**CondorShop — E-commerce Web**

Nombre Alumnos : Juan Rodríguez

Benjamín Huenupe

Carrera : Ingeniería en informática

Asignatura : Portafolio de Título

Docente : Marco Valenzuela

**Índice**

[**Análisis de Caso para CondorShop — E-commerce Web 3**](#_heading=h.cd5xkcyjlv4x)

[Descripción del Caso 3](#_heading=h.vp4pba5utfxh)

[Objetivos 3](#_heading=h.z32tfm1qhsxm)

[Análisis de Requisitos 4](#_heading=h.nd2s694x6wbu)

[Requisitos No Funcionales 5](#_heading=h.wld739woft44)

[Recopilación de Datos 5](#_heading=h.qzcxbq4ztlfd)

[Análisis de Problemas 6](#_heading=h.1qew3ju825u1)

[Propuestas de Solución 7](#_heading=h.k5jb17a33w8i)

[Evaluación de Alternativas 7](#_heading=h.18xgfcjcharl)

[Recomendaciones 8](#_heading=h.jdezs0hcvj5d)

[Plan de Acción 9](#_heading=h.mw0mpr36bgpe)

[Conclusión 9](#_heading=h.acn5oq9vcwwx)

**Análisis de Caso para CondorShop — E-commerce Web**

**Descripción del Caso**

* CondorShop es una tienda en línea que ofrece un catálogo de productos (tecnología, moda, hogar, etc.). El objetivo es proveer una experiencia de compra completa desde el navegador: buscar, filtrar, ver detalle, añadir al carrito y pagar. La plataforma registra órdenes y stock en MySQL, y muestra confirmaciones y número de orden. El sitio es responsive (móvil y desktop) y optimizable para SEO (páginas con metadatos por producto).

**Objetivos**

* Habilitar catálogo, carrito y checkout con persistencia en MySQL.
* Implementar seguridad básica (validación de entradas, sanitización, TLS en producción).
* Preparar SEO (URLs por producto, <title>, <meta description>, sitemap.xml, robots.txt).
* Integrar pagos reales (Stripe Checkout) y correo de confirmación (SendGrid/Mailgun).
* Asegurar usabilidad (responsive, accesible) y rendimiento (tiempos de respuesta aceptables).

Análisis de Requisitos

**Requisitos Funcionales**

1. Catálogo: listar productos con búsqueda, filtros por categoría y ordenamiento.
2. Detalle de producto: ver nombre, precio, stock, descripción e imagen.
3. Carrito: añadir/editar/eliminar ítems, calcular envío (gratis sobre CLP 50.000).
4. Checkout: formulario de cliente; crear orden y descontar stock (transacción en MySQL).
5. Confirmación: mostrar número de orden y estado de pago.
6. Admin (básico): crear/editar/eliminar productos (protegido por ADMIN\_KEY).
7. API REST: /api/products, /api/products/:id, /api/orders.
8. (Opcional) Cuentas de usuario (registro/login) para historial de compras.

Requisitos No Funcionales

1. Compatibilidad: Web responsive (móvil/desktop), navegadores modernos.
2. Seguridad: OWASP básica (validación, sanitización, uso de parámetros SQL, protecciones en cabeceras, TLS).
3. Usabilidad: UI clara, accesible (contraste, foco, etiquetas).
4. Rendimiento: TTI bajo; endpoints < 300 ms en promedio; paginado si catálogo crece; cache de CDN en estáticos.
5. Disponibilidad (en producción): ≥ 99.9% con proveedor gestionado.

Recopilación de Datos

* Fuentes primarias: entrevistas breves a usuarios y pruebas de usabilidad (5–7 participantes) sobre el flujo buscar → carrito → checkout en LlamaShop.
* Fuentes secundarias: benchmark de 3–5 e-commerce locales (búsqueda, filtros, ficha, envío, checkout).
* Instrumentos: guion de entrevista, 3 tareas de usabilidad, encuesta corta (≤10 preguntas).
* Métricas: tasa de éxito por tarea, tiempos, errores/abandono, comentarios clave.
* Procedimiento: sesiones presenciales/remotas con consentimiento; registro en planilla y síntesis de hallazgos + mejoras priorizadas.

Análisis de Problemas

* Pagos reales ausentes: el flujo es simulado; no hay conciliación ni recibos oficiales.
* SEO limitado: no existen URLs únicas por producto con metadatos; falta robots.txt/sitemap.xml.
* Administración mínima: no hay panel con login/roles para crear/editar productos o ver pedidos de forma segura.
* Concurrencia de stock: riesgo de “over-selling” sin bloqueo transaccional/consistencias más estrictas en la BD.
* Observabilidad: escasez de logs estructurados, auditoría y métricas (no hay alertas ante caídas o errores).
* Backus: plan de respaldos y restauración no formalizado.
* Seguridad: validación/sanitizaciones incompletas; falta hardening (cabeceras, rate-limit, manejo de secretos).

Propuestas de Solución

* Arquitectura: mantener Express + MySQL (monolito) y fortalecer API/estáticos.
* Pagos: Transbank Webpay Plus: flujo create → redirect → return → commit; guardar buy\_order y token\_ws; manejar integration/production y errores/abandonos.
* SEO: URLs por producto con metadatos + robots.txt y sitemap.xml.
* Admin: panel con login (JWT) para CRUD de productos y vista de pedidos.
* Stock/consistencia: órdenes en transacción SQL (verificación final de stock antes de confirmar).
* Operación: logs/métricas, backups automáticos y hardening (helmet, rate-limit, validación de inputs).
* Operación: logs/métricas, backups automáticos y hardening (helmet, rate-limit, validación de inputs).

Evaluación de Alternativas

Opción 1: Desarrollo Interno (Express + MySQL + Transbank)

* Costo: Bajo–Medio
* Viabilidad Técnica: Alta; control total del backend, API y flujo Webpay (create → redirect → return → commit).
* Beneficios: Personalización completa (búsqueda, filtros, SEO), integración directa con Transbank, mejor manejo de stock/órdenes y datos, sin lock-in.

Opción 2: Outsourcing / Plataforma (Shopify/WooCommerce + plugin Transbank)

* Costo: Moderado (suscripciones/plugins/comisiones)
* Viabilidad Técnica: Depende del proveedor/tema; limita cambios finos en checkout/SEO e integraciones específicas.
* Beneficios: Menor carga interna y salida rápida; catálogo y pagos acelerados con soporte del proveedor

Recomendaciones

* Consolidar Express + MySQL como base, agregando Stripe, SEO, panel admin y observabilidad.
* Formalizar backups y plan de recuperación.
* Establecer pipeline de despliegue (Render/Railway) con variables de entorno seguras y logs centralizados.
* Programar pruebas de usabilidad y mejorar continuamente búsqueda/filtros y checkout (móvil primero).

Plan de Acción

Fase 1 — Fundaciones (1–2 semanas)

* Stripe (sandbox): endpoint de sesión + webhook; guardar payment\_intent\_id.
* SEO: /producto/:id con metadatos, robots.txt, sitemap.xml; registrar dominio en Search Console.
* Seguridad base: helmet, rate-limit, validación/sanitización, .env y rotación de claves.
* Observabilidad: logs JSON, manejo centralizado de errores.

Fase 2 — Admin & Calidad (1–2 semanas)

* /admin con login (JWT + roles). CRUD de productos; listado de pedidos; exportación CSV.
* Email de confirmación de pedido.
* Estrategia de stock (transacciones y verificación final antes de capturar).
* Backups automáticos y prueba de restauración.

Fase 3 — Producción y Escalamiento (1 semana)

* Despliegue (API en Render/Railway + dominio + HTTPS).
* CDN para estáticos e imágenes (S3/R2).
* Monitoreo (alertas por 5xx/latencias) y revisión post-lanzamiento (métricas de conversión).

Entregables y criterios de aceptación

* Pasarela operativa en sandbox con órdenes “paid”.
* Productos indexables con metadatos y sitemap aceptado en Search Console.
* Panel admin funcionando y protegido; auditoría básica de cambios.
* Backups verificados; plan de restauración documentado.

Conclusión

La adaptación del caso al contexto de CondorShop consolida una tienda web funcional y escalable sobre Node.js (Express) + MySQL. Al priorizar pagos reales, SEO, panel de administración, seguridad y observabilidad, el proyecto queda preparado para indexación en Google, operación confiable y crecimiento ordenado, manteniendo un time-to-market competitivo y control técnico total.