Informe Técnico de pruebas Drawing-Academy

27 de septiembre de 2022

Procedimientos en Laravel versión 9.21.5

Presentado por:

Rafael José Arenas Restrepo

Julian David Anturi Duque

Presentado al supervisor:

Carlos Andrés Mora Agudelo

Centro de Diseño e Innovación Tecnológico Industrial

Dosquebradas - Risaralda

Test de módulos del proyecto Drawing academy

En el presente documento se mostrarán de forma gráfica los módulos en los que se realizaron los pertinentes test.

Cada test hace referencia a lo que se pretende demostrar en cuanto a funcionamiento de los módulos en sus funciones.

1. **Modulo de cursos**

1.1. Creación del test unitario

Para crear las funciones necesarias para probar el módulo completo de cursos solo hace falta insertar el código **php artisan make:test CouseTest --unit**

```
PS E:\TGO ADSI 2341240\5to Trimestre\CARLOS MORA\CONSTRUIR -- LUNES-VIERNES\Laravel\Laravel 2341240\Ejercicio1> php artisan make:test CourseTest --unit

  INFO  Test created successfully.
```

1.2. Realizando test de almacenamiento

El test se hizo a través de una función llamada **test_save_new_course**, y **test_save_new_course2** lo que estas funciones pretenden hacer es corroborar que se almacenen los cursos creados en la base de datos con sus respectivas verificaciones.

```php
    public function test_save_new_course(){
        $respuesta = $this->post('/courses', [
            'name' => 'Dibujo Nivel 2',
            'description' => 'Técnicas de dibujo con boligráfo',
            'duration' => 60,
            'image' => 'img.png'
        ]);

        return $respuesta->assertRedirect('/');
    }
```

```php
public function test_save_new_course2(){
    $respuesta = $this->post('/courses', [
        'name' => 'Dibujo Nivel 3',
        'description' => 'Técnicas de dibujo con colores',
        'duration' => 80,
        'image' => 'img2.png'
    ]);

    return $respuesta->assertRedirect('/');
}
```

El comando utilizado para hacer el test de forma individual es: **vendor/bin/phpunit –filter test_save_new_course tests/Unit/CourseTest.php**

```
PS A:\proyectos\htdocs\Academy> vendor/bin/phpunit --filter test_save_new_course tests/Unit/CourseTest.php
PHPUnit 9.5.24 #StandWithUkraine

..                                                                  2 / 2 (100%)

Time: 00:00.173, Memory: 22.00 MB

OK (2 tests, 4 assertions)
```

Efectivamente el test se realizo de una manera exitosa. Se ve por el estado OK y el resalado de color verde.

1.3. Realizando test de duplicación

Estos 3 tests se realizaron bajo la finalidad que los nombres, las descripciones y las imágenes de los cursos no se repitan, de tal manera que no exista el mismo nombre de curso.

En la primera función se validan los nombres de los cursos evitando así una posible duplicación.

```php
public function test_course_duplication(){
    $course1 = Course::make([
        'name' => 'Dibujo Nivel 2',
    ]);

    $course2 = Course::make([
        'name' => 'Dibujo Nivel 3',
    ]);

    $this->assertTrue($course1->name != $course2->name);
}
```

La segunda función esta encargada de la validación de duplicación de un curso con el mismo nombre y la misma descripción.

```php
public function test_course_duplication2(){
    $course1 = Course::make([
        'name' => 'Dibujo Nivel 2',
        'description' => 'Técnicas de dibujo con boligráfo',
    ]);

    $course2 = Course::make([
        'name' => 'Dibujo Nivel 3',
        'description' => 'Técnicas de dibujo con colores',
    ]);

    $this->assertTrue($course1->name != $course2->name && $course1->description != $course2->description);
}
```

Mientras tanto el tercer test valida los puntos anteriores y la imagen, que no sea igual

```php
public function test_course_duplication3(){
    $course1 = Course::make([
        'name' => 'Dibujo Nivel 2',
        'description' => 'Técnicas de dibujo con boligráfo',
        'image' => 'img.png'
    ]);

    $course2 = Course::make([
        'name' => 'Dibujo Nivel 3',
        'description' => 'Técnicas de dibujo con colores',
        'image' => 'img2.png'
    ]);

    $this->assertTrue($course1->name != $course2->name && $course1->description != $course2->description && $course1->image !=
    $course2->image);
}
```

El resultado es el siguiente:

```
PS A:\proyectos\htdocs\Academy> vendor/bin/phpunit --filter test_course_duplication tests/Unit/CourseTest.php
PHPUnit 9.5.24 #StandWithUkraine

...                                                                 3 / 3 (100%)

Time: 00:00.171, Memory: 22.00 MB

OK (3 tests, 3 assertions)
```

1.4. Test de eliminación

Esta serie de test pretenden la comprobación efectiva de la eliminación de registros, para ello las funciones crean un curso, mediante factories, acto seguido comprueba que, si se haya creado y siendo el caso exitoso, lo elimina finalizando la acción con una respuesta booleana.

```php
public function test_delete_course(){
    $course = Course::factory()->count(1)->make();

    $course = Course::first();

    if($course){
        $course->delete();
    }

    $this->assertTrue(true);
}
```

```php
public function test_delete_course2(){
    $course = Course::factory()->count(2)->make();

    $course = Course::first();

    if($course){
        $course->delete();
    }

    $this->assertTrue(true);
}
```

```php
public function test_delete_course3(){
    $course = Course::factory()->count(3)->make();

    $course = Course::first();

    if($course){
        $course->delete();
    }

    $this->assertTrue(true);
}
```

Se realizo la respectiva prueba que se ejecutó satisfactoriamente.

```
PS A:\proyectos\htdocs\Academy> vendor/bin/phpunit --filter test_delete_course tests/Unit/CourseTest.php
PHPUnit 9.5.24 #StandWithUkraine

...                                                                 3 / 3 (100%)

Time: 00:00.709, Memory: 24.00 MB

OK (3 tests, 3 assertions)
PS A:\proyectos\htdocs\Academy>
```

2. Módulo de docentes

2.1. Creación del test unitario

Para crear las funciones necesarias para probar el módulo completo de cursos solo hace falta insertar el código **php artisan make:test TeacherTest –unit**

```
PS E:\TGO ADSI 2341240\5to Trimestre\CARLOS MORA\CONSTRUIR -- LUNES-VIERNES\Laravel\Laravel 2341240\Ejercicio1> php artisan make:test TeacherTest --unit

  INFO  Test created successfully.
```

2.2. Realizando test de almacenamiento

El test se hizo a través de una función llamada **test_save_new_teacher**, y **test_save_new_teacher2**, lo que estas funciones pretenden hacer es corroborar que se almacenen los docentes creados en la base de datos con sus respectivas verificaciones.

```php
class TeacherTest extends TestCase
{
    public function test_save_new_teacher(){
        $respuesta = $this->post('/teachers', [
            'name' => 'Andrea',
            'last_name' => 'Zapata',
            'college_degree' => 'Licenciada en artes plásticas',
            'age' => 42,
            'contract_date' => '2022-09-04',
            'image' => 'img.png',
            'identify_document' => 'doc.pdf',
            'course_id' => 1
        ]);
        return $respuesta->assertRedirect('/');
    }

    public function test_save_new_teacher2(){
        $respuesta = $this->post('/teachers', [
            'name' => 'Daniel',
            'last_name' => 'Zapata',
            'college_degree' => 'Licenciado en dibujo técnico',
            'age' => 41,
            'contract_date' => '2022-09-04',
            'image' => 'img2.png',
            'identify_document' => 'doc2.pdf',
            'course_id' => 2
        ]);

        return $respuesta->assertRedirect('/');
    }
```

El comando utilizado para hacer el test de forma individual es: **vendor/bin/phpunit –filter test_save_new_teacher tests/Unit/TeacherTest.php**

```
PS A:\proyectos\htdocs\Academy> vendor/bin/phpunit --filter test_save_new_teacher tests/Unit/TeacherTest.php
PHPUnit 9.5.24 #StandWithUkraine

..                                                                  2 / 2 (100%)

Time: 00:00.187, Memory: 22.00 MB

OK (2 tests, 4 assertions)
```

Efectivamente el test se realizó de una manera exitosa. Se ve por el estado OK y el resalado de color verde.

2.3. Realizando test de duplicación

Estos 2 tests se realizaron bajo la finalidad que los documentos de identidad y las imágenes de los docentes no se repitan, de tal manera que no exista el mismo documento de identidad en dos docentes, en la primera función se validan las imágenes de los docentes evitando así una posible duplicación.

```php
public function test_teacher_duplication(){
    $teacher1 = Teacher::make([
        'image' => 'img.png',
    ]);

    $teacher2 = Teacher::make([
        'image' => 'img2.png',
    ]);

    $this->assertTrue($teacher1->image != $teacher2->image);
}
```

La segunda función esta encargada de la validación de duplicación de un documento entre docentes.

```php
public function test_teacher_duplication2(){
    $teacher1 = Teacher::make([
        'image' => 'img.png',
        'identify_document' => 'doc.pdf',
    ]);

    $teacher2 = Teacher::make([
        'image' => 'img2.png',
        'identify_document' => 'doc2.pdf',
    ]);

    $this->assertTrue($teacher1->image != $teacher2->image && $teacher1->identify_document != $teacher2->identify_document);
}
```

El resultado es el siguiente:

```
PS A:\proyectos\htdocs\Academy> vendor/bin/phpunit --filter test_teacher_duplication tests/Unit/TeacherTest.php
PHPUnit 9.5.24 #StandWithUkraine

..                                                                  2 / 2 (100%)

Time: 00:00.172, Memory: 22.00 MB

OK (2 tests, 2 assertions)
```

2.4. Test de eliminación

Esta serie de test pretenden la comprobación efectiva de la eliminación de registros, para ello las funciones crean un docente, mediante factories, acto seguido comprueba que, si se haya creado y siendo el caso exitoso, lo elimina finalizando la acción con una respuesta booleana.

```php
public function test_delete_teacher(){
    $teacher = Teacher::factory()->count(1)->make();

    $teacher = Teacher::first();

    if($teacher){
        $teacher->delete();
    }

    $this->assertTrue(true);
}
```

```php
public function test_delete_teacher2(){
    $teacher = Teacher::factory()->count(2)->make();

    $teacher = Teacher::first();

    if($teacher){
        $teacher->delete();
    }

    $this->assertTrue(true);
}
```

```php
public function test_delete_teacher3(){
    $teacher = Teacher::factory()->count(3)->make();

    $teacher = Teacher::first();

    if($teacher){
        $teacher->delete();
    }

    $this->assertTrue(true);
}
```

3. Módulo de Estudiantes

3.1. Creación del test unitario

Para crear las funciones necesarias para probar el módulo completo de cursos solo hace falta insertar el código **php artisan make:test StudentTest –unit**

```
PS E:\TGO ADSI 2341240\5to Trimestre\CARLOS MORA\CONSTRUIR -- LUNES-VIERNES\Laravel\Laravel 2341240\Ejercicio1> php artisan make:test StudentTest --unit

  INFO  Test created successfully.
```

3.2. Realizando test de almacenamiento

El test se hizo a través de una función llamada **test_save_new_student**, y **test_save_new_student2**, lo que estas funciones pretenden hacer es corroborar que se almacenen los Estudiantes creados en la base de datos con sus respectivas verificaciones.

```php
public function test_save_new_student(){
    $respuesta = $this->post('/students', [
        'document_type' => 'CC',
        'document_number' => 94262139,
        'identify_document' => 'doc.pdf',
        'expedition_date' => '2012-03-27',
        'id_exped_muni' => 9,
        'names' => 'Orlando',
        'last_name1' => 'Arenas',
        'last_name2' => 'Restrepo',
        'gender' => 'M',
        'birth_date' => '1994-03-26',
        'id_birth_muni' => 7,
        'stratum' => 3,
        'id_course' => 1
    ]);
```

```php
public function test_save_new_student2(){
    $respuesta = $this->post('/students', [
        'document_type' => 'CC',
        'document_number' => 94262140,
        'identify_document' => 'doc2.pdf',
        'expedition_date' => '2012-03-27',
        'id_exped_muni' => 9,
        'names' => 'Luis',
        'last_name1' => 'Montoya',
        'last_name2' => 'Castro',
        'gender' => 'M',
        'birth_date' => '1994-03-26',
        'id_birth_muni' => 7,
        'stratum' => 3,
        'id_course' => 1
    ]);

    return $respuesta->assertRedirect('/');
}
```

El comando utilizado para hacer el test de forma individual es: **vendor/bin/phpunit –filter test_save_new_student tests/Unit/StudentTest.php**

```
vendor/bin/phpunit --filter test_save_new_student tests/Unit/StudentTest.php
PHPUnit 9.5.24 #StandWithUkraine

..                                                                  2 / 2 (100%)

Time: 00:00.172, Memory: 22.00 MB

OK (2 tests, 4 assertions)
```

Efectivamente el test se realizó de una manera exitosa. Se ve por el estado OK y el resalado de color verde.

3.3. Realizando test de duplicación

Estos 2 tests se realizaron bajo la finalidad que los números de identidad y los documentos de identidad de los estudiantes no se repitan, de tal manera que no exista el mismo documento de identidad en dos estudiantes, en la primera función se validan los números de documentos de los estudiantes evitando así una posible duplicación.

```php
public function test_student_duplication(){
    $student1 = Student::make([
        'document_number' => 94262139,
    ]);

    $student2 = Student::make([
        'document_number' => 94262140,
    ]);

    $this->assertTrue($student1->document_number != $student2->document_number);
}
```

La segunda función esta encargada de la validación de duplicación de un documento entre estudiantes.

```php
public function test_student_duplication2(){
    $student1 = Student::make([
        'document_number' => 94262139,
        'identify_document' => 'doc.pdf',
    ]);

    $student2 = Student::make([
        'document_number' => 94262140,
        'identify_document' => 'doc2.pdf',
    ]);

    $this->assertTrue($student1->document_number != $student2->document_number && $student1->identify_document !=
    $student2->identify_document);
}
```

El resultado es el siguiente:

```
PS A:\proyectos\htdocs\Academy> vendor/bin/phpunit --filter test_student_duplication tests/Unit/StudentTest.php
PHPUnit 9.5.24 #StandWithUkraine

..                                                                  2 / 2 (100%)

Time: 00:00.157, Memory: 22.00 MB

OK (2 tests, 2 assertions)
```

3.4. Test de eliminación

Esta serie de test pretenden la comprobación efectiva de la eliminación de registros, para ello las funciones crean un estudiante, mediante factories, acto seguido comprueba que, si se haya creado y siendo el caso exitoso, lo elimina finalizando la acción con una respuesta booleana.

```php
public function test_delete_student(){
    $student = Student::factory()->count(1)->make();

    $student = Student::first();

    if($student){
        $student->delete();
    }

    $this->assertTrue(true);
}
```

```php
public function test_delete_student2(){
    $student = Student::factory()->count(2)->make();

    $student = Student::first();

    if($student){
        $student->delete();
    }

    $this->assertTrue(true);
}
```

```php
public function test_delete_student3(){
    $student = Student::factory()->count(3)->make();

    $student = Student::first();

    if($student){
        $student->delete();
    }

    $this->assertTrue(true);
}
```

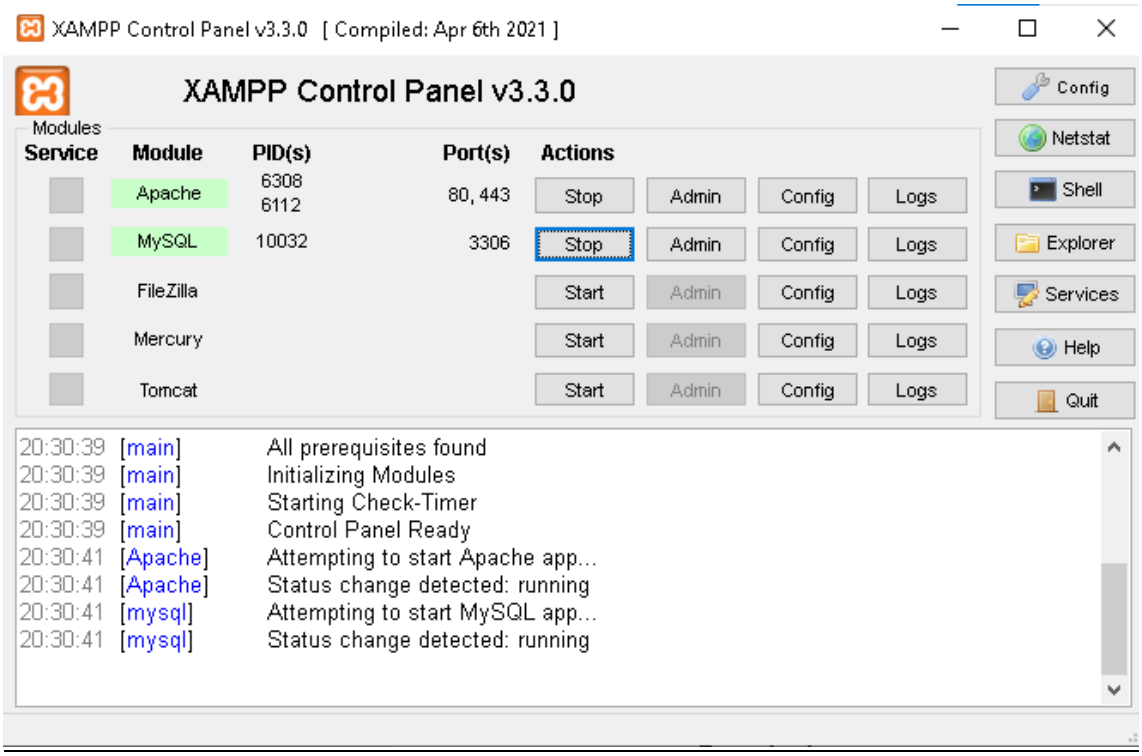
Se realizo la respectiva prueba que se ejecutó satisfactoriamente.

4. **Reporte de errores comunes**

Al realizar la prueba de eliminación del curso se presentó el siguiente error

```
PS A:\proyectos\htdocs\Academy> vendor/bin/phpunit --filter test_delete_course tests/Unit/CourseTest.php
PHPUnit 9.5.24 #StandWithUkraine

EEE                                                                 3 / 3 (100%)

Time: 00:06.243, Memory: 24.00 MB

There were 3 errors:

1) Tests\Unit\CourseTest::test_delete_course
Illuminate\Database\QueryException: SQLSTATE[HY000] [2002] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión (
SQL: select * from `courses` limit 1)
```

Resulta que, para crearse un curso, para eliminarse seguidamente se requiere conectarse a la base de datos, en este caso el servidor local de la base de datos estaba apagado, por lo tanto, no era posible almacenarlos para solucionarlo se ejecutó el programa XAMPP y se habilitaron los módulos de apache y MySQL



Una vez hecha esta corrección el test se ejecuto correctamente

```
PS A:\proyectos\htdocs\Academy> vendor/bin/phpunit --filter test_delete_course tests/Unit/CourseTest.php
PHPUnit 9.5.24 #StandWithUkraine

...                                                                 3 / 3 (100%)

Time: 00:00.709, Memory: 24.00 MB

OK (3 tests, 3 assertions)
PS A:\proyectos\htdocs\Academy>
```