Queopius ▪ https://www.queopius.com

DOSSIER DE SERVICIO

Auditoría Backend Laravel

SERVICIO TÉCNICO

Laravel
Health Check

Diagnóstico técnico para aplicaciones Laravel que necesitan más estabilidad, claridad arquitectónica y capacidad real de evolución.

El mejor momento para auditar no es cuando todo está roto. Es cuando la aplicación todavía funciona, pero ya aparecen fricción, bugs recurrentes y cambios menos previsibles.

STACK AUDITADO

- PHP
- Laravel
- APIs
- PHPUnit
- DDD
- Rendimiento

Documento para empresas, startups, SaaS, agencias y equipos técnicos.

RESUMEN

Una auditoría para decidir con menos intuición y más criterio técnico

No es una revisión estética del código. Es una lectura estructurada del backend para entender qué riesgos condicionan negocio, velocidad y mantenibilidad.

Claridad técnica

Lectura objetiva del estado real del backend y sus zonas críticas.

Roadmap accionable

Prioridades, quick wins y secuencia de mejora por impacto técnico.

Decisión defendible

Base clara para invertir, pausar, refactorizar o escalar con criterio.

Problema habitual	Riesgo si no se revisa	Valor de la auditoría
Cada cambio cuesta más	Más bugs y más tiempo de entrega	Prioridades técnicas por impacto
APIs inconsistentes	Integraciones frágiles	Contratos y respuestas más previsibles
Poco testing útil	Regresiones repetidas	Mapa de pruebas críticas con PHPUnit
Rendimiento irregular	Mala experiencia y costes ocultos	Hotspots y quick wins medibles

ENCAJE

Para quién es

Pensada para equipos que ya tienen una aplicación Laravel en producción y necesitan una lectura técnica clara antes de seguir creciendo o rehaciendo partes críticas.

Empresas con producto en marcha

Aplicaciones Laravel que ya sostienen negocio, pero empiezan a acusar fricción técnica en cada cambio importante.

Startups y SaaS

Equipos que necesitan validar si la base actual soporta nuevas funcionalidades, más volumen o una evolución de arquitectura sin romper ritmo.

Agencias que heredan proyectos

Contextos donde hace falta una revisión externa antes de presupuestar mejoras, asumir mantenimiento o comprometer nuevas entregas.

Equipos técnicos con deuda acumulada

Desarrollo interno que necesita priorizar refactor, testing, APIs o rendimiento sin abrir varios frentes a la vez.

BUEN ENCAJE

- Laravel en producción
- Necesidad de priorizar mejoras
- Interés real en mantenibilidad

NO ES EL MEJOR ENCAJE

- Solo se busca “mirar por encima”
- No hay acceso suficiente al código
- Se espera desarrollo completo sin diagnóstico

HALLAZGOS

Problemas que detecta

La auditoría no se queda en observaciones genéricas: baja a patrones concretos que suelen bloquear velocidad, estabilidad y mantenibilidad en proyectos Laravel existentes.

Responsabilidades mezcladas

Controladores grandes, modelos con demasiada lógica y servicios sin un límite claro entre aplicación, dominio e infraestructura.

APIs difíciles de mantener

Endpoints inconsistentes, validaciones desalineadas, respuestas poco previsibles y contratos que complican integraciones futuras.

Testing que no protege lo crítico

Cobertura baja o mal enfocada, suites lentas y ausencia de pruebas útiles sobre reglas de negocio, flujos sensibles o regresiones frecuentes.

Hotspots de rendimiento

Consultas pesadas, relaciones mal resueltas, N+1, jobs con fricción y decisiones de caché o colas que degradan la experiencia.

Deuda técnica que ya condiciona negocio

Duplicidades, acoplamiento, decisiones heredadas y zonas del proyecto que el equipo evita tocar porque cada cambio es una apuesta.

Falta de criterio operativo

Documentación dispersa, estructura poco legible y ausencia de prioridades claras para decidir qué arreglar primero y qué puede esperar.

COBERTURA

Cobertura 01

Qué reviso: arquitectura, código, APIs, datos y testing.

Arquitectura

Capas, módulos, responsabilidades, acoplamiento, límites de dominio y señales de monolito difícil de evolucionar.

Estructura del proyecto

Organización de carpetas, convenciones, servicios, acciones, DTOs, eventos, jobs y consistencia interna.

Controladores, modelos y servicios

Detección de lógica dispersa, modelos sobrecargados, controladores grandes y servicios ambiguos.

APIs

Diseño de endpoints, validación, contratos, errores, autenticación, autorización y documentación OpenAPI/Swagger.

Base de datos

Migraciones, relaciones, índices, consultas pesadas, N+1 y puntos de fricción para escalabilidad.

Testing con PHPUnit

Cobertura útil, equilibrio unitario/integración, velocidad de suite y protección de reglas críticas de negocio.

COBERTURA

Cobertura 02

Qué reviso: seguridad, rendimiento, deuda, documentación y entrega.

Seguridad básica

Validaciones, permisos, exposición de datos, dependencias y configuración sensible.

Rendimiento

Consultas, caché, colas, jobs, tiempos de respuesta, recursos y cuellos de botella.

Deuda técnica

Duplicación, acoplamiento, zonas frágiles, complejidad accidental y coste de cambio.

Documentación

README, onboarding, decisiones de arquitectura, contratos API y contexto crítico.

DevOps y entrega

Entorno local, Docker, CI/CD, despliegues, backups y reproducibilidad del proyecto.

DDD y modularidad

Límites de dominio, lenguaje ubicuo, casos de uso y separación entre aplicación e infraestructura.

CRITERIO

Cómo convierto observaciones técnicas en prioridades defendibles

Cada hallazgo se clasifica para que negocio y tecnología puedan decidir qué hacer primero, qué posponer y qué convertir en fase posterior.

Criterio	Pregunta que responde	Salida esperada
Impacto	¿Afecta estabilidad, clientes, equipo o ingresos?	Crítico / Alto / Medio / Bajo
Urgencia	¿Puede esperar o bloquea entregas inmediatas?	Ahora / Próximo sprint / Roadmap
Complejidad	¿Requiere refactor profundo o quick win?	Baja / Media / Alta
Riesgo	¿Puede generar regresiones o dependencia técnica?	Zona segura / Zona sensible
Valor	¿Mejora velocidad, mantenibilidad o confianza?	Prioridad técnica defendible

El informe no solo enumera problemas: agrupa hallazgos, impacto, riesgo, evidencia técnica y recomendación de acción para facilitar la decisión.

ENTREGABLE

Entregable

El objetivo no es entregarte una lista abstracta de observaciones, sino una lectura técnica accionable para tomar decisiones con criterio.

- **Informe técnico**
- **Riesgos priorizados**
- **Quick wins**
- **Roadmap de mejora**

PROCESO

Proceso de trabajo

La auditoría está planteada para ser directa, pragmática y útil: menos ceremonia, más señales claras y priorización accionable.

PASO DE TRABAJO

Contexto y acceso

PASO DE TRABAJO

Revisión técnica backend

PASO DE TRABAJO

Priorización y entregable

ALCANCE

Qué incluye y qué no incluye la auditoría

Definir límites desde el inicio evita malentendidos y permite que el diagnóstico sea útil, acotado y defendible.

INCLUYE

- Revisión técnica del backend Laravel
- Lectura de arquitectura y responsabilidades
- APIs, testing, datos y rendimiento
- Deuda técnica priorizada

NO INCLUYE

- Desarrollo de nuevas features
- Refactor completo dentro de la auditoría
- Garantía de ausencia total de errores
- Pentest o auditoría legal

Repositorio

Acceso al código y ramas relevantes.

Contexto

Objetivos, dolores, funcionalidades críticas y prioridades.

Entorno

Instrucciones de instalación, variables y entorno controlado.

Documentación

README, API docs, diagramas o decisiones existentes si las hay.

SIGUIENTE FASE

Tres caminos posibles según el diagnóstico

La auditoría no obliga a ejecutar. Sirve para decidir con claridad qué camino tiene sentido y cuál no.

Sprint de estabilización Laravel

Intervención corta para resolver quick wins, bugs críticos, endpoints frágiles y zonas prioritarias.

Encaja cuando hay problemas concretos que se pueden corregir con impacto visible.

Refactor o evolución técnica

Fase planificada para separar responsabilidades, introducir modularidad y reorganizar APIs.

Encaja cuando la base necesita una mejora estructural sin reescribir a ciegas.

Backend Partner mensual

Apoyo recurrente para mantenimiento evolutivo, revisión de PRs, arquitectura, testing y documentación.

Encaja cuando el equipo necesita continuidad técnica senior.

RESULTADO

Resultado para el cliente

Lo valioso no es solo detectar problemas, sino salir con claridad suficiente para decidir qué tocar, cuándo y con qué impacto esperable.

Más claridad técnica

Una lectura externa y estructurada del estado real del backend, sin depender solo de intuición interna.

Menos riesgo al priorizar

Capacidad de distinguir entre urgencias reales, deuda tolerable y mejoras que sí mueven estabilidad o velocidad.

SIGUIENTE PASO

Si tu Laravel ya mueve negocio, la base técnica también tiene que estar a la altura.

La auditoría te ayuda a ver dónde están los riesgos, qué conviene corregir primero y cómo recuperar capacidad de evolución sin rehacer a ciegas.

Reservar una conversación

También puedes escribir a dev.queopius@gmail.com.

Queopius
https://www.queopius.com
Queopius A.S.