AUDITORÍA PROFUNDA DE SEGURIDAD Y FUNCIONALIDAD

Plataforma Educativa

Reporte ejecutivo de hallazgos

Auditoría completa de seguridad y funcionalidad de la plataforma antes del lanzamiento comercial. 156 archivos PHP analizados, 47 tablas de base de datos, 60+ hallazgos identificados.

156
archivos PHP analizados

47
tablas BD revisadas

6
críticos de seguridad

25
críticos funcionales

28
hallazgos medios

12
hallazgos bajos

1. Resumen ejecutivo

Se realizó una auditoría profunda de la plataforma educativa SaaS multi-tenant antes del lanzamiento comercial. El análisis cubrió 156 archivos PHP, 47 tablas de base de datos, y todos los flujos críticos: registro del rector, importación de datos, creación de materias, y los 4 portales (rector, docente, estudiante, padre).

El resultado: se identificaron 60+ hallazgos entre críticos, medios y bajos. La buena noticia es que la mayoría son corrregibles en 3-5 días de trabajo enfocado. La mala noticia es que NO se debería lanzar la campaña de Facebook Ads hasta resolver al menos los 6 hallazgos críticos de seguridad y los 10 bugs funcionales que afectan más usuarios.

> *RECOMENDACIÓN PRINCIPAL: NO lanzar la campaña hasta resolver los 16 hallazgos críticos (6 de seguridad + 10 funcionales). Tiempo estimado: 3-5 días de trabajo enfocado. Después de eso, la plataforma estará sólida para recibir clientes reales.*

Hallazgos por severidad

Categoría	🔴 Críticos	🟡 Medios	🟢 Bajos
Seguridad	6	7	5
Funcionalidad	25	21	7
Total	**31**	**28**	**12**
% del total	44%	40%	16%

Hallazgos más graves detectados

- 🔴 **#1:** download.php permite descargar documentos de CUALQUIER institución sin autenticación. Un atacante con la URL puede robar documentos de admisión con cédulas de menores, recibos de pago, fotos de galería, etc.
- 🔴 **#2:** Webhook de Wompi (pagos) en modo sandbox acepta cualquier POST sin verificar firma. Permite marcar suscripciones como pagadas sin pagar.
- 🔴 **#3:** Importación de Excel SIN CSRF permite a un atacante inyectar estudiantes y padres con emails controlados, gaining acceso al portal de padres de una institución víctima.
- 🔴 **#4:** Estudiantes no se asignan automáticamente a materias después de importarlos. El docente ve "Sin estudiantes" aunque hay 200 cargados.
- 🔴 **#5:** Crear materia NO auto-asigna estudiantes. Hay que hacerlo manualmente materia por materia.
- 🔴 **#6:** El rector puede iniciar sesión aunque la institución esté suspendida (sin pagar).

2. Hallazgos críticos de seguridad

Estos 6 hallazgos representan riesgos inmediatos para la seguridad de los datos de los estudiantes y la integridad financiera de la plataforma. Deben corregirse ANTES de cualquier lanzamiento.

CRITICO download.php sin autenticación — acceso a documentos de cualquier institución

📍 download.php líneas 1-85

Descripción: El script no carga auth.php ni institution.php. Sirve cualquier archivo dentro de uploads/ a quien conozca la ruta, sin verificar login ni institución.

Impacto: Un atacante puede descargar documentos de admisión (con cédulas de menores), recibos de pago, fotos de galería, tareas entregadas, boletines, etc. de CUALQUIER institución. Las rutas son predecibles: uploads/{slug}/galeria/{album_id}/foto_*.jpg. Los slugs son públicos (aparecen en la URL del sitio).

Solución: Agregar require_login(), require_institution_access(), validar que el archivo pertenezca a la institución del usuario logueado, y requerir rol adecuado (padre solo puede ver archivos de sus hijos).

CRITICO Webhook de Wompi (pagos) en fail-open acepta cualquier POST en sandbox

📍 includes/payments.php líneas 407-411

Descripción: Si WOMPI_EVENTS_SECRET no está configurado y WOMPI_ENV !== "production", la función payments_wompi_event_valid() retorna true para cualquier payload recibido en /api/payments/webhook.php.

Impacto: Un atacante puede enviar un POST al webhook con status=APPROVED y marcar como pagada cualquier suscripción pendiente. Pérdida de ingresos directa: obtención gratuita de planes Premium/Integral ($180.000-$350.000 COP/mes).

Solución: Eliminar el fail-open. Si no hay secret configurado, el webhook debe rechazar todas las peticiones con HTTP 401. Implementar verificación HMAC-SHA256 del header x-signature de Wompi.

CRITICO Importación de Excel SIN CSRF permite inyección de padres con emails controlados

📍 admin/importar-datos.php líneas 194, 535, 613, 778, 981

Descripción: Los 5 handlers POST de importación (consolidado, padres, estudiantes, docentes, relaciones) NO llaman a require_csrf() o verify_csrf().

Impacto: Un atacante crea una página maliciosa con auto-submit al formulario de importación. El rector logueado la visita y se ejecuta la importación con un Excel crafted que contiene un acudiente con email del atacante. El atacante ahora tiene credenciales de un padre asociado a estudiantes reales de la institución.

Solución: Agregar require_csrf() al inicio de cada handler POST. Agregar <?= csrf_field() ?> en todos los formularios.

CRITICO CSRF en configuración institucional — permite modificar datos críticos

📍 admin/configuracion.php línea 38

Descripción: El POST que actualiza TODOS los datos de la institución (nombre, logo, colores, email, teléfono, escala de notas, NIT, DANE, etc.) no tiene protección CSRF.

Impacto: Un atacante puede CSRF-modificar el email oficial para phishing contra padres, cambiar nota_aprobatoria a 1.0 para que todos aprueben, o alterar el NIT/resolución (impacto legal/facturación).

Solución: Agregar require_csrf() al inicio del bloque POST y <?= csrf_field() ?> en el form.

CRITICO Demo "un-clic" permite acceso de rector sin contraseña en colegio-demo

📍 pages/portal.php líneas 155-205

Descripción: En la institución con slug colegio-demo, cualquier visitante puede enviar POST demo_role=rector y obtener acceso de rector completo sin contraseña. El flag DEMO_ACCESO_UN_CLIC por defecto es true.

Impacto: Un atacante obtiene acceso rector a la institución demo. Si por error se cargan datos reales en colegio-demo (común para testing), cualquier atacante los ve. Permite estudiar el panel admin para diseñar ataques contra otras instituciones.

Solución: Mover el flag a false por defecto. Exigir autenticación HTTP Basic adicional sobre el dominio para la institución demo.

CRITICO CSRF en suscripción y creación de pagos — permite generar órdenes de pago falsas

📍 admin/suscripcion.php línea 70

Descripción: El handler POST crear_pago no llama a require_csrf(). Permite iniciar checkout Wompi o generar orden de transferencia sin CSRF.

Impacto: Atacante puede generar masivamente referencias PE- que luego podría intentar marcar como pagadas vía el webhook vulnerable. DoS administrativo con cientos de pagos pendientes.

Solución: Agregar require_csrf() al inicio del handler y <?= csrf_field() ?> en el form de pago.

3. Top 10 bugs funcionales que afectan más usuarios

Estos son los 10 bugs funcionales que tienen mayor impacto en la experiencia del usuario. Están ordenados por prioridad de corrección.

CRITICO #1 — Estudiantes no se asignan automáticamente a materias

📍 admin/importar-datos.php, admin/materias.php, admin/usuarios.php

Descripción: Tras importar estudiantes o crearlos manualmente, NO se asignan a las materias de su grado. Hay que hacerlo manualmente materia por materia.

Impacto: El flujo "importar y ya funciona" está completamente roto. El docente ve "Sin estudiantes" aunque haya 200 cargados. El rector debe hacer trabajo manual repetitivo.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

CRITICO #2 — me_asignar_por_grado corta la búsqueda en el primer match

📍 includes/materia_estudiante_helper.php líneas 155-224

Descripción: La función tiene 4 intentos progresivos, pero solo ejecuta el siguiente si el anterior no encontró NADA. Si el Intento 1 encuentra algunos estudiantes, los demás con variantes del nombre del grado no se asignan.

Impacto: En una misma aula con nombres de grado inconsistentes (típico tras importar de Excel), el docente ve 5 de 30 estudiantes. Los demás no se vinculan.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

CRITICO #3 — me_normalizar_grados_institucion colisiona por número de grado

📍 includes/materia_estudiante_helper.php líneas 533-539

Descripción: Si dos grados oficiales comparten número (ej: "1° Primaria" y "1° Bachillerato" ambos tienen número 1), el mapa se sobrescribe. El último grado procesado gana.

Impacto: Estudiantes de primaria terminan migrados a bachillerato. Sus materias, horarios y calificaciones se mezclan.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

CRITICO #4 — admin/onboarding.php consulta tabla "estudiantes" que no existe

📍 admin/onboarding.php líneas 68-70

Descripción: La consulta SELECT COUNT(*) FROM estudiantes WHERE institucion_id = ? falla porque la tabla estudiantes NO existe (solo usuarios WHERE rol="estudiante"). El resultado siempre es 0.

Impacto: El item "Cargar estudiantes" del checklist de onboarding NUNCA se marca como completado, incluso si el rector ya cargó 500 estudiantes. Falsa sensación de configuración incompleta.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

CRITICO #5 — Renombrar un grado NO propaga el cambio a entidades que lo referencian

📍 admin/grados.php líneas 62-74

Descripción: Al cambiar grados.nombre, el UPDATE no afecta a usuarios.grado_actual, materias.grado, horarios.grado, tareas.grado, circulares.destino_grado. Todas referencian el grado por nombre VARCHAR.

Impacto: Renombrar "Grado 1°" a "Primero A" deja a todos los estudiantes con grado_actual="Grado 1°" que ya no matchea ningún grado oficial. La importación posterior los reporta como "grado no oficial". Los filtros no los muestran.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

CRITICO #6 — familias.php permite crear relaciones duplicadas

📍 admin/familias.php líneas 22-39

Descripción: El INSERT en padre_estudiante NO verifica si la relación ya existe. El schema no tiene UNIQUE KEY en (padre_id, estudiante_id).

Impacto: El rector puede crear 3 veces la misma relación "Ana → Juan" sin darse cuenta. El portal de padres mostraría al hijo 3 veces en la lista, las estadísticas se inflan, las autorizaciones se duplican.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

CRITICO #7 — Portal de padre multiplica circulares sin leer por número de hijos

📍 portal/padre/index.php líneas 37-45

Descripción: El foreach de hijos ejecuta la query de circulares no leídas para el usuario padre (no por hijo). Si el padre tiene 3 hijos, la query se ejecuta 3 veces con los mismos parámetros y se suman 3 veces el mismo conteo.

Impacto: Si hay 5 circulares sin leer, el padre ve "15 circulares sin leer" (3 hijos × 5). Confusión y falsa urgencia.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

CRITICO #8 — Portal de estudiante consulta tareas por grado, no por materia_estudiante

📍 portal/estudiante/index.php líneas 27-30

Descripción: La query de tareas pendientes filtra t.grado = ? con el grado_actual del estudiante. Si solo toma 5 de 8 materias del grado (asignación parcial via materia_estudiante), verá tareas de TODAS las materias del grado.

Impacto: Estudiante ve tareas de materias que no toma. No puede entregarlas. Se confunde.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

CRITICO #9 — Activación del trial fuera de transacción — estado inconsistente

📍 includes/onboarding.php líneas 225-239

Descripción: Tras el commit inicial, el bloque trial ejecuta dos UPDATEs (instituciones y altas_web) FUERA de transacción. Si el proceso muere entre los dos UPDATEs, la institución queda activa pero altas_web.estado sigue pendiente_pago.

Impacto: El rector puede loguearse y operar, pero el sistema interno cree que el alta sigue pendiente. El email de bienvenida no se envía. El soporte no tiene cómo detectar el caso.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

CRITICO #10 — Rector puede iniciar sesión aunque la institución esté suspendida

📍 includes/onboarding.php + pages/portal.php líneas 239-247

Descripción: En alta_web_iniciar() la institución se crea con activo=0 y suscripcion_estado="suspendida", pero el usuario rector se crea con activo=1. La consulta de login solo filtra usuarios.activo=1, sin verificar instituciones.activo.

Impacto: Entre la creación de la cuenta y la activación del pago/trial, el rector ve un panel "roto" (sin datos, sin funciones). Mensajes confusos. Si el pago nunca se confirma, la cuenta queda huérfana.

Solución: Ver detalles técnicos en el reporte completo de auditoría funcional.

4. Plan de remediación

El plan de remediación está organizado en 4 fases. Cada fase debe completarse antes de pasar a la siguiente. La idea es resolver primero los problemas que representan mayor riesgo y luego ir refinando.

Fase 1 — Críticos de seguridad (Día 1-2)

Estos 6 hallazgos deben resolverse ANTES de cualquier otra cosa. Representan riesgos inmediatos para los datos de estudiantes y la integridad financiera.

#	Hallazgo	Tiempo	Responsable
1	download.php sin auth	2h	Desarrollador
2	Webhook Wompi fail-open	1h	Desarrollador
3	CSRF en importar-datos.php	1h	Desarrollador
4	CSRF en configuracion.php	30min	Desarrollador
5	CSRF en suscripcion.php	30min	Desarrollador
6	Demo un-clic flag=false	15min	Desarrollador

⏱️ **Tiempo total estimado: 5 horas**

Fase 2 — Bugs funcionales críticos (Día 2-3)

Estos bugs son los que afectan la experiencia del usuario. Aunque no son riesgo de seguridad, son los que harán que un cliente se frustre y cancele su suscripción.

#	Bug funcional	Tiempo	Impacto
1	Auto-asignar estudiantes a materias (al crear/editar)	3h	Crítico
2	Fix me_asignar_por_grado (acumular intentos)	1h	Alto
3	Fix me_normalizar_grados colisión	1h	Alto
4	Fix onboarding tabla estudiantes	30min	Medio
5	Propagar renombrado de grados	2h	Alto
6	UNIQUE constraint en	30min	Medio

	padre_estudiante		
7	Fix portal padre conteo circulares	30min	Medio
8	Fix portal estudiante tareas por materia	1h	Alto
9	Transacción en activación trial	1h	Medio
10	Bloquear login institución suspendida	30min	Medio

⏱️ **Tiempo total estimado: 11 horas**

Fase 3 — Hallazgos medios (Día 4)

Estos hallazgos no bloquean el lanzamiento pero deben resolverse en la primera semana post-lanzamiento. Incluyen mejoras de UX, validaciones adicionales, y prevención de casos edge.

- CSRF en superadmin/leads.php y demos.php
- CSRF en migrar-excel.php y materia-estudiantes.php
- Validar longitud mínima de contraseña en edición de usuarios
- Marcar email_verificado=1 al crear usuarios manualmente
- Validar institución en relaciones padre-estudiante
- Soft-delete en vez de DELETE físico en familias.php
- Mostrar materias en portal del estudiante
- Filtrar calificaciones por periodo actual
- Ocultar calificaciones de materias desactivadas
- Fix fallback grados en usuarios.php cuando no hay grados
- Unificar tipos de documento entre estudiantes.php e import_helpers.php
- Mostrar contraseña temporal en pantalla (no en URL)
- Persistir credenciales más allá de la sesión
- Optimizar consultas SHOW COLUMNS en importación

⏱️ **Tiempo total estimado: 8 horas**

Fase 4 — Hallazgos bajos (Post-lanzamiento)

Mejoras cosméticas y de robustez. Pueden esperar 2-4 semanas sin afectar la experiencia del usuario.

- Implementar cabecera Content-Security-Policy
- Ocultar información sensible en mensajes de error PDO
- Forzar HTTPS con HSTS preload
- Agregar token expiration a verificación de email

- Estilos CSS para todas las relaciones de acudiente
- Mover niveles y jornadas a tabla de configuración
- Optimizar batch UPDATEs en normalización de grados
- Documentar comportamiento de validación de relaciones

⏱️ **Tiempo total estimado: 6 horas**

5. Cronograma y próximos pasos

Timeline estimado

Día	Actividad	Entregable
Día 1	Implementar Fase 1 (seguridad crítica)	Parche seguridad v1
Día 2	Implementar Fase 2 (bugs críticos) — Parte 1	Parche funcional v1
Día 3	Implementar Fase 2 (bugs críticos) — Parte 2 + Testing E2E	Parche funcional v2
Día 4	Implementar Fase 3 (hallazgos medios)	Parche mejoras v1
Día 5	Prueba E2E completa + Lanzamiento campaña	Plataforma lista

Próximos pasos inmediatos

- Revisar este reporte completo y priorizar según tu criterio
- Confirmar conmigo qué hallazgos vas a atacar primero (sugiero Fase 1 + Fase 2)
- Generar el paquete unificado de fixes (no parches sueltos)
- Aplicar fixes en servidor de staging o directamente en producción con respaldo
- Ejecutar prueba E2E completa siguiendo la guía actualizada
- Si todo pasa, lanzar campaña de Facebook Ads

> *COMPROMISO: Yo me encargo de generar TODOS los fixes en un solo paquete unificado y probado. Tú solo tendrás que subirlo al servidor y hacer la prueba final. Nada de parches sueltos.*

Lo que NO debes hacer

- 🚫**:** NO lances la campaña de Facebook Ads hasta completar Fase 1 y Fase 2
- 🚫**:** NO sigas aplicando parches sueltos uno por uno. Espera el paquete unificado
- 🚫**:** NO borres la institución de prueba hasta confirmar que todos los fixes funcionan
- 🚫**:** NO modifies código directamente en producción sin respaldo previo

Lo que SÍ debes hacer

- ✅**:** Respalda TODO antes de aplicar cualquier fix (código + BD)

- ✅**:** Prueba cada fix en una institución de prueba antes de producción
- ✅**:** Documenta cada fix aplicado (fecha, archivo, qué cambió)
- ✅**:** Mantén este reporte como referencia viva durante las próximas 2 semanas

6. Conclusión

Wilson, la auditoría confirma lo que sospechabas: la plataforma tiene bugs que la hacen no apta para recibir tráfico pago en este momento. Pero la buena noticia es que todos los hallazgos son corregibles y la mayoría en pocas horas.

La inversión de 3-5 días en aplicar estos fixes te dará la tranquilidad de que cuando un rector real encuentre un bug a las 2am un domingo, no será un bug crítico sino un detalle cosmético. Y esa es la diferencia entre una plataforma amateur y una profesional.

> *La pregunta no es "¿puedo permitirme 5 días sin lanzar la campaña?" La pregunta es "¿puedo permitirme lanzar la campaña y que los primeros 10 clientes encuentren bugs que los hagan cancelar en el primer mes?"*

Tu decisión de pausar la campaña y hacer la auditoría fue la correcta. Ahora aprovechemos este momentum para resolver todo de una vez y lanzar con confianza. Estoy a tu disposición para generar los fixes inmediatamente después de que me confirmes qué quieres atacar primero.