

Proyecto

Desarrollo de Aplicaciones Web

Entrega 3 - Parte 2

Pruebas de unidad y de integración.
Cuestiones y valoración.

Felipe Muñiz Peña
Desarrollo de Aplicaciones Web
Proyecto DAW - Grupo A
Curso 2024/2025

ÍNDICE

<u>Pruebas de unidad y de integracion</u>	<u>3</u>
<u>Planificación y permisos</u>	<u>5</u>
<u>Valoración</u>	<u>5</u>

Pruebas de unidad y de integración:

En primer lugar veo necesario matizar que, dada la naturaleza del proyecto, prácticamente cualquier acción o prueba posible se sirve de la base de datos en mayor o menor medida. Por ello, he intentado que las pruebas unitarias la utilicen lo mínimo posible, o nada.

No obstante, el proyecto cuenta con 33 tests en total.

Para la realización de estos tests, recomiendo la creación de una base de datos llamada `overdubs_test`. El archivo `.env.testing` y las pruebas están diseñados para que automáticamente lleven a cabo la creación de tablas y las migraciones pertinentes y de esta manera no hacer partícipe a la base de datos principal.

Para la ejecución de estas pruebas, realizadas con PHPUnit, basta con introducir el siguiente comando en la carpeta del proyecto:

php artisan test

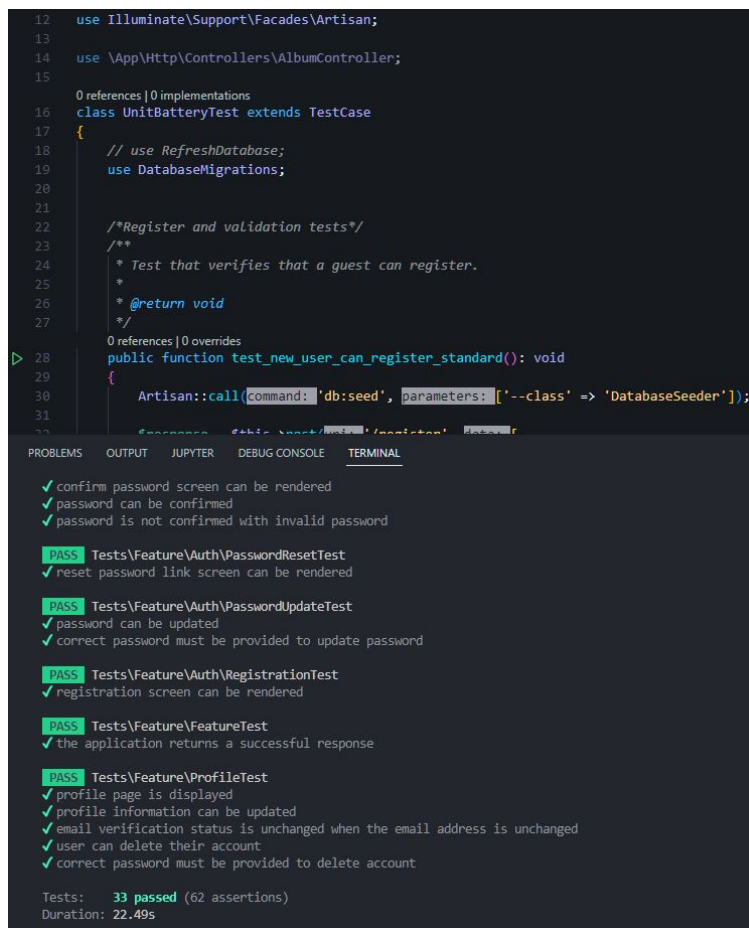
Las pruebas están divididas en tres carpetas, dentro del directorio `/tests/`:

- Unit tests: 10 pruebas. Ubicadas en ***/tests/Unit/UnitBatteryTest.php***.
- Integration tests: 4 pruebas. Ubicadas en ***/tests/Integration/IntegrationBatteryTest.php***.

En ambos archivos, así como en la documentación del proyecto, se encuentran definidas.

- Feature tests: 19 pruebas que proporciona Laravel por defecto. En su mayoría se realizan en torno al registro, login y autenticación de los usuarios. Ubicadas en ***/tests/Feature***.

La documentación del proyecto al completo se encuentra en la carpeta `.phpdoc`. Abriendo el archivo `index.html` se puede navegar a través de ella, la cual incluye por supuesto las pruebas. Esta documentación ha sido comentada en los archivos `.php` correspondientes y generadas con `phpDocumentor`.



```
12 use Illuminate\Support\Facades\Artisan;
13
14 use \App\Http\Controllers\AlbumController;
15
16 0 references|0 implementations
17 class UnitBatteryTest extends TestCase
18 {
19     // use RefreshDatabase;
20     use DatabaseMigrations;
21
22     /*Register and validation tests*/
23     /**
24      * Test that verifies that a guest can register.
25      *
26      * @return void
27      */
28     0 references|0 overrides
29     public function test_new_user_can_register_standard(): void
30     {
31         Artisan::call('db:seed', ['--class' => 'DatabaseSeeder']);
32     }
33 }
```

PROBLEMS OUTPUT JUPYTER DEBUG CONSOLE TERMINAL

- ✓ confirm password screen can be rendered
- ✓ password can be confirmed
- ✓ password is not confirmed with invalid password
- PASS** Tests\Feature\Auth\PasswordResetTest
- ✓ reset password link screen can be rendered
- PASS** Tests\Feature\Auth\PasswordUpdateTest
- ✓ password can be updated
- ✓ correct password must be provided to update password
- PASS** Tests\Feature\Auth\RegistrationTest
- ✓ registration screen can be rendered
- PASS** Tests\Feature\FeatureTest
- ✓ the application returns a successful response
- PASS** Tests\Feature\ProfileTest
- ✓ profile page is displayed
- ✓ profile information can be updated
- ✓ email verification status is unchanged when the email address is unchanged
- ✓ user can delete their account
- ✓ correct password must be provided to delete account

Tests: **33 passed** (62 assertions)
Duration: 22.49s

Pruebas unitarias:

- **test_new_user_can_register_standard:** comprueba que alguien puede registrarse como usuario estándar. Devuelve la confirmación de que puede autenticarse tras el proceso.
- **test_new_user_cant_register_without_password** y **test_new_user_cant_register_wrong_gender:** comprueba que alguien no puede registrarse como usuario estándar con datos no válidos. En el primer caso tratando un campo obligatorio, como es la contraseña, que está vacío. En el segundo caso, tratando un campo no obligatorio, como el género, introduciendo un valor no válido. Devuelven la confirmación de que la sesión contiene errores.
- **test_search:** comprueba que un usuario puede realizar una búsqueda. Devuelve la confirmación de que se lleva a cabo satisfactoriamente mediante un código 200.
- **test_user_cannot_access_admin_content:** confirma que un usuario estándar no puede acceder al contenido de administrador. Devuelve un código de error 404.
- **test_user_cannot_access_data_report:** comprueba que un usuario estándar no puede acceder al informe de datos. Devuelve un código de error 403.
- **test_follow:** comprueba que un usuario puede seguir o dejar de seguir a otro. Se lleva a cabo y se le redirige a la página pertinente, por lo que devuelve un código 302.
- **test_artist_cannot_follow:** comprueba que un artista no puede seguir o dejar de seguir a otra persona. Por ello, devuelve un código 403.
- **test_seeding_database:** comprueba que se puede realizar un *seed* sobre la base de datos.

Pruebas de integración:

- **test_user_sends_review:** comprueba que un usuario puede escribir y enviar una review. Se espera una redirección a la página del álbum una vez completado el procedimiento.
- **test_user_deletes_review:** comprueba que puede eliminar una review solo en caso de que haya sido escrita por él. Espera una redirección.
- **test_show_list:** comprueba que un usuario puede consultar una lista, ya sea creada por él o por otro usuario. Se espera un código 200.
- **test_cannot_delete_list:** comprueba que un usuario no puede borrar una lista creada por otro usuario. Se espera un código de error 405.

¿Ha podido cumplir la planificación? ¿Qué problemas ha tenido?

La planificación se ha podido cumplir satisfactoriamente, pese a los problemas ya mencionados en el otro documento adjunto. Los problemas o incidentes encontrados han afectado fundamentalmente a los tiempos, y por ende, a la planificación. La parte positiva de dichos problemas es que he aprendido a resolverlos. No obstante, debido a ello hay algunas funcionalidades que me gustaría implementar en el futuro, también comentadas previamente en el otro documento.

¿Su proyecto necesita algún tipo de permiso o autorización administrativa?

A la hora de poner en producción el proyecto habrá que adherirse a la normativa de protección de datos personales GDPR (reglamento general de protección de datos). También habría que detallar una política de privacidad, el uso de cookies y términos y condiciones generales a aceptar a la hora de usar Overdubs.

¿Ha establecido algún documento de prevención de riesgos laborales?

En este caso no. Cuando fuera necesario se plantearía un documento al respecto similar al de cualquier profesión realizada en entornos de oficina y utilizando equipo informático.

Realiza una valoración económica respecto a su ejecución.

Considero que este proyecto puede proporcionar ingresos a muy bajo coste. Entre estos costes se ha de incluir el hosting de la web y de la base de datos, pero incluso con las tarifas mínimas obtendremos capacidad más que de sobra para suplir las necesidades de este proyecto. Esto se traduce en que con el mismo coste podríamos desarrollar más proyectos web más adelante, por lo que a Overdubs solo habría que adjudicarle una fracción de estas tarifas.

Por otra parte, desde la concepción de la idea se ha tenido en cuenta la simplicidad como punto de partida, de cara a mantener el coste de mantenimiento lo más bajo posible.

Creo que, teniendo en cuenta que Overdubs está dirigido a un público muy especializado, hay bastante espacio para publicidad orientada a los usuarios, físicamente hablando en el layout de la propia web y figuradamente en cuanto a target, productos y servicios.

Podrían anunciarse, por nombrar algunos ejemplos: nuevos lanzamientos patrocinados, productos como discos, vinilos, equipamiento hi-fi, entradas de conciertos y festivales, etc. También se valoraría la redacción de publisreportajes y artículos.