

TRACKING

Tests de Funcionamiento

Versión 1.0 · Agosto 2026

1. Introducción

i Documento opcional. Los tests no son necesarios para ejecutar la aplicación. Este documento describe cómo montar y ejecutar la batería de pruebas automáticas para verificar que una instalación funciona correctamente. Si solo desea poner el sistema en marcha, consulte el **Manual de Instalación**.

El proyecto incluye una batería de tests automáticos construida sobre **PHPUnit**. Los tests actúan como un cliente HTTP real: lanzan peticiones contra la API instalada y verifican las respuestas, de modo que comprueban la instalación completa (servidor web, PHP, base de datos y configuración), no solo el código.

Los tests residen en `backend/tests`.

2. Requisitos previos

Antes de ejecutar los tests debe tener:

- La aplicación instalada y funcionando según el Manual de Instalación.
- Las dependencias de Composer instaladas (incluyen PHPUnit y Guzzle). Si la carpeta `vendor/` existe, ya las tiene.
- Al menos un usuario de prueba y un administrador de prueba dados de alta y operativos.

⚠ Los tests requieren que el servidor web esté **arrancado**: no se ejecutan contra el código en disco, sino contra la API servida por Apache.

3. Configurar el entorno de pruebas (`.env.testing`)

Los tests leen su configuración de un archivo independiente del de producción, situado junto a él:

```
backend/.env.testing
```

El proyecto incluye `backend/.env.testing` y `backend/.env.testing.example` como plantilla en caso de que no exista el primero; en cuyo caso cópielo con el nombre `.env.testing` y rellene los valores.

3.1 Campos

Campo	Descripción	Ejemplo
API_URL	URL base del backend a testear (debe terminar en /)	<code>http://localhost/tracking/backend/</code>
TEST_DEBUG	Muestra las trazas de petición y respuesta por consola	<code>false</code>
TEST_USER_LOGIN	Login del usuario de prueba (rol usuario)	<code>usuarioPrueba</code>
TEST_USER_PASS	Contraseña del usuario de prueba	<code>(la suya)</code>
TEST_ADMIN_LOGIN	Login del administrador de prueba	<code>adminPrueba</code>
TEST_ADMIN_PASS	Contraseña del administrador de prueba	<code>(la suya)</code>
TEST_CONEXION_ID	Conexión sobre la que se lanzan las pruebas (EVITAR CONEXIONES EN PRODUCCIÓN)	<code>1</code>

⚠ **Las contraseñas van en texto plano deliberadamente.** El test se comporta como un cliente real y debe enviar la credencial tal cual al endpoint de login. El cifrado reside en la base de datos, no en este archivo. Añada `.env.testing` a `.gitignore`.

⊖ **TEST_CONEXION_ID debe apuntar siempre a una base de pruebas, nunca a producción.** Algunos tests escriben en la base de datos.

4. Estructura de la batería

Los tests se organizan en tres suites que se corresponden con las tres capas del sistema:

Suite	Carpeta	Cobertura
general	<code>tests/general</code>	Contrato de la API, autorización y seguridad de sesión. No depende de datos concretos.
trackingweb	<code>tests/trackingweb</code>	Funciones sobre la base de control: usuarios, conexiones y permisos.
tracking	<code>tests/tracking</code>	Funciones de los módulos: pesadas, partidas, bodegas.

4.1 Tests incluidos en la suite general

Archivo	Qué verifica
<code>ApiContratoTest.php</code>	Que la API devuelve siempre JSON limpio con el Content-Type correcto, que solo admite POST y que las acciones desconocidas o malformadas devuelven 400 sin exponer detalles internos.
<code>AuthTest.php</code>	El ciclo de login, comprobación de sesión y logout, con credenciales correctas, incorrectas y ausentes.
<code>AutorizacionTest.php</code>	Que las acciones protegidas rechazan peticiones sin sesión y que el parámetro de conexiones se valida correctamente.

Archivo	Qué verifica
SeguridadSesionTest.php	Que la sesión queda ligada al navegador de origen, que el identificador se regenera al iniciar sesión y que un login fallido no deja sesión utilizable.

5. Ejecución

Abra una terminal en la raíz del proyecto y utilice los siguientes comandos.

5.1 Comandos habituales

Ejecutar toda la batería

```
vendor/bin/phpunit
```

Ejecutar una suite concreta

```
vendor/bin/phpunit --testsuite general
vendor/bin/phpunit --testsuite trackingweb
vendor/bin/phpunit --testsuite tracking
```

Ejecutar un grupo concreto

```
vendor/bin/phpunit --group contrato
vendor/bin/phpunit --group autorizacion
vendor/bin/phpunit --group seguridad
```

Ejecutar un único archivo

```
vendor/bin/phpunit backend/tests/general/AuthTest.php
```

i Para ver el detalle de cada petición y respuesta durante la ejecución, ponga `TEST_DEBUG=true` en el `.env.testing`. Manténgalo en `false` para ejecuciones normales.

5.2 Tests que escriben en base de datos

Los tests marcados con el grupo `escritura` modifican datos reales y están **excluidos por defecto**. Para lanzarlos hay que pedirlo explícitamente:

```
vendor/bin/phpunit --group escritura
```

⚠ No ejecute nunca el grupo escritura contra una base de producción. Incluye operaciones irreversibles como el reajuste masivo de pesadas.

6. Interpretación de los resultados

Símbolo	Significado	Acción
.	Test superado	Ninguna
S	Test omitido (skipped)	Normalmente faltan credenciales en <code>.env.testing</code>
F	Fallo de aserción	El sistema respondió algo distinto de lo esperado: revise el detalle
E	Error de ejecución	Excepción no controlada: revise el log de Apache

Al finalizar, PHPUnit resume el total de tests y aserciones ejecutadas. Una instalación correcta debe terminar sin fallos ni errores; los tests omitidos son aceptables si corresponden a credenciales que no ha configurado.

i Si un test falla, el mensaje indica el valor esperado y el obtenido. Cuando el fallo se produce contra la API, revise también el log de errores de Apache: la respuesta al cliente está deliberadamente resumida para no exponer detalles internos.