Laravel Boost - Problemas de Rendimiento y Seguridad

Problema

Laravel Boost es una herramienta MCP (Model Context Protocol) extremadamente útil para desarrollo asistido por IA, pero puede causar problemas graves si no se configura correctamente:

Problemas en Desarrollo

Rendimiento degradado

- Múltiples instancias del proceso `php artisan boost:mcp` ejecutándose simultáneamente
- El Browser Logs Watcher genera logs masivos (>1MB por llamada)
- Bloquea peticiones web causando timeouts
- Síntomas: páginas que tardan 30+ segundos en cargar o errores 502 de nginx

Consumo excesivo de recursos

- Procesos PHP acumulándose en memoria
- Logs creciendo sin control
- Sistema ralentizado general

Problemas en Producción (CRÍTICOS)

Exposición de información sensible

Laravel Boost expone información que NO debe estar disponible en producción:

- Credenciales de base de datos
- Tokens y secrets del archivo .env
- Estructura completa de la base de datos
- Rutas y configuración de la aplicación
- Logs con información sensible
- Permite ejecutar queries SQL arbitrarias

Riesgo de seguridad

Si Laravel Boost está activo en producción, un atacante podría:

- Leer variables de entorno (DB_PASSWORD, API_KEYS, etc.)
- Explorar la estructura de la base de datos
- Ejecutar consultas de solo lectura
- Acceder a logs con información de usuarios

Solución

1. Publicar Configuración

```
php artisan vendor:publish --tag=boost-config
```

Esto crea `config/boost.php` con opciones configurables.

2. Configurar Variables de Entorno

Añadir al archivo `.env`:

```
# Laravel Boost Configuration
# CRITICAL: MUST be false in production to prevent credential exposure
BOOST_ENABLED=true
BOOST_BROWSER_LOGS_WATCHER=false
```

Importante:

- `BOOST_ENABLED`: Activa/desactiva TODO Laravel Boost
- `BOOST_BROWSER_LOGS_WATCHER`: Monitoreo del navegador (causa problemas de rendimiento)

3. Modificar config/boost.php

```
<?php

return [
    // Se desactiva AUTOMÁTICAMENTE en producción
    'enabled' => env('BOOST_ENABLED', env('APP_ENV') !== 'production'),
```

```
    // Browser logs watcher DESACTIVADO por defecto
    'browser_logs_watcher' => env('BOOST_BROWSER_LOGS_WATCHER', false),
];
```

IMPORTANTE - Error Común:

NO uses `app()->isProduction()` en archivos de configuración:

```
// ❌ INCORRECTO - Causará error "Target class [env] does not exist"
'enabled' => env('BOOST_ENABLED', ! app()->isProduction()),

// ✅ CORRECTO - Usa solo env()
'enabled' => env('BOOST_ENABLED', env('APP_ENV') !== 'production'),
```

Razón: Los archivos de configuración se cargan ANTES de que la aplicación esté completamente inicializada. Solo puedes usar `env()` en estos archivos.

Esta configuración proporciona:

- **Fail-safe**: Se desactiva automáticamente si APP_ENV=production
- **Rendimiento**: Browser logs watcher desactivado por defecto
- **Seguridad**: Múltiples capas de protección

4. Configuración en Producción

En el servidor de producción, el `.env` DEBE tener:

```
APP_ENV=production
BOOST_ENABLED=false
```

NUNCA dejar Laravel Boost activo en producción.

Diagnóstico de Problemas

Si experimentas lentitud o errores 502:

1. Verificar procesos de Boost

```
ps aux | grep boost
```

Si ves múltiples instancias de `php artisan boost:mcp`, hay un problema.

2. Detener procesos problemáticos

```
pkill -f "boost:mcp"
pkill -f "boost:execute-tool"
```

3. Limpiar cache

```
php artisan cache:clear
php artisan config:clear
php artisan optimize:clear
```

4. Reiniciar servicios

Con Laravel Herd:

```
herd restart
```

Con otros entornos:

```
# Reiniciar PHP-FPM y nginx según tu configuración
```

Uso Recomendado

Desarrollo Normal

```
BOOST_ENABLED=true
BOOST_BROWSER_LOGS_WATCHER=false  # Mantener desactivado
```

Solo activa el browser logs watcher cuando necesites debug específico de frontend.

Debug Activo de Frontend

Si necesitas diagnosticar errores de JavaScript/navegador:

```
# Temporal - solo mientras debuggeas
BOOST_BROWSER_LOGS_WATCHER=true php artisan serve
```

Recuerda volver a `false` cuando termines.

Producción

```
APP_ENV=production
BOOST_ENABLED=false
```

Sin excepciones.

Beneficios de la Configuración Segura

- **Rendimiento**: Mejora de hasta 97% en tiempo de respuesta (de 30s a <1s)
- **Seguridad**: Protección automática contra exposición de credenciales
- **Estabilidad**: Previene acumulación de procesos y logs masivos
- **Fail-safe**: Desactivación automática en producción aunque olvides configurarlo

Referencias

- Laravel Boost Documentation
- Configuración recomendada basada en incidentes reales en entornos de desarrollo y producción

---

Última actualización: 2026-01-03

Revision #2
Created 2026-01-03 07:03:56 UTC by Abkrim
Updated 2026-01-03 07:14:41 UTC by Abkrim