Модульное тестирование

**EditProfile.php**

<?php

namespace Tests\Feature;

use App\Models\User;

use Illuminate\Foundation\Testing\RefreshDatabase;

use Illuminate\Foundation\Testing\WithFaker;

use Illuminate\Support\Facades\Auth;

use Laravel\Passport\Passport;

use Mockery\Generator\StringManipulation\Pass\Pass;

use Tests\TestCase;

class EditProfileTest extends TestCase

{

/\*\*

\* Тестируем удачное редактирование профиля

\*/

public function testSuccess()

{

//Авторизация пользователя для laravel-passport

$user = User::query()->first();

Passport::actingAs($user);

//Данные для авторизации

$data = [

'email' => $user->email,

'address' => 'Test address'

];

$response = $this->postJson('/api/editMe', $data);

//Проверка результата(200 статус и пользователь с обновленными данными)

$response->assertSuccessful();

$this->assertArrayHasKey( 'newUser', $response);

$this->assertEquals($response['newUser']['address'], 'Test address');

}

/\*\*

\* Тестируем пустой ввод

\*/

public function emptyTest()

{

//Авторизация пользователя для laravel-passport

$user = User::query()->first();

Passport::actingAs($user);

$response = $this->postJson('/api/editMe', []);

//Проверка результата(Ошибка обязательных полей)

$response->assertJsonValidationErrors(['email']);

$this->assertEquals($response['errors']['email'][0], "The email field is required.");

}

/\*\*

\* Тестируем с не валидными данными

\*/

public function testInvalid()

{

//Авторизация пользователя для laravel-passport

$user = User::query()->first();

Passport::actingAs($user);

$data = [

'email' => 'bademail'

];

$response = $this->postJson('/api/editMe', $data);

//Проверяем ошибку валидации(почта неверного формата)

$response->assertJsonValidationErrors(['email']);

$this->assertEquals($response['errors']['email'][0], "The email must be a valid email address.");

}

/\*\*

\* Тестируем неавторизированного пользователя

\*/

public function testUnauthorized()

{

$response = $this->postJson('/api/editMe', [

'email' => 'test@gmail.com'

]);

//Проверяем 403 статус

$response->assertUnauthorized();

}

}

**LoginTest.php**

<?php

namespace Tests\Feature;

use App\Models\User;

use Illuminate\Foundation\Testing\RefreshDatabase;

use Illuminate\Foundation\Testing\WithFaker;

use Tests\TestCase;

class LoginTest extends TestCase

{

private $user;

public function setUp(): void

{

parent::setUp(); // TODO: Change the autogenerated stub

$this->user = User::query()->first();

}

/\*\*

\* Тестируем удачный вход

\*

\* @return void

\*/

public function testSuccess()

{

$response = $this->postJson('/api/login', [

'email' => $this->user->email,

'password' => '20012007'

]);

//Проверяем 200 статус

$response->assertSuccessful();

}

/\*\*

\* Тестируем не валидную почту

\*

\* @return void

\*/

public function testInvalid()

{

$response = $this->postJson('/api/login', [

'email' => 'bademail',

'password' => '20012007'

]);

//Проверяем ошибку валидации почты

$response->assertJsonValidationErrors(['email']);

$this->assertEquals($response['errors']['email'][0], "The email must be a valid email address.");

}

/\*\*

\* Тестируем вход с пустой почтой

\*

\* @return void

\*/

public function testEmptyEmail()

{

$response = $this->postJson('/api/login', [

'email' => null,

'password' => '20012007'

]);

//Проверяем ошибку, что почта это обязательное поле

$response->assertJsonValidationErrors(['email']);

$this->assertEquals($response['errors']['email'][0], "The email field is required.");

}

/\*\*

\* Тестируем с пустым паролем

\*

\* @return void

\*/

public function testEmptyPassword()

{

$response = $this->postJson('/api/login', [

'email' => $this->user->email,

'password' => null

]);

//Проверяем ошибку валидации(пароль - это обязательное поле)

$response->assertJsonValidationErrors(['password']);

$this->assertEquals($response['errors']['password'][0], "The password field is required.");

}

/\*\*

\* Тестируем слишком короткий пароль

\*

\* @return void

\*/

public function testShortPassword()

{

$response = $this->postJson('/api/login', [

'email' => $this->user->email,

'password' => '11'

]);

//Проверяем ошибку валидации данных(Пароль слишком короткий)

$response->assertJsonValidationErrors(['password']);

$this->assertEquals($response['errors']['password'][0], "The password must be at least 8 characters.");

}

/\*\*

\* Тестируем слишком длинный пароль

\*

\* @return void

\*/

public function testLongPassword()

{

$response = $this->postJson('/api/login', [

'email' => $this->user->email,

'password' => '111111111111111111111111111111111'

]);

//Проверяем ошибку валидации данных(Пароль слишком длинный)

$response->assertJsonValidationErrors(['password']);

$this->assertEquals($response['errors']['password'][0], "The password may not be greater than 32 characters.");

}

/\*\*

\* Тестируем ввод данных для несуществующего пользователя

\*

\* @return void

\*/

public function testUnExistUser()

{

$response = $this->postJson('/api/login', [

'email' => 'test@mail.ru',

'password' => '200120072017'

]);

//Проверяем ошибку, что такая почта не существует

$response->assertJsonValidationErrors(['email']);

$this->assertEquals($response['errors']['email'][0], "The selected email is invalid.");

}

}

**ReportTest.php**

<?php

namespace Tests\Feature;

use App\Models\User;

use Illuminate\Foundation\Testing\RefreshDatabase;

use Illuminate\Foundation\Testing\WithFaker;

use Laravel\Passport\Passport;

use Tests\TestCase;

class ReportTest extends TestCase

{

public function testUser()

{

//Авторизируем пользователя

$user = User::query()->where('role', 50)->first();

Passport::actingAs($user);

$response = $this->get('/api/report');

//Проверяем статус 403

$response->assertForbidden();

}

public function testMoreRole()

{

//Авторизируем пользователя

$user = User::query()->where('role', '<=', 30)->first();

Passport::actingAs($user);

$response = $this->get('/api/report');

//Проверяем статус 200

$response->assertSuccessful();

}

}

**ViewProfile.php**

<?php

namespace Tests\Feature;

use App\Models\User;

use Illuminate\Foundation\Testing\RefreshDatabase;

use Illuminate\Foundation\Testing\WithFaker;

use Laravel\Passport\Passport;

use Tests\TestCase;

class ViewProfileTest extends TestCase

{

/\*\*

\* Тестируем получения информации неавторизированным пользователем

\*/

public function testUnauthorized()

{

$response = $this->getJson ('/api/users/1');

//Проверяем статус 401

$response->assertUnauthorized();

}

/\*\*

\* Test get info with role User

\*/

public function testSmallRole()

{

/\*\*

\* @var User $user

\*/

//Авторизируем пользователя

$user = User::query()->where('role', '50')->first()->getModel();

Passport::actingAs($user);

$response = $this->getJson('/api/users/1');

//Проверяем статус 403

$response->assertForbidden();

}

/\*\*

\* Тестируем получения отчета пользователем с ролью больше или равной Просматриватель

\*/

public function testSuccess()

{

/\*\*

\* @var User $user

\*/

//Авторизируем пользователя

$user = User::query()->where('role', '<=', 30)->first();

Passport::actingAs($user);

$response = $this->getJson('/api/users/2');

//Проверяем статус 200

$response->assertSuccessful();

}

}

**Результат**

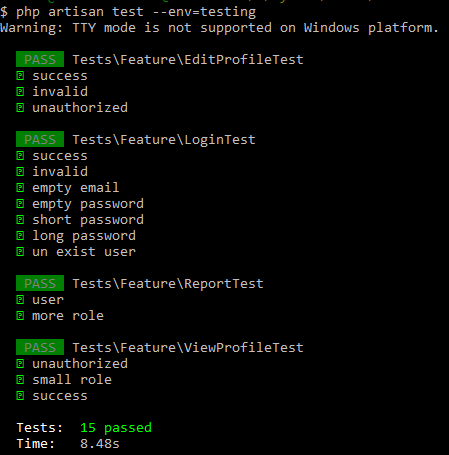


Рисунок 1 – Результат тестирования