP-status van 302 voor de `download` controller.

4. \*\*test\_job\_status\_controller\*\*: Verwacht een HTTP-status van 500 voor de `job-status` controller.

**Functionele tests**

1. \*\*testGetRecipe\*\*: Mockt Storage, creëert een nep afbeelding en test de `GetRecipe` methode van het `Chef` model.

2. \*\*testUpload\*\*: Mockt Storage en Queue, uploadt een nep afbeelding en controleert opslag en job aan de wachtrij. Verwacht een redirect naar `wait`.

3. \*\*testDownload\*\*: Creëert een nieuw recept en test de downloadfunctionaliteit (HTTP-status 200).

**Conclusie**

Deze tests verifiëren de correcte werking van routes, controllers, model eigenschappen, bestandsopslag en downloadfunctionaliteit in een Laravel-toepassing.